Spatially resolving the dynamics and structure of protein networks in adhesion sites

Rahuman S. Malik Sheriff

Department of Systemic Cell Biology
Max Planck Institute of Molecular Physiology

Supervisor: Dr.(IL) Eli Zamir

Reviewer 1: Prof. Dr. Philippe Bastiaens

Reviewer 2: Prof. Dr. Katja Ickstadt

A thesis submitted for the degree of
Philosophiae Doctor (PhD), Dr.rer.nat at Faculty of Chemistry

Technische Universität Dortmund
2014

Summary

A fundamental question in cell biology is how proteins interact and get locally self-assembled as macro-molecular structures that execute diverse functions. Cell matrix adhesion sites are macro-molecular assemblies, consisting of more than 150 proteins, involved in various cellular functions such as cell attachment, cell migration, cell morphogenesis, sensing the immediate environment and cell fate determination. A complex network of regulated interactions between adhesion site components generates highly dynamic and spatially heterogeneous adhesion sites that have distinct molecular composition and function. Collectively, these aspects pose fundamental challenges for monitoring how protein networks get assembled and function in adhesion sites. In this thesis novel concepts and approaches are developed to systematically address these challenges: (1) In order to monitor protein networks with high spatiotemporal resolution, sensitive four-color live cell image acquisition, aligning and correction approaches were developed (Chapter 2). (2) To analyze this four-color data I developed live cell compositional imaging approach to derive the spatiotemporal changes in the molecular composition of adhesion sites at a light resolution. Using this approach, the spatial organization of diverse protein-network states and their dynamics at sub-adhesion site resolution were resolved and visualized in spread fibroblasts as well as in fibroblasts responding to mechanical force perturbations (Chapter 3). (3) In order to understand the regulation and function of protein networks it is essential also to understand how the network components influence each other. Therefore, as a complementary data extraction approach, I developed object segmentation and tracking software that quantifies the relative changes in the levels of the four imaged components in individual adhesion sites. Using such time series and theoretical approaches, potential causal connections between proteins regulating assembling, disassembling and steady-state focal adhesion were inferred. In order to directly derive causal relations within complex biochemical systems it is required to perturb their components. Therefore I developed computational tools to quantify changes in the levels of proteins in focal adhesions in response to acute perturbations of their components and thereby spatially resolving causal relations between them (Chapter 4). (4) Finally, a fundamental problem in studying large and heterogeneous intracellular biochemical systems such as adhesion sites is the inability to co-monitor all the components, which can lead to distinct observed relations between the same subset of observed components. Therefore I was computationally involved in a collaborative development of a statistical method to unmix observations derived from a mixture of protein networks with distinct topologies (Chapter 5). To conclude, in this thesis the above mentioned challenges are addressed, novel tools are developed and thereby structure-function relationships of heterogeneous protein networks of adhesion sites are resolved with high spatial and temporal resolutions.

Zusammenfassung

Eine fundamentale Fragestellung der Zellbiologie ist, wie Proteine interagieren und sich selbst lokal als makromolekulare Strukturen assemblieren, die diverse Funktionen ausführen. Zell-Matrix-Adhäsionsstellen sind makromolekulare Anordnungen, die aus mehr als 150 Proteinen bestehen und in verschiedene zelluläre Funktionen wie Zelladhäsion, Zellmigration, Zellmorphogenese, die Wahrnehmung der direkten Umgebung sowie Entscheidungen über das zelluläre Schicksal involviert sind. Ein komplexes Netzwerk von regulierten Interaktionen zwischen den Komponenten generiert hochdynamische und räumlich heterogene Adhäsionsstellen, die unterschiedliche molekulare Kompositionen und Funktionen aufweisen. Insgesamt stellen diese Aspekte ein grundlegendes Problem für die Beobachtung darüber dar, wie die Proteinnetzwerke sich assemblieren und in den Adhäsionsstellen funktionieren. In dieser These werden neue Konzepte und Herangehensweisen entwickelt, um diese Problematik systematisch aufzuarbeiten: (1) Um Proteinnetzwerke mit hoher räumlicher und zeitlicher Auflösung zu beobachten, wurde ein sensitives 4-Farben-Bildaufnahmeverfahren an lebenden Zellen inklusive Alignement- und Korrekturmethoden entwickelt (Kapitel 2). (2) Um diese 4-Farbendaten zu analysieren, entwickelte ich ein kompositorisches Bildaufnahmeverfahren für lebende Zellen, mit dem die räumlich-zeitlichen Änderungen in der molekularen Komposition der Adhäsionsstellen mit der Auflösung des Lichtes abgeleitet werden. Mit dieser Technik visualisierte ich die räumliche Organisation diverser Proteinnetzwerkzustände und ihrer Dynamik mit einer Auflösung unterhalb der Größe der Adhäsionsstellen in Fibroblasten, die sich auf dem Substrat spreiten sowie solchen auf die mechanische Kräfte einwirken, auf (Kapitel 3). (3) Um die Regulation und Funktion von

Proteinnetzwerken zu verstehen, ist es auch notwendig zu begreifen, wie die Netzwerkkomponenten sich gegenseitig beeinflussen. Daher entwickelte ich als komplementären Datenextraktionsansatz eine Objektsegmentations- und Objektverfolgungssoftware, die die relativen Änderungen in den Leveln der beobachteten 4 Komponenten in individuellen Adhäsionsstellen quantifiziert. Aus den Zeitreihen und den theoretischen Ansätzen heraus, wurden potentielle kausale Verknüpfungen zwischen den Proteinen, die Aufbau, Abbau und stationäres Verhalten der fokalen Adhäsionsstellen regulieren, abgeleitet. Um direkt kausale Verbindungen innerhalb komplexer, biochemischen Systeme abzuleiten, müssen ihre Komponenten gestört werden. Daher entwickelte ich rechnergestützte Werkzeuge um die Änderungen in den Konzentrationen der Proteine in fokalen Adhäsionen nach akuter Störung ihrer Komponenten zu quantifizieren und damit die kausalen Zusammenhänge zwischen ihnen räumlich aufzulösen (Kapitel 4). (4) Ein grundlegendes Problem bei dem Studium großer und heterogener intrazellulärer biochemischer Systeme wie Adhäsionsstellen, ist die Unmöglichkeit alle Komponenten gleichzeitig zu beobachten. Dies kann zu unterschiedlichen beobachteten Abhängigkeiten zwischen der gleichen Untergruppe von beobachteten Komponenten führen. Daher war ich an der gemeinschaftlichen Entwicklung von statistischen, rechnergestützten Methoden beteiligt, mit denen aus der Beobachtung einer Mischung von Proteinnetzwerken mit unterschiedlichen Topologien, die einzelnen Netzwerke entmischt werden können (Kapitel 5). Im Rahmen dieser These wurden daher die zuvor benannten Herausforderungen angegangen, neue Methoden entwickelt und damit die Struktur-Funktionsbeziehungen heterogener Proteinnetzwerke in Adhäsionsstellen mit hoher räumlicher und zeitlicher Auflösung analysiert.

Dedicated to
ammi & abba

This thesis is lovingly dedicated to my mother, Rakheeb Sultana and my father, Malik Sheriff. Their support, encouragement, and constant love have sustained me throughout my life.

Acknowledgments

All praises and thanks to the Almighty, the most gracious and the most merciful.

I express my deep sense of gratitude to Dr. Eli Zamir for giving me an opportunity to work under his supervision. I am extremely grateful to his valuable guidance and constant support throughout my thesis. Without his continuous motivation, this thesis would not be successful.

I offer my sincere thanks to Prof. Hernan Grecco, for mentoring and guiding me in various collaborative projects. I am indebted to his kind scientific and academic support.

I am thankful to Prof. Philippe Bastiaens for his timely suggestions and accepting me as a PhD student in his department.

I am indebted to Prof. Katja Ickstadt for her cordial support. Additionally, I am also thankful to her and Mr. Jakob Wieczorek for their friendly collaboration and cooperation.

It is my pleasure to acknowledge Dr. Jana Harizanova for her immense help and timely suggestions.

I would like to express my sincere gratitude to Dr. Ruth Stricker and Dr. Aneta Koseska for their amiable collaborations and support.

I offer my special thanks to Sarah Imtiaz and Kondaiah Moganti for their scientific support to my thesis.

My sincere thanks to all the members of Department of Systemic Cell Biology, for offering a friendly ambiance during the course my Ph.D. thesis. I owe a debt of gratitude to each and every individual who supported me during my thesis.

I am very thankful to all my near and dear friends.

Most importantly, I express my wholehearted gratitude to my dear mom and dad and my sweet sister, Rishma and brother, Mufeeth for their caring love, boundless support, encouragement, prayers and blessings which molded me to face several hectic times in my life.

My acknowledge is incomplete without thanking two special persons in my life, my beloved wife, Dr. Durdana Parveen and my five month old daughter, Haniya Raqeeba, whom I am yet to meet. I am thankful to Almighty for giving me an understanding, supportive life partner and a beautiful daughter, whose photos have been my stress relievers.

Declaration

I, Rahuman Sheriff Malik Sheriff, hereby certify that this Ph.D. thesis, entitled 'Spatially resolving the dynamics and structure of protein networks in adhesion sites' which is approximately 200 pages in length, has been written by me, that it is the record of work carried out by me under the guidance of Dr. Eli Zamir, at the Department of Systemic Cell Biology, Max Planck Institute of Molecular Physiology, Dortmund. I was admitted as a PhD student in Sep, 2009 under the supervision of Prof. Philippe Bastiaens at Faculty of Chemistry, Technische Universitt Dortmund, Dortmund. This thesis contains contents from collaborative research and the assistance of the collaborators are exclusively mentioned. This thesis has not been previously submitted for the award of any degree, diploma, associateship, fellowship or its equivalent to any other University or Institution.

Place : Dortmund

Date : (Rahuman Sheriff Malik Sheriff)

Contents

List of Figures

Chapter 1

Introduction

Cells are the structural and functional units of living organisms. A single cell is an autonomous living form in case of primitive organisms like protozoans and bacteria and seldom interacts with other cells. Wherein, the multi-cellular organisms exist as a society of cells that essentially interact with others and maintain certain size and morphology. Several attempts have been made by the researchers in the past century to understand, how, in an embryonic development heterogeneous cells that arise from a seemingly simple zygote can organize themselves to form complex morphology. It can be now postulated that, the cells should be held attached to each other, should sense their environment, select their neighbors and migrate towards them in order to achieve a tight control over growth, size and morphology. One of the major factors that mediate all these major functions is cell-matrix adhesion sites. Cells secrete extra-cellular matrix which acts as a scaffold for cells and are attached to it through adhesion sites. Although cells can sense their broader environment through chemical cues, they can sense their immediate environment with higher sensitivity through physical contact by cell-matrix adhesion sites. Adhesion sites act as an anchoring points, that helps the cells pull itself towards the direction conducive for migration. Therefore, the micro role of adhesion sites in cell has a larger impact on the development and growth of the entire organism. In this thesis, the major focus is on studying the biology of cell-matrix adhesion sites using inter-disciplinary approaches that involves experimental and computational techniques.

1.1 Cell-matrix adhesion sites

Cell-matrix adhesion sites are macromolecular assemblies of proteins along the plasma membrane that mediate the attachment of the cell to the extracellular matrix (ECM) (Geiger et al., 2009) (Fig. 1.1). They are heterogeneous in their size, shape and molecular composition1.2. Adhesion sites, are highly dynamic multi-molecular structures which primarily regulates migration of cells during wound healing, embryonic development, immune cell infiltration, metastasis and other cellular processes (Geiger et al., 2009). Adhesion sites were initially reported in 1964, as patches of dense structures close to cellular substrate (Curtis, 1964). Later they were identified as connecting link between the ECM and actin cytoskeleton because the actin filament bundles stemmed or terminated at adhesion sites (Heath et al., 1978). Gradually, several proteins are reported to be localized in adhesion sites (Hanein and Horwitz, 2012). Adhesion sites are connected to the ECM through heterodimers of transmembrane integrins receptors, which control the formation of these sites upon binding with ECM ligands (Wehrle-Haller, 2012). Over last decade adhesion sites are reported to exhibit diverse functions and are involved in several biological processes, some of which are cell signaling regulation, sensing of environment and cell fate determination.

1.1.1 Molecular complexity and heterogeneity of cell- matrix adhesion sites

Recent molecular analyses of integrin adhesome implied that more than 200 distinct components are associated with adhesion sites, amongst which over a 100 are intrinsic constituents of these structures (Hanein and Horwitz, 2012; Geiger et al., 2009; Zaidel-Bar et al., 2007) (Fig. 1.1). Most of these components are multi-domain proteins that can interact with multiple molecular partners and thereby give rise to heterogeneous protein network, which is reflected as heterogeneity in composition of the adhesion sites (Zamir and Geiger, 2001a,b). The interactions between proteins are regulated by several mechanism, and few among them are competition between multiple proteins to interact with same domain of

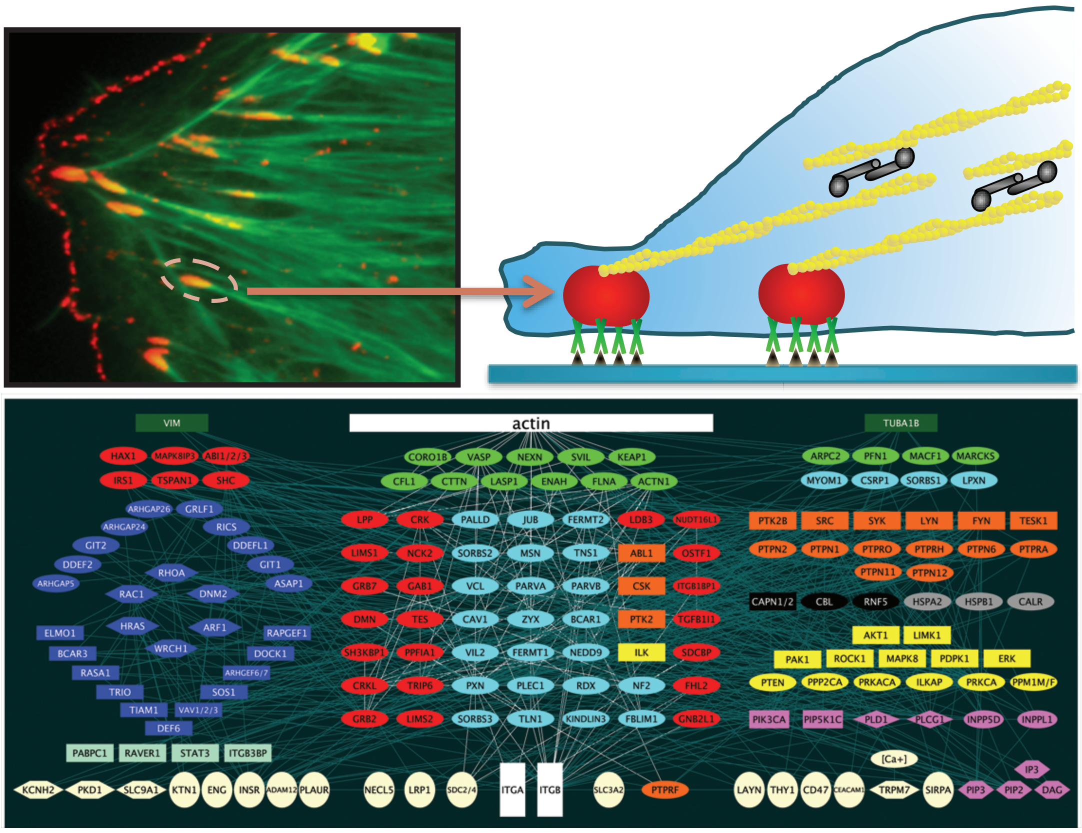


Figure 1.1: Molecular complexity of cell-matrix adhesion sites - Cell matrix adhesion sites are macromolecular complexes that mediate attachment of cells to their substrate. These multi-molecular structures are connected to the extracellular matrix (ECM) through integrins that span the plasma membrane. Adhesion sites also act as anchor point for actin filaments and thereby regulate cytoskeletal architecture of the cell. Actin filaments are crosslinked by homodimers of α-actinin, one of the components of adhesion sites. Adhesion sites are composed of more that 150 proteins. A snapshot of adhesion protein interaction network is displayed in the lower panel. Reproduced from: (Zamir and Geiger, 2013).

the target protein, activation or inhibition of binding by tyrosine phosphorylation and change in conformation of interacting proteins. Conformational changes can also be caused by force exerted over the component upon pulling by actin filament or phosphorylation or interaction with another component of the network. Conformational change in protein can uncover or generate a new binding site or retract an existing binding site (Schiller and Fässler, 2013; Zamir and Geiger, 2001b; Hanein and Horwitz, 2012; Zaidel-Bar and Geiger, 2010; Parsons et al., 2010). Diverse conditions at different spatial locations of the cell can favor different interactions to generate heterogeneous protein networks. Therefore, the highly diverse molecular content and diverse conditions within a cell allow adhesion sites to be immensely heterogeneous in their composition and function.

1.1.1.1 Types of cell-matrix adhesion sites

Adhesion sites can be broadly classified into four categories viz. nascent adhesions, focal complexes, focal adhesions and fibrillar adhesions, based on their morphology, molecular content and life span (Hanein and Horwitz, 2012). Nascent adhesion sites are the smallest structures with size $< 0.25\mu m$. These are structures with short lifespan of about 1 minute and they are formed at the edges of lamellipodia of a fibroblast. These structures are myosin II independent and their formation needs polymerization of actin (Zamir et al., 2008; Hanein and Horwitz, 2012; Zaidel-Bar and Geiger, 2010; Parsons et al., 2010). These nascent adhesion sites can mature focal complexes which are dot-like structures. These structures which are about $1\mu m$ in diameter are present at the interface between lamellipodium and lamellaum. Focal complexes which has a lifespan of several minutes require myosin II for their formation. Focal complexes can grow further into large, several micron long, oval shaped focal adhesions which has a half-life up to 20 minutes (Zaidel-Bar and Geiger, 2010). Focal adhesions are mostly present in the periphery of the cells from where the actin stress fibers originate. Focal adhesions are enriched with $\alpha5\beta3$ integrin as well as paxillin, vinculin and other adhesome proteins with phosphorylated tyrosine residues. Fibrillar adhesions are elongated or beaded structures, located more centrally in the cell and are associated with large actin filaments and fibronectin fibrils. These structures

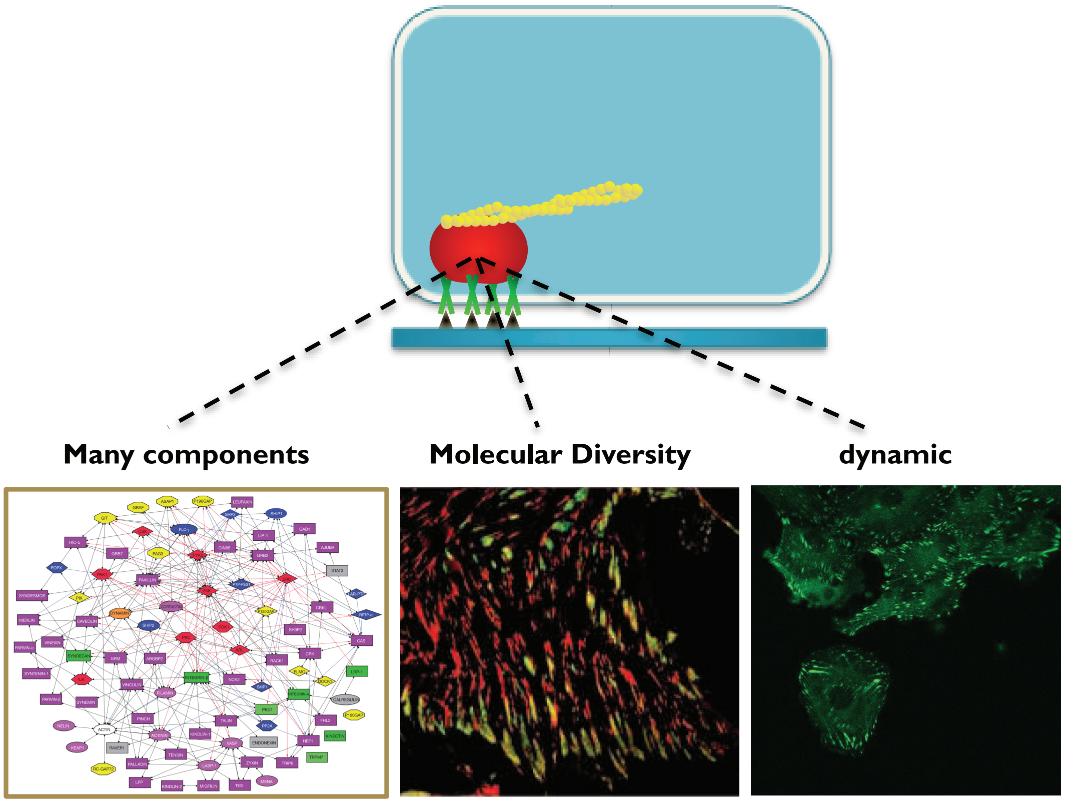


Figure 1.2: Properties of cell-matrix adhesion sites - Cell matrix adhesion sites can exhibit diverse cellular functions owing to their three major properties viz. multi-molecular nature, compositional heterogeneity and rapid dynamics. Partially adapted from: (Zaidel-Bar et al., 2007; Zamir and Geiger, 2001b).

are enriched in $\alpha5\beta1$-integrin, tensin and parvin (Zamir et al., 1999, 2000; Zamir and Geiger, 2001b).

1.1.2 Focal adhesion proteins

The complexity, robustness and sensitivity of the integrin adhesome network stems from multi-molecular proteins whose interaction with multiple binding partners is regulated by diverse mechanisms. An in silico analysis of adhesome consisting of 156 components connected by 690 interaction divulged several motifs and modules in the network that regulates the structural and signaling functions of adhesion sites (Zaidel-Bar et al., 2007). Adhesion sites play a vital role in triggering certain signaling process by recruiting both signaling enzyme and its substrate to the same scaffolding molecule (Zaidel-Bar et al., 2007; Geiger et al., 2009) and molecular complex. Mounting body of research in the past few years has revealed the structure, function and localization of several adhesion molecules. In the following section, the structure and function of adhesion site proteins that are considered in this thesis are briefly described:

α-actinin - α-actinin is the first identified adhesion sites component recognized in spread rat embryo cells as focally distributed plaques of proteins at the ends of actin stress fibers (Lazarides et al., 1975). α-actinin binds to actin and is essential for cross linking of actin filament and therefore a component of cellular cytoskeletal system (Fig. 1.1). α-actinin is a member of spectrin superfamily of proteins, and has four isoforms in mammalian cells, amongst which α-actinin1 and 4 are ubiquitously expressed where as the others are specific to muscle cells (Feng et al., 2013; Lek and North, 2010; Sjöblom et al., 2008). α-actinin interacts with vinculin, zyxin, integrins, and many other focal adhesion proteins (Feng et al., 2013; Sjöblom et al., 2008; Otey et al., 1990; Pavalko and LaRoche, 1993; Wachsstock et al., 1987) (Fig. 1.4). Acting binding property of α-actinin1 is regulated by Focal adhesion kinase(FAK) which phosphorylates α-actinin at Y12. This phosphorylation lowers the affinity of α-actinin for actin. In contrast PTP1B can dephosphorylate Y12 in order to restore its affinity (Izaguirre et al., 2001; Zhang et al., 2006; von Wichert et al., 2003). The functional α-actinin forms

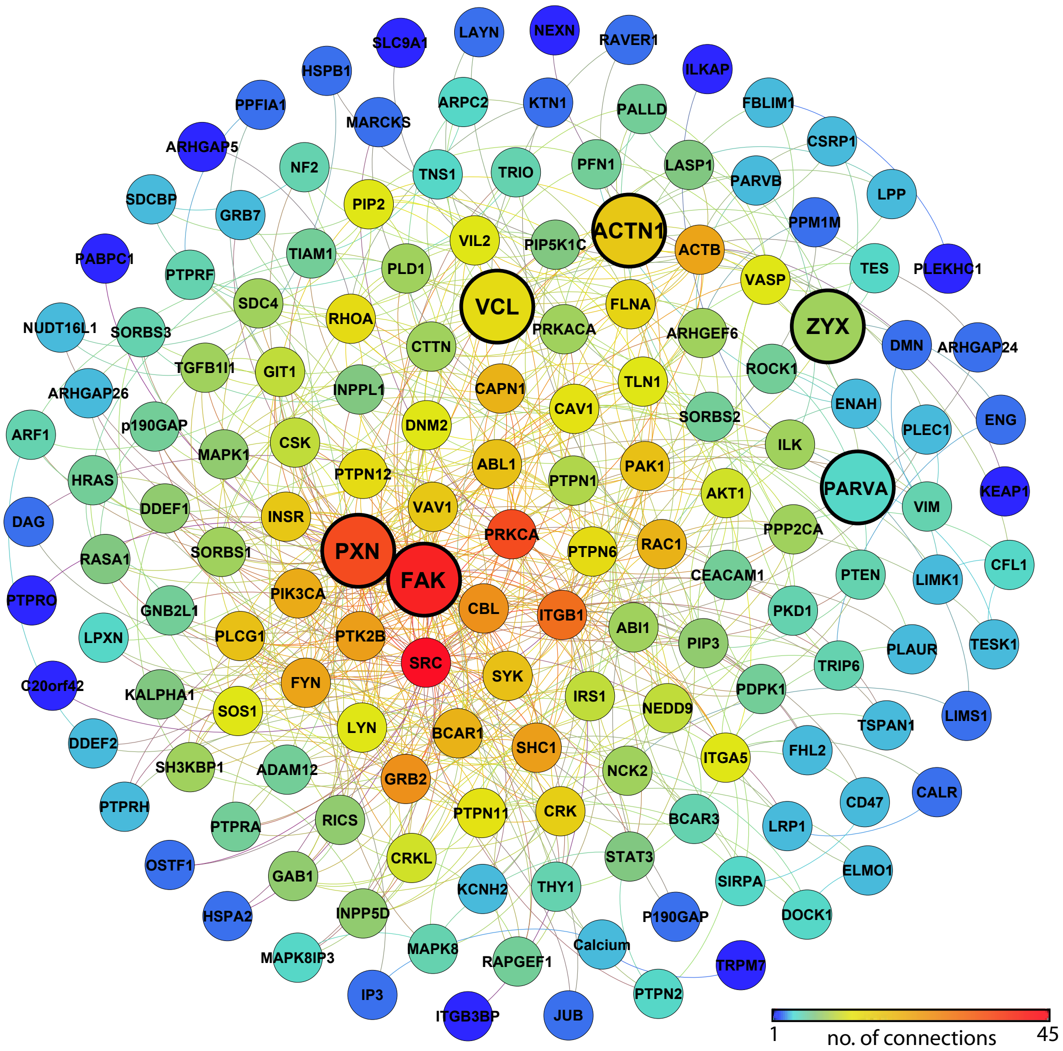


Figure 1.3: Components of the integrin adhesome network - Nodes in the network represent various components involved in protein interaction network of adhesion sites. The color of each node represents the number of its direct interaction partners in the network.

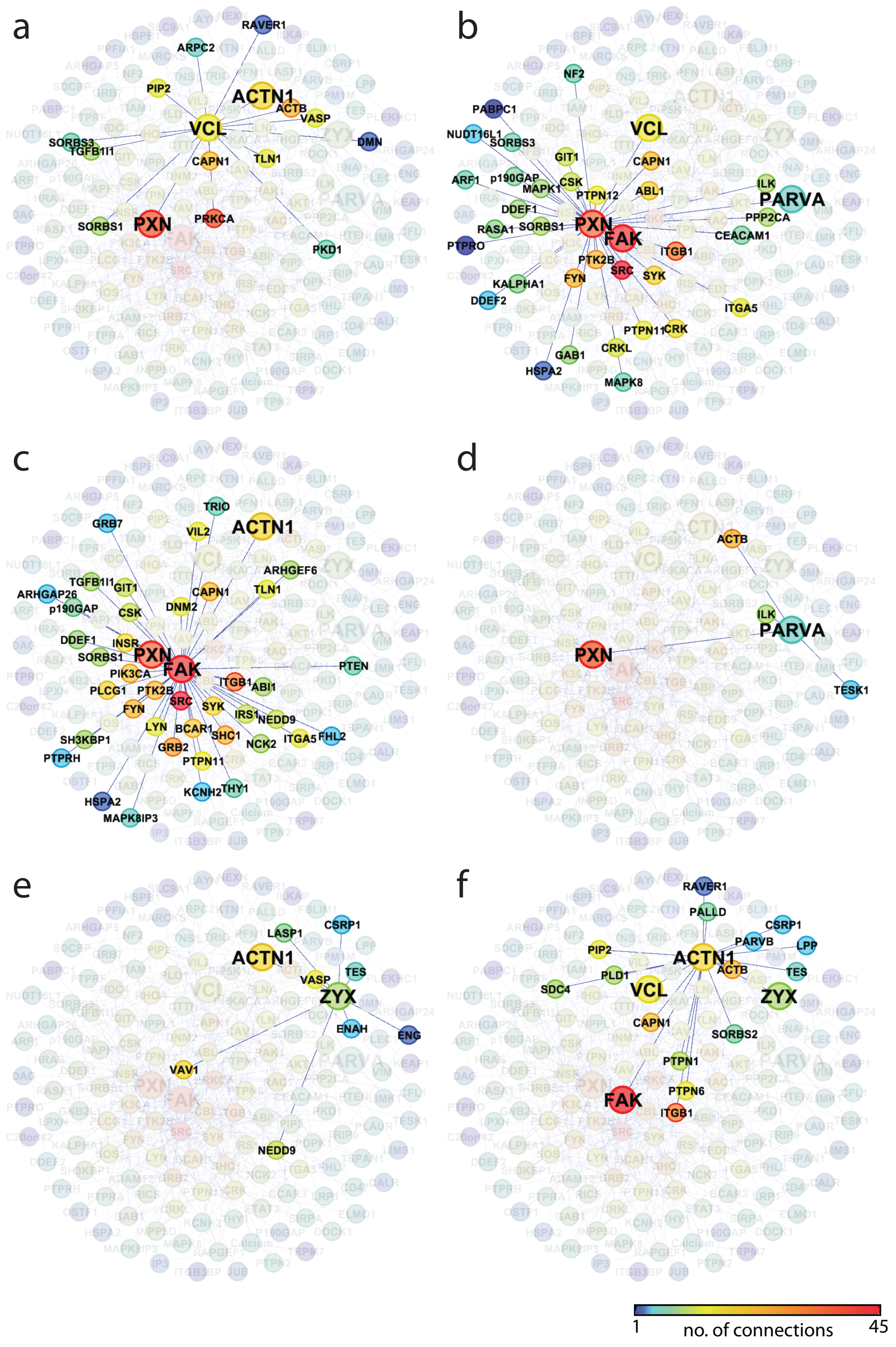


Figure 1.4: Interactome of adhesion site proteins - The interaction network showing direct interaction partners of (a) Vinculin (VCL) (b) Paxillin (PXN), (c) Focal adhesion Kinase (FAK), (d) α-Parvin (PARVA), (e) Zyxin (ZYX), and f) $\alpha-$Actinin1 (ACTN1). The color of each node represents the number of its direct interaction partners in the network.

an anti-parallel rod-shaped homo-dimers with two actin-binding domain (ABD) on either ends of the rod. This molecular architecture allows α-actinin dimer to bind to actin fibers at both the ends and thereby crosslink them (Sjöblom et al., 2008).

Focal adhesion kinase - Focal adhesion kinase (FAK) is a vital component of adhesion sites that sense and integrate various signals to regulate cell migration with its dual role as signaling kinase and as an adaptor/scaffold protein (Mitra et al., 2005). FAK is recruited to the focal contacts and activated upon integrins bind to the ECM and clusters. The activation firstly involves autophosphorylation of FAK at Tyr397 which generate binding site for SH2 domain of Src. Secondly Src phosphorylates FAK at Tyr576 and Tyr577 to endorse maximal catalytic activity. Focal adhesion targeting (FAT) domain of FAT interacts directly with paxillin and localizes in focal adhesion and this interaction is decreased by ERK2 mediated phosphorylation of FAK at Ser910 (Mitra et al., 2005; Parsons, 2003). FAK null mice exhibit defect in morphogenesis during embryonic development (Ilić et al., 1995). Fibroblasts lacking FAK display increased number of focal contacts suggesting the its role in disassembly of adhesion sites (Webb et al., 2004). The immobilized fraction of FAK in focal adhesion is directly coupled with the strength of the adhesion sites (Le Dévédec et al., 2012). FAK is also activated by growth factor stimulation(Schaller et al., 1994; Le Dévédec et al., 2012). FAK also regulates Rho-GTPases (e.g. Rho, Rac, Cdc42 and Ras) and hence loss of expression of FAK affects polarization of microtubules and focal contact turnover (Palazzo et al., 2004; Ren et al., 2000; Mitra et al., 2005). FAK phosphorylates Tyr12 of α-actinin to reduces cross-linking and thereby releases actin filaments from focal contacts. Moreover, the calpain-2 mediated cleavage of FAK controls the dynamics of adhesion in migrating cells (Chan et al., 2010).

α-Parvin - α-Parvin is one of the component of IPP, a heterotrimeric complex that regulated integrin-mediated signaling in adhesion sites. The other two components are integrin-linked kinase (ILK) and the adaptor protein PINCH. α-Parvin interacts with ILK via one of the two calponin homology (CH) domains at the C-terminal region (Legate et al., 2006). Binding of α-parvin and ILK is partly regulated by $PtdIns(3,4,5)P_3$. Additionally, it is also dependent on phosphorylation of α-parvin by CDC2 and MAPK (Attwell et al., 2003; Yang et al., 2005; Clarke et al., 2004; Legate et al., 2006). ILK- α-parvin interaction is inhibited by Lnk which forms complex with IKL (Devallière et al., 2012). α-parvin negatively regulates Rac and knockdown of α-parvin increased activity of Rac and formation of lamellipodium in HeLa cells. α-parvin can interact directly with F-actin. Furthermore, it also interacts paxillin which in turn binds to F-actin. α-parvin null mice are embryonically lethal. α-Parvin regulates the dynamics of focal adhesion and actin cytoskeleton and the motility of the cells (Devallière et al., 2012).

Paxillin - Paxillin is a multi-domain scaffold protein first identified as a part of integrin adhesome in 1990 (Turner et al., 1990; Deakin and Turner, 2008). Multiple protein binding domains of paxillin that are regulated by phosphorylation, enable paxillin to recruit several adhesion proteins and control the structural and functional properties of cell-matrix adhesion sites. Paxillin contains four LIM domains (LIM1-LIM4) in the C-terminal half and five leucine- and aspartate-rich LD motifs (LD1-DL5) in N-terminal half. LIM domains offer the binding sites for numerous structural and regulatory proteins which include non-adhesion site proteins tublin and tyrosine phosphatase PTPN12 (Côté et al., 1999; Herreros et al., 2000). N-terminal LD motifs serve as the binding site for vinculin and FAK (Brown et al., 1996; Turner and Miller, 1994). Additionally, the proline-rich region of N-terminus acts as a docking site for src, a non-receptor tyrosine kinase (Weng et al., 1993). Multiple tyrosine, serine and threonine phosphorylation sites of paxillin are targeted by various kinase including PAK, FAK, src, RACK, JNK, ERK, P38 MAPK, CDK5 and Abl (Deakin and Turner, 2008). Diverse phosphorylation state of paxillin controls the protein network state and composition of the adhesion sites by regulating various protein interactions and their phosphorylation. Interaction of integrin with fibronectin or collagen induces phosphorylation of paxillin at tyrosine residues Y31 or Y118 respectively (Bellis et al., 1997; Burridge et al., 1992; Petit et al., 2000; Deakin and Turner, 2008). Paxillin interacts with FAK with a stoichiometry of 1:1 as the LD2 and LD4 motifs of the former binds with the FAT domain of the latter (Brown et al., 1996; Turner et al., 1999; Bertolucci et al., 2005). This interaction is down regulated by phosphorylation of

LD4 motif at S273. Paxillin plays an essential role in assembly of adhesion sites as it is amongst the earliest protein detected in nascent adhesion sites formed in the leading edge of the cell (Digman et al., 2008; Deakin and Turner, 2008). Enrichment of paxillin specifically at the outer edge of the small dot-like adhesions sites is observed in spreading fibroblast cells (Zimerman et al., 2004). Paxillin also control cell migration by regulating the disassembly of adhesion sites. Cells lacking paxillin showed stabilization of adhesion sites (Webb et al., 2004). Moreover, paxillin with Y31 and Y118 mutated to non-phosphorylatable residues affected adhesion site disassembly in front edge of the migratory cell (Webb et al., 2004). Cleavage of paxillin by calcium-dependent protease calpain II is also shown to induce disassembly of adhesion sites (Carragher et al., 1999; Deakin and Turner, 2008).

Vinculin - Vinculin is one of the adhesion proteins with an important role in mechano-regulation of adhesion sites (Spanjaard and de Rooij, 2013). Vinculin is a 116kDa actin-binding protein that consist of a globular head and tail domain connected via a short proline-rich sequence (Ziegler et al., 2006). Head domain of vinculin binds to talin, that links integrin and F-actin whereas the tail domain binds to F-actin (Spanjaard and de Rooij, 2013). The proline-rich region and/or vinculin tail domain can bind and recruit other adhesion proteins like paxillin and VASP. The inactive auto-inhibited form of vinculin is present in the cytoplasmic region and exhibits an intra-molecular interaction between head and the tail domain. This interaction masks several ligand binding sites in the vinculin. The active state of vinculin that localize in the adhesion sites lack this head-tail interaction and expose all ligand-binding sites (Chen et al., 2005; Ziegler et al., 2006; Spanjaard and de Rooij, 2013). Exposure of high-affinity vinculin binding sites in tail of αactinin is sufficient to activate vinculin (Izard and Vonrhein, 2004; Izard et al., 2004; Bois et al., 2006). Vinculin null cells display defective spreading (DeMali, 2004). Additionally, an augmented recruitment of vinculin (along with talin and paxillin) and increase in size is observed in focal adhesions experiencing force by external pulling, stretching and shear stress or RhoA stimulation (Delanoë-Ayari et al., 2004; Balaban et al., 2001; Giannone et al., 2003; Rottner et al., 1999; Zaidel-Bar et al., 2003). Vinculin mutants causes significantly diminished dynamics of talin but not paxillin in focal adhesion (Cohen et al., 2006). Dominant positive mutants strongly stabilized focal adhesion by disengaging their normal actomyosin regulation (Humphries et al., 2007). Vinculin down-regulates the turnover of proteins in focal adhesion and represses the migration of the cell (Saunders et al., 2006; Xu et al., 1998; Ziegler et al., 2006). Mechanical tension dependent switching cycles of vinculin conformation is a key process that controls the stability as well the dynamics of focal adhesion (Humphries et al., 2007; Spanjaard and de Rooij, 2013).

Zyxin - Zyxin is an essential component of adhesion sites which plays a vital role in mechanotransduction system. It is also considered to be a key regulator of force dependent actin polymerization Hirata et al. (2008). Zyxin was initially identified in avian smooth muscle. Besides Adhesion sites, zyxin also localizes in cell-cell junction and stress fibers(Li et al., 2013; Hirata et al., 2008). Zyxin is LIM protein that regulates integrin-dependent cell motility and remodeling of actin filaments. Fibroblasts lacking zyxin showed an augmented adhesion and migration (Hoffman et al., 2006). Zyxin can interact with actin remodeling protein α-actinin and Ena VASP to regulate the organization of actin filaments (Li et al., 2013; Crawford et al., 1992; Drees et al., 1999) (Fig. 1.4). Lack of zyxin causes mislocalization of Ena and VASP (Hoffman et al., 2006). Accumulation of zyxin at focal adhesion determines the recruitment of Ena VASP (Hirata et al., 2008).

1.1.3 Stress fibers

Stress fibers or actin filaments are bundles of polymeric f-actin and contractile myosin and are major components of cellular cytoskeleton. They play an essential role in maintenance and regulation of cellular morphology. They also regulate maturation of adhesion sites and thereby participate attachment of cell with extracellular matrix (Cramer et al., 1997; Feng et al., 2013). The actin filaments or stress fibers can be classified into three types viz. ventral stress fibers, dorsal or radial stress fibers and transverse arcs, based on their sub-cellular localization and connection with focal adhesion (Naumanen et al., 2008; Feng et al., 2013; Pellegrin and Mellor, 2007). Ventral stress fibers stretch along the basal region of the cell and are connected to focal adhesion at both ends. The dorsal stress fibers are short non-contractile filaments that emerge from a focal adhesion and pass through dorsal region. In contrast to ventral stress fibers, the dorsal stress fibers are attached to focal adhesion only at one end. On the other hand, the transverse arcs are not directly interacting with focal adhesions. These transverse arcs are stress fibers which exhibit typical retrograde flow towards the nuclear region from leading edge. Ventral stress fibers play a vital role in controlling cell contraction, where dorsal fiber promotes maturation of focal adhesion (Heath, 1983; Small et al., 1998; Hotulainen and Lappalainen, 2006; Heath et al., 1978; Oakes et al., 2012; Feng et al., 2013). Dorsal fibers contain α-actinin which is sometimes displaced by myosin clusters. These fibers are elongated by polymerization of actin mediated by mDia1 (Feng et al., 2013).

1.2 Approaches and challenges in resolving the structure of protein networks

1.2.1 State-of-the-art approaches for inferring the topology of protein networks

With advancement in molecular biology research, and development of high - throughput measures, several attempts have been made to build up global maps of protein-protein interaction and DNA-protein interaction network. To understand the biological principles behind the global networks which are often noisy, a integrative model based approach is required (Friedman, 2004). High-throughput transcriptomics data from a static sample are analyzed by a clustering approach to detect groups with typical expression level and it may be assumed that all genes belonging to same group are governed by same condition (Friedman, 2004; Bansal et al., 2007). Although these groups imply a relevant biological context, additional time-resolved data is required to correctly reconstruct gene regulatory networks.

The gene regulatory network is an ensemble of influence interaction between genes which includes gene-to-gene interaction (gene expression network) and protein-to-gene interaction (transcriptional factor network). The data-driven protein network reconstruction from the analysis of multivariate time-resolved data is immensely valuable and is of supreme importance in systems biology research (Hempel et al., 2011a,b). Inferring or reverse-engineering gene networks from the experimental data through computational analysis can be achieved by several machine learning approaches and ready-to-use software. Some of them are reviewed and their performance is compared in Hempel et al. (2011b); Bansal et al. (2007). Nevertheless, the time-resolved data from the vast amount of genes is often short and noisy, imposing a challenge on network reconstruction. It is, therefore, essential to choose suitable method after understanding the principle of the approach and the limitation of the biological data. Some of the software available for inferring gene regulatory network include BANJO (Bansal et al., 2007; Yu et al., 2004), ARACNE (Basso et al., 2005; Margolin et al., 2006; Bansal et al., 2007), NIR/MNI (Gardner et al., 2003; di Bernardo et al., 2005; Bansal et al., 2007). A recently developed permutation-based measure termed as inner composition alignment or IOTA can efficiently identify causal connections between subsystems from short time series data of about 10 time points. This approach has been demonstrated to resolve statistically significant non-linear links from short time series of gene expression data (Hempel et al., 2011a).

Yet another multivariate approach that does not rely upon temporal data is Bayesian network based analysis. Bayesian networks are probabilistic graphical models that recently emerged as a propitious tool to infer interdependence relationship between multiple interacting components of complex networks of all the cell signaling cascade (Pearl, 1988; Sachs et al., 2002, 2005; Friedman et al., 2000; Friedman, 2004). Bayesian models inferred from the experimental data can capture complex stochastic non-linear influence relationship between the signaling molecules. The probabilistic measure of the connections accommodate the noise or variance innate to biological data. The Bayesian network can infer direct as well as indirect relationships mediated through unobserved components. Bayesian network computational methods resolved the causal links between signaling components of primary human T-cells (Sachs et al., 2005). The levels of multiple phosphorylated protein and phosholipid components are measured simultaneously at single cell level by flow cytometry, that offers large data. Bayesian analysis accurately predicted most of the inter-pathway network causality and pathway structure which are further confirmed experimentally by perturbation (Sachs et al., 2005). Moreover, the Dynamic Bayesian networks (DBN) based approach with enhanced prediction precision and computational time is developed to identify gene regulatory network from the time-series microarry data measured during the cell cycle of yeast cell (Zou and Conzen, 2005; Zou et al., 2009). Lately, a time varying dynamic Bayesian Network (TV-DBN) is developed for reconstructing the gene regulatory network for a mouse strain infected with influenza AH1N1 virus (Dimitrakopoulou et al., 2011). This is followed by development of dynamic cascaded method (DCM) which is based on an intra-state steady-rate and continuity assumption. DCM is used to elucidate significantly improved dynamic gene network during progression of hepato-cellular carcinoma (HCC) (Zhu et al., 2012).

Large protein networks function with several modules and it is essential to identify modules to grasp their functional organization. Recently a non-parametric Bayesian network (NPBN) approach has been used to recognize modules within the mitogen-activated protein kinase (MAPK) signaling cascade (Ickstadt et al., 2011). One of the fundamental advantages of identifying modules is its application in perturbation-based reverse-engineering approaches like modular response analysis (MRA) (Zamir and Bastiaens, 2008; Ickstadt et al., 2011). It is essential to perform perturbation experiments to elucidate the actual strength and sign of network connections, because the probabilistic and information theoretic measures can only infer potential network connection and topology. MRA is used to reverse engineer the topological difference in MAPK cascade that respond differentially to epidermal or neuronal growth factor (EGF or NGF) stimulation (Santos et al., 2007). MRA is sensitivity analysis to derive direct causal connectivity between the network components and it is based on experiments that measure a steady-state response to a perturbation (Santos et al., 2007; Kholodenko et al., 2002). The change in activity of all modules with respect to systematic perturbation of each module allows estimation of global response co-efficients. This estimate does not directly divulge the network structure as it reflects also the propagation of perturbation via other modules. Therefore, the local response co-efficient that indicates the direct strength and sign of influence of modules on each other, is computed citepSantos2007,Zamir2008. In this method the modules in the MAPK cascade Erk1/2 and Mek1/2 and Raf1/B-Raf are systematically perturbed by RNA interference mediated depletion of protein expression (Santos et al., 2007).

1.2.2 Fundamental challenges in studying protein networks in adhesion sites

The capability to reconstruct protein networks along space will provide a crucial step forward to enhance our understanding about high-level function of the cell at sub-cellular location. Approaches to resolve network structure along the space domain is seldom available. All the aforementioned approaches can resolve protein networks from the static and dynamic experimental data derived from a pooled population of cells. In order to perform network reconstruction by multivariate analysis, it is required to probe multiple components of the cell signaling network with high spatial and temporal resolutions. In the current state-of-the-art no experimental approach allows monitoring sensitively several components of the same signaling module with high spatiotemporal resolutions in live cells. Hence in this thesis the first challenge addressed is development of imaging approach to monitor multiple components of adhesion site protein network.

In order to capture spatial structure-function relationship of protein networks, it is necessary to develop computational approach that can visualize and resolve the network states along space and time. At present, a single approach that can spatially resolve protein network-state is compositional imaging (Zamir et al., 2008). This approach can resolve molecular signatures and sub-domains within adhesion sites that are generated by heterogeneous network states, but still it cannot divulge much information about the dynamics of the identified states. Hence, the next challenge is the development of computational tools for application of previously described compositional imaging on time-resolved multi-component imaging data, and development of additional concepts and approaches to resolve the dynamics of network-states. After the development of tools, the next goal is

Table 1.1: Summary of measures available for resolving topology of protein networks

Measure operating on	Similarity measure	Scoring scheme
Vector	L^s norm	ID
	Euclidean distance	ID
	Manhattan distance	ID
	Dynamic time warping (DTW)	ID
Random variables	Pearson correlation	CLR, MRNET
	Conditional Pearson correlation (CPC)	CLR, MRNET
	Partial Pearson correlation	CLR, MRNET
	Spearmans rank correlation	CLR, MRNET
	Kendalls rank correlation	CLR, MRNET
	Simple mutual information	CLR, MRNET
	Conditional mutual information	CLR, MRNET
	Mutual coarse-grained information rate(MCIR)	ID
	Conditional coarse-grained information rate(CCIR)	ID
	Inner compositional alignment (IOTA)	NA
Model-based measures	Simple Granger causality	ID
	Conditional Granger causality	ID
	Partial Granger causality	ID
	Bayesian Network (BN)	ID
	Dynamic Bayesian Network (DBN)	ID
	Time varying dynamic Bayesian Network (TV-DBN)	ID
Symbolic dynamics	Symbol sequence similarity	TS, AWE
	Mutual information of symbol vectors	TS, AWE
	Residual mutual information (RMI)	AWE

Table 1.2: ID: Identity, CLR: Context Likelihood of Relatedness, ARACNE: Algorithm for the Reconstruction of Accurate Cellular Networks, MRNET: Maximum Relevance/minimum redundancy NETwork, TS: Time Shift, AWE: asymmetric Weighting (AWE).Refer (Hempel et al., 2011b; Bansal et al., 2007; Friedman, 2004) for detailed information about each approach.

to resolve the dynamics of protein network states in adhesion sites of unperturbed and perturbed spread fibroblast cells, using the newly developed approaches.

The expended compositional imaging approach developed here identifies and visualizes the protein network states of the adhesion sites from snapshots of the live cell imaging data and thus derive compositional transitions. In order to understand the mechanism that controls the spatially heterogeneous network-state transition, it is important to elucidate the structure of the protein network at single focal adhesion level. The approaches for network reconstructions requires time resolved data; therefore, the next challenge is to develop computational tools for segmentation and quantification of dynamic changes in the levels of multiple components in individual focal adhesion from multicolor imaging data. Following this is the selection of appropriate theoretical approach and reconstruction of protein networks for each and every focal adhesion. As the theoretical approaches can resolve only probable connections, it is essential to establish approaches to infer causal links from perturbation experiment that enables reverse engineering of spatial resolved causal connections. Finally, as adhesion sites are heterogeneous structures, inferring a single causal network for all the adhesion sites would yield an average meaningless topology. It is therefore needed to unmix the mixtures of protein networks. As no methods so far has addressed the issue of unmixing, the subsequent challenge is development of an approach for the same.

1.3 Objective of the thesis

The objective of this thesis is exploring structure-function relationships of spatially heterogeneous protein networks in adhesion sites by developing novel concepts and approaches. Heterogeneity in composition of adhesion sites emanates from diversity in protein networks. Although compositional diversity is a known phenomena, the mechanism behind the generation of these diverse composition and networks is poorly understood. So far the composition of adhesion sites are characterized in fixed fibroblast cells (Zamir et al., 2008), which lacks information about the dynamics and the origination of diverse composition. As adhesion sites are highly dynamic structures, following the dynamics of protein networks is essential to acquire a deeper insight into the assembly and disassembly mechanism. New adhesion sites are assembled in the protruding edge of the cell and disassembled in the retracting edge. Additionally adhesion sites exhibit molecular sub-domains, suggesting the significance and importance of spatial aspect. This thesis is aimed at inferring the state, dynamics, topology and function of adhesion site protein networks along space and time in live fibroblast cells.

The goals of the thesis and the approaches and tools required to be developed for each goal are described in this section.

1.3.1 Imaging the dynamics of protein networks in adhesion sites

Specific aims:

- Monitoring large protein networks in cell-matrix adhesion sites with high spatial and temporal resolution.

  Tools development:

  - Development of 4-color imaging system which requires selection and building of fluorescent tags and optimization of imaging condition.
  - Development of image processing tools that allows correction of imaging artifacts (e.g. chromatic aberration in multicolor images and bleaching) and noise in the measurement.

1.3.2 Resolving the dynamics of protein networks in adhesion sites

Specific aims:

- Visualizing spatial organization of protein networks and their dynamics in adhesion sites of spread fibroblast cells.
- Characterizing steady-state transition and reconstructing pathways of protein network transitions in spread fibroblast cells.

- Investigating mechano-sensitivity and response of protein networks to force inhibition.

  Tools development:

  - Development of multi-dimensional approach to explore the dynamics of composition.

1.3.3 Reverse engineering protein networks in adhesion sites

Specific aims:

- Reconstruction of protein networks in individual adhesion sties from co-dynamics of proteins.

- Untangling actual protein causal network in adhesion sites by acute perturbation.

  Tools development:

  - Development of computational tool for segmentation and tracking of adhesion sites to quantify turnover of protein levels in individual adhesion sites.
  - Exploring theoretical approaches to infer causal connections in protein networks based on time series data.
  - Development of approaches for inferring causality by systematic acute perturbation.

1.3.4 Unmixing protein networks with distinct topologies

Specific aims:

- Separating mixture of focal adhesion with distinct protein networks from the mixed observation to extract unadulterated causal topology.

  Tools development:

- Development of approach to unmix protein network with distinct connections.

Chapter 2

Imaging the dynamics of protein networks in adhesion sites

Multi-molecular and highly dynamic nature of adhesion sites imply that the structural and functional properties of adhesion sites are determined by diverse molecular content and their dynamics within these structures.Therefore, monitoring a single protein in these structures is not sufficient to study their properties, functions and state and it is essential to monitor multiple component of the adhesion sites. Observing a single adhesion protein with fluorescent protein tag in live cell can only be helpful to study the localization and dynamics of the chosen molecule, not the entire the adhesion site. In contrast, immuno-staining techniques which allows examination of multiple components, cannot divulge any information about the dynamics. The multiple ingredients of the adhesions sites are distributed differentially in these structures resulting in spatial heterogeneity in the composition within an individual adhesion site. In current state-of-art, no technique exists which allows monitoring multiple components in adhesion sites along both space and time for quantitative analysis. Hence in this thesis, a multicolor imaging system is developed to monitor four adhesion sites proteins in live fibroblast cells with high spatial and temporal resolution.

2.1 Sensitive 4-color live cell imaging of adhesion sites

2.1.1 Selection of the protein set

In order to study the compositional dynamics of the adhesion sites and stress fibers, the foremost step is to select focal adhesion proteins which can demonstrate and represent well the heterogeneity and functional properties of adhesion sites. The four different adhesion sites proteins namely, zyxin, α-actinin, vinculin and paxillin are chosen for their central role in regulation of the adhesion sites as described in the previous chapter. These proteins have also been show to exemplify the diversity in the composition (Zamir et al., 2008) and exhibit significantly different focal adhesion dissociation kinetics (Lele et al., 2008). Amongst these four adhesion site proteins, zyxin and a-actinin localizes also in stress fibers and extend the scope of the study to the organization and dynamics of actin filaments.

2.1.2 Establishing the 4-color live cell imaging approach

To monitor the compositional changes, simultaneously all the four adhesion site proteins labeled with fluorescent protein tags are ectopically expressed in fibroblast cells and imaged. One of the main challenges of multicolor imaging is selection of fluorophores without spectral overlap, because all the adhesion site components reside within the same structure. After detailed examination of excitation/emission spectra of published fluorophores, four fluorescent proteins viz. mkate2, mCitrine, mTFP and mTagBFP are chosen. In order to achieve quantitative imaging it is required to optimize the microscopic spectral settings to acquire 4-color imaging of these fluorophores without spectral cross-contamination or bleed-through. Through selective laser excitation and stringent detection range, as described in figure (Fig. 2.1), 4-color images without any bleed-through are acquired with Zeiss LSM510 confocal microscope. As an effect of stringent detection range to avoid bleed-through, a low amount of photons are collected in the detector, resulting in dim and poor quality images. In order to enhance the brightness of the fluorescent tags, double fluorophores, termed as tandem (TD) fluorophores

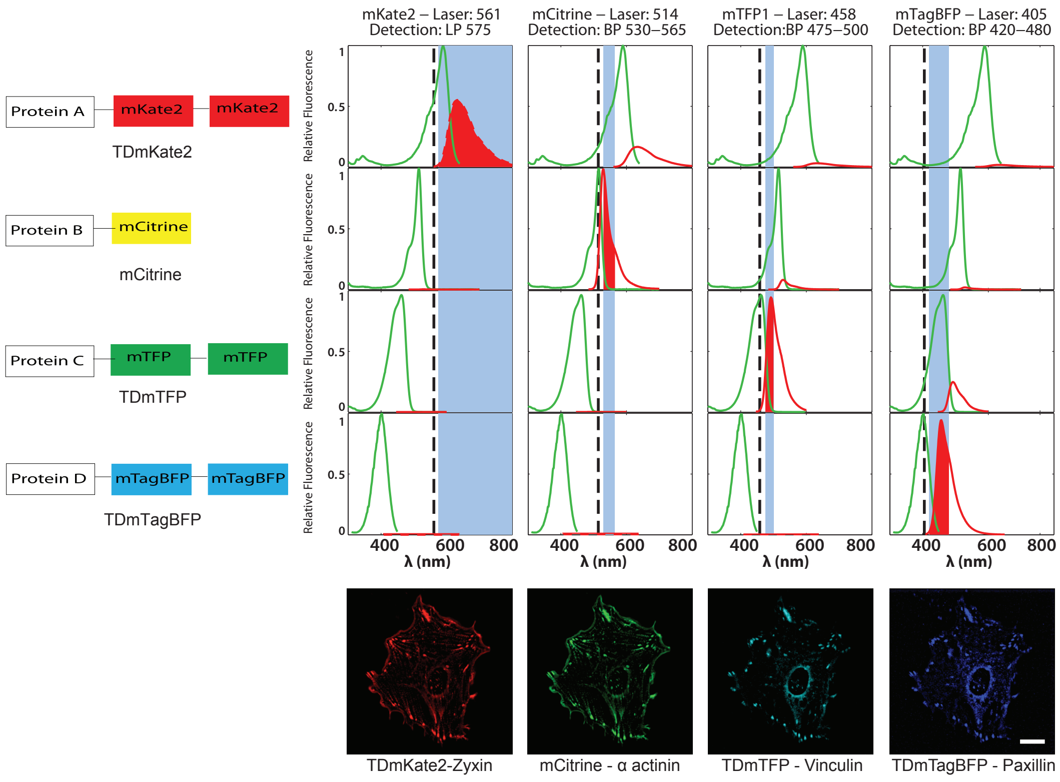


Figure 2.1: Sensitive multicolor imaging of cell-matrix adhesion sites - Optimization of reagents and microscopic parameters to achieve 4 color live cell imaging: Spectrally well-separated fluorophores (mKate2, mCitrine, mTFP and mTagBFP) are selected. Excitation (laser) and the detection settings are optimized for sequential imaging of the fluorophores. Tandem dimers (TD) for the relatively less-bright fluorophores (TDmKate2, TDmTFP and TDmTagBFP) are used to enhance their brightness under the given detection settings. Spectral properties of the chosen fluorophores under optimized imaging conditions (excitation/emission) is displayed to demonstrate the 4-color imaging of live fibroblast cells expressing focal adhesion proteins zyxin, α -actinin, vinculin and paxillin tagged with TDmKate2, mCitrine, TDmTFP and TDmTagBFP respectively. Scale bar: 20 μm.

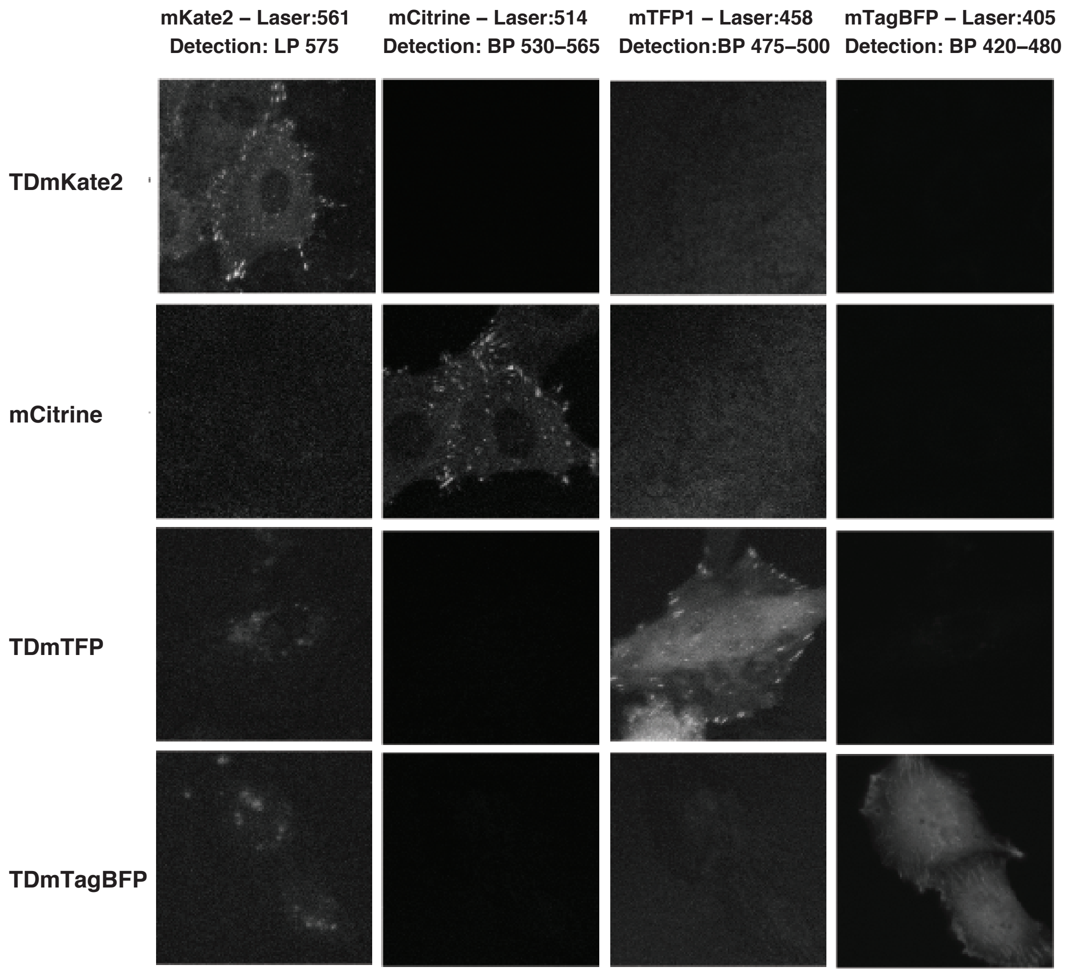


Figure 2.2: Bleed-through matrix of 4-color imaging system - Images of REF52 cells expressing focal adhesion protein tagged with TDmKate or mCitrine or TDmTFP or TDmTagBFP display detection setting specific signal without bleed-through into other imaging channels.

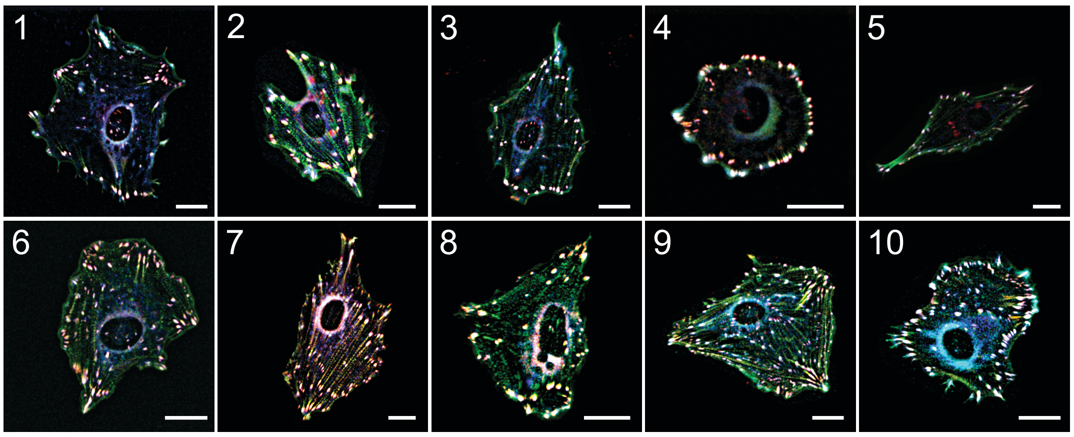


Figure 2.3: Imaging the steady state behavior of spread fibroblasts - Superimposed multicolor image s of spread REF52 cells transiently expressing TDmKate-zyxin (red), mCitrine-α actinin (green), TDmTFP-vinculin (cyan) and TDmTagBFP-paxillin (blue). Number denoted on the top left of the images represent cell ID. Scale bar: 20 μm .

are used to tag each protein. Plasmids of adhesion proteins zyxin, vinculin and paxillin with tandem (TD) mKate2, mTFP and mTagBFP respective developed in our research group are used. As mCitrine has a high quantum yield, mCitrine a-actinin is used without any tandem tag.

Having spectrally distinct tandem fluorophores tagged to adhesion proteins and optimal detection setting (Fig. 2.1), 4 color live cell imaging without spectral bleed-though (Fig. 2.2) is acquired. Steady state behavior as well as perturbation response of the spread fibroblast cells expressing TDmKate2-zyxin, mCitrine-α-actinin, TDmTFP-vinculin and TDmTagBFP-paxillin (Fig. 2.3) are monitored to reveal the dynamic compositional changes in cell-matrix adhesion sites and stress fibers of the fibroblast cells (Fig. 2.4).

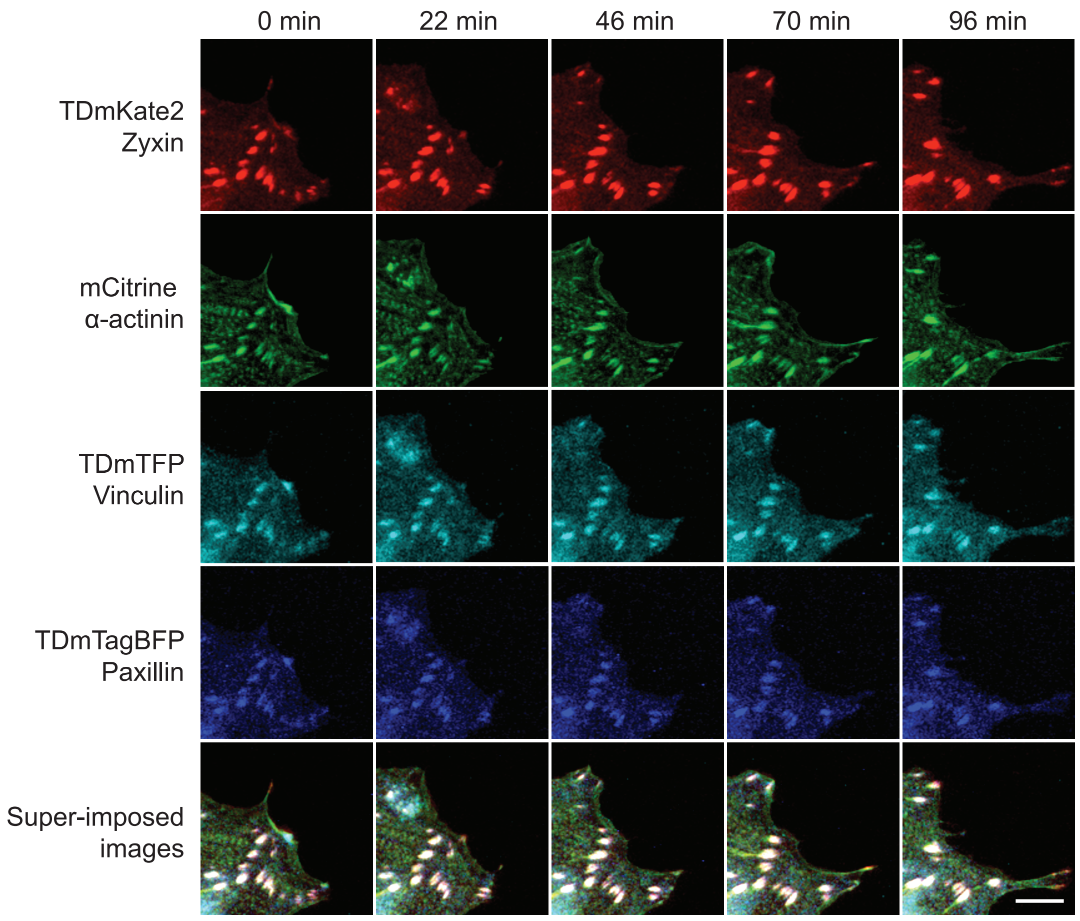


Figure 2.4: Imaging the multi-molecular dynamics of adhesion sites in live fibroblasts - Spread REF52 cells transiently expressing TDmKate-zyxin, mCitrine- α actinin, TDmTFP-vinculin and TDmTagBFP-paxillin are imaged every 2 minutes for about 60 to 90 min to monitor the dynamics of the chosen proteins during steady state assembly and disassembly of cell-matrix adhesion sites. A portion of the imaged cell is magnified to display the dynamics of the focal adhesion proteins at the time points (0, 22, 46, 70 and 96 minutes) sampled from the multicolor live cell movie. Scale bar: 10 μm.

2.2 Establishing approaches and tools to process the imaging data

Image acquisition procedures can sometimes generate unavoidable artifacts and noise in the data, which can rise from the deficit in optics or electronics of instrumentation, or deficit in fluorophores or other practical aspects. These artifacts and noise can be corrected during digital post processing of the images. Hence, it is crucial to properly process the image in order to obtain correct quantitative information from the microscopic data. Processing of images requires various steps to be performed in a rational order. As a part of this thesis, several custom-made computational tools were developed to execute these processing steps. A generic image processing pipeline 2.1 is developed to process multicolor time-lapse image sequences for quantitative analysis of adhesion sites. Different steps involved in processing and the rationale behind each step is described in the following section.

Table 2.1: Image Processing Pipeline

S.No.	Image processing steps
1	Correction of chromatic aberration in raw images
2	Image background correction
3	Bleaching correction
4	Low-pass filtration: Mean filtration of images
5	High-pass filtration and thresholding (spatial)
6	Lateral shift correction in time frames of images
7	2-way Kalman filtration (Temporal)
8	Low-pass filtration - nsmooth (spatial)

2.2.1 Correction of chromatic aberration

Chromatic aberration is one of the issues that arise due to the optical deficit in the multicolor imaging setup. Images of different imaging channels are distorted differentially along space. As adhesion sites are small structures which are few pixels ($< 6\mu m$) in length, these differential distortion dramatically affect their protein composition. Hence, it essential to correct the chromatic aberration to correctly capture the composition of adhesions at pixel level. Basically, there are two types of chromatic aberrations viz. longitudinal and lateral. The longitudinal chromatic aberration occurs due to focusing of light with different wavelength at different position along the optical axis. This aberration cannot be easily corrected. But its effect is overcome by imaging in wide-field mode. The lateral or radial chromatic aberration occurs because the lens focuses lights with different wavelength at different position in the focal plane to cause warping of the original image data. This can cause different distortion in each of the 4-color images, such that a pixel at a given x,y co-ordinate of the image, may correspond to different physical regions on the cell. The extent of distortion varies non-linearly along the entire focal plane of image. An objective that correct for a wide range of wavelength is still not available. But still this warping, when estimated carefully, can be digitally corrected by post processing. In order to achieve unwarping, multicolor beads are imaged under the imaging conditions of 4-color imaging system on every microscopic session. The beads encompass similar intensity profile in all 4 channels, with slight distortion in their position in different channels. By fixing, mCitrine image as standard, distortions in all other channels along the 2D space of the image are estimated. This estimate is used to correct the 4 color images, by unwarping the mKate2, mTFP and mTagBFP images in accordance with mCitrine images. All the image correction process are implemented in a custom-developed GUI with uses a java plugin bUnwarpJ Arganda-Carreras et al. (2006) to perform the same. The details description about 4-color beads image acquisition and usage of software for correction can be found under the Appendix 8.1.1.

2.2.2 Bleaching correction

In long time-lapse microscopic experiments, often the fluorophore is photo- chemically destroyed due to prolonged exposure to light. Moreover, in multicolor imaging system, different fluorophores bleach at different rate owing to their inherently different biophysical properties and the difference in the power of respective laser shined on them. Imaging the differentially bleached fluorophores generates multi-color images with false intensities resulting in incorrect representation of adhesion site composition. Any quantitative analysis of the bleaching time-lapse images can therefore lead to false interpretation of the data. In order to avoid bleaching, the exposure of the fluorophores to light can be reduced in following two ways. Firstly, reducing the laser power can be useful, but beyond certain limit it can decrease signal-to-noise ratio too much. Secondly, increasing the time interval between frames or decreasing the total number of time frames, can be helpful, but beyond certain limit it may conflict with the experimental interest to capture the dynamics in fine details for longer period. Even after optimization of the above mentioned aspects, it is not always possible to avoid photo-bleaching. Hence, the bleaching is corrected digitally in post-processing procedure. As the drop in intensity due to bleaching or the bleach curve is not always an exponential decay, bleaching correction is done by a simple-ratio method which assumes that the total intensity of a fluorescent protein within a cell remains unchanged during course of measurement. The total intensity of the cell in the first frame of a time-lapse movie of a given fluorescent channel is estimated, and the consequent frames corrected by multiplying the entire frame with a factor that restores the total intensity of the cell. A Matlab scrip is used to perform this correction is attached in Appendix 8.1.3. This approach will fail when cells move in and out of the frame. In frame with multiple cells, a polygon should be used to define the region of interests (ROI) for the cell which remains within the frame for the whole movie. In some cases with many moving cells, it is difficult to have a common ROI for all time frames. To handles such scenario, a Matlab GUI called TrackCell is developed (Appendix 8.1.4). It allows user to easily draw ROI for all the time-frames which is then used to properly correct bleaching.

2.2.3 Reduction of spatial low-frequency noise

Fluorescent-light microscopy images regularly contain a diffuse offset background, which adds up to the actual signal. Additionally, adhesion sites present in different regions of the cell experience different background which is contributed by the cytoplasmic component and therefore this background is dependent on the thickness of the cell. Consequently, the background cannot be removed by setting a uniform threshold for the entire image. As the background represents low frequency noises, they can be removed by high-pass filtration that can be achieved by two round of box filtration. In box filtration step (1), mean intensity within a box around each pixel is subtracted from it. A box size larger than the adhesion site is used in order to flatten the background. Box size is determined by number of pixels corresponding to $6\mu m$ is used to determine the box size. An uniform threshold is found iteratively to identify and flag the background pixels (Zamir et al., 1999). The second box filtration is applied on the image, but the average intensity within a box around each pixel is now calculated only from the pixels which are labeled as background. As a result, the actual background is subtracted. A second uniform threshold is found iteratively and the pixels below it are set to zero to exclude them from further analysis. An image filtration tool is developed to perform these processes and is described in Appendix 8.1.5.

2.2.4 Reduction of spatial and temporal high-frequency noise

Microscopic images acquired in laser scanning confocal microscopes often contain high frequency noise along the space and time domain due to deficit in the electronics that lack very highly sensitivity detection and cannot maintain the positions of point illumination with very high (sub-micrometer level) precision. Moreover, images that are acquired at less optimal imaging conditions (e.g. high power excitation for weakly expressed proteins) to obtain long time lapse movies, are inherently susceptible to augmented noise in the measurement. This high-frequency noise is observed in the intensity of time-lapse images along both space and time dimension. Eliminating this noise greatly helps to improve the quality of spatio-temporal analysis of the images. The high frequency noise along the space in the microscopic images is reduced by two rounds of low-pass filtration. The first round of filtration is performed before high-pass filtration, where the high-frequency noise is reduced by performing a mean filtration of the image, at the least possible (single pixel) resolution. Each and every pixel in the image is replaced by the mean of the pixels within 1 pixel radius. This reduces noisy pixels which are different from its neighbors. The high-frequency noise along the time domain is removed using a bidirectional Kalman filter for stacks. This is followed by the second low-pass filtration along the space domain which involves a robust interpolation-like 2D smoothing performed by a recent multidimensional smoothing algorithm Garcia (2010). The noise along space and time, when removed ensures smooth changes in intensity levels along space and time, and thereby enhance the quality of compositional imaging of adhesion site dynamics.

2.2.5 Correction of lateral shifts between time-lapse frames

During the course of time lapse microscopy, it is possible that the imaging field is slightly shifted due to certain unavoidable reasons. It is essential to correct these shifts by image registration methods, to properly analyze the dynamics. The extent of shift in the images can be estimated by cross correlation of images and registered. Registration of all consecutive time frames may not be useful, as it could also distort the dynamics when a cell is moving. Hence, the time-frames of the movie with abrupt shifts in the imaging field is manually noticed, and the selected frames are corrected for the shift using an image registration algorithm (Guizar-Sicairos et al., 2008) wrapped using a Matlab code as described in the Appendix 8.1.6.

Detailed protocol for step by step processing of the data is described in the Materials and methods section 7.3. Additionally the user instruction and brief description of image processing tools and their source code can also be found in Appendix 8.

Chapter 3

Resolving the dynamics of protein networks in adhesion sites

Compositional diversity is an elementary property of adhesions sites. High diversity in the composition of adhesion sites emanates from the diversity in the protein network states and function. Hence it is intriguing to analyze and visualize the large protein network states in adhesion sites. Multicolor images of adhesion site proteins, acquired and processed can be used to resolve the composition and thereby the network state. Inferring molecular composition from the superimposed multi-color images is challenging when more than two components are imaged. To address this fundamental issue, compositional imaging, was developed to analyze and display multi-component composition of adhesion sites (Zamir et al., 2008). It is the first approach to resolve molecular signatures of cell matrix adhesion sites through quantitative multicolor imaging of adhesion site composition. This approach revealed variation in the composition between as well as within adhesion sites, offering a strong evidence for inter and intra adhesion site heterogeneity. Molecular organization of adhesion sites with distinct sub-domain composition were visualized. Additionally, sensitivity of various compositions to rho-kinase inhibition was also evaluated (Zamir et al., 2008). In this thesis, the compositional imaging approach is improved and extended to analyze the live cell multicolor imaging data, to study dynamics and compositional heterogeneity of adhesion sites with high-spatial and temporal resolution.

3.1 Establishing the approach: live cell compositional imaging

Composition imaging is a methodology based on clustering of pixels from multi-color microscopic images, to identify clusters with similar composition and map their cellular distribution. In order to perform compositional imaging, the multi-color images are processed as described in the previous chapter. A computational tool based on Matlab GUI is developed to perform compositional analysis of time lapse images. Some important amendments are incorporated in the previously described method (Zamir et al., 2008) and are described in the following section. The detailed step by step processing of the data and compositional analysis is described in Appendix 8.2.

3.1.1 Data normalization and noise exclusion

In multi-color imaging, imaged cells are transiently transfected with plasmids with four fluorescent labeled proteins and it is difficult to control the amount of plasmid incorporated into each cell and there by the protein expression. Different cells may have different intensity range. Dynamic range of pixel intensity is also affected by imaging acquisition setting, such as signal amplification and detector gain, which are usually tailored for each channel according to the expression level of proteins. Intensity of the pixels harvested from images of multiple cells should be normalized to a common dynamic range to remove the difference due to imaging and/or expression level within proteins and within cells. In order to achieve this, pixels intensity for each cell is normalized by dividing the distribution of each fluorescent protein intensity by the median of its non-zero elements (i.e. Intensity normalization). For each pixel the magnitude is calculated as the distance between the origin and the normalized protein intensity level coordinates in the multidimensional composition space. Fig. 3.1 shows distribution of distribution of pixels in 3D composition space, color coded with their magnitude. The magnitude of each pixel is used to eliminate noisy pixels. As many of the very low magnitude pixels are noisy, they are removed iteratively from the time-lapse movie. For data without noisy pixels, the amplitude threshold is set to zero. The usefulness of this process is explained in the subsequent section.

3.1.2 Spherical distance hierarchical clustering

In addition to intensity normalization, a second normalization called spherical or composition normalization is performed in order to cluster pixels based on their stoichiometry, as described previously (Zamir et al., 2008) (also refer methodology 7.3 for details). The normalized factional intensity for each pixel is calculated in such a way that the sum of square of intensity level in all channels is one i.e. the Euclidean norm is one. As a result of spherical normalization, all the pixels with same stoichiometry are considered equally irrespective of their absolute intensity. For instance a pixel with Paxillin:Vinculin intensity as 20:10 is considered equally as 2:1. At the same time, a noisy background pixel with low magnitude, for e.g. 0.002:0.001 will also be considered equally. So, it is essential to remove very low amplitude noisy pixels from the clustering data, as they interfere with the clustering process. Pixels are object in the multidimensional space with coordinates corresponding to the levels of protein contained in the pixel. A spherical normalization basically projects these objects to N-dimensional sphere and therefore distance between any two objects should be measured along the N-sphere (FIG. 3.2). This is a fundamentally important amendment against the previous compositional clustering methodology (Zamir et al., 2008) that uses Euclidean distance. A multi-thread ward hierarchical clustering algorithm that uses spherical distance measure is developed by Prof. Hernan Grecco and is used to cluster the pixels and it is hereafter referred as spherical clustering.

3.1.3 Resolving the compositional dynamics

As adhesion sites are highly dynamic structure, analyzing the multidimensional protein dynamics is needed to understand how adhesion site assemble, disassembly and steadily slide through. In Zamir et al. (2008), compositional imaging approach was applied on multicolor images of fixed cells stained with antibody

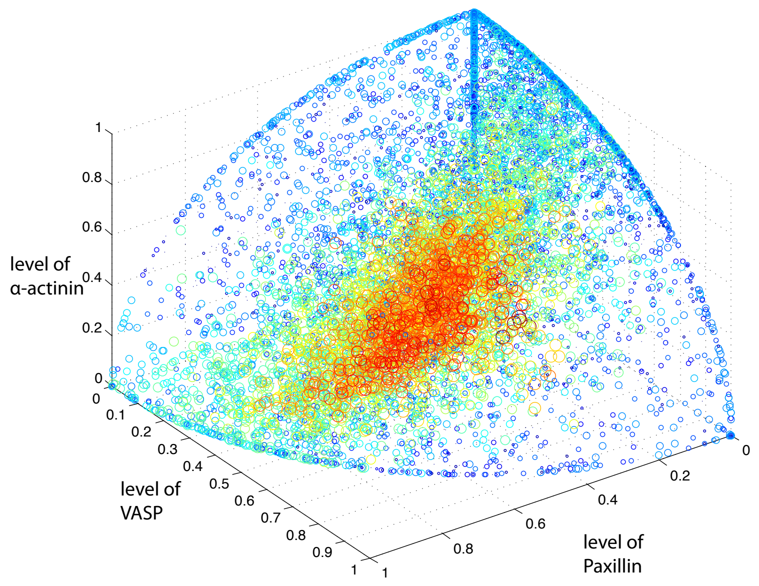


Figure 3.1: Distribution of pixels and their amplitude in composition space - Composition normalized pixels are projected in the multi-dimensional composition space with its dimensions corresponding to the relative levels of paxillin, VASP and α-actinin. Composition normalization is performed considering considering the forth imaging component, zyxin. The color represents the amplitude of each pixels from low (blue) to high (red).

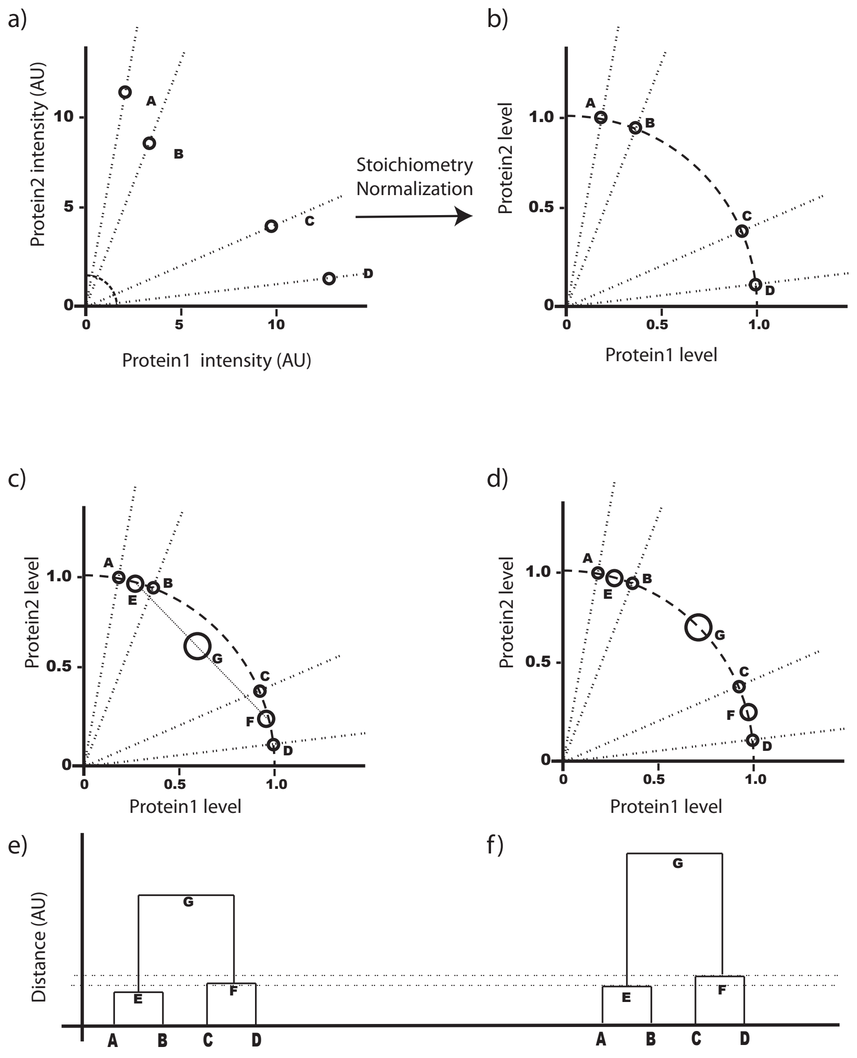


Figure 3.2: Stoichiometry normalization and spherical distance - Each pixel from multicolor image yields the level of multiple proteins. Hence each pixel can be considered as an object in multidimensional space corresponding to the level of each protein. In order to cluster pixels based on their composition and not on the absolute levels proteins, each pixel is repositioned in the stoichiometric composition space. This is achieved by stoichiometry normalization i.e. normalizing the vector of proteins levels in a pixel such that the Euclidean norm equals to one. This normalization projects the pixels (e.g. A,B,C,D) into a multidimensional arc (a,b). Clustering of stoichiometry normalized pixels by considering Euclidean distance disturbs the stoichiometry when pixels are merged to create new clusters (c) and thereby distorts the dendrogram (e). Hence pixels are clustered by adapting the spherical distance i.e. the distance measure along the multidimensional arc. This clustering approach preserves the stoichiometry of protein levels when two pixels are merged to form new clusters (d) and therefore generates an undistorted dendrogram for hierarchical clustering (f).

for adhesion proteins. Hence, the molecular signatures identified from the snapshot of single time point, does not divulge any information about the dynamics of these molecular domains. In this thesis, the compositional imaging approach is applied on dynamic multi-color time-lapse image series, to reveal spatio-temporal regulation in composition of adhesion sites.

3.1.3.1 Visualizing high-dimensional compositional dynamics

Compositional analysis of dynamics involves the canonical compositional imaging step, where pixels from images of same cell at different time point are co-clustered. In order to demonstrate this, a 3 color live cell imaging system is chosen. REF52 cell transiently expressing mTagBFP-Paxillin, mCitrine Zyxin and mKate2-VASP is treated with actomyosin inhibitor after the first acquisition time point and the dynamics is imaged at an interval of 1 minute per frame (Fig. 3.3 a). Pixels harvested from these 3 color images are normalized and clustered using hierarchical clustering approach. Obeying the hierarchy of the dendrogram, the nodes are explored to choose the four nodes or clusters A,B,C and D in the dendrogram generated from the clustered data (Fig. 3.3 b). The chosen clusters are spatially mapped back to the images by color-coding the corresponding pixels. Cluster selection process and cluster visualization is manually iterated several times until compositional signatures are identified. The final compositional movie allows to visualize the spatial and temporal dynamic behavior of the chosen composition (Fig. 3.3 c).

3.1.3.2 Unraveling transitions in composition

In addition to dynamics visualization, several other aspects of composition dynamics are proposed and extracted. 1) Contextual response of compositions: From the time-lapse compositional movie, the change in abundance of chosen clusters along the time can be estimated (Fig. 3.4 a). This reveals functional response of different molecular domains (A,B,C and D) of adhesion sites disassembly induced by inhibition of actomyosin contractility. 2) Compositional transition and its probability: The compositional movie displays dynamic changes in the composition at pixel level. This change in composition in pixels at a given

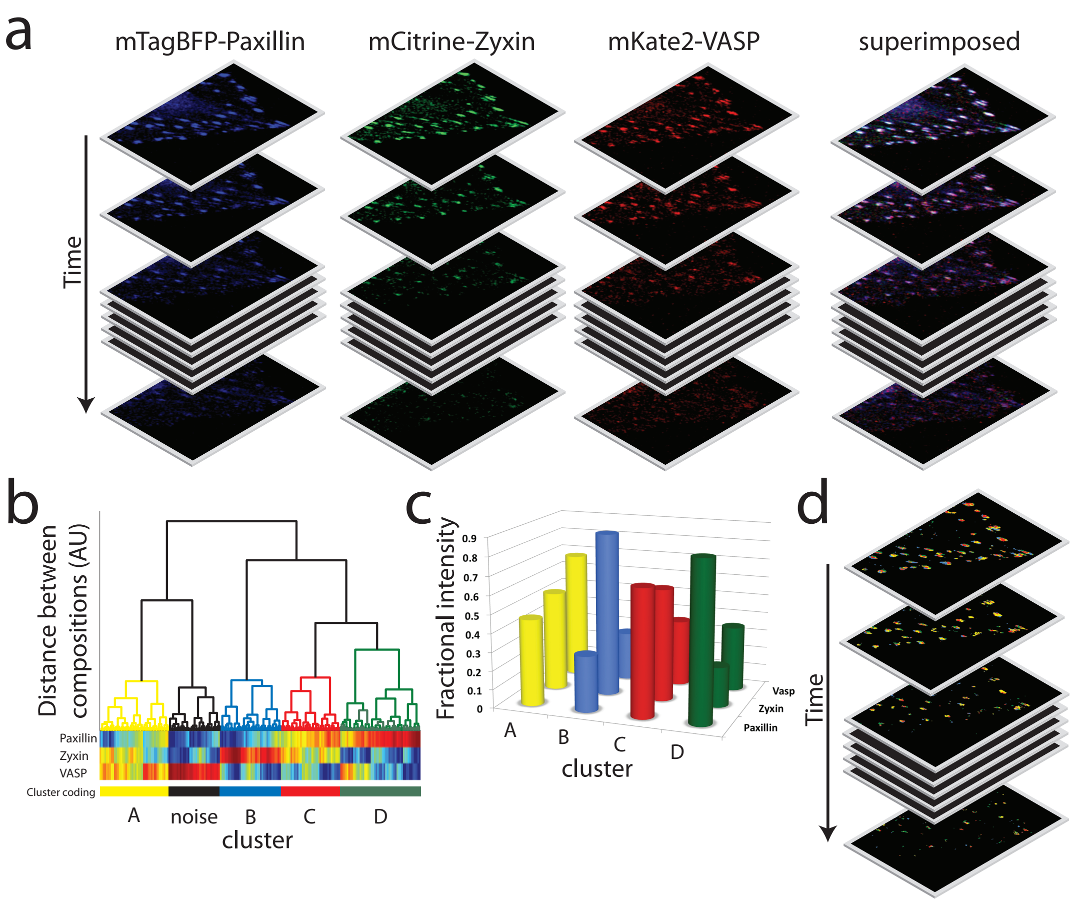


Figure 3.3: Demonstration of compositional imaging of 3 component data - Time lapse multicolor images of live REF52 cells expressing mTagBFP-paxillin, mCitrine-zyxin and mKate2-VASP are acquired at an interval of 1 minute.(a) REF52 cells were treated with Y27639 after the first frame to induce disassembly of focal adhesions. (b) Pixels harvested from the images of multicolor movie are clustered in 3D composition space by hierarchical clustering to generate dendrogram. Nodes are iteratively chosen in the dendrogram to identify clusters (A,B,C and D) of pixels with similar composition. (c) Compositional signature of each identified cluster is extracted. (d) Clusters are mapped back to their physical location in the images by color coding the pixels to generate compositional image series, which divulge spatial organization and temporal behavior of the identified clusters.

physical location in the image of the cell from one type to another (e.g. A to C) is termed as transition in composition. The probability of all possible transitions can be estimated for every pair of consecutive time frames. By setting a cut-off, the compositional transition can be visualized (Fig. 3.4 b). The topology of transitions evolve with time. As these are transition of composition in three-dimenstional space, they can be easily visualized in the composition space (Fig. 3.4 c). Initially, before the treatment of the drug (time points: 0 min to 1 min), all compositions exhibit transition with the same type, but later response specific changes in the transition network can be observed. At time 3 min to 4 min, composition B is the fist to disappear, hence an arrow is pointed from B to empty transition in the 3D composition space (Fig. 3.4 b, c). The change in topology of compositional transition in adhesion sites implicitly converge the mechanism and different intermediate stages of adhesion sites disassembly.

3.2 Results I: Compositional dynamics of adhesion sites in spread fibroblasts

To characterize and visualize diverse composition of cell-matrix adhesion sites in spread fibroblast, 10 REF52 cells, ectopically expressing TDmKate-zyxin, mCitrine-α-actinin, mTFP-vinculin and TDmTagBFP-paxillin are imaged at a regular interval of 2 minutes under steady-state conditions (Fig. 2.3, 2.4). The multicolor time-lapse images are processed for quantitative analysis as described in the chapter 2. Compositional imaging is then performed at two levels: 1) the whole cell level, by considering all the pixels from adhesion sites, stress fibers and cell membrane and 2) adhesion site level, considering pixels pertaining to adhesion sites alone.

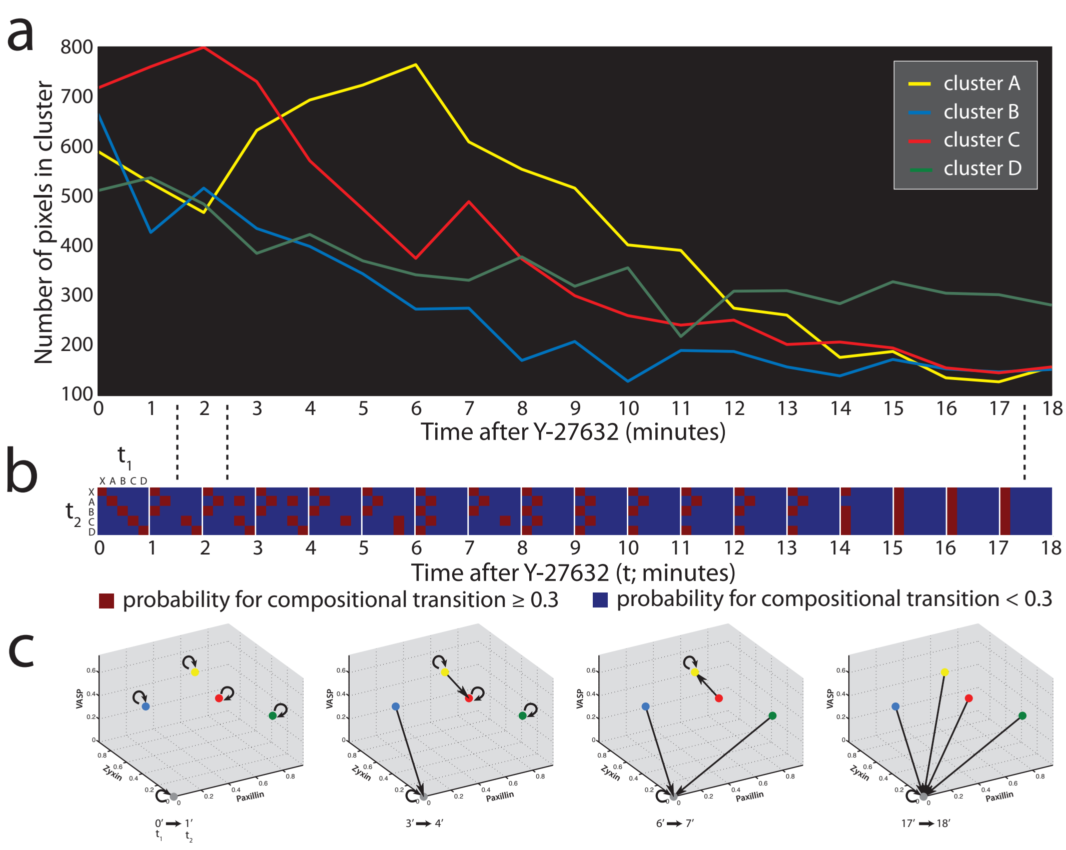


Figure 3.4: Divulging compositional transition and their pathways in 3 component data - The movie of compositional images of REF52 cells allows quantification of abundance of clusters as well as probability of transition of one cluster into another along all the time frames. (a) Different clusters (3.3 respond differentially to the actomysoin inhibition, showing differential decline in the abundance. (b) The topology of compositional transition also changes along time. (c) Transition of clusters is visualized in the 3D compositional space for sampled time points.

3.2.1 Resolving compositional dynamics within the whole cell

3.2.1.1 Visualizing spatiotemporal dynamics of composition

Multicolor images of REF52 cells in addition to adhesion sites (zyxin, α-actinin, vinculin and paxillin) also visualizes stress fibers (zyxin, α-actinin) and cell membrane (α-actinin). To resolve the compositional dynamics of all these structures, the pixels harvested from whole cell is normalized, binned and clustered by spherical hierarchical clustering. Nodes are iteratively chosen from the dendrogram of the clustered pixels and significant compositions (A to H) are identified (Fig. 3.5 a b). In order to ensure the robustness of the chosen composition, their relative abundance in different cells is estimated (Fig. 3.5 c). Identified compositions are color mapped on the multicolor movies to generate compositional images (Fig. 3.6). The compositions revealed distribution and spatial organization of molecular (domains) composition in various structures such as adhesion sites, stress fibers and cell membrane. This allowed identification of structure specific composition. Additionally, the compositional movies visualized the dynamics of various molecular domains during assembly and disassembly of adhesion sites, treadmilling of stress fibers, steady-state protrusion and retraction of cell membrane.

3.2.1.2 Resolving pathways of compositional transitions

Compositional images reveal dynamic changes in the compositions from one type to another. This suggests that the underlying protein networks also dynamically change. In order to understand how these protein networks are assembled and disassembled, it is required to estimate the probability of all possible compositional transitions. Probability of transition of composition is estimated in all spread fibroblast cells to investigate the robustness of the composition in multiple cells (Fig. 3.7 a). The pair-wise correlation of probability of transition revealed high degree of regularity in all observed fibroblast cells (Fig. 3.7 b). To derive a representative topology of transitions, probability of transition is averaged over all the cells (Fig. 3.8). The probability of transitions in compositions reflects the probability of transition of underlying protein network. This matrix of average

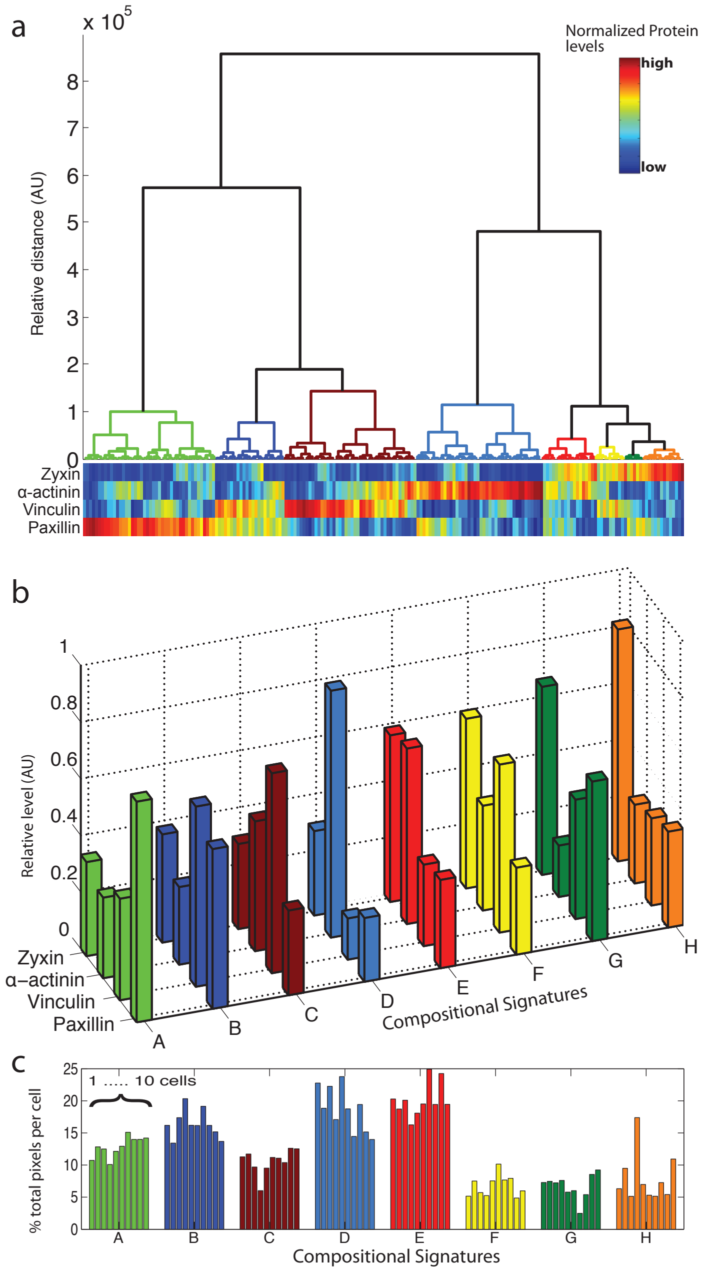
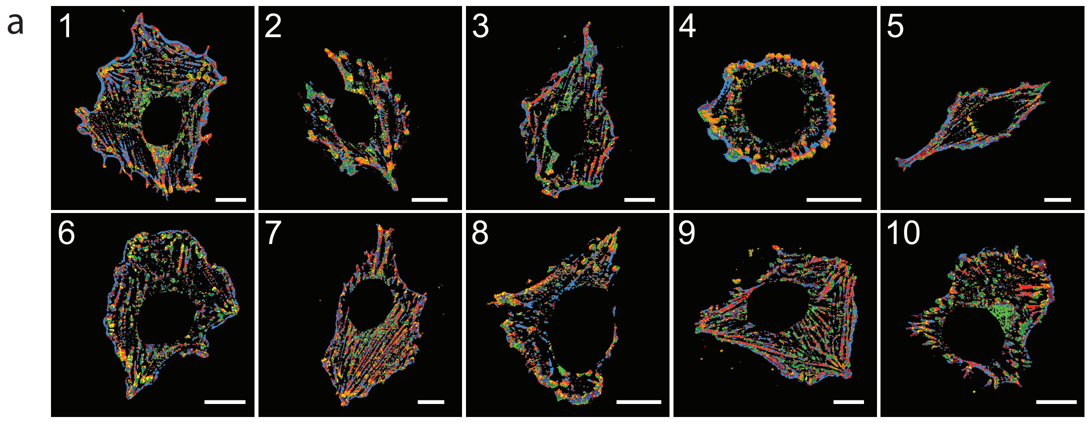


Figure 3.5: Compositional analysis of whole fibroblast cells reveals compositional signatures consistently prevalent in all the cells - Pixels from the time-lapse multicolor images of 10 REF52 cells are co-clustered by hierarchical clustering to identify groups with similar levels of zyxin, α-actinin, vinculin and paxillin. (a) Nodes corresponding to the identified clusters are color coded on the dendrogram of clustered pixels. (b) Compositional signatures of the identified clusters (A to H) are represented by the bar plot. (c) The relative abundance of clusters (A to H) in a cell is comparable between all 10 cells signifying their robustness.



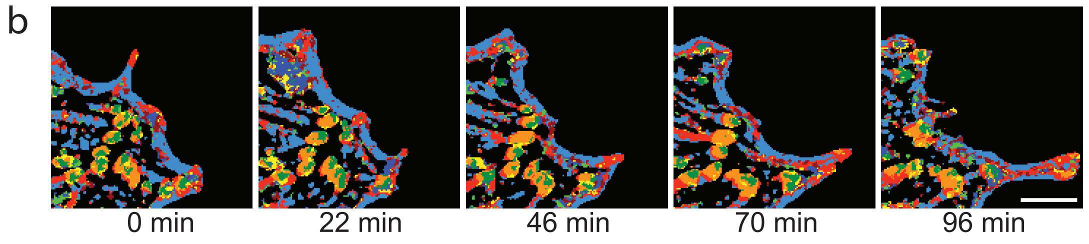


Figure 3.6: Compositional imaging of whole fibroblast cells visualizes spatial and temporal behavior of molecular domains in adhesion sites and stress fibers - Molecular signatures identified by compositional analysis 3.5 are mapped on the images of cells to generate compositional images of the spread REF52 cells. (a) Compositional images (cell ID 1 - 10) reveal spatial organization of identified compositional signatures on various structures that includes adhesion sites, stress fibers and cellular protrusions. Cell ID: top right corner of the image. Scale bar : 20μm .(b) Compositional images series also visualize the dynamic changes in the composition. A portion of compositional images of cell (1) for sampled time points (0, 22, 46, 70 and 96 minutes), is magnified to display molecular subdomains and dynamic changes in composition during steady state protrusion of fibroblast cell. Scale bar : 10 μm.

probability of transition divulge information about pathways of protein network transitions. These pathways of compositional inter-conversion offer insight into construction and destruction of network-states. Moreover, it also unveils flux strength of the identified paths. Further, to understand the temporal pattern of the compositional transitions, the probabilities of transition is estimated for all the imaged time frames for each cell. The probabilities of transitions are stable along time in all cells during steady state (Fig.3.9).

3.2.2 Resolving compositional dynamics exclusively within focal adhesions

3.2.2.1 Visualizing spatiotemporal compositional dynamics

To resolve the dynamics of focal adhesions at a higher compositional resolution, pixels pertaining to adhesion sites are chosen and all other structures are excluded from the analysis. The normalized pixels harvested from the whole cell are filtered by choosing an amplitude threshold high enough to select pixels from focal adhesions. Filtered pixels are then clustered by spherical hierarchical clustering. Nodes are iteratively chosen from the dendrogram of the clustered pixels and significant compositions (A to F) are identified (Fig. 3.10 a b). Similar to the previous analysis relative abundance of identified composition in different cells is estimated (Fig. 3.10 c). Compositional images (Fig. 3.11 a) generated by color mapping pixels revealed molecular domains and their spatial organization in adhesion sites with higher resolution. As all the compositions localize in focal adhesion, compositional images rendered deeper insight in compositional patterns at sub-adhesion site level (Fig. 3.12). As a corollary the compositional movies also visualized the dynamics of molecular domains during assembly and disassembly of adhesion sites at various sub-cellular regions of the cell including regions of cellular protrusion and retraction (Fig. 3.11 b). This enabled opportunity to infer multi-molecular mechanism behind the focal adhesion assembly and disassembly (Fig. 3.13). The mechanosensitivity and sliding property is also reflected by the spatial patter of molecular composition within the focal adhesion (Fig. 3.14).

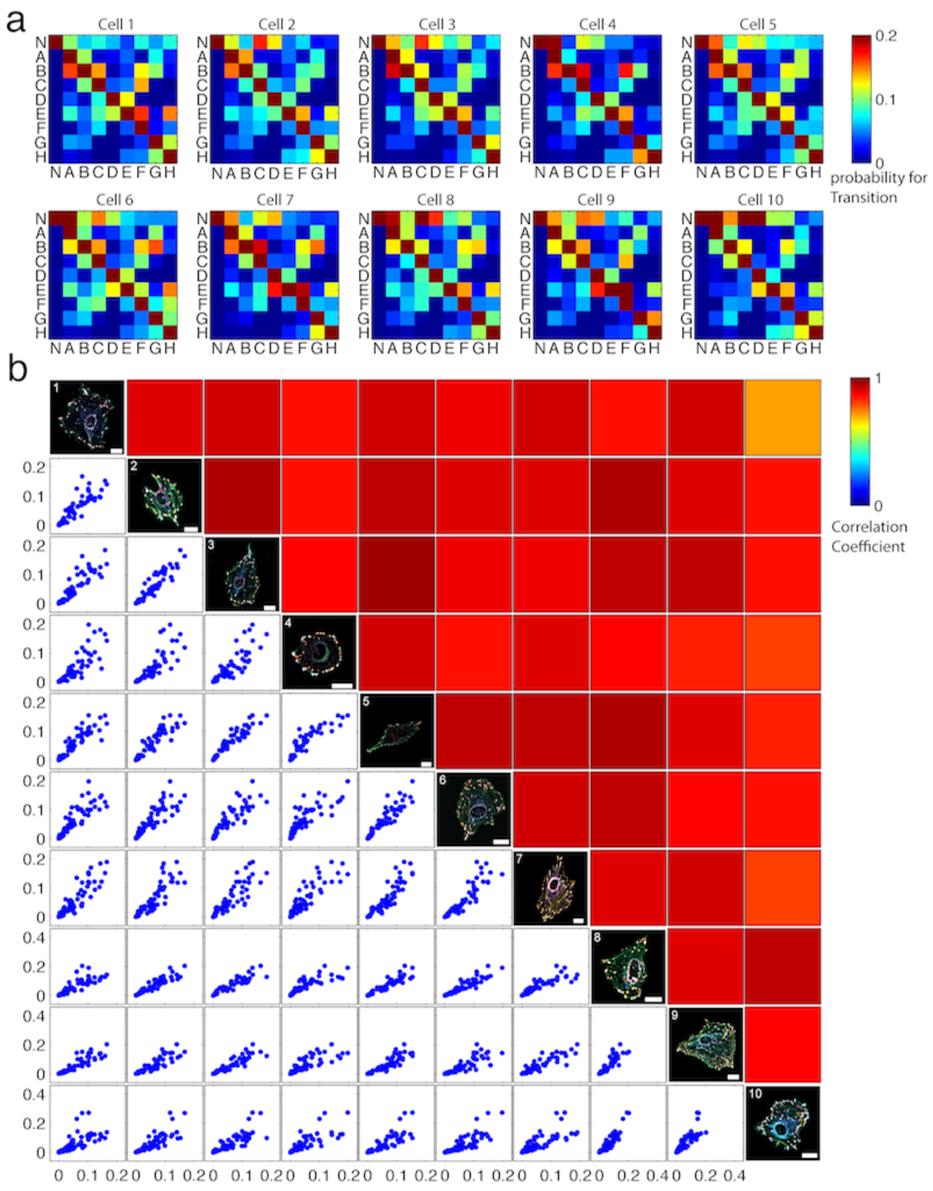


Figure 3.7: Compositional transitions are comparably uniform in spread fibroblast cells - (a) The heat maps represent the probability for all possible transition between the compositions (A to H) calculated as described in the methods section (Sec. 7.3) from the compositional movies of spread fibroblast cells (Fig. 3.6). N is an empty composition i.e. region without any composition. Rows are composition at t_{i+1} , columns are composition at t_i. (b) The consistency in the probability for transition between the 10 cells is presented as the scatter plots showing comparison of probabilities for transitions between all pairwise combinations of the cells. The diagonal of the matrix plot represents superimposed multicolor images of all 10 cells which are compared. Scale bar: 20 μm. The heatmaps above the diagonal represent the correlation coefficients for the corresponding pairwise comparisons of transition probabilities between the cells.

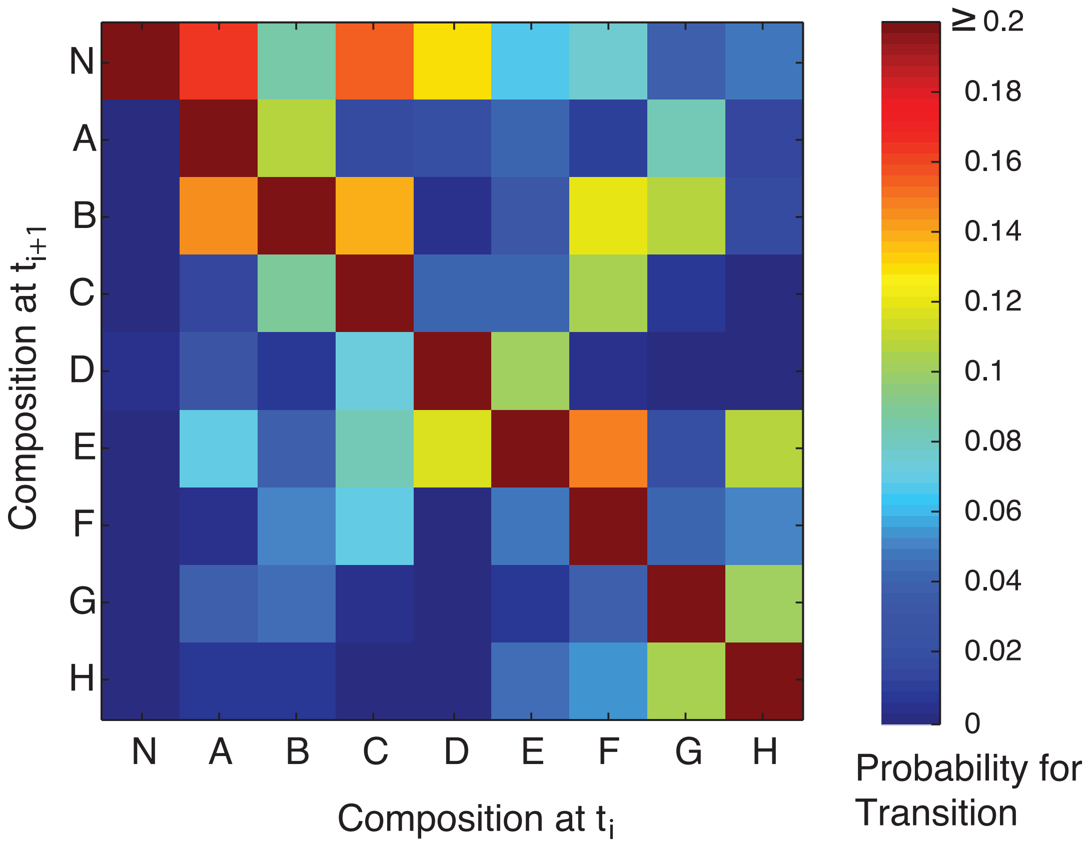


Figure 3.8: Average probabilities of compositional transition in adhesion sites - The heat maps represent the mean probability of all possible transition between the compositions (A to H) observed in 10 spread fibroblast cells. N is an empty composition i.e. region at which the fluorescence level of all imaged proteins is below their corresponding background thresholds. Transition of any of the composition to N means disappearance of a composition and vice versa.

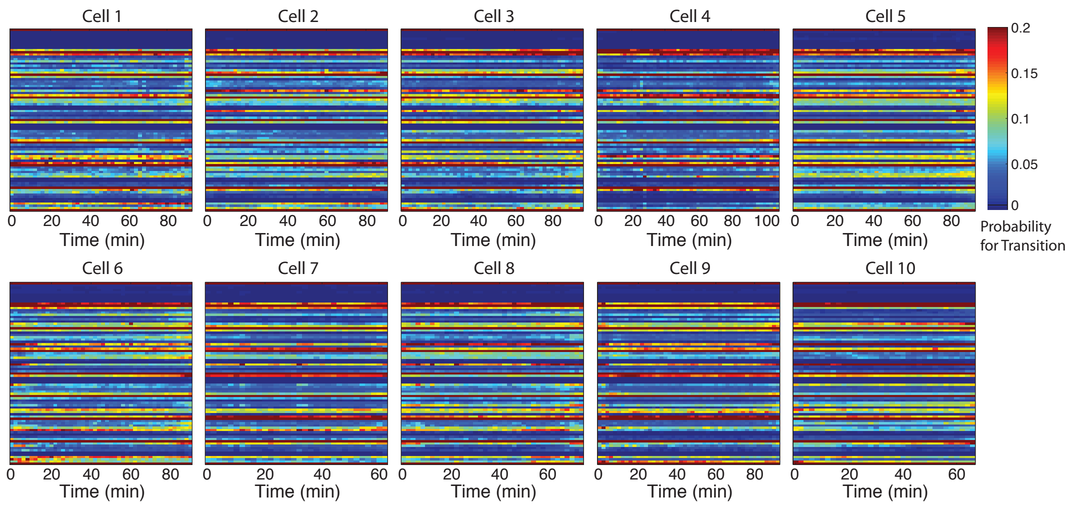


Figure 3.9: The occurrence probabilities of compositional transitions is generally uniform overtime in unperturbed fibroblasts - The heat map represents the probabilities for transition between all possible compositions, estimated for each two consecutive frames along the acquired period. The probabilities remain mostly unaltered along time in all of the spread fibroblast cells.

3.2.2.2 Resolving pathways of compositional transition

Analogous to the whole-cell analysis, the probability of transition of composition specific to focal adhesion is estimated in all spread fibroblast cells (Fig. 3.15 a). Comparable with the previous analysis, the pair-wise correlation of probability of transition revealed high degree of uniformity in all observed fibroblast cells (Fig. 3.15 b). The matrix of average probability of transition divulge information about pathways of protein-network transitions pertaining to focal adhesions (Fig. 3.16). Resembling the previous analysis, probability of all transitions are mostly stable along time in adhesion sites of all cells (Fig. 3.17). But additionally, minor fluctuations in the probability are also observed for some of the compositional transitions.

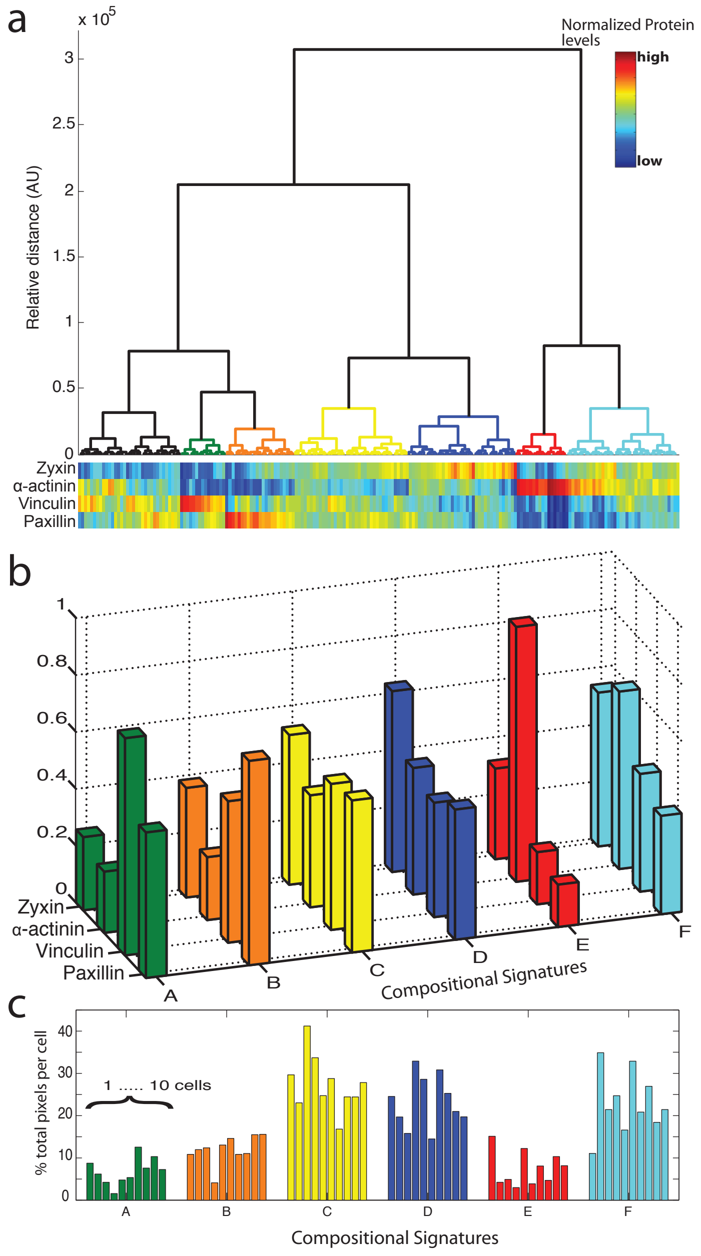


Figure 3.10: Compositional analysis of focal adhesions in fibroblast cell reveals compositional signatures comparably prevalent in all the cells - Pixels of focal adhesion segmented from the time-lapse multicolor images of 10 REF52 cells are co-clustered by hierarchical clustering to identify groups with similar levels of zyxin, α -actinin, vinculin and paxillin. (a) Nodes corresponding to the identified clusters are color coded on the dendrogram of clustered pixels. (b) Compositional signatures of the identified clusters (A to F) are represented by the bar plot. (c) Although the relative abundance of clusters (A to F) in a cell is comparable between all 10 cells, it shows certain degree of variability.

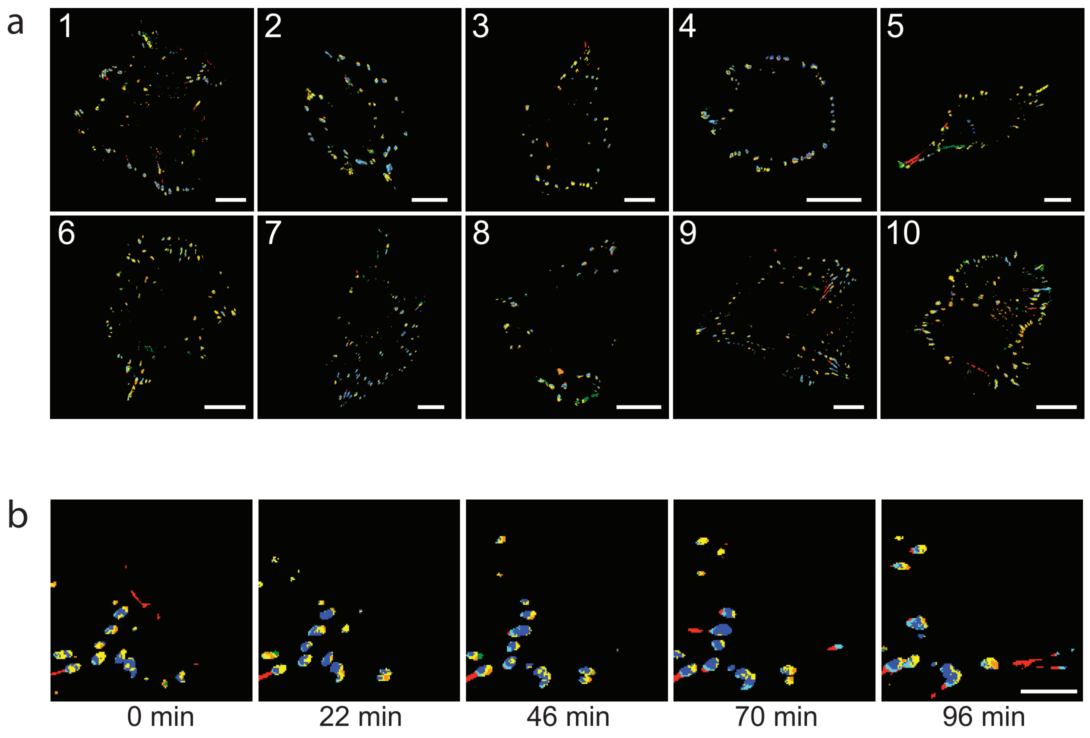


Figure 3.11: Compositional imaging of focal adhesions in fibroblasts visualizes spatial organization and dynamic regulation of molecular subdomains - (a) Molecular signatures identified by compositional analysis (Fig. 3.10) are mapped on the images of cells to generate compositional images of the spread REF52 cells. Compositional images reveal spatial organization of identified compositional signatures on adhesion sites and visualize subdomains of these structures with higher resolution. Cell ID: top right corner of the image. Scale bar : 20 μm. (b) Compositional images series also visualize the dynamic changes in the composition. A portion of compositional images of cell (1) for sampled time points (0, 22, 46, 70 and 96 minutes), is magnified to display molecular subdomains and dynamic changes in composition during protrusion of fibroblast cell showing steady state focal adhesion assembly and disassembly. Scale bar: 10 μm.

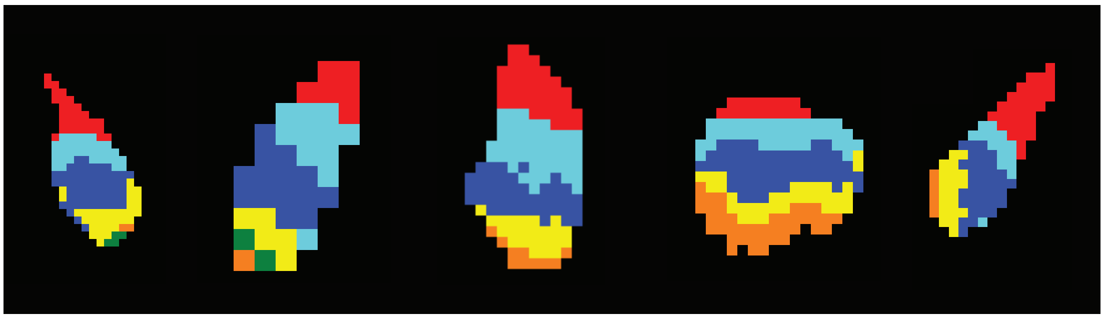


Figure 3.12: Spatial organization of network states within adhesion sites - Adhesion sites exhibit various subdomains with distinct network states. Additionally, the protein networks states also follow particular order of spatial organization. The color code of network-state or composition corresponds to the compositional imaging in the Fig. 3.10.

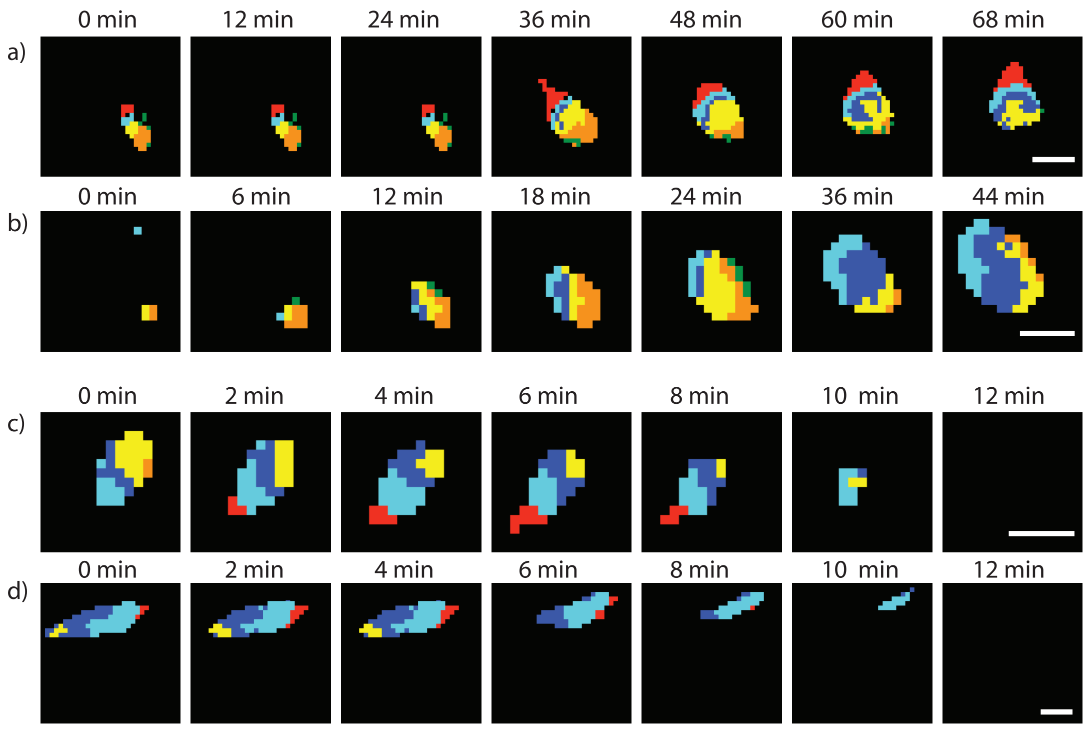


Figure 3.13: Compositional imaging reveal pathways of focal adhesion assembly and disassembly - Adhesion sites assembly and disassembly exhibit various stages and orderly appearance/disappearance of sub-domains of protein network-states within adhesion sites. Scale bar: 2 μm.

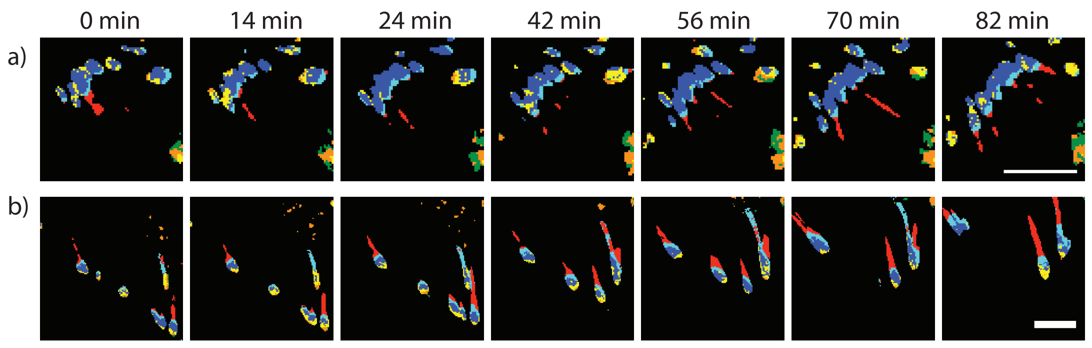


Figure 3.14: Spatial organization of network states within adhesion sites reflect mechanical force exerted by stress fibers - Adhesion sites that are not connected to stress fibers slide very slow (a) in contract those that are connected to stress fibers(b). The former adhesion sites therefore exhibit only blue(D) and cyan(F) (a), where as latter experiencing mechanical pulling by stress fibers show peculiar spatial ordering (yellow(C), cyan(F), blue (D) and red (E)) of network states (b).The color code of network-state or composition corresponds to the compositional imaging in the Fig. 3.10. Scale bar: 10 μm.

3.3 Results II: Analyzing the compositional changes in response to actomyosin inhibition

An important fundamental property of adhesions sites is mechano-sensitivity. Maturation of focal adhesion (FA) is dependent on the force excreted on the adhesion sites by pulling of actin filaments. Therefore inhibition of force by rho kinase inhibitor (Y27362) that disrupts actomyosin contractility in actin filaments, causes FA disassemble. Response of adhesion sites to force inhibition is a reversible process. Washing out the drug allows recovery of actin filaments and focal adhesion. By repeating cycles actomyosin inhibitor treatment, adhesion sites repeated assembly and disassembly can be monitored in the same cells (Fig. 3.18). Pixels yielded from the 4-color time lapse images are clustered in a 4-D composition space with each dimensions corresponding to the levels of proteins. Compositional analysis of pixels revealed clusters (A-G) with diverse compositional signatures. Compositional imaging is used to understand the dynamic behavior of various composition. The temporal abundance of the chosen com-

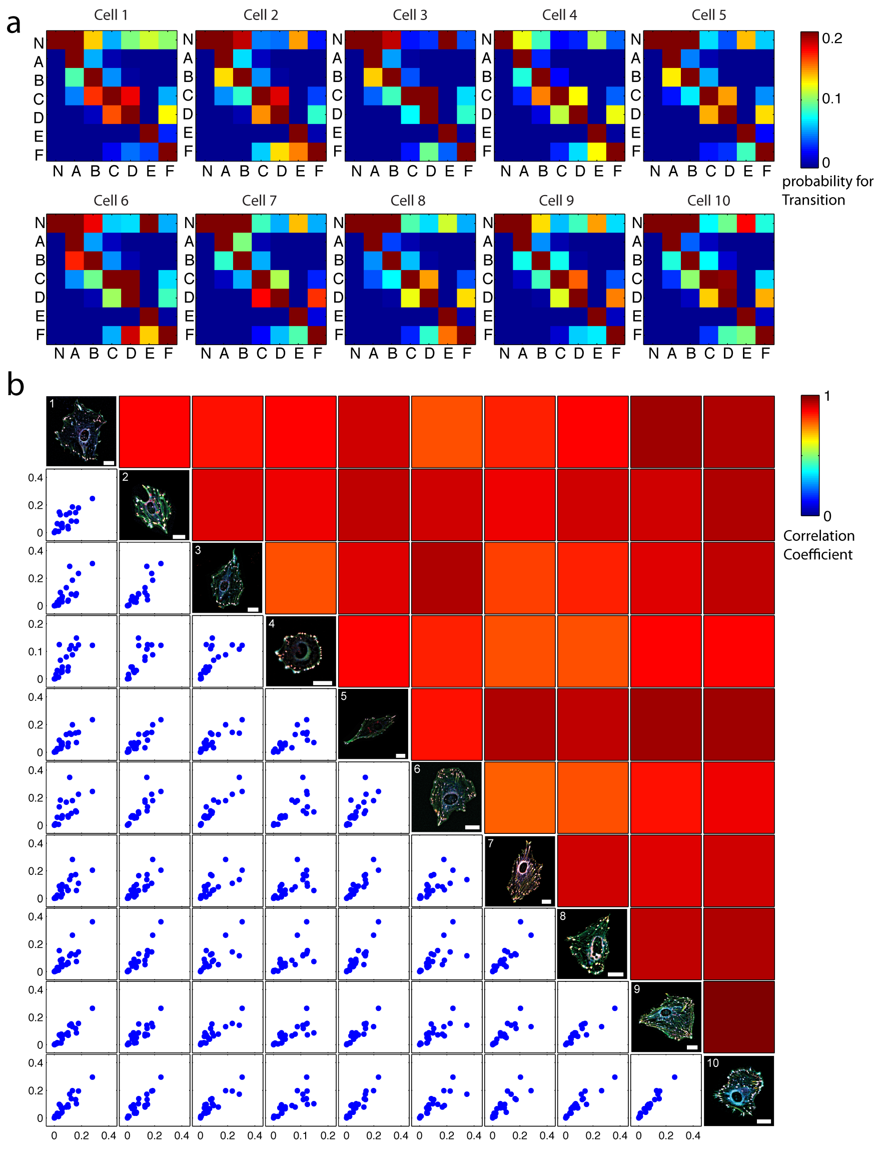


Figure 3.15: Compositional transitions within focal adhesions are comparably uniform in spread fibroblast cells - (a) The heat maps represent the probability for all possible transition between the compositions (A to H) calculated from the compositional movies of spread fibroblast cells (Fig. 3.11) as described in methods section (Sec. 7.3). N is an empty composition i.e. region without any composition. Rows are composition at t_{i+1} , columns are composition at t_i. (b) The consistency in the probability for transition between the 10 cells is presented as the scatter plots showing comparison of probabilities for transitions between all pairwise combinations of the cells. The diagonal of the matrix plot represents superimposed multicolor images of all 10 cells which are compared. Scale bar: 20 μm . The heatmaps above the diagonal indicate the correlation coefficients for the corresponding pairwise comparison of probabilities between the cells.

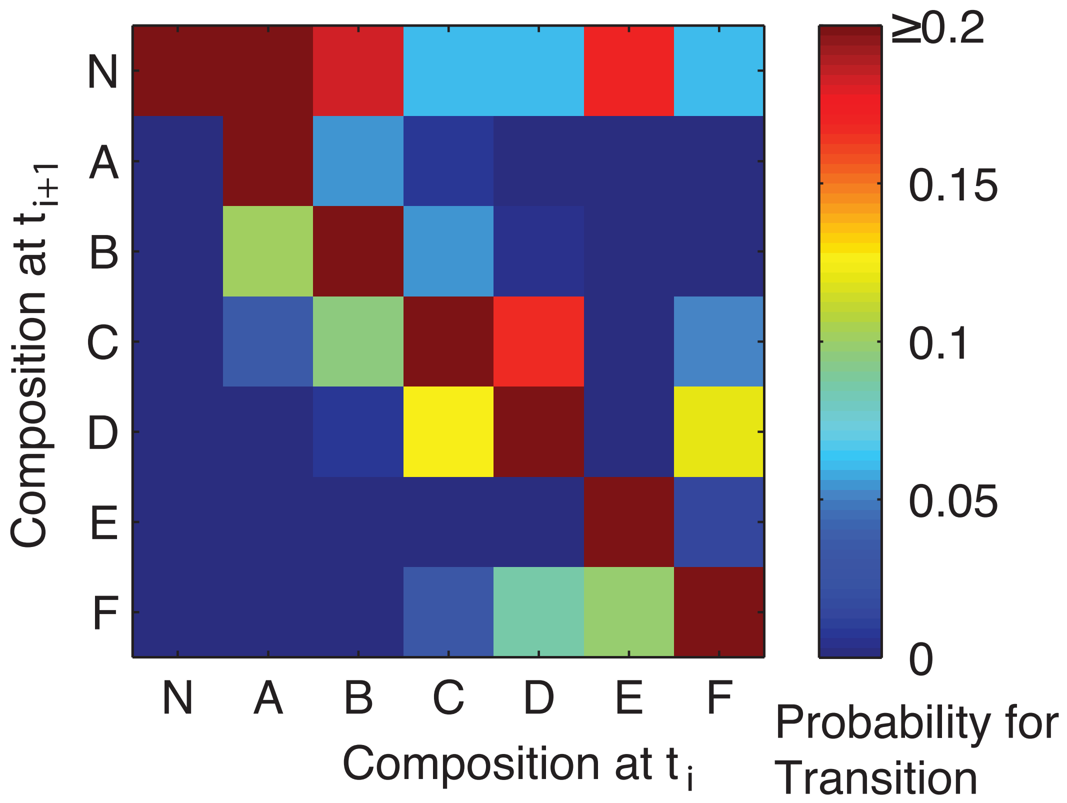


Figure 3.16: Average probabilities of all compositional transitions within focal adhesions divulge pathways of compositional transitions - The heat map represent the mean probability of all possible transition between the compositions (A to H) observed in focal adhesion of 10 spread REF52 cells. N is an empty composition i.e. region at which the fluorescence level of all imaged proteins is below their corresponding background thresholds. Transition of any of the composition to N means disappearance of a composition and vice versa.

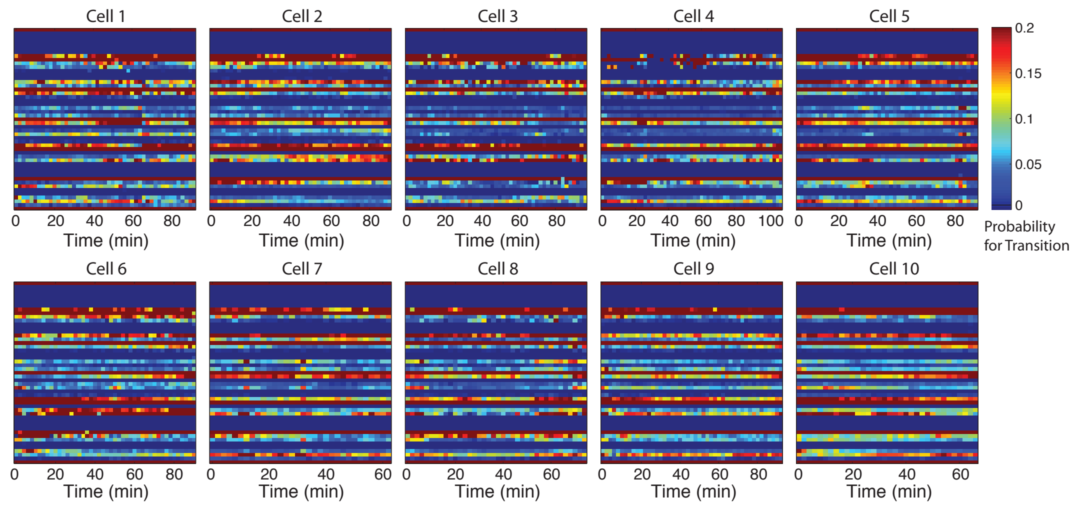


Figure 3.17: The occurrence probabilities of compositional transitions within focal adhesion is generally uniform overtime in unperturbed fibroblasts. - The heat map represents the probabilities for transition between all possible compositions, estimated for every imaging time frame of compositional movies. The probabilities mostly remain unaltered along time in all the spread fibroblast cells.

position in response the force inhibition-recovery cycles, offer functional insight into each composition. Additionally, temporal profile of abundance of all possible compositional transitions, revealed groups with similar pattern. The difference in temporal profile of each group concord with the disassembly-recovery state of the cell (Fig. 3.19).

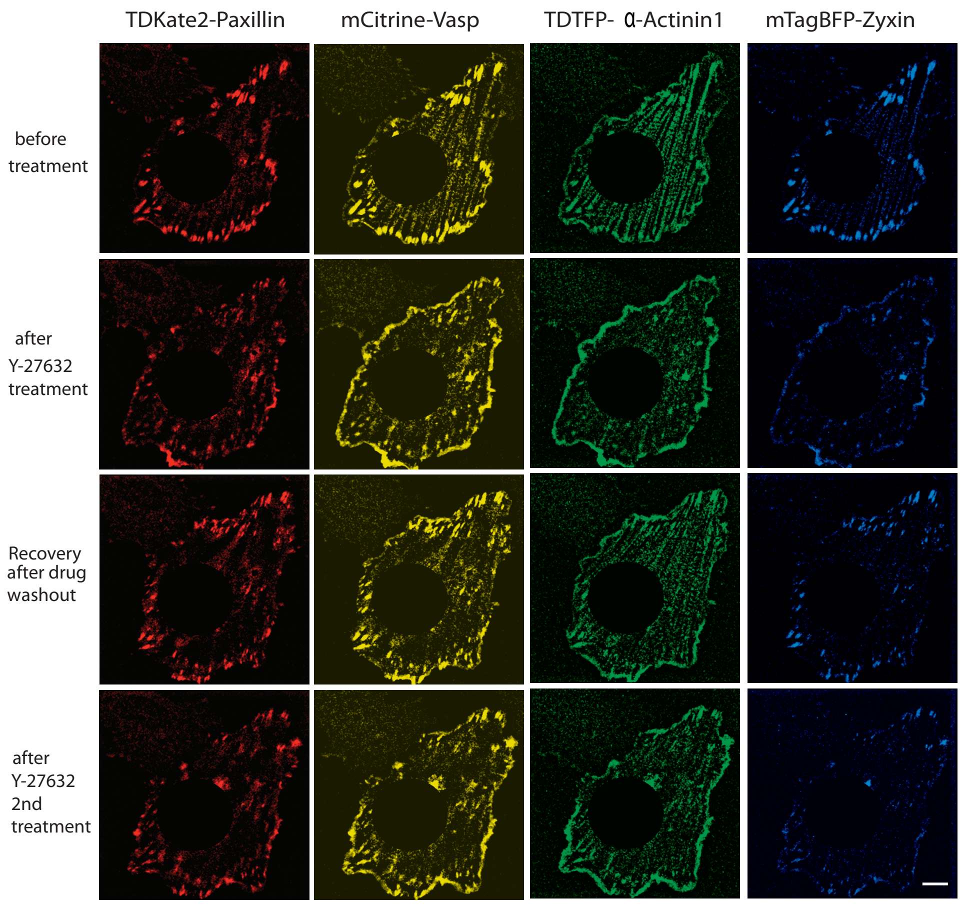


Figure 3.18: Multicolor imaging of cell-matrix adhesion sites responding to actomyosin inhibition cycles - REF52 cells expressing dmKate2-Paxillin, mCitrine-VASP, dmTFP-α-actinin and mTagBFP Zyxin, imaged every 2nd minute, were treated with 50 μM Y-27362 to inhibit actomyosin contractility at 22nd minute; drug is washed out at 66th minute allowing the cell to recover until second similar treatment at 119th minute of imaging. Scale bar: 20 μm.

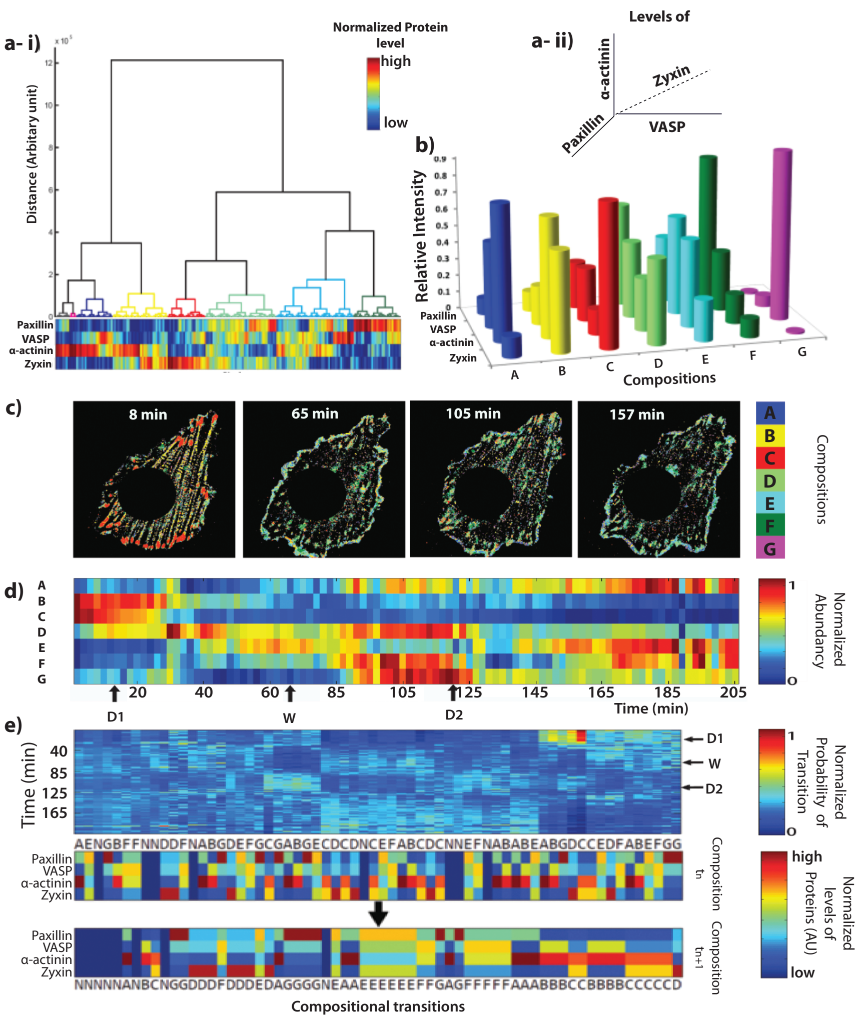


Figure 3.19: Compositional analysis of actomyosin inhibition reveals mechanosensitive properties of adhesion sites - Pixels harvested from the time lapse images of REF52 cells subjected to actomyosin inhibition as indicated in Fig. 3.18 are clustered by hierarchical clustering in the compositonal space (a-ii), in which each dimension corresponds to the level of one of the labeled components. By sampling through the dendrogram (a-i), significant nodes are selected to identify compositional signatures (b). Spatial mapping of compositional signatures visualized the spatial organization as well as context specific dynamic response of network-states in adhesion sites (c). Heatmap representing the normalized abundance of composition revealed the perturbation and recovery specific signatures (d). Clustering the normalized temporal profile of abundance of various compositional transition helps to identify groups with similar similar dynamics (e). D1, W and D2 indicate first treatment with Y-27632, the washout and the second treatment with Y-27632.

Chapter 4

Reverse engineering protein networks in adhesion sites

In the previous chapter, the spatially regulated compositional diversity and compositional dynamics of adhesion sties in spread fibroblast cells indicate functionally distinct protein network states. These networks of proteins essentially control the functional behavior of focal adhesion in the cells. To understand this control mechanism, it is crucial to quantify the causality between the proteins to reverse engineer protein networks in adhesion sites. Causality or cause-effect relationship between proteins of the network can be quantified by tracking the changes in the level of proteins in the adhesion sites with respect to each other. Causal protein network can be inferred through two methods viz. (1) by quantifying temporal co-changes in network components (i.e. time series of protein levels in focal adhesion) in steady state or (2) by quantifying co-changes in protein levels on perturbation of one of the network component. The quantification of protein levels in adhesion sites and elucidation of causal protein networks by both the methods are described in the following sections.

4.1 Development of computational tools for segmentation and tracking of adhesion sites

Inferring causal relationship between the proteins in adhesion sites requires quantification of dynamic changes in the level of proteins in individual adhesion sites. In order to quantify the level and turnover of proteins, individual adhesion sites should be segmented from the multicolor time lapse images and tracked along time. To achieve this, an object segmentation tool and an object tracking tool are developed as described below.

4.1.1 Object segmentation tool

The object segmentation tools is a Matlab based GUI developed as a part of the thesis for sophisticated segmentation of cell-matrix adhesion sites. The tool uses the high-pass (HP) filtered images as an input, to recognize the focal adhesions. Segmentation is based on the water algorithm (Zamir et al., 1999). Additionally, the software allows user to draw polygon to specifically include or seclude adhesion sites of interest or disinterest, respectively. The tool also allows batch processing of multiple images with similar parameter and region of interests (ROI). User instruction and complete source code of the tool is available in the appendix (Sec. 8.3.1).

4.1.2 Object tracking software

In order to track the segmented adhesion sites, object tracking tool, also based on Matlab GUI is developed. This tool allows user to perform both fully as well as semi-automated tracking of adhesion sites. Firstly, adhesion sites segmented from different channels of multicolor images (output of object segmentation tool) are merged to generate single common mask. The segmented adhesion sites in the common mask is then tracked along the time domain of the movie. Matching of adhesion sites between the consecutive time frames is achieved by the following criteria

1. Overlapping adhesion sites are straightly matched.

2. Adhesion sites with multiple overlap, are matched based on highest percentage overlap.
3. When two focal adhesions in a time frame overlap totally with single focal adhesions in next frame; the former are merged and matched with latter.
4. In the absence of any overlap a focal adhesion is matched to another found proximal within the specified radius in the next frame.
5. Focal adhesion which failed to match with another in the next time by all the above criteria are considered discontinued from further tracking.
6. Newly appearing focal adhesions are appended to the list of focal adhesions and tracked further.

User instruction and complete source code of the tool is available in the appendix (Sec. 8.3.2).

4.2 Reconstructing the structure of protein networks in focal adhesions based on protein dynamics

Compositional analysis of focal adhesions in fibroblast cells revealed spatially heterogeneous protein network states. To study the regulation and function of the diverse networks, it is required to elucidate how the network components influence each other in individual adhesion site. Towards this end, about 200 focal adhesion are segmented from the multicolor time lapse images of five spread fibroblast cells (cell ID: 1-5) (Fig. 2.3) and tracked over time using object segmentation and object tracking software. The levels of four labeled proteins (Zyxin, α-actinin, vinculin and paxillin) for each focal adhesion is quantified as a function of time (i.e. time series) (Fig. 4.1). The temporal dynamics of four proteins (time series) vary depending on the dynamic state (i.e. assembly, disassembly or steady-state) of the adhesion sites. Hence, the causal protein network inferred from the time series data could offer insight into the mechanisms behind these three

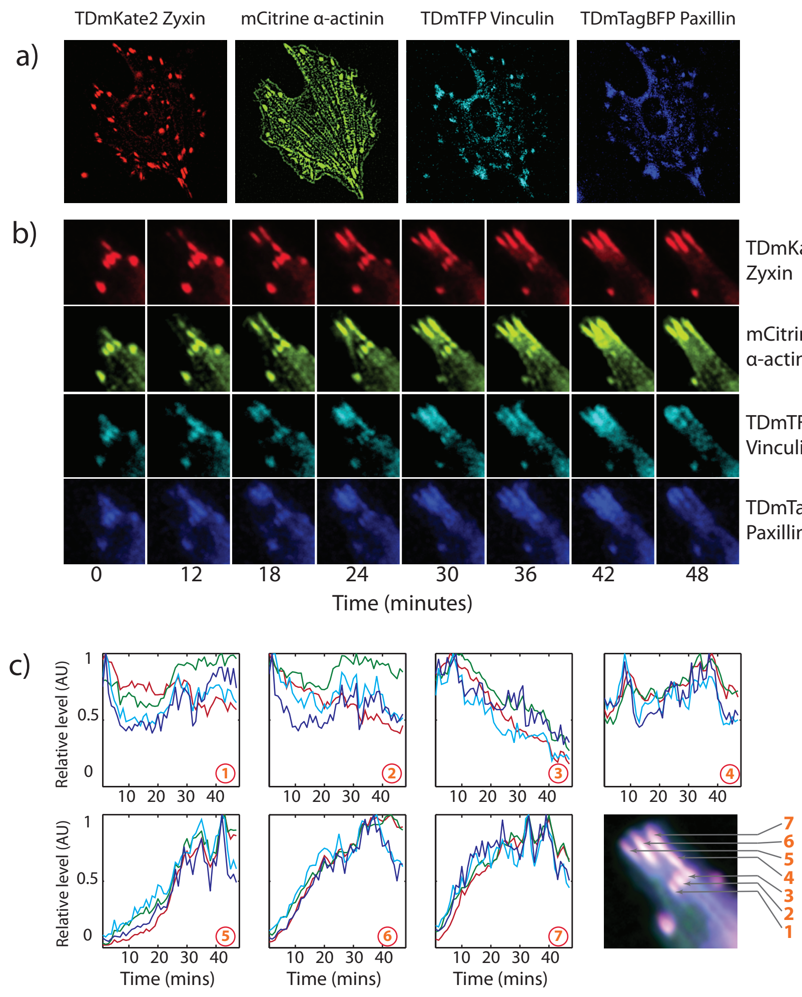


Figure 4.1: Monitoring changes in protein levels in adhesion sites of unperturbed fibroblasts - (a)4-color images of spread REF52 cell (ID :4) expressing TDmKate-zyxin (red), mCitrine-αactinin (green), TDmTFP-vinculin (cyan) and TDmTagBFP-paxillin (blue). Live cells are imaged every 2 minutes for about 90 minutes to monitor the dynamics of the chosen proteins during steady state assembly and disassembly of cell-matrix adhesion sites. (b) A portion of the imaged cell is magnified to display the dynamics of the focal adhesion proteins at the time points (0, 24, 36, 48, 60, 72, 84, 96 minutes) sampled from the multicolor live cell movie. (c) Focal adhesions are segmented and tracked over the multicolor movie to extract the time series intensity profile of all four proteins. Bottom right panel is the time-channels superimposed image.

dynamic states. To uncover how network structure is controlled by intracellular local conditions that generate heterogeneous adhesion sites and dynamic states, it is necessary to infer causal protein network for individual adhesion sites by an appropriate approach. To find a suitable network reconstruction approach for the extracted time series data, Dr. Koseska, screened through various approaches described in (Hempel et al., 2011b) on the basis on the length of time series data and the principle and prediction accuracy of the algorithm and identified that a combination of partial Pearson correlation and IOTA (Hempel et al., 2011a) to be the most appropriate. Therefore, causal connections are extracted from the time series of individual adhesion sites through the conjugated theoretical approaches as described in the following sections.

4.2.1 Inferring directed relationship between proteins

Reconstruction of protein networks requires identification of causal links between the proteins and it can be achieved in two steps as described. In the first step, the directed links are identified from the time series data of individual adhesion sites, by inner composition alignment (IOTA), a permutation-based asymmetric association measure algorithm developed by Dr. Koseska. This algorithm identifies the directionality of both unidirectional and bidirectional direct network connections based on the co-dynamics of proteins. In the second step, the probabilities of direct connection between all the proteins is estimated by partial Pearson correlation between time series data in individual adhesion sites. Here direct connection refers to links between proteins that are not mediated through other measured network component. Moreover, as partial Pearson correlation is a measure of directionless connections, it is coupled with IOTA. Therefore, a causal link detected by IOTA is considered to be significant when the probability of direct connection is higher. Using two different threshold values of direct connection probabilities, significant connections between the network components (zyxin, α-actinin, paxillin and vinculin) are estimated (Fig. 4.2). The abundance of identified connection in all analyzed focal adhesion revealed most and the least prevalent significant causal links observed in focal adhesions (Fig. 4.2).

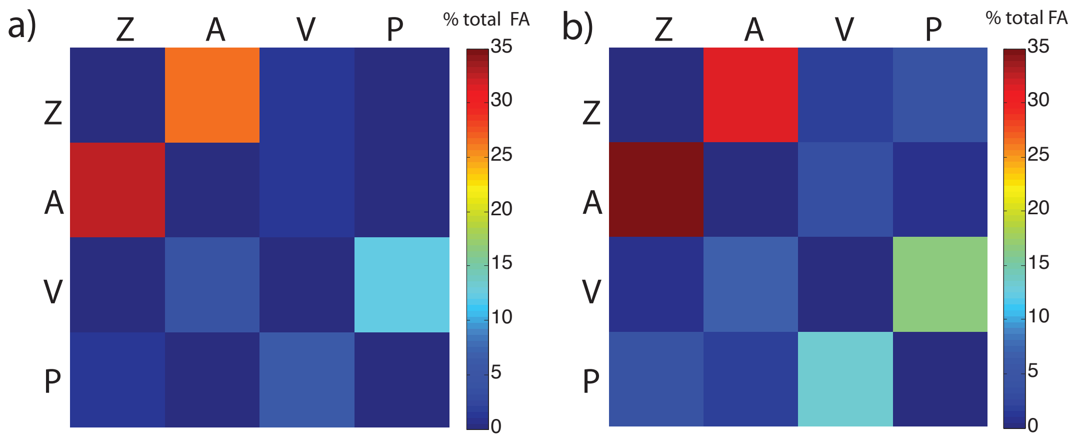


Figure 4.2: Resolving potential causal relations between proteins in focal adhesions from time series data - Intensity profiles of proteins zyxin (Z), α-actinin(A), vinculin(V) and paxillin (P) along time are used to calculate significant causal connections for each tracked adhesion site. The heatmaps represent the abundance of significant causal connections of proteins observed in the analyzed (n = 200) focal adhesions. Significant connections are those which are detected by both IOTA and partial Pearson correlation in the time series data of intensity profiles of protein levels. The threshold P-value used for partial Pearson correlation are $P > 0.75$ (a) and $P > 0.66$ (b).

4.2.2 Resolving network connectivity in assembling, disassembling and steady-state focal adhesion

Focal adhesions have three growth phases i.e. assembly, steady-state and disassembly phase. Focal adhesions are generally assembled in the leading edge of the migrating cell and disassembled from the rear edge. A cell may contain several adhesion sites belonging to different categories of the growth phase. Therefore, under a given timeframe focal adhesions in a cell can be broadly classified based on their dynamics into three categories of dynamic state. Resolving the causal links between proteins in these three distinct dynamic states of focal adhesion is necessary to understand the underlying regulatory mechanism. All the focal adhesions are tracked and therefore categorized based on their protein dynamics into three aforementioned groups by affinity propagation (Frey and Dueck, 2007) clustering of time series data (Fig. 4.3). The abundance of identified significant connection in each category of focal adhesion revealed common as well as dynamic state-specific causal links (Fig. 4.4).

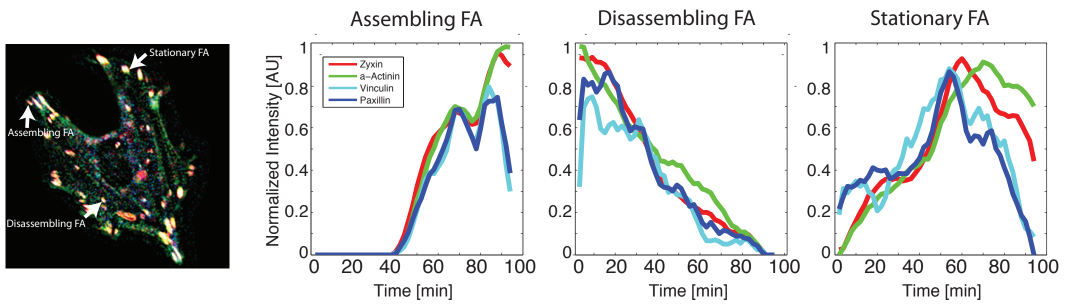


Figure 4.3: Classification of adhesion site dynamics by clustering time series - Time series intensity profiles are clustered by affinity propagation algorithm to classify segmented adhesion sites into three categories: assembling, disassembling and stationary adhesion sites. For each of these categories, representative (normalized) intensity traces are shown.

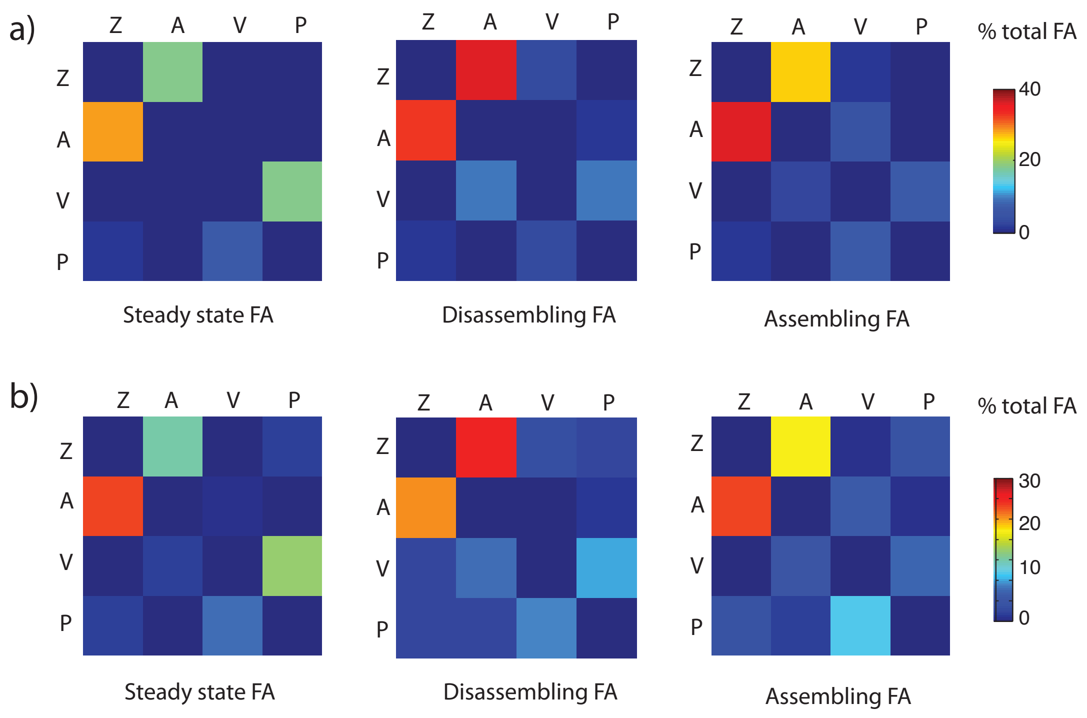


Figure 4.4: Inferring state-specific potential causal relations between proteins in focal adhesions from time series data - Intensity profiles of proteins zyxin (Z), α-actinin(A), vinculin(V) and paxillin (P) along time are used to calculate significant causal connections for each tracked adhesion site. The heatmaps represents the abundance of significant causal connections of proteins observed in steady-state, disassembling and assembling focal adhesions. Significant connections are those which are detected by both IOTA and partial Pearson correlation. The threshold P-value used for partial Pearson correlation are $P > 0.75$ (a) and $P > 0.66$ (b).

4.2.3 Resolving protein network topology in individual adhesion sites

Spatially heterogeneous adhesion sites are generated by heterogeneous protein network with distinct topology. It is essential to resolve the topology of the causal protein networks in individual adhesion sites to acquire deeper insight into the structure-function relationship. Topology of a causal protein network is an ensemble of all causal links between the network components. In order to construct the network topology the most significant and prevalent causal links are chosen (Fig. 4.5) and all possible sub-networks are generated. Abundance of all sub-networks observed in tracked focal adhesion are quantified (Fig. 4.5 b). Amongst all sub-networks, the most prevalently observed (top 5) protein networks are spatially mapped on the images of adhesion sites. These color mapped images directly visualize the spatial distribution of distinct protein causal networks observed in focal adhesion (Fig. 4.5 c).

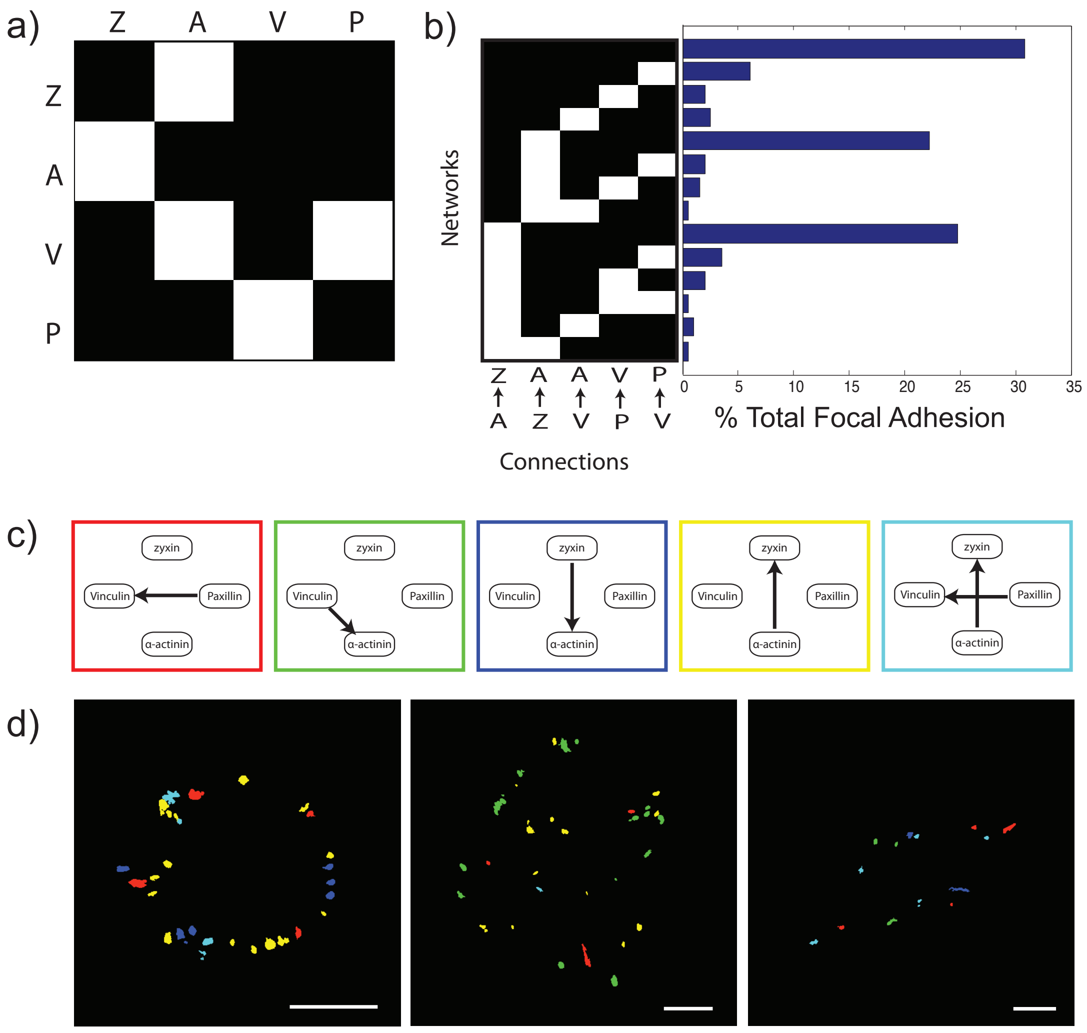


Figure 4.5: Spatially resolving potential causal relations between proteins in adhesion sites - (a) Protein causal network is built based on the most abundant connections (top 5) observed in focal adhesion. The binary plot (b) represents all sub-networks (built from the 5 connections) observed in focal adhesions. The white block denotes presence of one the 5 connections. The bar plot shows the relative abundance of the subnetworks in the analyzed focal adhesions. Top 5 abundant networks (c) are color coded and spatially mapped in the images of cells (d).

4.3 Resolving causal topologies of protein networks in adhesion sites by acute perturbations

In the previous section, the causal links and network topology of adhesion sites are elucidated from co-dynamics of the proteins participating in the network. This approach revealed potential network connections. In order to reconstruct actual protein causal networks that offers mechanistic insight into adhesion site dynamics, it is necessary to perform a perturbation experiment. A protein network component can be acutely perturbed and the changes in the levels of all components can be monitored in adhesion sites. The strength of influence of the perturbed component over the unperturbed ones divulges the actual causal connection. Therefore, the actual causality between the perturbed and unperturbed proteins can be derived from perturbation measurement i.e. quantifying co-changes in the level of latter with respect to the former. By systematically perturbing each network component and inferring the its actual causal links with other components, the entire protein network can be constructed (Fig. 4.6) (Zamir and Bastiaens, 2008).

4.3.1 Spatially resolved causal relations between two proteins

4.3.1.1 Acute perturbation system

A reverse engineering approach based on acute perturbation system is developed in collaboration with Dr. Stricker to resolve causal connections between proteins in adhesion sites. Taking advantage of advent in chemical biology era, a chemically inducible acute perturbation system is developed by Dr. Stricker to acutely deplete an adhesion site protein (or 'bait protein') from the adhesion sites and monitor its effect on other components (or 'prey protein') in the system. The depletion is achieved by recruiting the bait protein to the mitochondrial region. This system is based on inducible FKBP-FRB dimerization system (Banaszynski

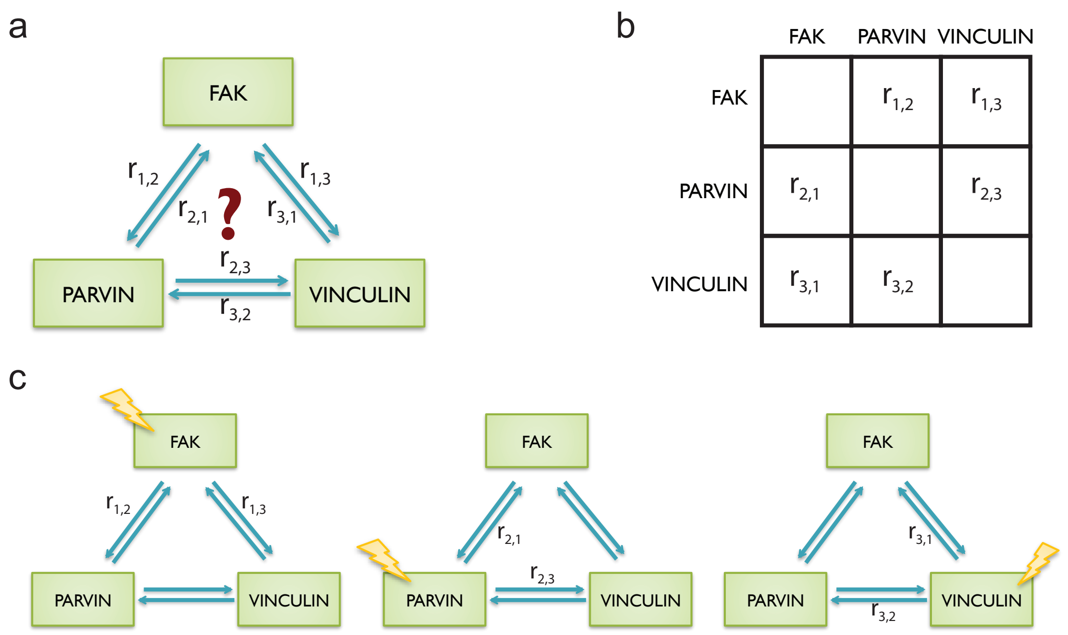


Figure 4.6: Resolving FAK-parvin-vinculin causal protein network topologies by systematic perturbations - To resolve a protein causal network firstly, the network components (FAK, parvin and vinculin) and the relationship $r_{i,j}$ is defined (a). The causal relationship $r_{i,j}$ for the network can be built by filling the causality matrix (b) by perturbation experiments. These experiments require perturbation of one of the network component and monitoring its effect on other network components (c). For e.g. by perturbing FAK, the causal effect of FAK on parvin ($r_{1,2}$) can be estimated as log (Δparvin /ΔFAK) where Δparvin and ΔFAK are ratio between the levels of respective proteins after and before perturbation (e.g. Δparvin = parvin level after perturbation / parvin level before perturbation).

et al., 2005), in which the interaction between the FKBP and FRB is mediated by external chemical component like Rapamycin or Rapalog (Fig. 4.7 A) Appropriate quantification of perturbation of bait and its effect on prey proteins unveils causal connection from bait to prey. This approach can be used to validate the previously identified network relationship. It can also be used to elucidate the causal relationship between any proteins of interest. Dr. Stricker developed the reagents of acute perturbation system and performed most of the wet experiments. In this part of the thesis, approaches developed for the quantitative analysis of the perturbation data to uncover causal relationship between proteins are described.

4.3.1.2 Resolving causal relation between vinculin and FAK

In order to understand the causal relationship between vinculin and FAK, each of the protein is perturbed systematically and its effect over the other protein is monitored in every individual adhesion sites. A FKPB domain is fused to bait protein and the FRB-domain is fused to MOA, a mitochondrial localization signal. On addition of rapamycin, the FBPB domain interacts with FRB, accumulating the bait protein in the mitochondrial region. This depletes the bait protein in the cytoplasmic pool, which in turns depletes the protein level in adhesion site to achieve the equilibrium of bait protein between adhesion site and cytoplasm. To detect the causal effect of vinculin (bait protein) over FAK (prey protein), the former is perturbed and latter as co-monitored. Cells expressing, FKBP-Vinculin (bait), FAK (prey) and FRB-MOA (mitochondrial recruiter) with three different fluorescent tag are imaged continuously before and after rapamysin addition (Fig. 4.7 B). Upon, rapamycin addition, the recruitment of Vinculin to mitochondria (Fig. 4.7 C,E)and thereby decrease in FAK present in FA is observed (Fig. 4.7 D). These experiments performed by Dr. Stricker, have to be quantitatively analyzed to precisely extract the causal effect. The images are processed by tools image processing tools to prepare images for quantification of adhesion site dynamics. The object segmentation and object tracking tools are originally developed to quantify the adhesion sites for this project are previously described. The levels of the vinculin (bait) and FAK (prey) in individual adhesion sites are

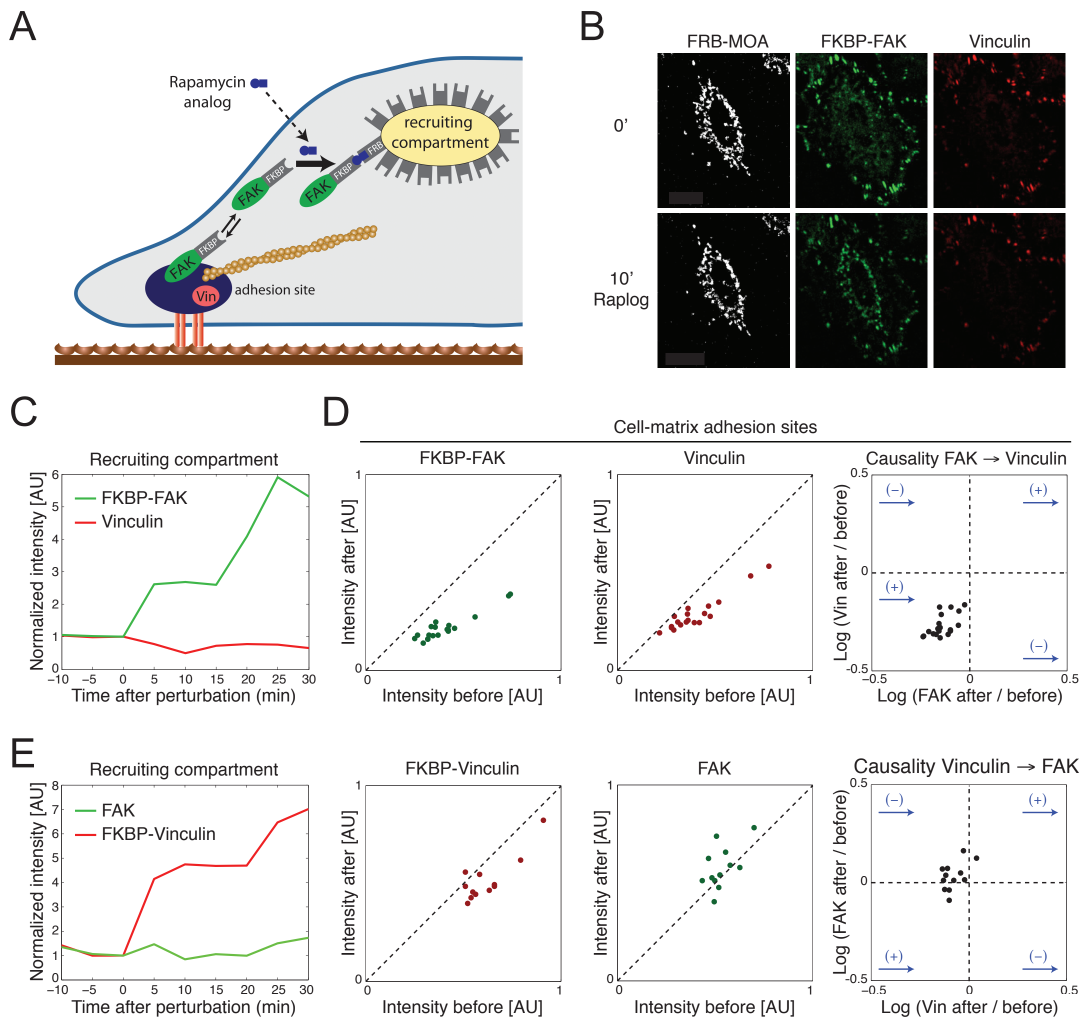


Figure 4.7: Resolving FAK-Vinculin causal relationships by acute perturbations - (A)A chemically inducible acute perturbation system based on binding FKBP-FRB dimerization is used to acutely recruit FAK (the bait protein or perturbed protein) to the mitochondria. Simultaneously its effect on vinculin (vin) (the prey protein or observed protein) is simultaneously monitored. (B) On addition of rapalog (Ariad Pharmaceuticals, Inc. and Clontech Laboratories, Inc.), the FKBP-FAK is recruited to mitochondrial region with FRB-MOA marker. (C) The mitochondrial region is masked and the increase in fluorescent level due to recruitment is quantified. (D) Relative change in the protein levels in individual adhesion sites are quantified by computational tools and the causal effect of FAK on Vinculin, which is the relative change in Vinculin with respect to FAK is estimated. (E) Similar quantifications are also performed by recruiting FKBP-Vinculin to mitochondria and monitoring its effect on vinculin.

quantified before and after the perturbation from the images (Fig. 4.7 D) and the causal relation $r_{i,j}$ is measured as logarithmic fractional change in FAK (prey) with respect to fractional change in vinculin (bait) for every adhesion site (Fig. 4.7 D). The corollary experiment with FAK-FKBP as bait and vinculin as prey performed by Dr. Stricker, is quantitatively analyzed to detect the causal effect of FAK over vinculin (Fig. 4.7 C,D,E). Unlike the theoretical measure of causality based on dynamics, the directionality is clearly defined in this approach due to the advantage of acute perturbation of adhesion sites.

4.3.2 Resolving causal network of Parvin, FAK, Vinculin by reversible acute perturbations

In order to resolve causal network structure of FAK-vinculin-parvin, each of the network components are perturbed one at a time and their effect on other components is observed (Fig. 4.6). This perturbation experiment consists of one bait and two prey proteins. To resolve the topology of causal network during assembly and disassembly of adhesion sites, yet another dimerization system, functionally similar to the previously described one is adapted for acute perturbation. This system was developed in Dr. Yaowen Wu's group. This system consist of A and D domain similar to FKBP and FRB respectively. The advantage of this system is the reversibility of the protein dimerization to recover the perturbation. The dimerization between A and D can induced by a ligand and released by a competitor molecule. Parvin fused with A domain (bait protein) is recruited to the mitochondria by the D-MOA on the addition of ligand (i.e. depletion) and later released on addition of competitor molecule (i.e. repletion).

REF52 cells expressing A-Parvin (bait), D-MOA (mitochondrial recruiter), FAK (prey 1) and vinculin (prey 2) tagged with four spectrally separated fluorophores, are imaged to monitor the effect of parvin on FAK and Vinculin (Fig. 4.7). Individual focal adhesion are segmented and tracked by the computational tools to quantify the level of the bait and the prey proteins. The causal effect of the bait protein (parvin) over the prey proteins (FAK and vinculin) is estimated from the logarithmic change prey level with respect to prey level in individual

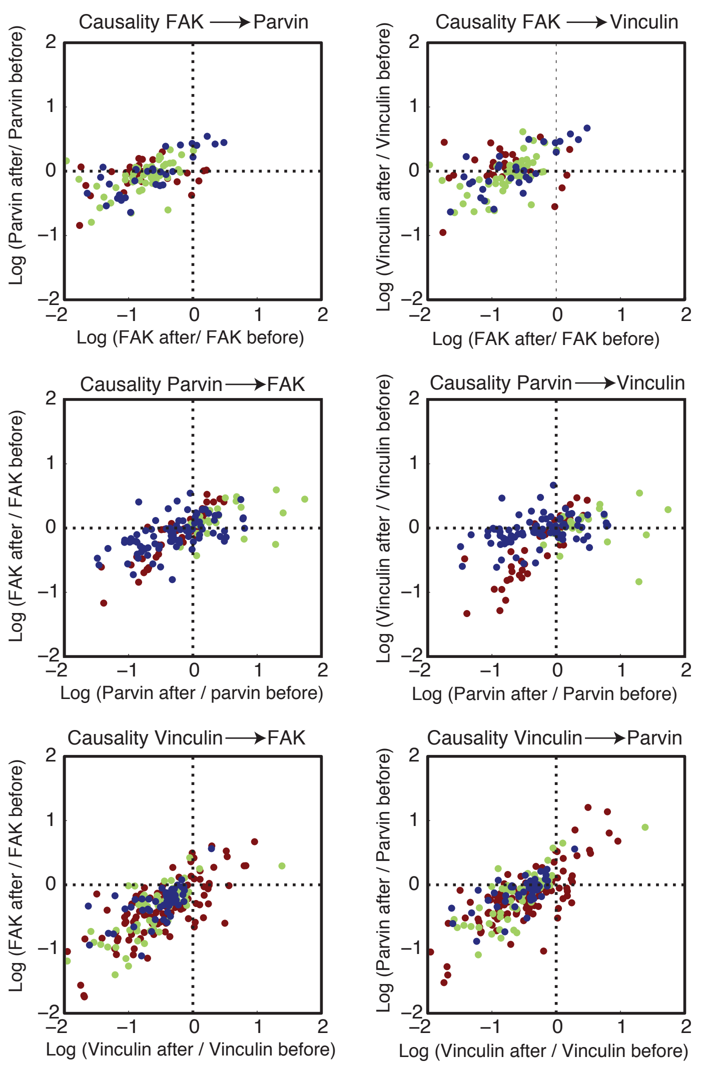


Figure 4.8: Resolving FAK-parvin-vinculin causal connections by depletion perturbations - A chemically inducible A-D dimerization system is used to acutely perturb one of the network components and its effect on other components is monitored. The network component (e.g. FAK) is depleted from the focal adhesion via mitochondrial recruitment after addition of ligand and its effect on other network components (e.g. Parvin and Vinculin) in the focal adhesion are monitored. Adhesion sites from the perturbation experimental images are segmented, tracked and the protein levels of all the network components are quantified to elucidate causal effect at individual focal adhesion. 3 rows of plots represent 3 perturbation experiments, one for each network component. For each perturbation experiment, data from 3 cells (colored dots) are displayed. Each data point is estimated from quantification of protein levels in single focal adhesion.

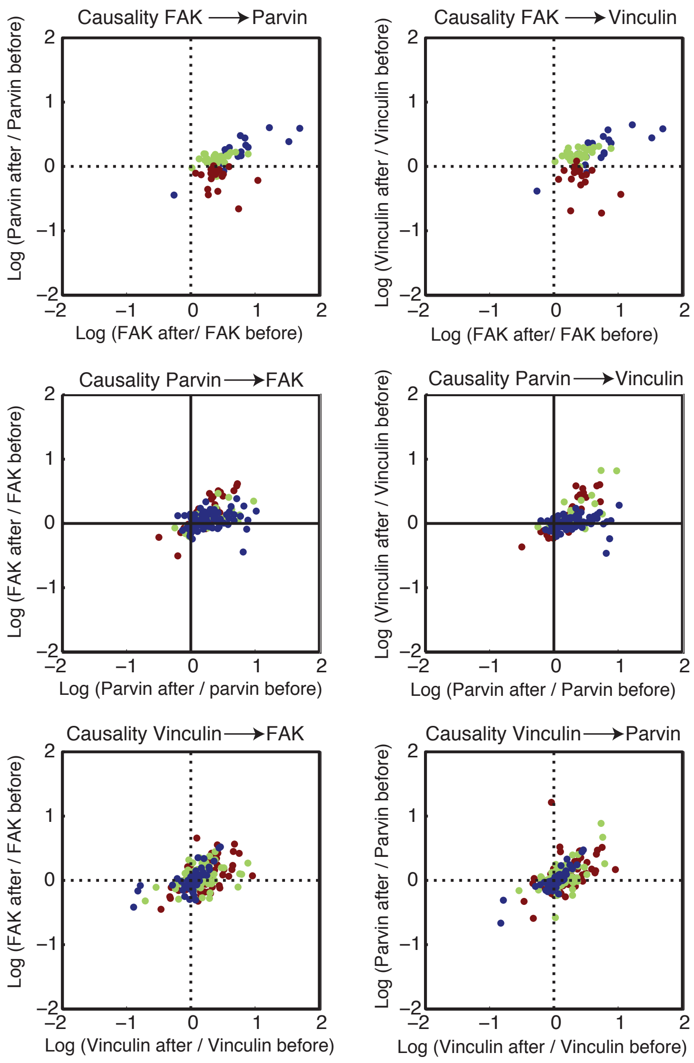
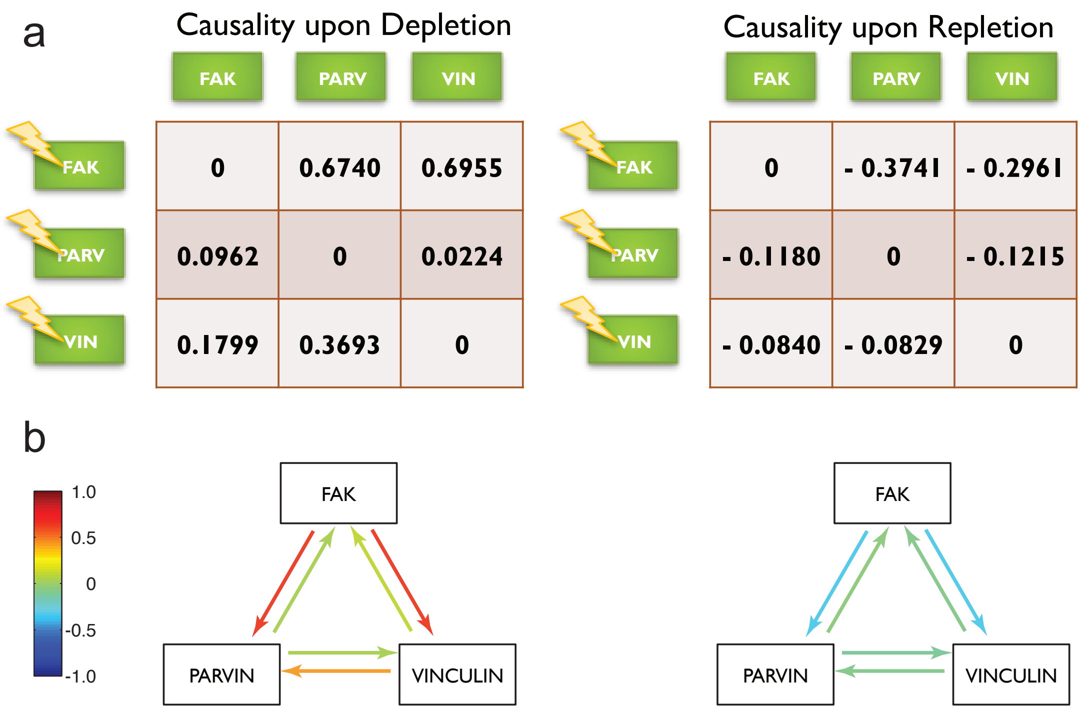


Figure 4.9: Resolving FAK-parvin-vinculin causal connections by repletion perturbations - A chemically inducible A-D dimerization system is used to acutely perturb one of the network components and its effect on other components is monitored. The depleted network component (e.g. FAK) is released from the mitochondria after addition of competitor and its effect on other network components (e.g. Parvin and Vinculin) in the focal adhesion are monitored. Adhesion sites from the perturbation experimental images are segmented, tracked and the protein levels of all the network components are quantified to elucidate causal effect at individual focal adhesion. 3 rows of plots represent 3 perturbation experiments, one for each network component. For each perturbation experiment, data from 3 cells (colored dots) are displayed. Each data point is estimated from quantification of protein levels in single focal adhesion.

adhesion sites (Fig. 4.8, 4.9). Adhesion sites disassemble during depletion perturbation and therefore the causal link inferred reflects the causal effect during disassembly (Fig. 4.8). During the repletion, the adhesion sites recover, and therefore the inferred causal link reflects the causal effect during assembly (Fig. 4.9). The causal link from parvin to FAK $r_{2,1}$ and vinculin $r_{2,3}$ is measured for individual adhesion sites during depletion and repletion perturbation. The variance of $r_{i,j}$ in adhesion sites revealed heterogeneity in the protein causal network of different adhesion sites. The global causal relation in the causality matrix is measured as median $r_{i,j}$ for all the adhesion sites. Only experiments pertaining to parvin perturbation are performed as a part of the thesis. Further Dr. Stricker performed experiments with FAK, Vinculin as bait protein in addition to few more experiments on parvin perturbation. Systematic perturbation of all three proteins, are quantified to estimate the causal relation $r_{i,j}$ between all protein pairs. Thereby the causality matrix is filled (Fig. 4.10 a) to reconstruct the causal protein network (Fig. 4.10 b). As depletion causes disassembly and repletion favors assembly, the difference in causal network during repletion and depletion convey difference in logic of adhesion site assembly and disassembly mechanisms, respectively.



Causality upon Depletion

	FAK	PARV	VIN
FAK	0	0.6740	0.6955
PARV	0.0962	0	0.0224
VIN	0.1799	0.3693	0

Causality upon Repletion

	FAK	PARV	VIN
FAK	0	- 0.3741	- 0.2961
PARV	- 0.1180	0	- 0.1215
VIN	- 0.0840	- 0.0829	0

Figure 4.10: Resolved FAK-parvin-vinculin causal network topologies in adhesion sites suggest asymmetric assembly and disassembly mechanisms - (a) Causality matrix is estimated for both depletion as well as repletion perturbation experiment. The difference in the causal network (b) suggests that the causal effect between the network components FAK, parvin and vinculin are distinct during disassembly and assembly of adhesion sites.

Chapter 5

Unmixing intracellular mixtures of protein-network structures

In the previous chapter, it is shown that the connectivity of Parvin (α -Parvin), FAK and Vinculin network in focal adhesions (FA) is resolved using reverse engineering approach. The fundamental limitation in the approach is number of components that can be co-monitored at the same time. As adhesion sites are complex multi-component structures, the connectivity between the three chosen proteins can be mediated by one or more components that are unmeasured. The spatial heterogeneity in the composition of the adhesion sites suggests the possibility of heterogeneous connectivity between any two components at different subcellular as well as sub-adhesion site location. Inferring a single causal network for data with heterogeneous connectivity can result in incongruous average relationship. This fundamental issue hold also true when relationship between proteins in heterogeneous cell population is inferred. The causal relation and correlation between proteins can be quantitative different for different subpopulation due to the dissimilarity in their intrinsic molecular physiology. For instance, any attempt to resolve functional relationship between cell cycle regulatory proteins in an unsynchronized population would yield meaningless connection. Similarly, any effort to infer connection of in mixed population of tumor and normal cells, would also fail. The connections between measured proteins in cancer cells can either stronger or weaker than the normal cells, due to genetic or epigenetic modification of the in the measured components or unmeasured components that mediate the crosstalk between the measured components (Fig. 5.1). Hence, in order to properly infer the causal relationship between proteins, it is essential to unmix the subpopulation of adhesion sites or cells; so that, the causal connectivity can be profoundly resolved for each of the subpopulation.

5.1 Unmixing-via-non parametric Bayesian network (UNPBN)

Toward this end, in collaboration with Mr. Jakob Wieczorek and Prof. Katja Ickstadt, a statistical method is developed. This method recursively couples an iterative unmixing process with non-parametric Bayesian network model analysis (Ickstadt et al., 2011) of subpopulation and hence termed as unmixing-via-NPBN (UNPBN). The UNPBN algorithm can not only detects the number of subpopulation, unmix their corresponding observation, but also can also determine the network topology of each subpopulation. The UNPBN algorithm developed by Mr. Wieczorek and Prof. Ickstadt takes advantage of multi-dimensionally of high-dimensional measurements usually acquired in techniques like flow-cytometry (Sachs et al., 2005; Krutzik et al., 2008; Zamir et al., 2005) or toponome imaging (Friedenberger et al., 2007; Schubert et al., 2006). UNPBN exploits the fact that the stochastic FA-to-FA or cell-to-cell variability in protein localization or expression level gives rise to high-order probability distribution in a multidimensional space with dimensions corresponding to the level of proteins measured. A number of random Gaussian Bayesian network models are generated to fit the data and the better fitting model is iteratively adapted and coupled with the unmixing process to reach point convergence, where the data can unmixed with a best fitting model for each data subset. The performance and the scope of the developed method are demonstrated using a simulated data as described in the following sections.

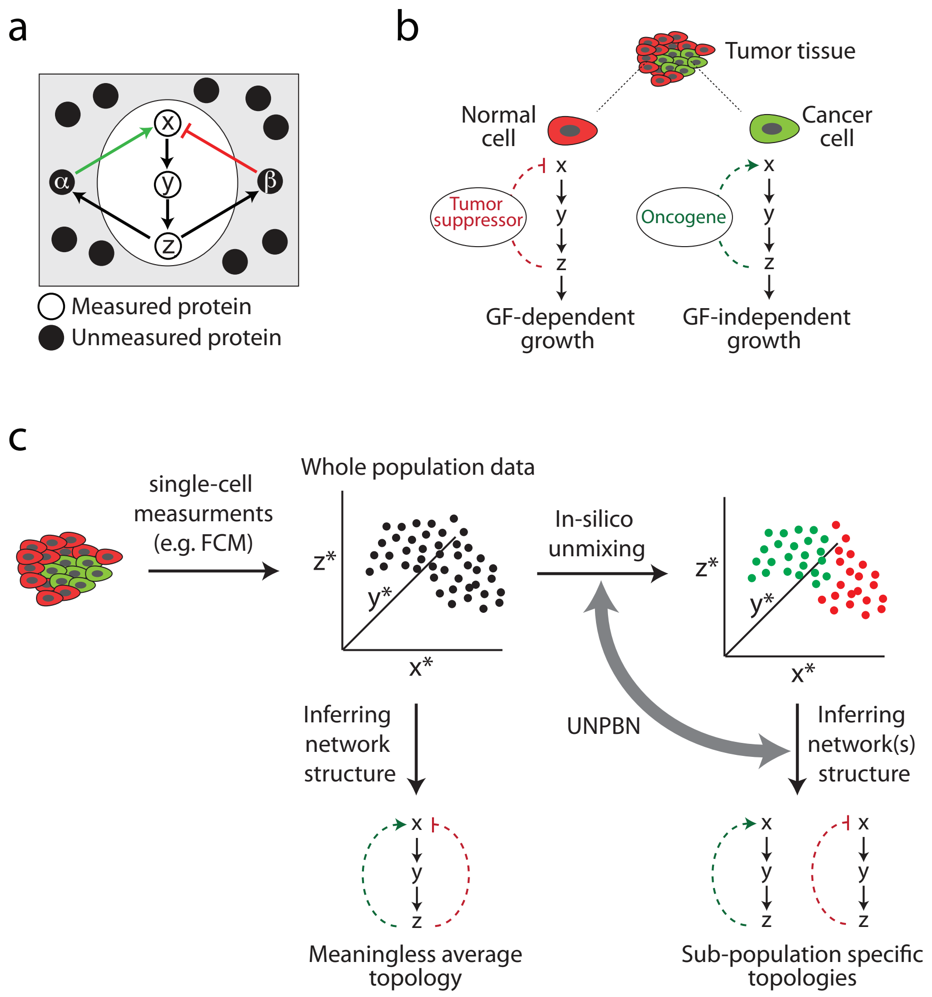


Figure 5.1: Unmixing signaling circuits with distinct topologies - The network reconstruction approaches are fundamentally limited by the number of experimentally measured components. In a simple biochemical system (a), where only a part of the system (x,y,z) is measured, the influence between the components (x,y) mediated through unmeasured components (α and β), can be different depending of the state and level of the unmeasured component. (b) For example in a cancerous tissue, the network components (x,y) can exhibit different causal relation in cancer and normal cells depending on the unmeasured expression levels of oncogene and tumor suppressor component. (c) In such conditions, inferring a single network topology from the cells en masse can yield an average meaningless and biologically implausible topology. UNPBN, a novel approach developed in collaboration with Prof. Ickstadt and Mr. Wieczorek, unmix subpopulations of cells from the mixture and infer true topology of networks in each subpopulation.

5.2 Simulation of heterogeneous cell population with distinct MAPK networks

In order to develop, assess and demonstrate the UNPBN method, it is required to simulate signaling network that exhibit distinct network topology at different conditions. Therefore, a MAPK cascade that has been show to exhibit two different topologies of Raf-Mek-Erk kinases network in PC12 cells depending on the stimulation of epidermal growth factor (EGF) or nerve growth factor (NGF), is chosen for simulation Sasagawa et al. (2005). A previously described model of extended MAPK network is used for simulation (Fig. 5.3) . This dynamic model can reproduce the experimentally demonstrated transient and stable temporal profile of Erk activation upon EGF and NGF stimulation respectively. It is essential to incorporate the cell-to-cell variability to mimic the real experimental data and generates multi-dimensional probability distribution that UNPBN essentially requires. Therefore in order to introduce intra-population variability (hereafter referred as noise), a stochastic Gaussian noise in the total protein levels is added, mimicking natural cell-to-cell variance in expression of proteins. Additionally, inter-population variability in the system is achieved by simulation of model with either EGF or NGF stimulation. The dynamic simulation at lower noise level resulted in two higher-order patterns in distribution of levels of active Raf, Mek and Erk, as determined by the network structure. With increasing noise, the overlap between the two distribution increased, making it harder to designate the (EGF or NGF simulated) network for the observations in the distribution (Fig. 5.4). The temporal difference in the levels of active Raf, Mek and Erk upon two different simulation is lost with increasing noise as shown in the figure (Fig. 5.3). Additionally, a modified model corresponding to mutant Mek with reduced activity, is simulated for both EGF and NGF simulation. This simulation offered additional two subpopulation, which allowed to demonstrate the ability of UNPBN to unmix, the mixture of two and four subpopulations (Fig. 5.6).

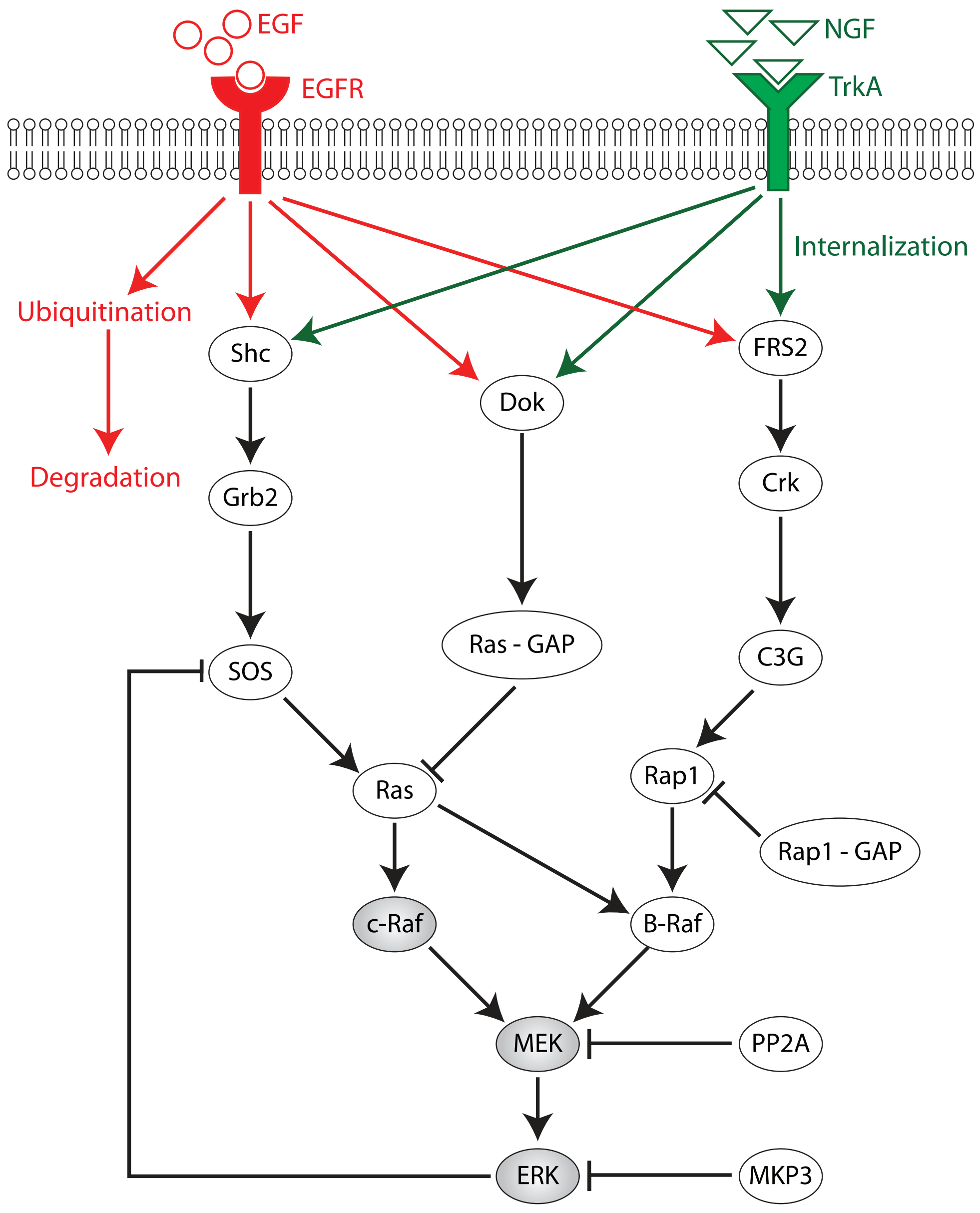


Figure 5.2: Cell signaling network sensing EGF and NGF in PC12 cells - In order to simulate single cell measurements of protein networks, a published MAPK signaling model (Sasagawa et al., 2005), that reproduced differential response of EGF and NGF stimulation in PC12 cells, is used. The elements (proteins, interactions and processes) that are specific to either EGF or NGF are colored red and green, respectively. The components that are consider as the measured components of the system (i.e. Raf, Mek and Erk) are colored grey. Cell-to-cell variability is specifically introduced into the total levels of these measured components in simulation.

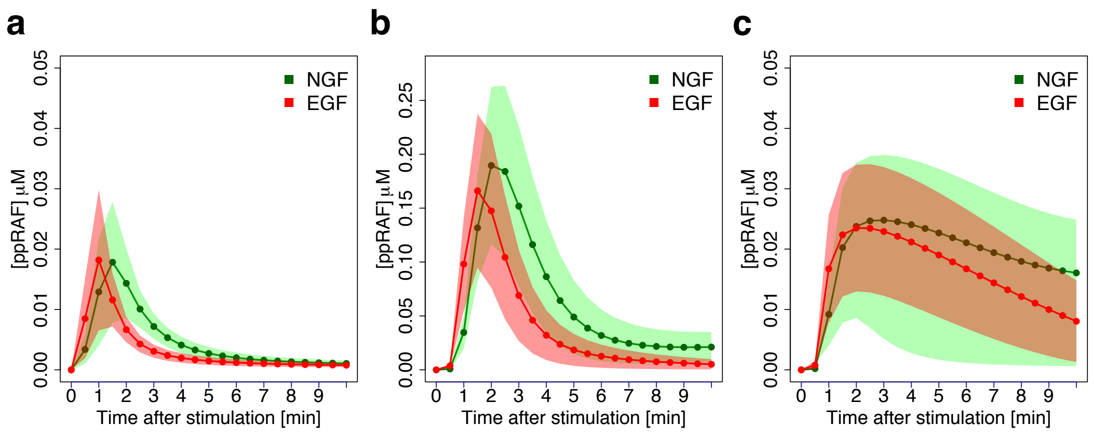


Figure 5.3: Simulations of the protein network sensing EGF and NGF in PC12 cells - (a) Simulated time profiles of pRaf level as a function of time after NGF (green) or EGF (red) stimulations, without noise (solid line) or with 0.7 noise level (greed and red shadows). (b) and (c) are same as (a) but for ppMek and ppErk, respectively.

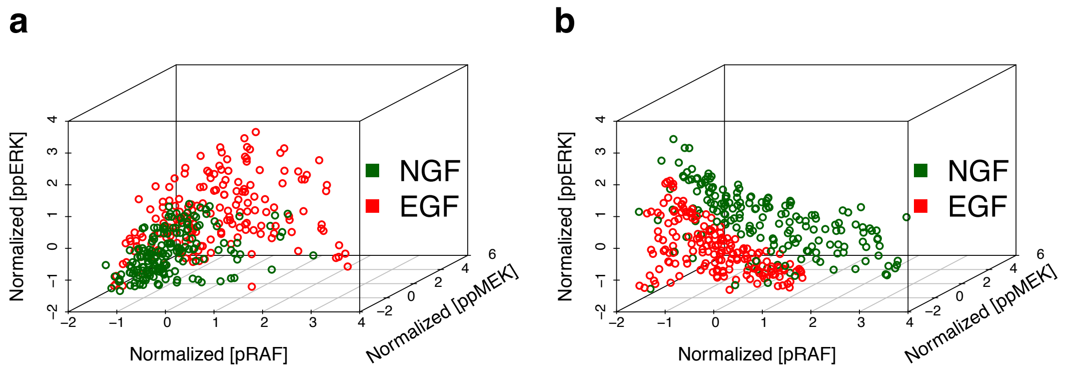


Figure 5.4: UNPBN algorithm unmix observations from different signaling networks - UNPBN unmixed observations of mixed cell population stimulated with NGF (green) or EGF (red). The 3-dimensional scatter plots show the levels of active Raf, Mek and Erk in each observation at 2 minutes (a) and 9 minutes (b) after growth factor stimulation, with a noise level of 0.7.

5.3 Unmixing subpopulation of cells with distinct cell signaling circuits

Simulated observations from EGF and NGF stimulated cells were randomly selected and mixed to create a heterogeneous mixture of cells. The UNPBN was applied only on the levels of active Raf,Mek and Erk, to imitate the limitation in the experimental setup to measure a part of the system. As the UNPBN algorithm developed by Mr. Wieczorek and Prof. Ickstadt is computationally expensive, a parallel matlab implementation of the algorithm is used to unmix heterogeneous cells (2 or 4 population mixture) with different levels of noise. The accuracy of correct allocation of observation is about 100%, 93% and 85% for the noise levels 0.1, 0.5 and 0.7, respectively (Fig. 5.5 a). In order to evaluate the performance of UNPBN, Mr. Wieczorek tested unmixing through two commonly used clustering approaches viz. hierarchical clustering and k-means clustering. At lower noise levels, the two subpopulations are quite correctly allocated by all three methods where as at higher noise level UNPBN out performed other methods (Fig. 5.5 a). The UNPBN also out performed other approaches, when the unmixing evaluated for simulation data along different time points. This suggest the ability of UNPBN to unmix subpopulations on the basis of high-dimensional relation of observations (Fig. 5.4). Mr. Wieczorek developed another algorithm which used the outcome of UNPBN analysis to correctly identify the number of subpopulation contained in the mixture. To compare the performance of UNPBN, Mr. Jakob unmixed the same data using other clustering approaches like hierarchical and k-means clustering. The comparison of unmixing performance showed that UNPBN surpass other clustering approaches in unmixing at higher noise level.

5.4 UNPBN reveals distinct topology of each subpopulation

The dynamic simulation of model shows, typical temporal pattern in the levels of pRaf, ppMek and ppErk upon EGF or NGF stimulation. This suggest that the statistical connection between these three components constantly changes.

This in turns propose other unmeasured components in the network. The level of those unmeasured components, mediate and influence the connection between the active pRaf, ppMek and ppErk. UNPBN algorithm offers insight into statistical strength of connectivity. The Bayesian model identified for each subpopulation is an acyclic graph with connection strength defined by posterior edge probability. UNPBN uncovers the posterior edge probability for connections between Raf-Mek-Erk for distinct subpopulation along the different time points after growth factor stimulation (Fig. 5.7). The posterior edge probabilities of mixed population when estimated though a standard Gaussian Bayesian Network (GBN) approach without unmixing, an average behaviors was observed. The average connection strength is incorrect when the two subpopulation display a very distinct posterior edge probability (PEP) (Fig. 5.7, 2 minute). This demonstrates the inevitable need to separate subpopulations to capture their actual connectivity and network topology of protein-protein interaction.

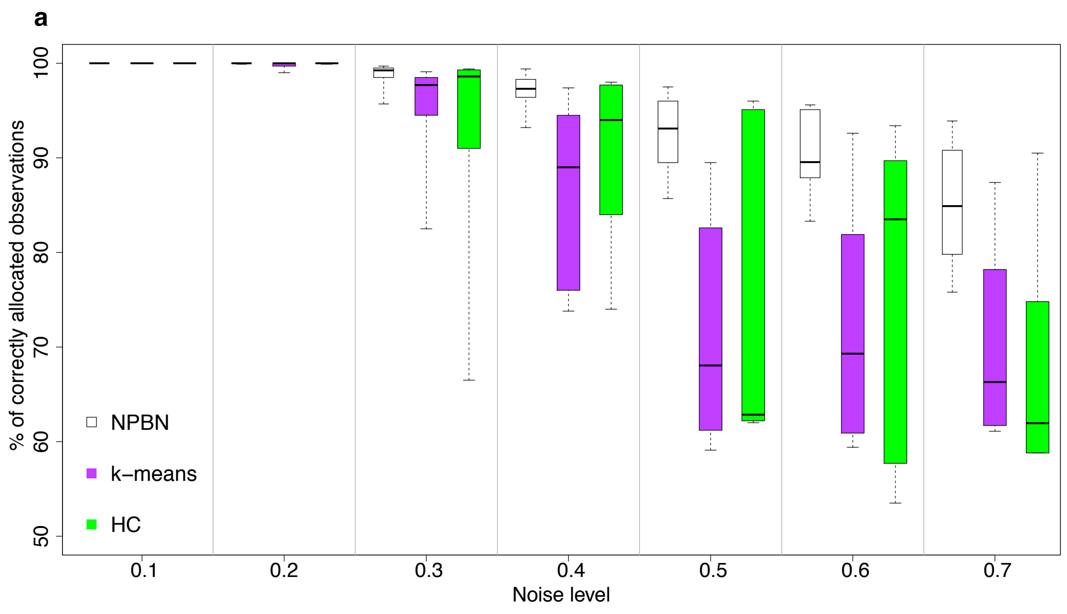


Figure 5.5: Efficiency of unmixing by UNPBN in comparison to clustering approaches - Mixtures of observations of EGF and NGF treated cells are simulated with different noise levels. For each noise level, observations are unmixed using UNPBN, k-means clustering (with $k = 2$) and hierarchical clustering (choosing final two clusters). The percentage of observations correctly allocated by different approaches are indicated by boxplots for various noise level.

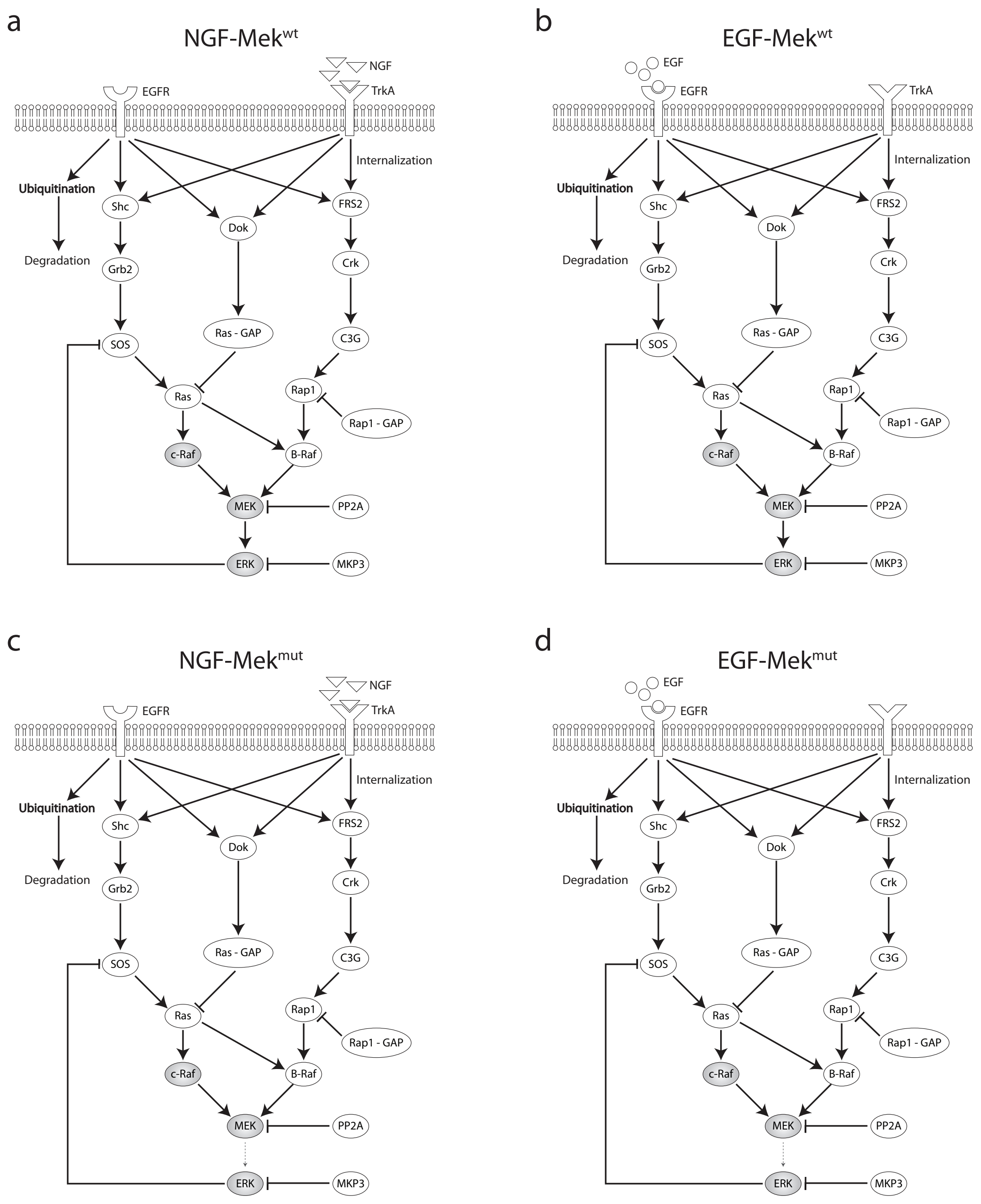


Figure 5.6: Simulation of EGF and NGF signaling networks with four distinct topologies - NGF-Mek^{wt} (a) and EGF-Mek^{wt} (b) are the wild-type networks with NGF and EGF stimulation, respectively. NGF-Mek^{mut} (c) and EGF-Mek^{mut} (d) are aforementioned networks with an alteration in SBML parameter corresponding to k_{cat} of Mek (J136) that depicts a mutant Mek with a lower activity (indicated as dotted arrow from Mek to Erk).

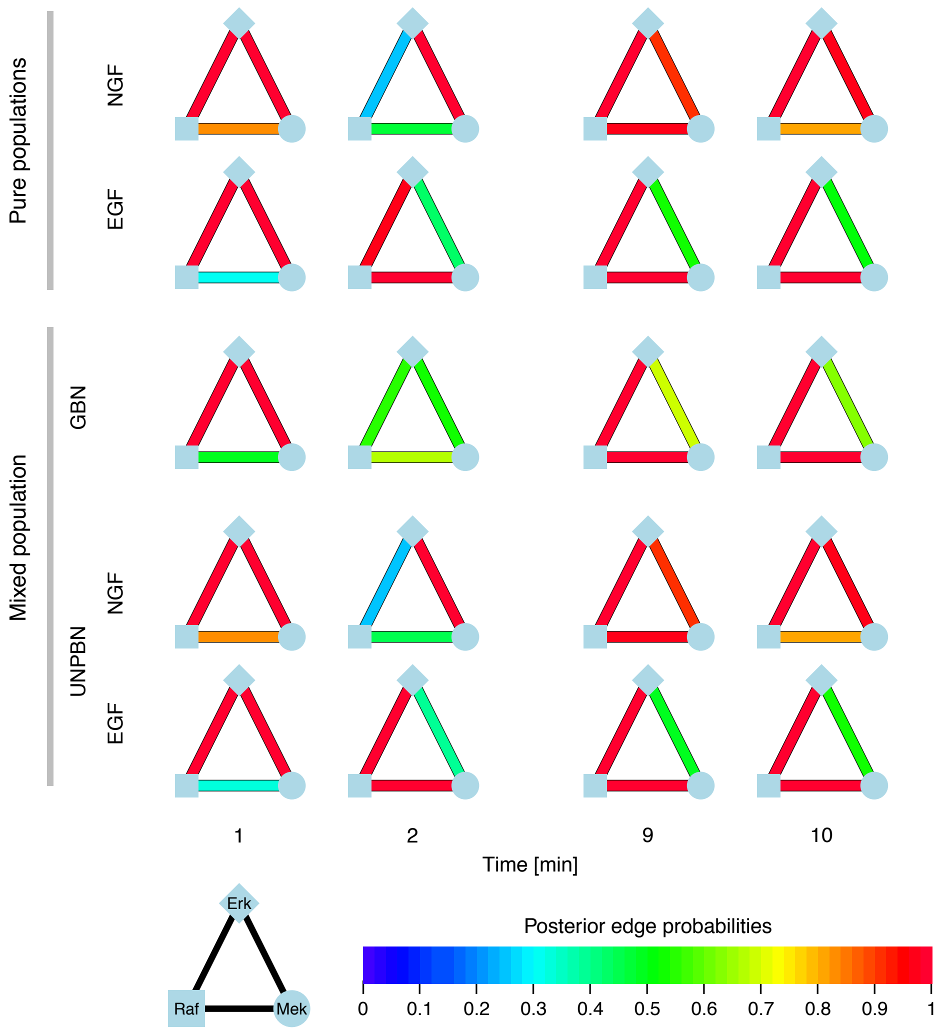


Figure 5.7: Recovering correctly the edge probabilities network between pRaf, ppMek and ppErk for each distinct cell subpopulation in a mixed population - The color code of the edges between pRaf, ppMek and ppErk represents the posterior edge probabilities. The top two rows show the edge probabilities derived by UNPBN analysis of pure NGF-and EGF-stimulated cell population. The edge probabilities of the mixture of population derived without unmixing by GBN show an average implausible edge probability that does not represent any of the subpopulation. In contrast, the UNPBN analysis of mixed population, yield two networks with edge probabilities corresponding to the edge probabilities of each pure population. Thereby the UNPBN analysis can recover correctly the network topology from observations of different time points after stimulation of growth factor (NGF or EGF).

Chapter 6

Discussion and outlook

A fundamental question in cell biology is how a large number of interdependent signaling proteins communicate robustly to self-regulate and execute diverse cellular functions at spatially distinct regions in response to local extracellular cues. Answering this question, requires resolving structure of protein networks at sub-cellular resolution. Despite having a paramount biological significance, inference of protein network topology at sub-cellular resolution remained a formidable task due to several basic challenges. This thesis addressed the fundamental challenges in detection of protein networks, grasping their dynamics and regulation and inferring their structure with spatial and temporal resolution by developing novel concepts and approaches. More importantly, the structure-function relationship between the protein network of adhesion sites and their dynamic regulation is resolved at sub-cellular resolution.

6.1 Compositional imaging resolve the dynamics of spatially heterogeneous protein network states in adhesion sites

Cells sense spatially resolved cues about biochemical and biophysical properties of extracellular microenvironment and in response form adhesion sites with spatially heterogeneous protein networks and function. The local signal induces spatially diverse condition sets within the cell that favors condition-specific protein-protein

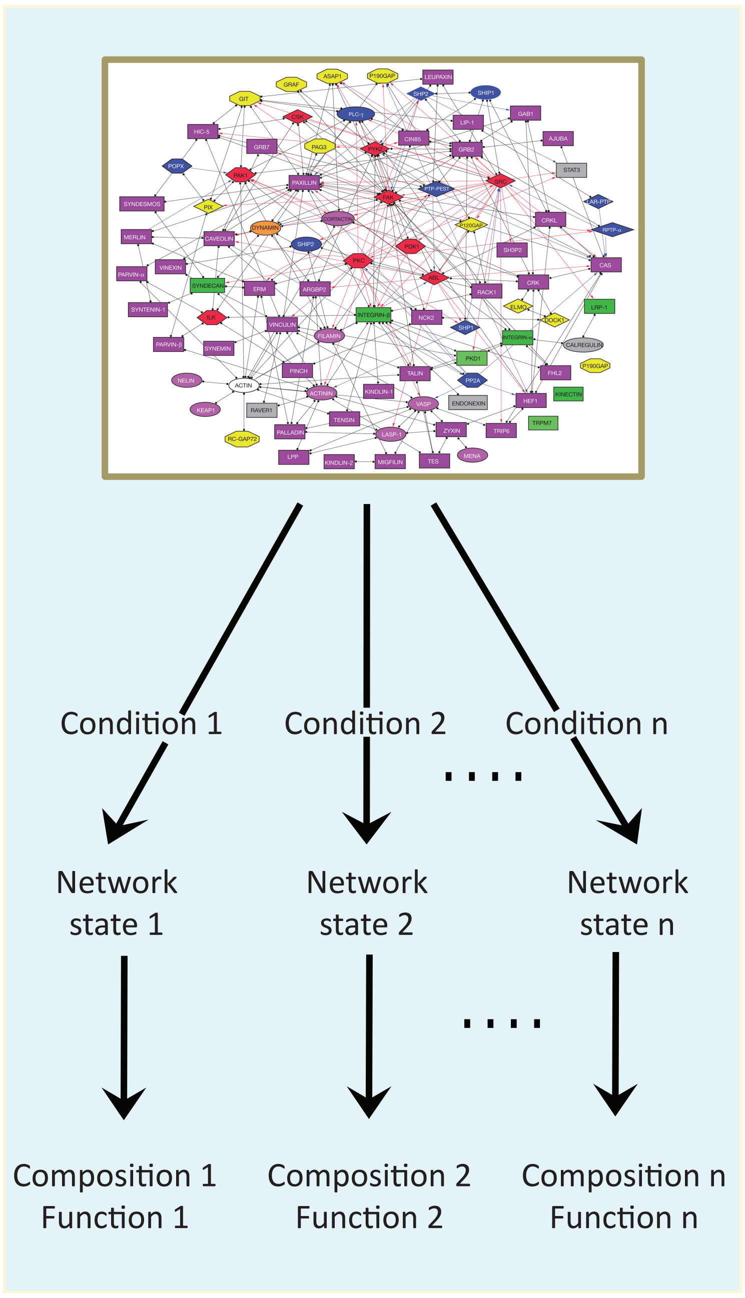


Figure 6.1: Spatillay heterogeneous protein networks emerge from varying local conditions within the cell - Many condition sets spatially vary within the cell in response to immediate extracellular cues. The diverse local intracellular conditions favor condition-specific interactions and therey persuade the protein network to assume specific state that are spatially heterogeneous. Each protein network perform certain function and each network state exhibit a characteristic composition (map of protein-network adapted from Zaidel-Bar et al. 2007).

interaction. This results in formation of adhesion sites with diverse protein network that can perform distinct locally required function. Each protein-network state exhibit a characteristic composition of network components (Fig. 6.1). By identifying the compositions of adhesion sites through compositional imaging approach, the underlying network states that generate such composition can be indirectly captured.

The challenge to detect of protein network in few micron long adhesion sites is surmounted by developing multicolor live cell imaging system (Section 2.1). Furthermore, several computational tools are developed to systematically process the multicolor time lapse images to remove any imaging artifacts and noise and prepare the data for quantitative analysis (Section 2.2). To capture the dynamics of protein network states with spatial resolution, the compositional imaging approach is applied and the transition of composition. Even though clusters of composition are manually selected, they are chosen from dendrogram through several round of iteration following the objective rule sets as described in the method section. Hence, the identified clusters of composition are robust. This is evident from the fact that the relative abundance of the composition is apparently uniform in all 10 cells analyzed (Fig. 3.5 c). Additionally, the probabilities of transition in composition is also comparably uniform in all 10 cells (Fig. 3.7 b). Despite varying amount of the ectopic proteins expressed in the cells, the compositions are robustly observed in different cells. This could be because the ectopic proteins complete only with their endogenous counter part to localize to adhesion sites and the sites available for these protein to bind other endogenous protein remains unaffected. Therefore, localization of any addition ectopic protein in the adhesion site is limited by the levels of the endogenous adaptor proteins.

Compositional imaging of adhesion-cytoskeleton system of spread fibroblast cells expressing Zyxin, α-actinin, vinculin and paxillin (Fig. 2.3) revealed compositions specific to adhesion sites, stress fibers and membrane protrusion. The results (Section 3.2.1) revealed the compositions pertaining to adhesion sites (A,F,G,H), stress fibers (A,B,D,E) and membrane protrusion (D). Composition 'B' is observed in the cytoplasmic region around the adhesion sites. Composition 'D' and 'E' form striation in stress fibers. The α-actinin rich composition 'H (and sometimes 'F') is present in the proximal region of the adhesion site from which the stress fibers emanates. Occasionally, it is also observed in complete adhesion site. The tip or the distal region of the adhesion site away from the actin-filament often exhibit composition ‘G’, with relatively higher level of paxillin and vinculin in comparison to zyxin. Composition ‘C’ is the least abundant composition observed in the outer membrane regions and around stress fibers.

6.2 Spatial organization of molecular composition reflects state of the focal adhesion

In order to investigate the network-states within adhesion sites at higher spatial and composition resolution, pixels pertaining specifically to adhesion sites are clustered and compositional imaging is performed (Section 3.2.2). Pixels are considered as an object in multidimensional compositional space with its dimension corresponding to the level of each protein (zyxin, α-actinin, vinculin and paxillin). The adaptor proteins that localize in adhesion site are in their active state (achieved by conformational change and/or phosphorylation or specific interaction with other component). Here the active state refer to the ability of a protein to interact with its partners. For example only vinculin with open active conformation can localize in adhesion sites. Vinculin present in the cytoplasm is inactive due to its auto-inhibited closed conformation (Mierke, 2009). Therefore the dimension of the multidimensional composition space is level of active protein present in adhesion sites. So, the compositional space may also be referred as network-state space and the compositional analysis indeed captures spatially heterogeneous protein-network states.

The compositional imaging (Section 3.2.2, Fig. 3.11) revealed multiple domains within the adhesion sites with distinct network states (Fig. 3.12). These distinct protein network states always follow certain order of spatial organization. The order starting from the proximal end of focal adhesion that is connected to actin filament is 1) ‘E (red)’, ‘F (cyan)’, ‘D (blue)’ , ‘C (yellow)’, ‘A (green)’ and/or ‘B (orange). The proximal end of the focal adhesion is comprised of α-actinin rich composition ‘E (red)’. This is the region connected to the actin filaments and hence it is characterized with higher level of α-actinin and moderate level of zyxin in comparison to low levels of vinculin and paxillin. This composition is observed in many focal adhesion but not all and its presence signifies stronger connection of focal adhesion with stress fibers. The next network-state that follows is 'F (cyan)' composition, that contains slightly higher levels of α-actinin and zyxin and moderate levels vinculin and paxillin. This domain is pretty much observed in all focal adhesion. Zyxin and α-actinin are both component of actin filaments (Hirata et al., 2008; Sjöblom et al., 2008) and hence this region may also acts as major region junction for stress fibers and focal adhesion. The composition 'D (blue)' is observed in all the mature adhesion sites and it contains slightly higher level of zyxin and moderate amount of other three proteins. This composition also form the central core of many adhesion sites. This composition is preceded by 'C (yellow)' composition that contains almost equal and moderate levels of all four proteins. It is observed in majority of the adhesion sites. The adhesion sites are mostly capped with the composition 'B (orange)' at the tip or the distal region. This region has higher levels of paxillin and moderate levels of vinculin and zyxin and low levels of α-actinin. This composition with higher paxillin has also been previously reported (Zamir et al., 2008). The low levels of α-actinin suggests that this region lack any stress fibers above it. The composition 'A (green)', is not often found in adhesion sites, they are found in rare structures around adhesion sites. When present, this composition proceed or precede the composition 'B (orange)'. Both the latter composition contain moderate or higher level vinculin. In contrast it has been previously observed that vinculin localized in the proximal tip of the focal adhesions (Chen et al., 2005) and this different might be due to the consideration of relative levels of vinculin.

6.3 Molecular mechanisms of focal adhesion assembly and disassembly

One of the basic question about adhesion sites that is still not clearly understood is how these multi-molecular structures assemble and disassemble. So far, no approaches have investigated this query by monitoring multiple component dynamics. In this compositional imaging approach is used to address this issue. The compositional imaging visualized the spatial and temporal dynamics of protein networks within adhesion sites. This offered a valuable opportunity to examine how adhesion sites with multiple protein network states within are formed and destroyed. Assembly and disassembly of adhesion sites exhibit an asymmetric spatial and temporal order of network-states appearance and disappearance (Fig. 3.13).

The focal adhesion assembly initiates with formation of focal complexes. Interestingly these focal complexes that are less than a micron in diameter also contain two domains (with composition 'B (orange)' and 'C (yellow)') and are spatially polarized with the compositions 'B' at the distal region. Composition 'B (orange)' is paxillin-rich and 'C (yellow)' also has relatively more paxillin than all other compositions except 'B'.This concord with the reports that suggest, paxillin is among the early proteins that appear in focal complexes by binding to FAK (Bertolucci et al., 2005). When the focal complex start to mature, both theses domains expand towards the proximal region. Followed by which the other composition domains 'D (blue)' and 'F (cyan)' are subsequently formed again at the proximal region. Some adhesion sites additionally form composition 'E (red)' in the distal region after composition 'F (cyan)' and these adhesion sites grow relatively faster and larger due to higher pulling force. Subsequently, the domain with composition 'C (yellow)' increases in size and the distal cap of composition 'B (orange)' becomes prominent. Additionally, the composition domain 'A (green)' also appear. Finally, the mature adhesion sites is formed with transition of major portion of the composition domain 'C (yellow)' into 'D (blue)', a characteristic composition of mature adhesion sites. The distal cap composition 'B' in some cases is lost and in some retained.

During assembly, the initial focal complex gradually grow in the proximal region toward the interior of the cells to form mature focal adhesion. In contrast, during disassembly, the focal adhesion start disassembling from the distal end to the proximal end. Firstly in the disassembly, the composition 'B (orange)' and 'C (yellow)' begin to disappear gradually. This is followed by disappearance of the core composition 'D (blue)' and the proximal most composition 'E (red)'.

The composition 'D (blue)' may also undergo a transition to 'F (cyan), which finally remains for a while before disappearing completely. To summarize, the focal adhesions exhibit asymmetric spatial polarity in assembly and disassembly process. Additionally, focal adhesions also adapt a temporal pattern in formation and destruction of molecular domains or network-states within them during assembly and disassembly processes.

6.4 Actomyosin contractility controls the dynamics and organization of protein networks within adhesion sites

Mechanosensitivity is one of the basic properties of focal adhesion. Focal adhesion rely upon pulling force exerted by actomyosin contractility for maturation (Geiger et al., 2009). On the other hand, the actin filaments depend of focal adhesion as their anchoring points. Focal adhesions exhibit different spatial pattern and organization of network-states or compositions (Fig. 3.14) possibly linked with their mechanosensitivity. The focal adhesion that consists of proximal 'E (red)' and distal 'B (orange)' composition may be experiencing a strong pulling force. The α-actinin low composition imply that the pulling by actin filaments has caused less α-actinin distal zone of the focal adhesion. It can be observed from the compositional images that such adhesion sites either grow in their size or slide rapidly or both (Fig. 3.14 b). In contrast, the adhesion sites, that lack both the proximal 'E (red)' and distal 'B (orange)' composition are either stationary or slide very slow (Fig. 3.14 a). The latter type of adhesion sites, predominantly contain the composition 'D (blue), that consist of slightly high zyxin and moderate and equal amount of all other component. Additionally, the latter adhesion sites also contain a small proximal region of composition 'F (cyan)' with slightly high zyxin and α-actinin, so that it can connect weakly with stress fibers. The functional role of theses adhesion sites could be to serve as strong anchoring points, where as the former may have role in sensing the stiffness of the extracellular microenvironment as well as cell migration. Based on these observations the two type of focal adhesions with distinct molecular organization can be inferred viz. one that is strongly linked with stress fibers and the other that is weakly or not linked to stress fibers (Fig. 6.2).

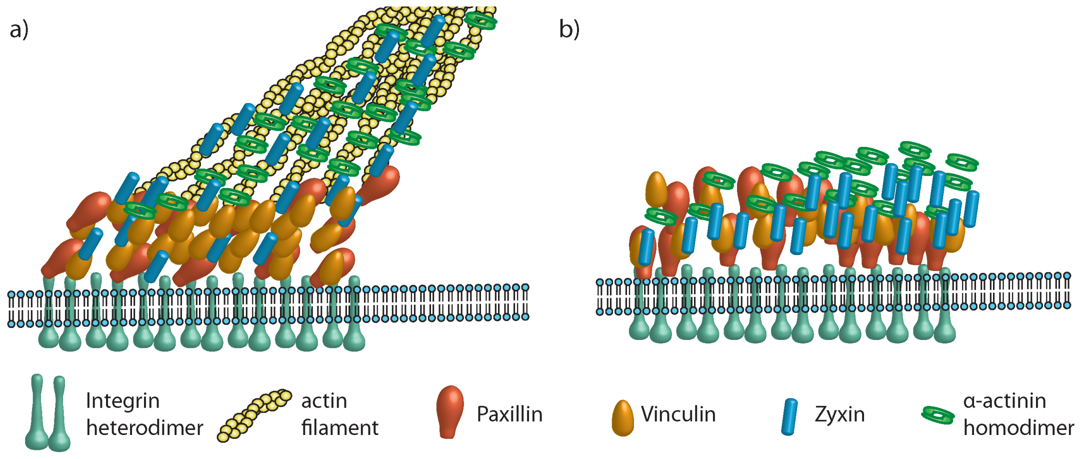


Figure 6.2: Models of focal adhesion architecture - Focal adhesion connected to stress fibers (a) and those not (b) exhibit different molecular organization. The tip of the focal adhesion connected to the actin bundles are α-actinin rich, and region preceding it is rich in both α-actinin and zyxin. The distal end consist of relative higher levels of paxillin and vincullin and low levels of zyxin and α-actinin. The focal adhesion that are not connected to actin bundles exhibit a distal region (towards membrane) with equal amount of all proteins, and the frontal region with relatively higher amount of zyxin and α-actinin.

6.5 Relationship between molecular composition and mechanosensistivity of adhesion sites

To further study the mechanosensitive properties of focal adhesion, compositional imaging is performed on fibroblast cells (expressing paxillin, VASP, α-actinin and zyxin) subjected to cycles of actomyosin inhibition and recovery. The results (Section 3.3, Fig. 3.19) revealed context (perturbation / recovery) specific compositional signatures (Fig. 3.19) d). The focal adhesion and the stress fibers displayed the composition 'C' and 'B' respectively (Fig. 3.19) b) in the unperturbed state. But after actomyosin inhibition, the adhesion sites and stress fiber disassemble and later reassemble following the drug washout. Interestingly, these structures formed after recovery displayed a different composition i.e. 'D' and 'G' for adhesion sites and stress fibers respectively, that contain low level of zyxin in comparison to their counterpart in unperturbed cell. When actomyosin is inhibited for the second time, these adhesion sites with composition 'D' did not disassembly as expected, suggesting a gain of resistance to force perturbation. This new property of adhesion sites could have emerged from the new composition, that lacks zyxin, an important mechanosensing component (Hirata et al., 2008). Although this observation is shown in single cells, it has also been observed in many other cells (data not included). This experiment should repeated in multiple cells to validate this observed phenomenon.

6.6 Compositional transition facilitates reconstruction of pathways of network-state transitions

Adhesion sites with distinct molecular composition and cellular functions are generated in response to the external cues. Additionally, cell-matrix adhesion sites can alter network-state and thereby their molecular composition to perform locally desired function. The dynamic changes in network-state or the composition is estimated as compositional transition. The probabilities of compositional transition divulge the physiologically favored flux of protein network states. The probability of transition between all compositions of adhesion sites (Fig. 3.16) can be represented as directed network (Fig. 6.3). This network correspond to the pathways a protein-network state can possibly take during steady state adhesion site assembly or disassembly. The pathways of network-state transitions suggest that the cells adapt a conservative mechanism to generate network-states. The current network-state of a focal adhesion with certain function can be altered by the cell in response to the local extracellular cues to produce focal adhesion with different network-state and function. This will allow cells to rapidly respond to the external stimulus. The pathway of compositional transitions uncover various valuable information about regulation of protein-network states or composition.

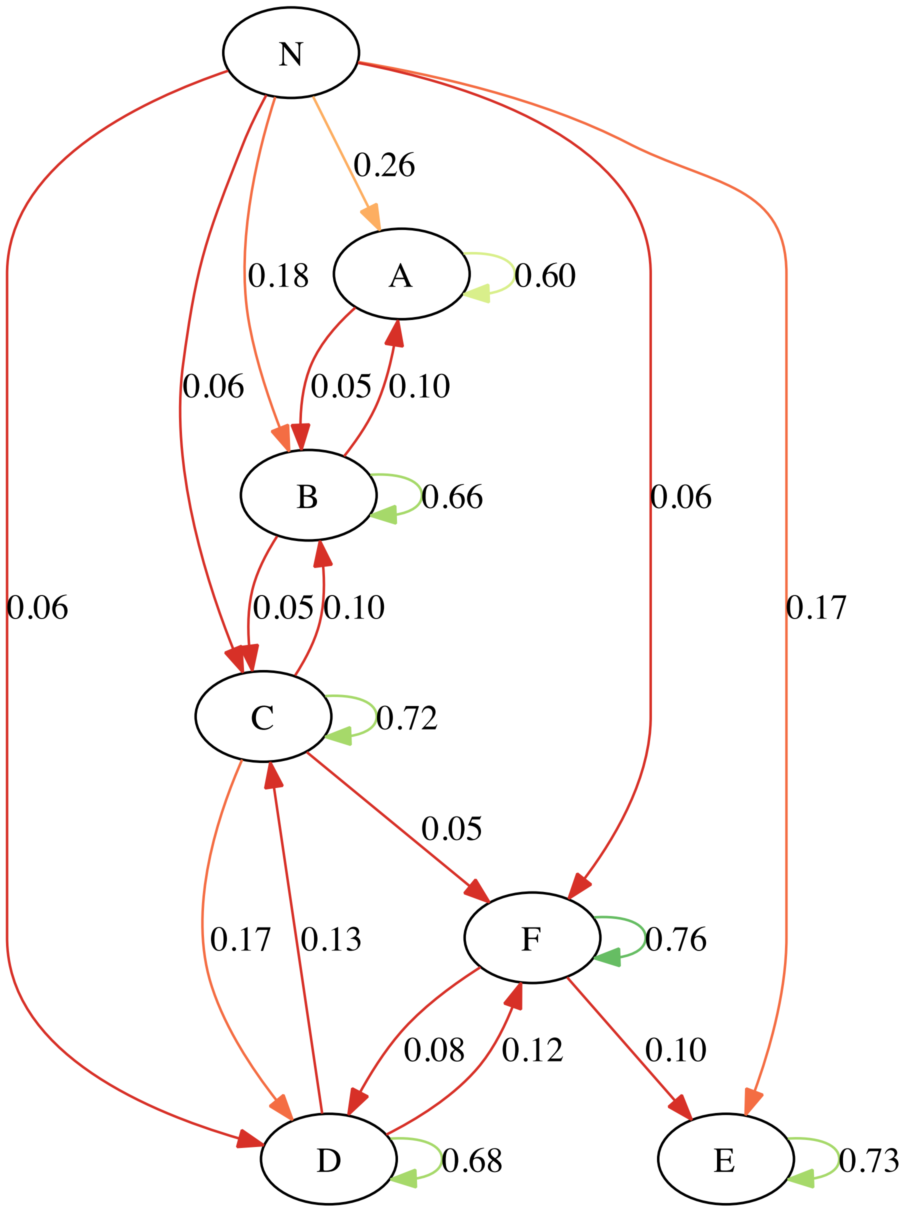


Figure 6.3: Untangling the pathways of protein-network dynamics - The matrix of probability of transition from Fig. 3.16 can be graphically represented to display the pathways of protein network inter-conversion in steady state fibroblast cells.

The composition ‘A’, ‘B’ and ‘E’ are assembled directly from the empty cytoplasmic region of the cell, where as the other clusters should be generated though one of theses composition. Composition ‘C’ can convert into ‘D’ during maturation of adhesion site and the vice versa occurs during adhesion disassemble. This approach helps to understand the dynamic behavior network states which is reflected by the ensemble of changes in network structure. Additionally, it would be interesting to develop approach that can dissect pathways specific to dynamic state such as assembly or disassembly of adhesion sites.

6.7 Causal connections between proteins determine the dynamic state of focal adhesions

Although the protein composition of adhesion sites, indirectly represent the network states, it cannot divulge structure of protein network. It is essential to elucidate the protein network structure to understand functional properties of protein network. The focal adhesion segmentation tool allowed automatic segmentation and tracking of adhesion sites, to extract the timeseries of multi-protein level. The significant network connection observed in the focal adhesion include reciprocate causal relations between zyxin - α-actinin pair and Vinculin-paxillin pair. Additionally, vinculin also affects α-actinin. This matches with the localization and their interaction behavior. The reciprocate causal link between zyxin and α-actinin is observed in all three dynamic state i.e. assembly, disassembly and stationary state of adhesion sites. Vinculin affects paxillin steady state as well as disassembling adhesion sites more prominently that assembling adhesion. In contrast, paxillin weakly affects vinculin almost equally in all three states. Vinculin affects α-actinin only in disassembling focal adhesion. One of the limitation of the approach used is that it cannot infer the strength of the connection. Moreover the methodology adapted to detect significant connection captures a limited number of connections (sometimes only one) in each adhesion sites. This seems to be biologically implausible. The number of significance of the causal connection inferred can be increased by removing high-frequency noise in the time series in future analysis. The spatial mapping of protein network also doesn't divulge protein network due to aforementioned limitation. Nonetheless, an interesting observation profoundly found is that the vinculin affects α-actinin more abundantly in sliding focal adhesion. In this approach the causal connections refer to the change in the level of one protein upon change of other in focal adhesion. Therefore, in sliding focal adhesion, vinculin may recruit α-actinin to make stronger connection with stress fibers and thereby get pulled efficiently.

As the theoretical approaches can only divulge potential causal connection, it is essential to carry out perturbation experiments to resolve actual causal strength and network topology. The perturbations experiments subject the focal adhesion to diverse condition i.e. the depletion perturbations causes disassembly of the focal adhesion, where as the repletion perturbation allows the focal adhesion to recover and reassemble. The topology of protein network inferred by both the perturbation can therefore offer insight into assembly as well as disassembly processes. The preliminary results show that the topology of protein networks during assembly and disassembly are different implying asymmetry in the regulatory mechanism (Fig. 4.10). This observation is also consistent with the compositional analysis as previously discussed. FAK, parvin and vinculin indirectly interact through the common adaptor protein, paxillin. The causal effect between these proteins is therefore mediated via paxillin. FAK interacts with paxillin with a stoicheometry of 1:1 (Zimerman et al., 2004) and therefore, it can be observed that, FAK strongly influence both Parvin and Vinculin positively during disassembly. Additionally, FAK and Vinculin also have other common interaction partners which include α-actinin and Talin1 (Fig. 1.4). The negative influence of FAK on Vinculin during assembly of focal adhesion may arise from the fact that both FAT domain of FAK and tail of vinculin compete with LD4 domain of paxillin (Turner et al., 1999). Although the initial analysis offers interesting observations, it is required to analyze more data to derive at strong biological conclusions.

6.8 Potential implication of unmixing in protein network-reconstruction

Heterogeneity of protein network is a commonly observed phenomenon as the protein networks differ at intercellular (e.g. in subpopulation of cells) as well as intracellular (e.g. adhesion sites) level. Although, several methods are developed to infer the structure of causal protein network, the fundamental concept of separating the mixture of protein networks is never addressed. Therefore all the approaches (Table: 1.1) resolved an average meaningless topology of protein network whenever more than one population of protein network exists. To address this fundamental challenge, a novel approach UNPBN is developed in collaboration with statistic department of TU Dortmund. Moreover, the need and usefulness of this approach is demonstrated using a MAPK signaling network that exhibit two different topology depending on the growth factor stimulation (Santos et al., 2007). UNBPN in addition to unmixing also offers a network structures for each unmixed population. One of the major advantages of this Bayesian networks based approach is the estimation of edges strength of the network by conditional probabilities that captures high-dimensional relationship between proteins. This approach can unmix protein networks efficiently from small data sets (300 observations). On the other hand, the current UNPBN algorithm is computationally too expensive to analyze data set with large number of observations. Moreover, the computation time also increases with increasing number of network components considered for the analysis. Therefore, the algorithm requires further optimization by incorporating relevant sampling based techniques to handle large data set and data dimension.

UNPBN, when coupled with FACS can offer a powerful potential to sort subpopulation from conglomerated tissue mass. The UNPBN based cell sorting, can open up opportunity to investigate unadulterated biophysical and biochemical properties of each subpopulation with distinct protein network. As, UNPBN can unmix protein networks from any multivariate measure, the protein levels quantified from multicolor images of adhesion sites can be used to unmix spatially heterogeneous protein networks with distinct causal connections. Although this was the original motivation behind the development of UNPBN approach, it cannot be achieved within the timeframe of this thesis. As the processed data and the approach is readily available from the thesis, applying UNPBN to unmix protein networks of focal adhesions and spatially mapping the subpopulation on the images to infer the structure-functional relationship, is an important task that should be subsequently addressed. As UNPBN is applied on static data, it cannot resolve directed connections in the protein networks. UNPBN cannot also resolve autoregulatory edges in the protein network as it cannot fit cyclic graph models to the data. Hence, it would be of immense advantage to take this approach to the next level by developing a novel approach that combines UNPBN with dynamic Bayesian network approach. The time traces quantifying the multiprotein level in segmented focal adhesion can be a useful data set for this approach and the novel analysis can offer insight into directed protein network structure and autoregulation of proteins in addition to unmixing.

Chapter 7

Materials and methods

7.1 Reagents

7.1.1 Plasmids

The plasmids used in this thesis were derived from the PEGFP vectors (Clonetech). In this vector the GFP is replaced with other flurophores i.e., TDmTagBFP, TDmTFP, mCitrine and TDmKate2 and the adhesion proteins paxillin, vinculin, α-actinin and zyxin were cloned into the vectors respectively by Mr. Moganti and Dr. Stricker. The adhesion proteins are cloned with the same restriction sites BglII and EcoRI in all four different vectors. Plasmids used for multicolor imaging of actomyosin inhibition cycles in REF52 cells include TDmKate2-paxillin, mCitrine-VASP, TDmTFP α -actinin and mTagBFP-zyxin. The plasmids used for acute perturbation experiments performed as a part of the thesis include D-mTagBFP-FLAG-MOA-N1 (recruiting compartment), A-mTFP-mTFP-parvin-C1 (bait protein) mCitrine-FAK-C1 (prey protein 1) and TDmKate2-vinculin-C1 (prey protein 2) are cloned by Dr. Stricker in a backbone derived from PEGFP vector (Clonetech).

7.1.2 Cell culture and transfection

Rat embryonic fibroblast (REF52) cells were maintained in DMEM supplemented with 10 % fetal calf serum. REF52 cells that are $60-70\%$ confluent in Lab-Tech

4 chambers slides are transfected with 0.5 μg of each of the four plasmids for e.g. TDmTagBFP-paxillin, TDmTFP-vinculin, mCitrine-α-actinin and TDmKate2-zyxin following the protocol of Lipofectamine 2000 (catalog number: 11668-019, Invitrogen).

7.1.3 Inhibition of actomyosin contractility

Acute perturbation of actomyosin contractility is achieved by addition of pharmacological inhibitor to the medium of the cells at a given time point during imaging. For inhibition of ROCK, cells were treated with 100 μM Y-27632 (catalog number Y0503, Sigma-Aldrich Logistik GmbH, Schnelldorf, Germany).

7.2 Microscopy

7.2.1 Four color live cell imaging

Live cell imaging was performed using a Zeiss LSM 510 confocal microscope (Zeiss Microsystems GmbH, Germany) with a 40x water immersion objective. Before imaging cells, the growth medium in their glass bottom dishes was replaced by imaging medium (catalog number P04-01163, PAN Biotech, GmbH). Cells were then maintained on the microscope stage at 37 and 5% CO_2. Multicolor imaging is achieved by sequential excitation and detection of TDmKate2-Paxillin (excitation at 561 nm and emission detection range at 575-800 nm), mCitrine-VASP (excitation at 514 nm and detection at 530 -565 nm), TDmTFP-α-actinin1 (excitation at 458 nm and detection at 470-500 nm) and TDmTagBFP-Zyxin (exitation at 405nm and detection at 420-480 nm).

7.2.2 Imaging 4-color beads

To correct lateral chromatic aberration, multicolor beads (TetraSpeck, catalog number T14792, Life Technologies) with 1 μm diameter are imaged after every 4-color microscopic session under same imaging setup and zoom factors used. A relatively higher laser power is used to enhance the signal-to-noise ratio of the multicolor images of beads .

7.2.3 Acute perturbation of focal adhesion

REF52 cells are transfected with D-mTagBFP-FLAG-MOA-N1 (recruiting compartment), A-mTFP-mTFP-parvin-C1 (bait protein), mCitrine-FAK-C1 (prey protein 1) and TDmKate2-vinculin-C1 (prey protein 2) are imaged at Olympus Cell-R wide field microscope. Multicolor imaging is achieved by sequential excitation and detection of TDmKate2 (excitation at 545-58 nm and emission detection range at LP610 nm), mCitrine (excitation at 512-515 nm and detection at 528 -555 nm), mTFP (excitation at 450-467 nm and detection at 484-504 nm) and TDmTagBFP-Zyxin (exitation at 381-399 nm and detection at 425-442 nm). Multicolor time-lapse images are acquired at an interval of 5 minutes between two imaging frame. After initial 10 imaging time frames, 1 μM ligand is added to the imaging medium to induce dimerization of A and D domain and thereby recruitment of the prey protein to the mitochondrial region. The depletion perturbation is monitored for next 35 minutes (7 time frames) and 1 μM competitor molecule is added to release the perturbation. The repletion perturbation is monitored for another 30 minutes (6 time frames).

7.3 Computational protocols

7.3.1 General image processing

Step 1: Raw Images Multicolor live cell imaging data is acquired, resulting in a set of raw (pixels) data $P^{RAW}(x_i, y_j, t_k, \vec{I})$, where x and y denote the position of a pixel in the image, t is the time point and $\vec{I}$ is a vector indicating the fluorescence intensity of each of the labeled protein in that pixel. Typically images were acquired with 512 X 512 pixel size and 16-bits fluorescence intensity resolution.

Step 2: Chromatic aberration correction Lateral chromatic aberration in the raw images are corrected by unwarping the distortion with respect to one of the labelled proteins, estimated from the multicolor images of beads, to generate aberration corrected images $P^{AC}(x_i, y_j, t_k, \vec{I})$.

Step 3: Background correction The offset in the intensity of the aberration corrected images $P^{AC}(x_i, y_j, t_k, \vec{I})$ is removed by substracting the average background intensity from each image to produce background corrected images $P^{BgC}(x_i, y_j, t_k, \vec{I})$.

Step 4: Bleaching correction The artificial drop in the intensity levels of different proteins due to differential photobleaching of the fluorophores is corrected for each labeled protein $\vec{I}$ by multiplying all the pixels in each time frame t_k by a numerical factor that keeps total intensity of images cell constant along time.This step generates bleaching corrected images $P^{BlC}(x_i, y_j, t_k, \vec{I})$.

Step 5: Spatial low-pass (mean) filtration Spatial very-high frequency noise in the bleaching corrected images $P^{BlC}(x_i, y_j, t_k, \vec{I})$ is removed by low-pass mean filtration in which the value of each pixel get the value of the mean intensity of the pixel and its 8 nearest neighboring pixels, resulting in $P^{LP}(x_i, y_j, t_k, \vec{I})$.

Step 6: Spatial high-pass filtration Spatial low-frequency background noise $P^{LP}(x_i, y_j, t_k, \vec{I})$ is removed by high-pass filtration in which the value of each pixel is subtracted by the mean intensity of the pixels within a desired box size that is significantly bigger than the structures of interest, resulting in $P^{HP}(x_i, y_j, t_k, \vec{I})$.

Step 7: Image shift correction Lateral shift in the imaging frames of time-lapse images $P^{HP}(x_i, y_j, t_k, \vec{I})$ are corrected by restoring the shift in the pixels estimated by image registration of high-pass filtered images of one the the proteins $P^{HP}(x_i, y_j, t_k, I_{pr})$ to generate shift corrected images $P^{SC}(x_i, y_j, t_k, \vec{I})$.

7.3.2 Compositional imaging

To perform compositional imaging, additional image processing steps that removes spatial and temporal high-frequency noises are performed. Followed by which pixels harvested from the processed images are clustered and the clusters are analyzed and the compositional transitions are identified.

7.3.2.1 Method-specific image processing

Step 8: Temporal low-pass (Kalman) Filtration Temporal high-frequency noise in the pixels of images $P^{SC}(x_i, y_j, t_k, \vec{I})$ are removed by 2-way kalman filtration of intensity profile of each pixel along time for each labeled protein, resulting in $P^{KF}(x_i, y_j, t_k, \vec{I})$.

Step 9: Spatial low-pass (nsmooth) filtration Spatial high-frequency noise in the pixels of each image of $P^{KF}(x_i, y_j, t_k, \vec{I})$ is removed by robust smoothing of pixel intensity along space by nsmooth algorithm, resulting in $P^{NS}(x_i, y_j, t_k, \vec{I})$.

7.3.2.2 Cluster analysis

Step 10: Intensity thresholding To exclude noisy pixels outside the structures of interest, from the clustering analysis, a threshold (intensity) level is set for each protein in the processed images $P^{NS}(x_i, y_j, t_k, \vec{I})$. Pixels with intensity level below the threshold for all proteins are excluded from the further analysis. The resulting pixels vector $P^{T}(x_i, y_j, t_k, \vec{I})$ is smaller than that of $P^{NS}(x_i, y_j, t_k, \vec{I})$. Additionally, the pixels in $P^{T}(x_i, y_j, t_k, \vec{I})$ with negative intensity due to processing steps are set to zero (Zamir et al., 2008).

Step 11: Intensity normalization Now each pixel in the vector $P^{T}(x_i, y_j, t_k, \vec{I})$ can be considered as an object in the multidimensional space with co-ordinates corresponding to $\vec{I}$ i.e. intensity of each labeled protein. The dynamic range of intensity levels of different fluorescent proteins are normalized to similar range without affecting their distribution by dividing each component (I_{pr}) of $\vec{I}$ by the median intensity μ_{pr} of all objects with non-zero intensities ($P^{T}(x_i, y_j, t_k, I_{pr}) > 0$) resulting in the vector $P^{IN}(x_i, y_j, t_k, \vec{I})$.

$$P^{IN}(x_i, y_j, t_k, \vec{I}) = P^{T}(x_i, y_j, t_k, I_{pr})/\mu_{pr}$$

Step 12: Amplitude thresholding A multi-dimensional thresholding is done to exclude noisy pixels and/or segment regions of interest. The amplitude of each intensity normalized pixels $P^{IN}(x_i, y_j, t_k, \vec{I})$ in the multidimensional space is the distance from the origin. A higher amplitude threshold can be used to selectively harvest pixels pertaining focal adhesion and alternatively a low amplitude threshold can be used to harvest all non-noisy pixels from $P^{IN}(x_i, y_j, t_k, \vec{I})$. Therefore resulting amplitude thresholded pixels vector $P^{AT}(x_i, y_j, t_k, \vec{I})$ is smaller than $P^{IN}(x_i, y_j, t_k, \vec{I})$.

Step 13: Composition normalization In order to consider pixels/objects in the multidimensional compositional space by the stoichiometric ratio between the components, irrespective of their total amount, another step of normalization between the components ($\vec{I}$) of each pixel $P^{AT}(x_i, y_j, t_k, \vec{I})$ is applied such that euclidean norm (sum of the square of intensity levels of all component) for any pixel is 1 (i.e. ΣI_{pr}^2) resulting in vector $P^{CN}(x_i, y_j, t_k, \vec{I})$. It can be achieved by dividing each component (I_{pr}) of the vector $\vec{I}$ by square root of sum of squares of all component in $\vec{I}$ for a given pixel ($\sqrt{(\Sigma I_{pr}^2)}$)

$$P^{CN}(x_i, y_j, t_k, \vec{I}) = P^{IN}(x_i, y_j, t_k, \vec{I_j}) / \sqrt{\Sigma I_j^2 \forall j}$$

Step 14: Clustering All the pixels in $P^{CN}(x_i, y_j, t_k, \vec{I})$ are objects in the multidimensional composition space of labeled protein levels. Objects with similar stoichiometric composition are grouped by hierarchical clustering with spherical-distance measure and the dendrogram of clustered pixels is generated.

Step 15: Cluster selection and data visualization Significant clusters of pixels $P^{CN}(x_i, y_j, t_k, \vec{I})$ with similar composition and spatial localization and pattern can be empirically and iteratively identified by following the set of instructions mentioned below.

1. Clusters are selected by sampling the nodes with distinct composition from the dendrogram, irrespective of their hierarchical level, without violating dendrogram-topology. Each pixel in $P^{CN}(x_i, y_j, t_k, \vec{I})$ is assigned to one of the chosen cluster C resulting a new vector $P^{CL}(x_i, y_j, t_k, \vec{I}, C)$.

2. Average composition of the selected clusters are calculated. Clusters with similar compositions are merged by choosing their parent node in dendrogram instead.

3. In order to visualize spatial organization of compositional signatures from $P^{CL}(x_i, y_j, t_k, \vec{I}, C)$, all chosen clusters C are color coded and spatially mapped onto the image of the cell. Temporal abundance of chosen clusters at all the time points are estimated.

   (a) Clusters with differential localization and similar temporal abundance pattern are kept unmerged.

   (b) Clusters with mixed or overlapping localization but with distinct temporal abundance pattern are kept unmerged.

   (c) Clusters with mixed or overlapping localization and similar temporal abundance pattern are merged and proceeded to step 2.

4. Unmerged clusters are split into two clusters by choosing their child nodes and proceeded to step 2.

This process is iterated until optimal cluster selection is achieved resulting in clusters with distinct spatial and temporal pattern. Finally, in $P^{CL}(x_i, y_j, t_k, \vec{I}, C)$ each chosen clusters corresponds to a group of pixels with a given compositional signature. The biological role of each cluster or composition is inferred from the time lapse compositional images visualizing spatial and temporal dynamics of the identified composition (C).

7.3.2.3 Resolving compositional transition

Step 16: Estimating probability of compositional transition From the vector $P^{CL}(x_i, y_j, t_k, \vec{I}, C)$ the dynamic changes in the abundance of different clusters in a cell with time is calculated. For every consecutive time points ($t = n, n+1$), transition in the cluster of the pixels at every x,y is calculated. Thereby, probability of transitions $PT(C_{a,tn}, C_{b,tn+1}, \vec{t})$ between all clusters are estimated . The probability of transition of a cluster from C_a to C_b is the fraction of total pixels from cluster $C_{a,tn}$ that changes into cluster $C_{b,tn+1}$ at a given consecutive time frames. A threshold can be used to identify significant transitions. Probability of transitions of the clusters for every consecutive time frames pairs is visualized as directed graph to show the dynamics changes in the topology of transition.

7.3.3 Simulation of MAPK signaling network

In order evaluate the unmixing algorithm based on non-parametric Bayesian network (UNPBN) model, an established *in silico* dynamic model of Erk Signaling network (Sasagawa et al. 2005) showing differential response to EGF and NGF treatment, is used to simulate data. SBML model (BIOMD 0000000049), downloaded from www.ebi.ac.uk, is imported into Matlab Symbiology toolbox to simulate *in silico* kinetics of the signaling network (using ode15s (stiff/NDF) solver). To introduce cell-to-cell variability, the levels of unphosphorylated cRaf, MEK and ERK from a normal distribution $N(\mu, \sigma)$ is sampled around the respective initial concentration for a given set of percentage deviation from the mean ($\sigma = \mu \cdot pd$, where $pd \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$) and simulated with sampled (n = 1000) initial conditions. The values of *pd* represents cell-to-cell variability. To simplify notation this aspect will be referred with the term 'noise' . Dynamic response to EGF and NGF alone treatments were simulated for 600 seconds (i.e. for 10 minutes after growth factor stimulation). From the simulation data, the levels of active Raf (cRaf_Ras_GTP), double phosphorylated MEK (ppMEK) and double phosphorylated Erk (ppERK) every minute until 10th minute are used for unmixing. The current model offered two different signaling response, one for EGF and another for NGF treatment. The mixed simulation data from the two different responses is used to demonstrate unmixing of two component system. Additionally, a SBML model parameter (J136_k = 0.015) is modified, to depict a mutant (defective) MEK kinase with reduced activity resulting in reduced ERK activation. The modified model yielded another two different signaling response to EGF and NGF stimulation. The mixed simulation data from original and modified model is used demonstrate the unmixing of four component system.

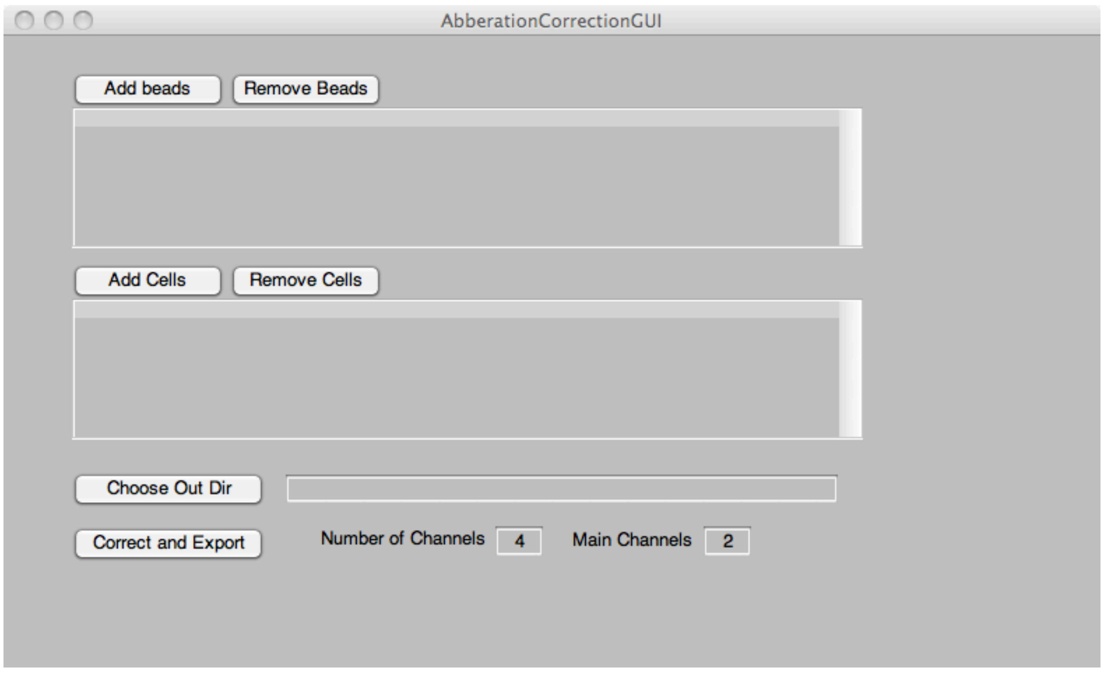
Chapter 8

Appendix: Computational tools development

8.1 Image processing tools

8.1.1 Aberration correction

- Run Matlab GUI “AbberationCorectionGUI.fig” in Matlab
- Press “Add beads” and select images corresponding to the cell that be corrected
- Press “Add cell” and select hyperstack (TIFF or lsm) of image of cell
- Select the output directory
- Click on “Correct and Export”
- The tiff image series of the corrected images are exported to selected output directory



Number of Channels 4
Main Channels 2

Figure 8.1: Chromatic aberration correction tool -

```
function corrabbsimple(beads,chn,mainchn, indirk,outdir)
pluginsrc = 'java -Xmx1000m -cp /Applications/MBF_ImageJ/ij.jar: ...
    /Applications/MBF_ImageJ/plugins/bUnwarpJ_.jar bunwarpj.bUnwarpJ_ ';
mkdir('tempfiles');
fnames = fullfile(beads,'*.tif');
lst   = dir(fnames);
nn = size(lst,1);
noimg = nn/chn;
imgn = reshape({ lst.name },chn, noimg)';
for ia = noimg:-1:1
    for ib = chn:-1:1
        img(:,:,ib,ia) = imread(fullfile(beads,imgn{ia,ib}));
    end
end
k = img(:,:,1,1);
simg  =  uint32(zeros([size(k) chn]));
img = uint32(img);
for ia = 1:chn
    for ib = 1:noimg
        simg(:,:,ia) = simg(:,:,ia) +  img(:,:,ia,ib);
    end
end
for ib = chn:-1:1
    tif32write(simg(:,:,ib),['tempfiles/tempSourceImg0' num2str(ib) '.tif']);
```

```
end
otherchn = setdiff(1:chn,mainchn);
targetImg = ['tempfiles/tempSourceImg0' num2str(mainchn) '.tif'];
cimg(:,:,mainchn) = im2double(imread(targetImg));
for ib = otherchn
    sourceImg = ['tempfiles/tempSourceImg0' num2str(ib) '.tif'];
    transfname = ['tempfiles/_transf_field_ch'  num2str(ib) '.txt'];
    func = ['—align ' targetImg ' NULL ' sourceImg ' NULL 0 3 0 0 0 1 10   ...
        tempfiles/_tempout.tif notrequired —mono'];
    system([ pluginsrc func]);
    cimg(:,:,ib) = im2double(imread('tempfiles/_tempout.tif'));
    java.io.File('tempfiles/_tempout_transf.txt').renameTo(java.io.File( ...
        transfname));
end
if(isequal(indirk,''))
    indirk = uigetdir('','Choose the folder of Data') ;
elseif (¬(ischar(indirk)))
    warndlg('Enter the correct pathway as string','Invalid Input','Replace');
elseif(¬exist(indirk,'dir'))
    warndlg('Incorrect directory. Try again','Invalid Path','Replace');
end
fnames = fullfile(indirk,'*.tif');
lst  = dir(fnames);
nn = size(lst,1);
noimg = nn/chn;
imgn = reshape({ lst.name },chn, noimg)';
if(¬exist(outdir,'dir'))
    mkdir(outdir);
end
for ii = 1:noimg
    targetImg = fullfile(indirk,imgn{ii,mainchn});
    dtargetimg = imread(targetImg); %display taget image
    figure(234);
    subplot(1,chn,mainchn);
    imagesc(dtargetimg);
    axis image
    tif32write(uint32(dtargetimg), fullfile(outdir, imgn{ii,mainchn}));
    for jj = otherchn
        sourceImg = fullfile(indirk,imgn{ii,jj});
        [ ¬,nm] = fileparts(imgn{ii,jj});
        out = fullfile(outdir,[nm '_AC.tif']);
        transf = ['tempfiles/_transf_field_ch'  num2str(jj) '.txt'];
        func = ['—elastic_transform ' targetImg ' ' sourceImg ' '  transf ' ' ...
            out];
        system([ pluginsrc  func]);
        outimg = imread(out);
        figure(234);
        subplot(1,chn,jj);
        imagesc(outimg);
        axis image
    end
```

```
end
rmdir('tempfiles','s');
```

```
function varargout = AbberationCorrectionGUI(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @AbberationCorrectionGUI_OpeningFcn, ...
    'gui_OutputFcn',  @AbberationCorrectionGUI_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function AbberationCorrectionGUI_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = AbberationCorrectionGUI_OutputFcn(hObject, eventdata, ...
    handles)
varargout{1} = handles.output;
function add_beads_Callback(hObject, eventdata, handles)
[fname,fpathname] = ...
    uigetfile({'*.tif;*.lsm','ImageFiles(*.tif,*.lsm)';'*.tif','*.Tiff'; ...
    '*.lsm','*.LSM'},'Choose Beads Images', 'MultiSelect','on','');
if(¬isequal(fname,0))
    List = get(handles.beads_list, 'string');
    if(iscell(fname))
        for ia = 1:length(fname)
            List{end+1} =  fullfile(fpathname,fname{ia});
        end
    else
        List{end+1} = fullfile(fpathname,fname);
    end
    set(handles.beads_list, 'string', List);
end
function remove_beads_Callback(hObject, eventdata, handles)
currentItems = get(handles.beads_list, 'String');
rowToDelete = get(handles.beads_list, 'value');
set(handles.beads_list, 'Value', []);
newItems = currentItems;
newItems(rowToDelete) = [];
set(handles.beads_list, 'String', newItems);
set(handles.beads_list, 'Value', 1);
function beads_list_Callback(hObject, eventdata, handles)
function beads_list_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function add_cells_Callback(hObject, eventdata, handles)
[fname,fpathname] = ...
    uigetfile({'*.tif;*.lsm','ImageFiles(*.tif,*.lsm)';'*.tif','*.Tiff'; ...
    '*.lsm','*.LSM'},'Choose Beads Images', 'MultiSelect','on','');
if(¬isequal(fname,0))
    List = get(handles.cells_list, 'string');
    if(iscell(fname))
        for ia = 1:length(fname)
            List{end+1} =  fullfile(fpathname,fname{ia});
        end
    else
        List{end+1} = fullfile(fpathname,fname);
    end
    set(handles.cells_list, 'string', List);
end
set(handles.outdirk,'String',[ fpathname(1:end−1) 'AC']);
function remove_cells_Callback(hObject, eventdata, handles)
currentItems = get(handles.cells_list, 'String');
rowToDelete = get(handles.cells_list, 'value');
set(handles.cells_list, 'Value', []);
newItems = currentItems;
newItems(rowToDelete) = [];
set(handles.cells_list, 'String', newItems);
set(handles.cells_list, 'Value', 1);
function cells_list_Callback(hObject, eventdata, handles)
function cells_list_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function outdir_Callback(hObject, eventdata, handles)
pat = get(handles.outdirk,'String');
if(¬exist(pat,'dir'))
    pat = fileparts(pat);
end
handles.outpath = uigetdir(pat);
set(handles.outdirk,'String',handles.outpath);
function outdirk_Callback(hObject, eventdata, handles)
function outdirk_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function corr_exp_Callback(hObject, eventdata, handles)
beads_list = get(handles.beads_list, 'String');
cells_list = get(handles.cells_list, 'String');
beadspath = '_tmp/Beadsfiles';
```

```
cellspath = '_tmp/Cellfiles';
mkdir(beadspath);
mkdir(cellspath);
nochn = str2num(get(handles.nochannels,'String'));
mchn = str2num(get(handles.mainchannel,'String'));
localsave(beads_list,beadspath,nochn);
localsave(cells_list,cellspath,nochn);
outdirk = get(handles.outdirk,'String');
corrabbsimple(beadspath,nochn,mchn,cellspath,outdirk);
rmdir(beadspath,'S');
rmdir(cellspath,'S');
function localsave(list,tpath,nochn)
nob = size(list,1);
for ia = 1:nob
    beadsorcell = tiffread32(list{ia});
    [pat,nam,ext] = fileparts(list{ia});
    if(isfield(beadsorcell,'lsm'))
        nochn = beadsorcell(1).lsm.DimensionChannels;
        nt = beadsorcell(1).lsm.DimensionTime;
        if(nt > 1)
            msgbox(sprintf('check file %s. Only first time point ...
                considered',nam));
        end
        for ib = 1:nt
            for ic = 1:nochn
                tif32write(uint32(beadsorcell(ib).data{ic}), ...
                    fullfile(tpath,sprintf('t%03d_%s_C00%d.tif',ib,nam,ic)));
            end
        end
    else
        nt = size(beadsorcell,2)/nochn;
        idx = 1;
        for ib = 1:nt
            for ic = 1:nochn
                tif32write(uint32(beadsorcell(idx).data), ...
                    fullfile(tpath,sprintf('t%03d_%s_C00%d.tif',ib,nam,ic)));
                idx = idx + 1;
            end
        end
    end
end
function nochannels_Callback(hObject, eventdata, handles)
function nochannels_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function mainchannel_Callback(hObject, eventdata, handles)
function mainchannel_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

8.1.2 Background subtraction

- Import the aberration corrected image sequence in ImageJ
- Convert stack into hyperstack according to the number of channels.

  To estimate the background, draw a polygon in the region outside the cell for each channel.
- For each time frame, estimate the mean background intensity from the polygon and subtract it from the entire image.
- Save the hyperstack of images for next step, bleach correction .

8.1.3 Bleaching correction

- Draw polygon outlining the cell using “TrackCell.fig” tool.
- Correct bleaching in image series for selected or all channels by using the function “bleachCorrByRatio()” in Matlab

```
function bleachCorrByRatio(indirk,dat,name_chn)
if ¬exist('name_chn','var')
    name_chn = {'c001','c002','c003','c004'};
end
outdirk = [indirk 'BC'];
mkdir(outdirk);
for ib = 1:size(name_chn,2)
    [img, imgn] = imreadseries(indirk,name_chn(ib));
    if(exist('dat','var'))
        bleachCurve = ROImean4stack(img,dat); %use mask of cell based on the ...
            struct dat
    else
        bleachCurve = reshape(mean(mean(img,1),2),1,[]); %Calculate mean ...
            without mask
    end
    CorrFactor = bleachCurve./bleachCurve(1);
    for ia = 1:size(img,3)
        tif32write(img(:,:,ia) ./CorrFactor(ia) ,fullfile(outdirk,imgn{ia}));
    end
end
```

```
function bleachcurve = ROImean4stack(img,dat)
disdat = cell2mat({dat(:).framept}');
[m,n,noimg] = size(img);
mask = false(size(img));
for ii = 1:noimg
    [tempmask,k] = fetchmask(dat,disdat,m,n,ii);
    if(k)
        mask(:,:,ii) = reshape(tempmask,m,n);
    end
end
fimg = img;
fimg(¬mask) = NaN;
for ia = noimg:−1:1;
    timg = fimg(:,:,ia);
    bleachcurve(ia) = nanmean(timg(:));
end
function [mask,k] = fetchmask(dat, disdat, m,n,num)
k = find(disdat(:,1)≤num & disdat(:,2)≥num);
if(k)
    mask = roipoly(m,n, dat(k).polyg(:,1), dat(k).polyg(:,2));
    mask = double(reshape(mask,m*n,1));
else
    mask = 0;
end
```

8.1.4 Cell outlining tool

- Run Matlab GUI “TrackCell.fig”
- Note: The cell required to be bleach corrected should be saved as image series
- click on “Image series” button in “Load Data” panel. Load only images from single channel.
- click on the option “Draw and Add polygon” and and draw a polygon around the cell of interest. Check the image frames one by one by sliding timeslider. When the cell moves out of the polygon, click on “Edit and Add polygon” button and edit the polygon to fit the cell. Check the polygon on all the frames and edit if required.
- To draw a new polygon after certain time frame instead of editing the previous one, click “End point” and then click “Draw and Addpolygon”.
- Click on “End point” at the last frame.

- Table showing the starting and ending frame number for each polygon can also be manually edited if required.
- Export the data containing the polygons and corresponding frames by clicking “Export Data”.

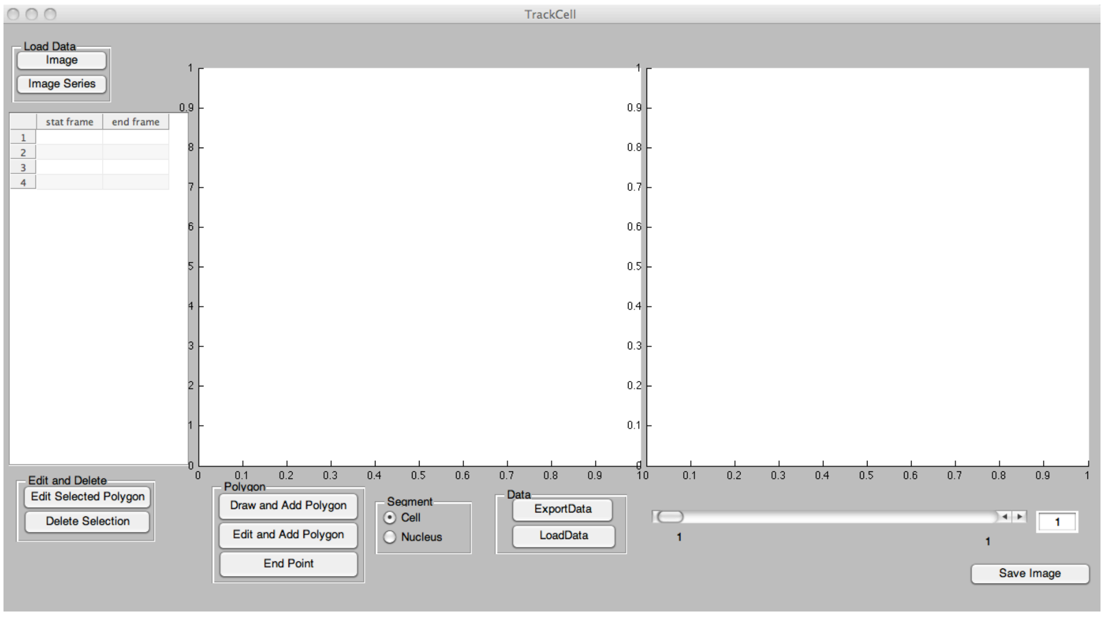


Figure 8.2: Cell outlining tool: TrackCell -

```
function varargout = TrackCell(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @TrackCell_OpeningFcn, ...
                   'gui_OutputFcn',  @TrackCell_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function TrackCell_OpeningFcn(hObject, eventdata, handles, varargin)
handles.tmax = 1;
handles.disdat = [];
handles.sel = 1;
handles.flag =1;
colormap(gray);
handles.flnm = '';
handles.flnm2 = '';
tmp = dir('trackcell_log.mat');
if(size(tmp,1))
    load('trackcell_log.mat');
    handles.flnm = tmp{1};
    handles.flnm2 = tmp{2};
end
handles.output = hObject;
guidata(hObject, handles);
function varargout = TrackCell_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function tslider_Callback(hObject, eventdata, handles)
sliderValue = round(get(handles.tslider,'Value'));
set(handles.timept,'String', num2str(sliderValue));
handles.img = imread(fullfile(handles.flnm, handles.imgn{sliderValue}));
if(isfield(handles,'dat'))
    if size(handles.dat,2) > 1
    num = find(handles.disdat(:,1)≤sliderValue & ...
        handles.disdat(:,2)≥sliderValue ); %make sure if its r8
    else
        num = 1;
    end
    if(num)
        pos = handles.dat(num(1)).polyg;
        disp_images(hObject,handles,pos);
        handles = guidata(hObject);
        guidata(hObject, handles);
    end
end
function timept_Callback(hObject, eventdata, handles)
tpt = str2num(get(hObject, 'String'));
if (isempty(tpt) || tpt < 0 || tpt > handles.tmax)
    tpt = 1;
end
set(hObject,'String', round(tpt));
set(handles.tslider,'Value',tpt);
tslider_Callback(hObject, eventdata, handles)
function loadimg_Callback(hObject, eventdata, handles)
[filename, handles.flnm] = uigetfile('*.tif', 'Choose a TIFF image',handles.flnm);
if(filename)
removeold(hObject,handles);
handles.tmax = 1.1;
handles.imgn = fullfile(handles.flnm,filename);
handles.img = imread(handles.imgn);
init_image(hObject,handles);
handles = guidata(hObject);
```

```
guidata(hObject, handles);
end
function loadimgseries_Callback(hObject, eventdata, handles)
[¬, handles.flnm] = uigetfile('*.tif', 'Choose a TIFF image from the Image ...
    Series',...
    handles.flnm,'MultiSelect', 'on');
if(handles.flnm)
removeold(hObject,handles);
handles = guidata(hObject);
lst = dir([handles.flnm '*.tif']);
    nn = size(lst,1);
    prompt={[ num2str(nn) ' images found.',...
        ' Channel name containing:']};
    name='Input for Channel';
    numlines=1;
    defaultanswer={''};
    name_chn = inputdlg(prompt,name,numlines,defaultanswer);
    if(¬isempty(name_chn))
    if(strcmp(name_chn{1} , ''))
        handles.imgn = reshape({ lst.name },1, size(lst,1))';
    else
        imgn = reshape({ lst.name },1, size(lst,1))';
        ind = ¬logical(cellfun('isempty',strfind(imgn,name_chn{1})));
        handles.imgn = imgn(ind);
    end
    if(size(handles.imgn,1))
        handles.tmax = size(handles.imgn,1);
        handles.img = imread(fullfile(handles.flnm, handles.imgn{1}));
        init_image(hObject,handles);
        handles = guidata(hObject);
        msgbox(sprintf('%d Images are read',size(handles.imgn,1)),'Images ...
            found','replace');
    else
        msgbox('Zero Images are read','No Images found','warn','replace');
    end
    guidata(hObject, handles);
    end
end
function init_image(hObject,handles)
set(handles.tslider,'Value',1);
set(handles.tslider,'Max', handles.tmax);
set(handles.tslider,'SliderStep', [1/handles.tmax 0.1]);
axes(handles.axes1);
handles.fig = imagesc(handles.img);
guidata(hObject, handles);
function GetPolygon_Callback(hObject, eventdata, handles)
axes(handles.axes1);
h = impoly;
pos = h.getPosition;
if(isfield(handles,'dat'))
    handles.dat(end+1).polyg = pos;
```

```
else
    handles.dat = struct();
handles.dat(end).polyg = pos;
end
handles.dat(end).framept(1) = str2num(get(handles.timept,'String'));
n = size(handles.dat,2);
handles.disdat(n,:) = [handles.dat(end).framept(1), handles.tmax];
set(handles.uitable,'Data',handles.disdat);
disp_images(hObject,handles,pos);
handles = guidata(hObject);
guidata(hObject, handles);
function disp_images(hObject, handles,pos)
BW = roipoly(handles.img,pos(:,1),pos(:,2));
if(get(handles.nucleus,'Value'))
    BW = ¬BW;
end
perim = bwperim(BW);
axes(handles.axes1);
img = handles.img;
img(perim) = max(handles.img(:));
imagesc(img);
axes(handles.axes2);
handles.dimg = double(handles.img).* double(BW);
handles.dimg(perim) = max(handles.img(:));
imagesc(handles.dimg);
guidata(hObject, handles);
function removeold(hObject,handles)
if(isfield(handles,'img'))
handles = rmfield(handles, {'imgn','img'});
if(isfield(handles,'dat'))
handles = rmfield(handles, {'dat','disdat'});
set(handles.uitable,'Data','');
end
end
guidata(hObject, handles);
function uitable_CellEditCallback(hObject, eventdata, handles)
handles.disdat = get(hObject,'Data');
r = eventdata.Indices(1);
c = eventdata.Indices(2);
handles.dat(r).framept(c) = handles.disdat(r,c);
guidata(hObject, handles);
function eGetPolygon_Callback(hObject, eventdata, handles)
handles.dat(end).framept(2) = str2num(get(handles.timept,'String'))−1;
handles.disdat(end,2) = handles.dat(end).framept(2);
axes(handles.axes1);
h = impoly(gca,handles.dat(end).polyg);
pos = h.wait;
handles.dat(end+1).polyg = pos;
handles.dat(end).framept = [handles.dat(end−1).framept(2)+1 handles.tmax];
handles.disdat(end+1,:) = [handles.dat(end).framept(1) handles.tmax];
set(handles.uitable,'Data',handles.disdat);
```

```
disp_images(hObject,handles,pos);
handles = guidata(hObject);
guidata(hObject, handles);
function endPoint_Callback(hObject, eventdata, handles)
handles.dat(end).framept(2) = str2num(get(handles.timept,'String'));
handles.disdat(end,2)= handles.dat(end).framept(2);
set(handles.uitable,'Data',handles.disdat);
guidata(hObject, handles);
function export_dat_Callback(hObject, eventdata, handles)
[filename, handles.flnm2] = uiputfile( {'*.mat'},'Save Data as',handles.flnm2);
dat = handles.dat;
save(fullfile(handles.flnm2,filename),'dat');
function uitable_CellSelectionCallback(hObject, eventdata, handles)
if(handles.flag)
    if(size(eventdata.Indices))
handles.sel = eventdata.Indices(1);
    end
end
handles.flag = 1;
guidata(hObject, handles);
function delsel_Callback(hObject, eventdata, handles)
handles.disdat(handles.sel,:) = [];
handles.dat(handles.sel) = [];
handles.flag = 0;
set(handles.uitable,'Data',handles.disdat);
guidata(hObject, handles);
function loaddat_Callback(hObject, eventdata, handles)
[filename, handles.flnm2] = uigetfile('*.mat', 'Choose Matlab Polygon ...
    Data',handles.flnm2);
if(filename)
load (fullfile(handles.flnm2,filename));
handles.dat = dat;
handles.disdat = cell2mat({dat(:).framept}');
set(handles.uitable,'Data',handles.disdat);
guidata(hObject, handles);
end
function figure1_CloseRequestFcn(hObject, eventdata, handles)
tmp{1} = handles.flnm;
tmp{2} = handles.flnm2;
save('trackcell_log.mat','tmp');
delete(hObject);
function EditPoly_Callback(hObject, eventdata, handles)
axes(handles.axes1);
h = impoly(gca,handles.dat(handles.sel).polyg);
pos = h.wait;
handles.dat(handles.sel).polyg = pos;
guidata(hObject, handles);
function saveimg_Callback(hObject, eventdata, handles)
[filename, handles.flnm2] = uiputfile( {'*.tif'},'Save Image as',handles.flnm2);
if(filename)
imwrite(handles.dimg,jet,fullfile(handles.flnm2,filename),'tif');
```

```
guidata(hObject, handles);
end
```

8.1.5 Image filteration tools

- High-pass filtration: Run the Matlab GUI “$ImgProcTool\_S''$

- Load the images series of one Channel at a time.

- Type the box size in pixels for filtration of the image. The box size is number of pixels corresponding to $6/mum$ (approximate length of large focal adhesion).

- Click the check the box on right corner of the thresholded image to visualize the applied high-pass filter and threshold.

- Slide through the slide bar or type the value of primary threshold. Iterate over different values to find optimal threshold.

- Click “Secondary threshold” button to iteratively choose the value for secondary threshold.

- Slide through the time slider visualize the application of filtration on different time frames.

- After finalizing the box size and primary and secondary threshold values, Click “Export Images” button, to apply the final parameters on all the images and export image series.

- To reduce temporal high-frequency noise apply low-pass filtration kalman4stack on time-lapse image stack of each channel To reduce spatial high-frequency noise apply low-pass filtration smoothn Garcia (2010) on individual images

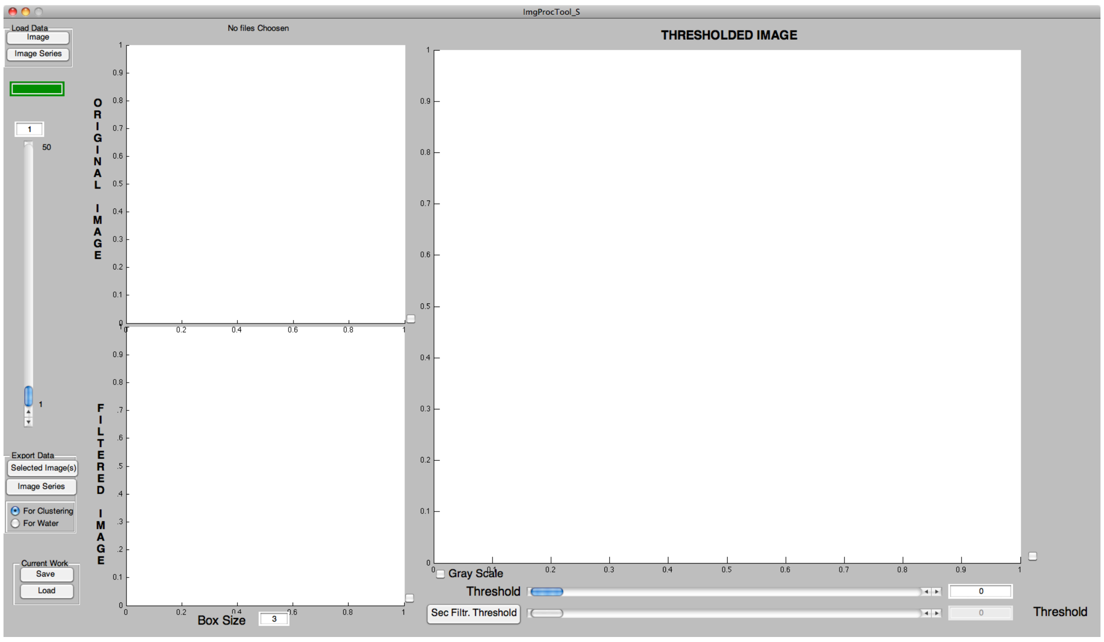


Figure 8.3: Image processing tool - HighPass filtration -

```
function pic3  = filterimg(pic1,w)
tot_pixels = w * w;
[m n] = size(pic1);
cw = floor(w/2);
npic1 = nan(size(pic1) + 2*cw);
npic1(1+cw:m+cw,1+cw:n+cw) = pic1;
dim = size(npic1);
m = dim(1);
n = dim(2);
H = fspecial('average', [w w]);
filt = imfilter(npic1,H,0);
pic3 = npic1 − filt;
edgi1 = [1       (m−w−cw)   1    m−cw−w];
edgi2 = [cw           m−w+1   cw   n−w+1];
edgj1 = [1        1          cw+1    cw+1];
edgj2 = [n−w+1   n−w+1   m−cw−w+1 m−cw−w+1];
for z = 1:4
        for i = edgi1(z):edgi2(z)
            sum1 = zeros(1,n);
            tot_pixels = zeros(1,n);
            j = edgj1(z);
            subpxl = npic1(i:i+w−1,j:j+w−1);
            sum1(edgj1(z)) = nansum(subpxl(:));
            tot_pixels(edgj1(z)) = sum(¬isnan(subpxl(:)));
            irange=i:i+w−1;
            smvec=nansum(npic1(irange,1:n),1);
            totpvec = sum(¬isnan(npic1(irange,1:n)),1);
            icw=i+cw;
            for j = edgj1(z)+1:edgj2(z)
                sum1(j) =  sum1(j−1) − smvec(j−1) + smvec(j+w−1);
                tot_pixels(j) = tot_pixels(j−1) − totpvec(j−1) + totpvec(j+w−1);
            end
            j = edgj1(z):edgj2(z);
            pic3(icw,j+cw) = npic1(icw,j+cw)−sum1(j)./tot_pixels(j);
        end
        if (z==2)
            pic3 = pic3';
            npic1 = npic1';
            n = m;
        end
end
 pic3 = pic3';
[m n] = size(pic1);
pic3 = pic3(1+cw:m+cw,1+cw:n+cw);
```

```
function pic3  = secfilterimg(pic1,mask,w)
[m n] = size(pic1);
cw = floor(w/2);
npic1 = nan(size(pic1) + 2*cw);
npic1(1+cw:m+cw,1+cw:n+cw) = pic1;
newmask = false(size(mask) + 2*cw);
newmask(1+cw:m+cw,1+cw:n+cw) = mask;
dim = size(npic1);
m = dim(1);
n = dim(2);
pic3 = npic1;
mpic = npic1;
mpic(newmask) = NaN;
for i = 1:m−w+1
    j = 1;
    subpxl = mpic(i:i+w−1,j:j+w−1);
    sum1 = zeros(1,n);
    tot_pixels = zeros(1,n);
    sum1(1) = nansum(subpxl(:));
    tot_pixels(1) = sum(¬isnan(subpxl(:)));
    avg = sum1/tot_pixels;
    pic3(i+cw,j+cw) = npic1(i+cw,j+cw) − avg;
    irange=i:i+w−1;
    smvec = nansum(mpic(irange,1:n),1);
    totpvec = sum(¬isnan(mpic(irange,1:n)),1);
    icw=i+cw;
    sum1(n−w+1) = 0;
    tot_pixels(n−w+1) = 0;
    for j = 2:n−w+1
        sum1(j) =  sum1(j−1) − smvec(j−1) + smvec(j+w−1);
        tot_pixels(j) = tot_pixels(j-1) - totpvec(j-1) + totpvec(j+w-1);
    end
    j = 2:n-w+1;
    pic3(icw,j+cw) = npic1(icw,j+cw)-sum1(j)./tot_pixels(j);
end
[m n] = size(pic1);
pic3 = pic3(1+cw:m+cw,1+cw:n+cw);
```

```
function varargout = ImgProcTool_S(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @ImgProcTool_S_OpeningFcn, ...
    'gui_OutputFcn',  @ImgProcTool_S_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function ImgProcTool_S_OpeningFcn(hObject, eventdata, handles, varargin)
handles.tmax = 1;
handles.thdmax = 100;
handles.nch = 1;
enabling(handles);
handles.flag = 1;
handles.filterKernel = fspecial('average', [3 3]);
handles.flnm = '';
handles.flnm2 = '';
handles.logfile = 'imgproc_log.mat';
tmp = dir(handles.logfile);
if(size(tmp,1))
    load(handles.logfile);
    handles.flnm = tmp{1};
    handles.flnm2 = tmp{2};
end
handles.output = hObject;
guidata(hObject, handles);
function varargout = ImgProcTool_S_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function enabling(handles)
set(handles.time_slider,'Enable','on');
cn = '12345';
for ia = 1%:handles.nch
    eval(['set(handles.oickbox' cn(ia) ' , ''Enable'', ''on'');']);
    eval(['set(handles.fickbox' cn(ia) ' , ''Enable'', ''on'');']);
    eval(['set(handles.tickbox' cn(ia) ' , ''Enable'', ''on'');']);
    eval(['set(handles.bsize' cn(ia) ' , ''Enable'', ''on'');']);
    eval(['set(handles.thd' cn(ia) ' , ''Enable'', ''on'');']);
    eval(['set(handles.thdslider' cn(ia) ' , ''Enable'', ''on'');']);
end
for ia = 1
    eval(['set(handles.oickbox' cn(ia) ' , ''Value'', 0);']);
    eval(['set(handles.fickbox' cn(ia) ' , ''Value'', 0);']);
    eval(['set(handles.tickbox' cn(ia) ' , ''Value'', 0);']);
    eval(['cla( handles.oiaxes' cn(ia) ',''reset'' )']);
    eval(['cla( handles.fiaxes' cn(ia) ', ''reset'')']);
    eval(['cla( handles.tiaxes' cn(ia) ', ''reset'')']);
    eval(['handles.oi' cn(ia) ' =0; handles.fi' cn(ia) ' = 0; handles.ti' ...
        cn(ia) ' = 0;']);
end
function init_image(hObject,handles)
enabling(handles);
colormap(gray);
cn = '12345';
for ia = 1%:handles.nch
    oimage(hObject,handles, cn(ia),1)
    handles = guidata(hObject); %dont know if required
end
set(handles.time_slider,'Value',1);
set(handles.time_slider,'Max',handles.tmax);
set(handles.maxt,'String',num2str(handles.tmax));
handles.thdmax = max(max(max(handles.img)))/2; %set limit for threshold
for ia = 1
    eval(['set(handles.thdslider' cn(ia) ',''Value'', 0);']);
    eval(['set(handles.thdslider' cn(ia) ',''Max'', handles.thdmax);']);
end
set(handles.secthd_slider,'Value',0);
set(handles.secthd_slider,'Max',handles.thdmax);
guidata(hObject, handles);
function loadimg_tb1_Callback(hObject, eventdata, handles)
[filename, handles.flnm] = uigetfile({'*.tif';'*.lsm'}, 'Choose a TIFF image', ...
    handles.flnm);
if(handles.flnm)
    handles.tmax = 1.1;
    removeold(hObject,handles);
    handles = guidata(hObject);
    handles.imgn{1,1} = filename;
    init_image(hObject,handles);
    handles = guidata(hObject);
    guidata(hObject, handles);
end
function loadimg_tb2_Callback(hObject, eventdata, handles)
[filename, handles.flnm, FilterIndex] = uigetfile({'*.tif';'*.lsm'}, 'Choose a ...
    TIFF image from the Image Series',handles.flnm,'MultiSelect', 'on');
if(handles.flnm)
    removeold(hObject,handles);
    handles = guidata(hObject);
    if(FilterIndex == 1 || FilterIndex == 3)
    lst = dir([handles.flnm '*.tif']);
    else
        lst = dir([handles.flnm '*.lsm']);
    end
    nn = size(lst,1);
    prompt={[ num2str(nn) ' images found.',...
        ' Channel name containing:']};
    name='Input for Channel';
    numlines=1;
    defaultanswer={''};
    name_chn = inputdlg(prompt,name,numlines,defaultanswer);
    if(¬isempty(name_chn))
        if(strcmp(name_chn{1} , ''))
            handles.imgn = reshape({ lst.name },1, size(lst,1))';
        else
            imgn = reshape({ lst.name },1, size(lst,1))';
            ind = ¬logical(cellfun('isempty',strfind(imgn,name_chn{1})));
            handles.imgn = imgn(ind);
        end
        if(size(handles.imgn,1))
            handles.tmax = size(handles.imgn,1);
            init_image(hObject,handles);
            handles = guidata(hObject);
            msgbox(sprintf('%d Images are read',size(handles.imgn,1)), 'Images ...
                found','replace');
        else
            msgbox('Zero Images are read','No Images found','warn','replace');
        end
        guidata(hObject, handles);
    end
end
function removeold(hObject,handles)
if(isfield(handles,'imgn'))
    handles = rmfield(handles, 'imgn');
    if(isfield(handles,'img'))
        handles = rmfield(handles,{'img'});
    end
    if(isfield(handles,'fimg'))
        handles = rmfield(handles,{'fimg'});
    end
end
guidata(hObject, handles);
function time_slider_Callback(hObject, eventdata, handles)
sliderValue = round(get(handles.time_slider,'Value'));
set(handles.timept,'String', num2str(sliderValue));
nm = '1'; %nm ='12345';
for ia = 1:handles.nch
    eval(['if(get(handles.oickbox' nm(ia) ','''Value''))' ...
        'oimage(hObject,handles, nm(ia),sliderValue);'...
        'handles = guidata(hObject);'...
        'end']);
    eval([ 'if(get(handles.fickbox'  nm(ia) ',''Value''))'...
        'fimage(hObject,handles, nm(ia));'...
        'handles = guidata(hObject);'...
        'end']);
    if(get(handles.tickbox1,'Value'))
        thdv = get(handles.thdslider1, 'Value');
        timage(hObject,handles, nm(ia) ,thdv);
    if(get(handles.secfilt,'Value'))
        handles.secfimg = secfilterimg(handles.img(:,:,1), handles.fimg ...
            >round(thdv), str2double(get(handles.bsize1,'String')));
        secthdv = get(handles.secthd_slider, 'Value');
        secfimage(hObject,handles,secthdv);
    end
    end
end
guidata(hObject,handles);
function timept_Callback(hObject, eventdata, handles)
tpt = str2num(get(hObject, 'String'));
if (isempty(tpt) || tpt ≤ 0 || tpt > handles.tmax)
    tpt = 1;
end
set(hObject,'String', round(tpt));
set(handles.time_slider,'Value',tpt);
time_slider_Callback(hObject, eventdata, handles)
function bsize_Callback(hObject, eventdata, handles)
bz = get(hObject,'Tag');
eval(['set(handles.fickbox' bz(end) ',''Value'',1);']);
fimage(hObject,handles, bz(end));
handles = guidata(hObject);
guidata(hObject, handles);
function oickbox_Callback(hObject, eventdata, handles)
val = get(hObject,'Value');
if(val)
    sldv = round(get(handles.time_slider,'Value'));
    oimage(hObject,handles,'1',sldv)  %oimage(hObject,handles,cb(end),sldv);
    handles = guidata(hObject);
end
guidata(hObject, handles);
function oimage(hObject,handles, cb,sldv)
if( strcmp(handles.imgn{sldv,str2num(cb)}(end-2:end),'lsm'))
    img = imread( fullfile(handles.flnm ,  handles.imgn{sldv,str2num(cb)}));
    img = img(:,:,1);
    handles.img(:,:, str2double(cb)) = imfilter(img, ...
        handles.filterKernel,'symmetric');
    handles.lsminf = lsminfo(fullfile(handles.flnm ,  ...
        handles.imgn{sldv,str2num(cb)}));
else
handles.img(:,:, str2double(cb)) = imfilter(imread( fullfile(handles.flnm ,  ...
    handles.imgn{sldv,str2num(cb)})), handles.filterKernel,'symmetric');
```

```
end
ax = 'axes(handles.oiaxes1)';
ax(end−1) = cb;
eval(ax);
imagesc(handles.img(:,:,  str2num(cb) ));
axis off image
set(handles.flname,'String',handles.imgn{sldv});
guidata(hObject,handles);
function fickbox_Callback(hObject, eventdata, handles)
cb = get(hObject,'Tag');
val = get(hObject,'Value');
if(val)
    fimage(hObject,handles, cb(end));
    handles = guidata(hObject);
end
guidata(hObject,handles);
function fimage(hObject,handles, cb)
eval(['if(¬ get(handles.oickbox' cb ',''Value'')),'...
    'set(handles.oickbox' cb ',''Value'',1); '...
    'sldv = round(get(handles.time_slider,''Value''));'...
    'oimage(hObject,handles,cb,sldv);'...
    'handles = guidata(hObject);'...
    'end']);
ax = 'axes(handles.fiaxes1)';
ax(end−1) = cb;
eval(ax);
eval(['k = str2double(get(handles.bsize' cb(end) ',''String''));']);
nn = str2num(cb(end));
set(handles.time_slider,'Enable','off');
set(handles.statusbar,'String','Please Wait!!!','Backgroundcolor',[1 0 0]);
handles.fimg(:,:,nn) = filterimg( handles.img(:,:,nn),k);
imagesc(handles.fimg(:,:,nn));
axis off image
set(handles.time_slider,'Enable','on');
set(handles.statusbar,'String','Done!!','Backgroundcolor',[0 1 0]);
if(get(handles.tickbox1,'Value'))
    timage(hObject,handles,cb,get(handles.thdslider1,'Value'));
end
guidata(hObject, handles);
function tickbox_Callback(hObject, eventdata, handles)
cb = get(hObject,'Tag');
val = get(hObject,'Value');
eval(['thdv = str2double(get(handles.thd' cb(end) ',''String''));']);
if(val)
    timage(hObject,handles,cb(end),thdv);
    handles = guidata(hObject);
end
guidata(hObject, handles);
function thdslider_Callback(hObject, eventdata, handles)
thdv = round(get(hObject,'Value'));
cb = get(hObject,'Tag');
```

```
eval(['set(handles.thd' cb(end) ',''String'', num2str(thdv));']);
val = get(handles.tickbox1,'Value');
if(val)
    timage(hObject,handles,cb(end),thdv);
    handles = guidata(hObject);
    axis off image
end
guidata(hObject, handles);
function thd_Callback(hObject, eventdata, handles)
cb = get(hObject,'Tag');
thdv = str2num(get(hObject, 'String'));
set(hObject,'String', round(thdv));
if (isempty(thdv) || thdv < 0 || thdv > handles.thdmax)
    thdv = 0;
end
eval(['set(handles.thdslider' cb(end) ',''Value'',thdv);']);
timage(hObject,handles,cb(end),thdv);
handles = guidata(hObject);
guidata(hObject, handles);
function timage(hObject,handles,cbe,thdv)
eval(['if(¬get(handles.fickbox' cbe ',''Value''))'...
    'set(handles.fickbox' cbe ',''Value'',1); '...
    'fimage(hObject,handles,cbe);'...
    'handles = guidata(hObject);'...
    'end']);
ax = 'axes(handles.tiaxes1)';
ax(end−1) = cbe;
eval(ax);
handles.timg = handles.fimg(:,:,str2num(cbe));
handles.timg(handles.timg≤thdv) = 0;
if(get(handles.secfilt,'Value')==0)
    if(get(handles.grayscale,'Value')== 1)
        imagesc(handles.timg);
    else
        imagesc(im2bw(handles.timg,0));
    end
else
    secfilt_Callback(handles.secfilt,'', handles);
    handles = guidata(hObject);
end
axis off image
guidata(hObject, handles);
function secfimage(hObject,handles,secthd)
axes(handles.tiaxes1);
handles.tsecfimg = handles.secfimg;
handles.tsecfimg(handles.tsecfimg≤secthd) = 0;
if(get(handles.secfilt,'Value')≠0)
    if(get(handles.grayscale,'Value')== 1)
        imagesc(handles.tsecfimg);
    else
        imagesc(im2bw(handles.tsecfimg,0));
    end
end
axis off image
guidata(hObject, handles);
function  export_buttons_Callback(hObject,eventdata, handles)
switch get(hObject,'Tag')
    case 'export_imgseries'
        init = 1;
        fin = handles.tmax;
    case 'export_img'
        init = round(get(handles.time_slider,'Value'));
        fin = init;
end
cn = '12345';
handles.flnm2 = uigetdir(handles.flnm2, 'Choose a folder to Save Image Series');
set(handles.bsize1,'Enable','off');
set(handles.time_slider,'Enable','off');
set(handles.statusbar,'String','Please Wait!!!','Backgroundcolor',[1 0 0]);
if(handles.flnm2)
    for ia = 1:handles.nch
        eval([' x = get(handles.tickbox' cn(ia) ',''Value'');']);
        if(x)
          fltbxsize = str2double(get(handles.bsize1 ,'String'));
                pthreshold = str2double(get(handles.thd1,'String'));
            threshold = str2double(get(handles.secthd,'String'));
            if(get(handles.forclus,'Value'))
                [¬,flname,¬] = fileparts( handles.imgn{init,1});
                save(fullfile(handles.flnm2, [flname '.mat']),'threshold');
            end
             suff = [ '_F' num2str(fltbxsize) 'T' num2str(pthreshold)] ;
            if(get(handles.secfilt,'Value'))
                suff = [ suff 'ST' num2str(threshold)];
            end
imgn = handles.imgn;
flnm = handles.flnm;
flnm2 = handles.flnm2;
filterKernel = handles.filterKernel;
yn = get(handles.secfilt,'Value');
            for ib = init:fin
                [¬,flname,ext] = fileparts( imgn{ib,ia});
                img = imfilter(imread(fullfile(flnm , imgn{ib,ia})), ...
                    filterKernel,'symmetric');
                fimg = filterimg(img, fltbxsize);
                if(yn)
                fimg = secfilterimg(img,fimg>threshold,fltbxsize);
                end
                fimg(fimg <0) = 0;
                tif32write(fimg, fullfile(flnm2,[flname suff ext]));
                waitbar(ib./fin);
            end
        end
    end
    guidata(hObject, handles)
end
set(handles.bsize1,'Enable','on');
set(handles.time_slider,'Enable','on');
set(handles.statusbar,'String','Done!!','Backgroundcolor',[0 1 0]);
function save_set_Callback(hObject, eventdata, handles)
[filename,handles.flnm2] = uiputfile('*.fig','Save your GUI settings');
if handles.flnm2 == 0
    return
end
saveDataName = fullfile(handles.flnm2,filename);
hgsave(saveDataName);
function load_set_Callback(hObject, eventdata, handles)
[filename, handles.flnm2] = uigetfile('*.fig', 'Choose the GUI settings file ...
    to load');
if handles.flnm2 == 0
    return
end
loadDataName = fullfile(handles.flnm2,filename);
theCurrentGUI = gcf;
hgload(loadDataName);
close(theCurrentGUI);
function figure1_CloseRequestFcn(hObject, eventdata, handles)
tmp{1} = handles.flnm;
tmp{2} = handles.flnm2;
save(handles.logfile,'tmp');
delete(hObject);
function grayscale_Callback(hObject, eventdata, handles)
thd_Callback(handles.thd1, eventdata, handles)
function forclus_Callback(hObject, eventdata, handles)
function forwater_Callback(hObject, eventdata, handles)
function statusbar_Callback(hObject, eventdata, handles)
function secthd_slider_Callback(hObject, eventdata, handles)
thdv = round(get(hObject,'Value'));
set(handles.secthd,'String', num2str(thdv));
    secfimage(hObject,handles,thdv);
    handles = guidata(hObject);
    axis off image
guidata(hObject, handles);
function secthd_Callback(hObject, eventdata, handles)
secthdv = str2double(get(hObject, 'String'));
set(hObject,'String', round(secthdv));
if (isempty(secthdv) || secthdv < 0 || secthdv > handles.thdmax)
    secthdv = 0;
end
set(handles.secthd_slider,'Value',secthdv);
secfimage(hObject,handles,secthdv);
handles = guidata(hObject);
guidata(hObject, handles);
function secfilt_Callback(hObject, eventdata, handles)
```

```
if(get(hObject,'Value'))
yn = 'on';
if(get(handles.tickbox1,'Value'))
    pthdv = get(handles.thdslider1,'Value');
    handles.secfimg = secfilterimg(handles.img(:,:,1), ...
        handles.fimg>round(pthdv), str2double(get(handles.bsize1, 'String')));
    secfimage(hObject,handles,get(handles.secthd_slider,'Value'));
    handles = guidata(hObject);
    guidata(hObject, handles);
end
else
    yn = 'off';
    thdslider_Callback(handles.thdslider1, eventdata, handles);
end
set(handles.secthd_slider,'Enable',yn);
set(handles.secthd,'Enable',yn);
```

8.1.6 Shift correction

Use the function “shiftcorrection()” in Matlab to correct lateral shift in the timeframes for e.g. shiftcorrection(“ ”,4,2,11) refers “ ” selection the folder through input panel, number of channels (4), reference channels for correcton (2), shift occurred after time frame 11

```
function shiftcorrectionKON(infname,chn,mainchn,ptofchg)
addpath('/Users/mrsheriff/Documents/MATLAB/Jana/Image shift correction');
lst_flnm = '';
tmp = dir('shiftcorrectionKON_log.mat');
if(size(tmp,1))
    load('shiftcorrectionKON_log.mat');
end
if(isequal(infname,''))
    indirk = uigetdir('','Choose the folder of Data') ;
elseif(¬(ischar(infname)))
warndlg('Enter the correct pathway as string','Invalid Input','Replace');
elseif(¬exist(infname,'dir'))
    warndlg('Incorrect Image. Try again','Invalid Path','Replace');
    error('try again');
else
    indirk = infname;
end
mkdir(fullfile(indirk,'Shift_Corrected'));
fnames = fullfile(indirk,'*.tif');
lst  = dir(fnames);
nn = size(lst,1);
noimg = nn/chn;
imgn = reshape({ lst.name },chn, noimg)';
```

```
otherchn = setdiff(chn:-1:1,mainchn);
otherchn = otherchn(end:-1:1);
for ib = chn:-1:1
    for ia = noimg:-1:1
        img{ib}(:,:,ia) = imread(fullfile(indirk,imgn{ia,ib}));
    end
    sc_img{ib}(:,:,1:ptofchg(1)) = img{ib}(:,:,1:ptofchg(1));
end
for ib = chn:-1:1
    for ia = ptofchg(1):-1:1
        imwrite(uint16(sc_img{ib}(:,:,ia)), ...
            fullfile(indirk,['Shift_Corrected/sc_' imgn{ia,ib}]) );
    end
end
for ic = 1:size(ptofchg,2)
    ic
   st = ptofchg(ic)+1;
   if(size(ptofchg,2)>ic)
       fn = ptofchg(ic+1);
   else
       fn = noimg;
   end
    [output Greg] = dftregistration(fft2(sc_img{mainchn}(:,:,st-1)), ...
        fft2(img{mainchn}(:,:,st)),1);
    for ia = st:fn
        ia
        for ib = chn:-1:1
        sc_img{ib}(:,:,ia) = pixelshiftcorrect(output, img{ib}(:,:,ia));
        imwrite(uint16(sc_img{ib}(:,:,ia)), ...
            fullfile(indirk,['Shift_Corrected/sc_' imgn{ia,ib}]) );
        end
    end
end
lst_flnm = indirk;
'done'
save('shiftcorrectionKON_log.mat','lst_flnm');
function src_im = pixelshiftcorrect(output, im )
[l, m] = size(im);
shift_r = round(output(3));
shift_c = round(output(4));
sr_im = zeros(l,m);
src_im = zeros(l,m);
[vec_r1 vec_r2] = findshift(shift_r,l);
sr_im(vec_r1,:) = im(vec_r2,:);
[vec_c1 vec_c2] = findshift(shift_c,m);
src_im(:,vec_c1) = sr_im(:,vec_c2);
function [vec1 vec2] = findshift(shift,sz)
if(shift ≤ 0 )
    vec1 = 1:sz-abs(shift);
    vec2 = (1+ abs(shift)):sz;
else
```

```
    vec1 = (1+abs(shift)):sz;
    vec2 = 1:sz—abs(shift);
end
```

8.2 Compositional imaging of live cell dynamics

Compositional analysis of multicolor images can be achieved using cluster analysis tools. This package allows sophisticated investigation of multi-molecular composition and transition of composition. This tool requires certain organization of the input data, as described below. A folder has to be created with the following contents:

i A subfolder named “Processed_Images” containing multicolor processed image series. The folder should also contain the “.mat” file exported by the HighPass filtration tool. Each image channel requires one “.mat” file. This mat file consists of image threshold value.

ii Output file of CellOutliningTool names as “cell.mat” containing the polygon that outlines the cell along the entire time lapse image series.

iii Output file of CellOutliningTool names as “nucleus.mat” containing the polygon that outlines the nucleus of cell along the entire time lapse image series. Compositional Imaging can be performed by the following steps.

Step 1: Click Choose Folder button to load the folder created as described above containing all required files. The default number of leaves in the dendrogram is set to 200. This parameter can be changed in the edit box before loading the files. Once the data files are loaded, the following operations are performed automatically by the software.

a) All the processed Images are read and pixel intensities are extracted, excluding nuclear and cell background region based on the polygon set in the files “nucleus.mat” and “cell.mat” respectively.

b) The pixels intensities are normalized, large data are binned to reduce the size and the metadata are exported to a subfolder named “NData”. The metadata includes normalized pixels, index of the pixels, bin size of reduced data, dimension of image, etc.

c) The normalized data is clustered by spherical hierarchical clustering algorithm (developed by Hernan) and the linkage matrix is exported as "link.mat" under the chosen head folder.

d) Using the linkage matrix, a dendrogram is generated and displayed. Each and every node and leaf is labeled with an ID (number). This dendrogram is also saved under the same head folder as figure111.fig, so that it can be directly loaded later.

In order to perform all these above mentioned operations, sufficient time is taken. These aforementioned calculations are performed for the first time of loading. Later, when the folder is chosen, the dendrogram is displayed instantly to continue with the further steps of analysis.

Step 2: Type the ID of the nodes/clusters chosen from the dendrogram in the first column of the editable table in the software.

Step 3: Select colors for the chosen nodes by clicking "choose color" button. Colors can be chosen randomly for all nodes or specifically selected nodes by clicking the appropriate radio button. Same color can be chosen for all nodes and the color of node of interest can be specifically changed to highlight and visualize certain cluster. Colors for the clusters can also be chosen by directly typing the value for R,B and G in the respective columns of the editable table in the software.

Step 4: Press "update composition and dendrogram" to color code the nodes in the dendrogram. Additionally, the heatmap of composition of the chosen nodes is also displayed adjacent to cluster ID in table. Bar plot of the composition of the chosen nodes is also displayed. Carefully examine the composition and merge clusters with similar compositions.

Step 5: Press "Generate and Export data" button after checking the radio box "composition images". Compositional images and movies are exported. The abundancy of various composition along time is displayed as line plot and heat map.

Step 6: Press "Compositional Image Movie" button to load the compositional image stack in the Matlab via ImageJ. Carefully, investigate the spatial distribution of the clusters and temporal pattern of the abundance. Merge two spatially mixed clusters that show similar temporal pattern by choosing the parent node. Split large clusters if necessary by choosing two child nodes. Repeat again from step 2 to step 6.

Thoroughly iterate over the cluster/node selection by merging/ splitting to finally identify empirically significant clusters.

Step 7: Export or import the table of cluster selection and color code by clicking "export" or "import" button under the Table panel, respectively.

Step 8: To estimate the transition in composition, select "state Transition data" radiobutton in the "Generate data" panel and press "Generate and Export data" button. This calculation will take a while depending on the size of the data. Press "State Transition Movie" button under "Display" panel to visualize the dynamic compositional transition movie.

Additional Options:

Press "Compositional Space" button under "Display" panel to visualize the distribution of clusters in composition space (all 2D and 3D space).

To exclude noisy clusters, type "1" in the column "Noise?".

The checkbox "Exclude noise from Display" is to remove noisy cluster from the plots of composition space, temporal profile and barplot of composition and transition matrix.

```
function varargout = ClusteringTool(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @ClusteringTool_OpeningFcn, ...
    'gui_OutputFcn',  @ClusteringTool_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function ClusteringTool_OpeningFcn(hObject, eventdata, handles, varargin)
handles.flnm = '';
if(exist('clusteringtool_log.mat','file'))
    load('clusteringtool_log.mat');
    handles.flnm = tmp{1};
```

```
end
paths;
handles.dsel = 1;
handles.nolf = 200;
handles.rbsel2 = 'rb4rand';
handles.figh = 1; %dendrogram figure handle
handles.norows = 15; %no. of rows in the gui table. 15 bcoz hard to choose ...
    more that 15 cluster due to color limitation
handles.output = hObject;
guidata(hObject, handles);
function varargout = ClusteringTool_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function norm_dend(hObject,eventdata,handles)
if(exist([handles.flnm '/Processed_Images'],'dir'))
    if(exist([handles.flnm '/cell.mat'],'file')) %&& ...
        exist(fullfile(handles.flnm,'/Processed_Images/chn.mat'),'file'))
        if(exist([handles.flnm '/NData/FnNormPixelslist_bean.mat'],'file'))
            set(handles.status,'String','Normalized Data Exists');
            set(handles.status,'BackgroundColor',[0 0 0.8]);
        else
            set(handles.status,'String','Data being normalized.. Please ...
                wait.....');
            set(handles.status,'BackgroundColor',[0.8 0 0]);
            msgbox('Pixels normalization started!!!  kindly wait until it is ...
                finished','Notice','replace');
            normpixels4cell([handles.flnm '/']); %for analyzing the movie of a ...
                single cell
            set(handles.status,'String','Data normalization Completed.');
            set(handles.status,'BackgroundColor',[0 0.8 0]);
        end
        load(fullfile(handles.flnm,'chn.mat'));
        handles.chn = chn;
        if(exist([handles.flnm '/link.mat'],'file'))
            set(handles.status,'String','Linkage File Exists');
            set(handles.status,'BackgroundColor',[0 0 0.8]);
            load(fullfile(handles.flnm,'link.mat'));
            handles.link = link;
        else
            set(handles.status,'String','Linkage for clustering is being ...
                carried out.. Please wait.....');
            set(handles.status,'BackgroundColor',[0.8 0 0]);
            msgbox('Linkage Calculation started!!!  kindly wait until it is ...
                finished','Notice','replace');
            [link cord] = hircluster(handles.flnm);
            save(fullfile(handles.flnm,'link.mat'),'link','cord');
            set(handles.status,'String','Linkage Completed');
            set(handles.status,'BackgroundColor',[0 0.8 0]);
            handles.link = link;
        end
        clear link;
        clear cord;
        set(handles.status,'String','Dendrogram being generated. Please ...
            wait.....');
        set(handles.status,'BackgroundColor',[0.8 0 0]);
        load(fullfile(handles.flnm,'NData/FreqFnPixelslist.mat')); %loading freq
        if(isempty(setdiff(handles.figh,findobj('Type','figure'))))
            close(handles.figh);
        end
        flg(1) = exist([handles.flnm '/figure111.fig'],'file');
        flg(2) = 0;
        if(exist(fullfile(handles.flnm,'inicomp.mat'),'file'))
            load(fullfile(handles.flnm,'inicomp.mat'));
            handles.discomp = inicomp;
            if(isequal(size(inicomp), [ handles.nolf, handles.chn]))
                flg(2) = 1;
            end
        end
        if(isequal(flg,[2 1]))
            handles.figh = openfig([handles.flnm '/figure111.fig']);
            curH = findobj(handles.figh,'color','blue');
            load(fullfile(handles.flnm,'dendrogramdata.mat'));
            handles.H=  curH(end:-1:1); handles.T = T; handles.Z = Z; ...
                handles.aclustwt = aclustwt;
            handles.perm = perm;
        else
            msgbox('Dendrogram Calculations started!!!  kindly wait until it ...
                is finished','Notice','replace');
            handles.figh = figure(handles.figh);
            subplot(10,1,1:9);
            [handles.H, handles.T,handles.perm,handles.Z,handles.aclustwt] = ...
                mydendrogram5(handles.link,handles.nolf,'freq',freq); %include ...
                frequence Dendrogram5 is the file
            handles.comp = ...
                cluscompimg(handles.T,zeros(handles.nolf,3),'c',handles.flnm);
            handles.discomp = zeros(handles.nolf,handles.chn);
            handles.discomp(1:handles.nolf,:) = handles.comp(handles.perm,:)*63;
            inicomp = handles.discomp;
            save(fullfile(handles.flnm,'inicomp.mat'),'inicomp');
            figure(handles.figh);
            subplot(10,1,10);
            image(handles.discomp');
            axis off
            colormap(jet(64));
            freezeColors();
            saveas(handles.figh,fullfile(handles.flnm,'figure111.fig'));
            H= handles.H; T = handles.T; Z = handles.Z;aclustwt = ...
                handles.aclustwt; perm = handles.perm;
            save(fullfile(handles.flnm,'dendrogramdata.mat'),'H', 'T', 'perm', ...
                'Z','aclustwt');
        end
        set(handles.status,'String','Dendrogram Displayed.');
        set(handles.status,'BackgroundColor',[0 0.8 0]);
```

```
    else
        warndlg('cell.mat or chn.mat file is missing!!!','Files Missing');
    end
else
    warndlg('Processed_Images folder is missing!!!','Files Missing');
end
guidata(hObject, handles);
function load_fld_Callback(hObject, eventdata, handles)
handles.flnm = uigetdir(handles.flnm, 'Choose a folder of Analysis');
if(handles.flnm)
    set(handles.text1,'String',handles.flnm);
    handles.dat = zeros(1,6);
    set(handles.uitable,'Data',handles.dat);
    norm_dend(hObject,eventdata,handles);
    handles = guidata(hObject);
    guidata(hObject, handles);
end
function choose_color_Callback(hObject, eventdata, handles)
handles.dat = get(handles.uitable,'Data');
noc = size(handles.dat,1);
switch handles.rbsel2
    case 'rb4sel'
        color = uisetcolor();
        handles.clmp(handles.sel,:) = color;
        handles.dat(handles.sel,3:5) = color;
    case 'rb4all'
        color = uisetcolor();
        for ia = 1:noc
            handles.clmp(ia,:) = color;
            handles.dat(ia,3:5) = color;
        end
    otherwise
        handles.clmp = jet(noc);
        handles.clmp = handles.clmp(randperm(noc),:);
        handles.dat(1:noc,3:5) = handles.clmp;
end
set(handles.uitable,'Data',handles.dat(1:end,:));
disp_colormap(hObject,eventdata,handles);
guidata(hObject, handles);
function disp_colormap(hObject,eventdata,handles)
handles.disclmp = zeros(handles.norows,1);
sz = size(handles.clmp,1);
handles.disclmp(1:sz) = 2:(sz+1);
axes(handles.axes3);
image(handles.disclmp);
colormap([0 0 0;handles.clmp]);
freezeColors();
if(get(handles.ex_noise,'Value'))
    handles.dat = get(handles.uitable,'Data');
    handles.clmp(logical(handles.dat(:,6)),:) = [];
end
```

```
sz = size(handles.clmp,1);
axes(handles.axes2);
image((2:(sz+1))');
colormap([0 0 0;handles.clmp]);
freezeColors();
function update_Callback(hObject, eventdata, handles)
handles.dat = get(handles.uitable,'Data');
handles.dat = handles.dat(¬isnan(handles.dat(:,1)),:);
clist = handles.dat(:,1);
handles.clist = clist;
sz = size(handles.clist,1);
clmp = handles.dat(1:size(handles.clist,1),3:5);
handles.clmp = clmp;
disp_colormap(hObject,eventdata,handles);
handles.T1 = mycluster(handles.clist',handles.Z,handles.T,handles.H, ...
    handles.clmp);
handles.comp = cluscompimg(handles.T1,handles.clmp,'c',handles.flnm);
nfig = figure(22);
for ia = 1:size(handles.comp,1),
    subplot(1,size(handles.comp,1),ia),bar(handles.comp(ia,:), ...
        'FaceColor',clmp(ia,:)); ylim([0 1]);
end
saveas(nfig,fullfile(handles.flnm,'figure22.fig'));
handles.discomp = zeros(handles.norows,handles.chn);
handles.discomp(1:sz,:) = handles.comp*63;
axes(handles.axes1);
image(handles.discomp);
colormap(jet(64));
freezeColors();
for ia = 1:sz
    if(handles.clist(ia))
        handles.dat(ia,2) = handles.aclustwt(handles.clist(ia))';
    end
end
set(handles.uitable,'Data',handles.dat);
guidata(hObject, handles);
function gen_data_Callback(hObject, eventdata, handles)
if(get(handles.rb1,'Value'))
    opt = 'i';
    str = 'Compositional Images are';
elseif(get(handles.rb2,'Value'))
    opt = 't';
    str = 'State Transition Matrix is';
elseif(get(handles.rb4,'Value'))
    opt = 'n';
    str = 'Neighbourhood Matrix is';
else
    opt = 'a';
    str = 'Both Compositional Images and State Transition Matrix is';
end
set(handles.status,'String', [str ' being generated. Please wait.....']);
set(handles.status,'BackgroundColor',[0.8 0 0]);
msgbox([ str ' being generated and Exported!!!  kindly wait until it is ...
     finished'],'Notice','replace');
[¬, clustfreq] = cluscompimg(handles.T1,handles.clmp,opt,handles.flnm);
comp = handles.comp;
save(fullfile(handles.flnm,'composition.mat'), 'comp');
if(opt=='i' || opt == 'a')
    if(get(handles.ex_noise,'Value'))
        handles.dat = get(handles.uitable,'Data');
        clustfreq(logical(handles.dat(:,6)),:) = [];
        handles.clmp(logical(handles.dat(:,6)),:) = [];
    end
    temp_clustfreq = clustfreq;
    figure(189);
    clf(189);
    for ia = 1:size(handles.clmp,1),
        hold on;
        plot(1:size(clustfreq,2),clustfreq(ia,:), 'color', ...
            handles.clmp(ia,:),'LineWidth',2);
    end
    xlabel('Time');
    ylabel('Intensity');
    clustfreq = clustfreq./repmat(clustfreq(:,1),1,size(clustfreq,2));
    figure(199);
    clf(199);
    for ia = 1:size(handles.clmp,1),
        hold on;
        plot(1:size(clustfreq,2),clustfreq(ia,:), 'color', ...
            handles.clmp(ia,:),'LineWidth',2);
    end
    xlabel('Time');
    ylabel('Normalized Intensity');
    clustfreq = temp_clustfreq;
    for ia = 1:size(handles.clmp,1)
        minval = min(clustfreq(ia,:));
        maxval = max(clustfreq(ia,:));
        clustfreq(ia,:) = (clustfreq(ia,:) − minval)*64/( maxval − minval );
    end
    axes(handles.axes4);
    image((clustfreq));
    colorbar;
    colormap(jet(64));
    freezeColors();
    figure(8586961);
    subplot(1,10,1);
    sz = size(handles.clmp,1);
    image((2:(sz+1))');
    axis off
    ylabel('Compositions');
    colormap([0 0 0;handles.clmp]);
    freezeColors();
    h = subplot(1,10,2:10);
    image((clustfreq));
    set(h,'ytick',[]);
    xlabel('Time');
    colorbar;
    colormap(jet(64));
    freezeColors();
end
set(handles.status,'String', [str ' Exported. ']);
set(handles.status,'BackgroundColor',[0 0.8 0]);
function show_comp_imgs_Callback(hObject, eventdata, handles)
MIJ.start;
mijread(fullfile(handles.flnm,'ClusCompImg','CompositionalImage.tif'));
function show_state_trans_Callback(hObject, eventdata, handles)
load(fullfile(handles.flnm,'state_change.mat'));
opt = get(handles.ex_noise,'Value');
handles.dat = get(handles.uitable,'Data');
disp_transition(remove_noise(statechangeProb_new(state_chg), ...
    handles.dat(:,6),opt),handles.flnm,'StateTrans');
function status_Callback(hObject, eventdata, handles)
function status_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function figure1_CloseRequestFcn(hObject, eventdata, handles)
tmp{1} = handles.flnm;
save('clusteringtool_log.mat','tmp');
delete(hObject);
function uitable_CellSelectionCallback(hObject, eventdata, handles)
if(eventdata.Indices)
    handles.sel = eventdata.Indices(1);
    disdat = get(handles.uitable,'Data');
    if(handles.sel> size(disdat,1))
        disdat(handles.sel,2) = 0;
        set(handles.uitable,'Data',disdat);
    end
    guidata(hObject,handles);
end
function noleaf_Callback(hObject, eventdata, handles)
handles.nolf = str2num(get(hObject,'String'));
guidata(hObject,handles);
function load_table_Callback(hObject, eventdata, handles)
[filename, handles.flnm] = uigetfile('*.mat', 'Choose Table Data',handles.flnm);
if(filename)
    load (fullfile(handles.flnm,filename));
    handles.nolf = nolf;
    handles.dat = dat;
    set(handles.uitable,'Data',dat);
    guidata(hObject, handles);
end
```

```
function export_table_Callback(hObject, eventdata, handles)
[filename, handles.flnm] = uiputfile( {'*.mat'},'Save Table Data ',handles.flnm);
if(filename)
    dat = get(handles.uitable,'Data');
    nolf = handles.nolf;
    save(fullfile(handles.flnm,filename),'dat','nolf');
end
function gen_graph_Callback(hObject, eventdata, handles)
load(fullfile(handles.flnm,'state_change.mat'));
mat = sumstates(state_chg);
c2cytoscape(mat,'d',handles.flnm);
c2cytoscape(mat,'u',handles.flnm);
function color_code_SelectionChangeFcn(hObject, eventdata, handles)
handles.rbsel2 = get(eventdata.NewValue,'Tag');
guidata(hObject, handles);
function show_comp_space_Callback(hObject, eventdata, handles)
k = handles.T1;
load(fullfile(handles.flnm,'NData','FnNormPixelslist_bean.mat'));
n = size(k,1);
rnd = randperm(n);
if(n>1e5)
    rnd = rnd(1:1e5);
end
k = k(rnd,:);
upixelslist = upixelslist(rnd,:);   %Sample 1e5 poins
mrksz = 5;
sz = size(handles.clist,1);
figure(234);
for ia = sz:-1:1
    hold on;
    lst{ia} = upixelslist(k==ia,:);
    plot3(lst{ia}(:,1),lst{ia}(:,2),lst{ia}(:,3),'.', 'color', ...
        handles.clmp(ia,:),'MarkerSize', mrksz);
end
if(size(upixelslist,2)==4)
    figure(235);
    for ia = sz:-1:1
        hold on;
        lst{ia} = upixelslist(k==ia,:);
        plot3(lst{ia}(:,4),lst{ia}(:,2),lst{ia}(:,3),'.', 'color', ...
            handles.clmp(ia,:),'MarkerSize', mrksz);
    end
end
figure(4556)
for ia = 2:handles.chn
    for ib = 1:ia-1
        subplot(handles.chn-1,handles.chn-1, ((ia-1)*handles.chn)+ib  - ...
            (handles.chn+ia-2)  );
        for ic = 1:sz
            hold on;
            plot(lst{ic}(:,ib),lst{ic}(:,ia),'.','color', ...
                handles.clmp(ic,:),'MarkerSize', mrksz);
        end
        xlabel(sprintf('Channel %d',ib));
        ylabel(sprintf('Channel %d',ia));
    end
end
function uipanel1_SelectionChangeFcn(hObject, eventdata, handles)
function thd_Callback(hObject, eventdata, handles)
function thd_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function disp_top_Callback(hObject, eventdata, handles)
opt = get(handles.ex_noise,'Value');
handles.dat = get(handles.uitable,'Data');
thd =  str2num(get(handles.thd,'String'));
if(get(handles.trans_top,'Value'))
    load(fullfile(handles.flnm,'state_change.mat'));
    state_chg_prob = remove_noise(statechangeProb_new(state_chg), ...
        handles.dat(:,6),opt);
    Topo = (state_chg_prob ≥ thd);
    disp_transition(Topo,handles.flnm,'StateTransTopology');
    save(fullfile(handles.flnm,'State_Trans_Top.mat'),'Topo','thd');
else
    load(fullfile(handles.flnm,'Neighbourhood_matrix.mat'));
    neighb_prob = remove_noise(NeighbFrac(neighb_mat),handles.dat(:,6),opt);
    Topo = (neighb_prob ≥ thd);
    disp_transition(Topo,handles.flnm,'Neighbourhood Topology');
    save(fullfile(handles.flnm,'Neighb_Topo.mat'),'Topo','thd');
end
function disp_neighb_Callback(hObject, eventdata, handles)
load(fullfile(handles.flnm,'Neighbourhood_matrix.mat'));
opt = get(handles.ex_noise,'Value');
handles.dat = get(handles.uitable,'Data');
[m, n] = ...
    size(imread(fullfile(handles.flnm,'ClusCompImg/CompositionalImage.tif')));
disp_transition(  remove_noise(  ((NeighbProb(neighb_mat) * m*n)−1), ...
    handles.dat(:,6),opt) , handles.flnm, 'Neighbourhood_Prob');
disp_transition(remove_noise( NeighbFrac(neighb_mat), ...
    handles.dat(:,6),opt),handles.flnm, 'Neighbourhood');
function show_3dtrans_Callback(hObject, eventdata, handles)
function tslider_Callback(hObject, eventdata, handles)
sliderValue = round(get(handles.tslider,'Value'));
set(handles.tpt,'String', num2str(sliderValue));
draw3d_trans(handles.nwcomp, handles.nwclmp, ...
    handles.state_trans_topo(:,:,sliderValue), handles.pos_cam);
function tpt_Callback(hObject, eventdata, handles)
tpt = round(str2num(get(hObject, 'String')));
if (isempty(tpt) || tpt < 0 || tpt > get(handles.tslider,'Max'))
```

```
    tpt = 1;
end
set(hObject,'String', tpt);
set(handles.tslider,'Value',tpt);
draw3d_trans(handles.nwcomp,handles.nwclmp, ...
    handles.state_trans_topo(:,:,tpt),handles.pos_cam);
function set_campos_Callback(hObject, eventdata, handles)
figure(586);
f = gca;
handles.pos_cam  = get(f,'cameraPosition');
guidata(hObject, handles);
function export_trans3d_Callback(hObject, eventdata, handles)
noimg = size(handles.state_trans_topo,3);
ind = (1:noimg) + 1000;
ind = num2str(ind');
ind(:,1) = 't';
if(¬exist(fullfile(handles.flnm,'3D State Transition Movie'),'dir'))
    mkdir(fullfile(handles.flnm,'3D State Transition Movie'));
end
aviobj = avifile(fullfile(handles.flnm, '3D State Transition ...
    Movie.avi'),'compression','None');
for ia = 1:noimg
    draw3d_trans(handles.nwcomp,handles.nwclmp,handles.state_trans_topo(:,:,ia),handles.pos_cam);
    F = getframe;
    aviobj = addframe(aviobj,F);
    print('-f586','-dtiff', fullfile(handles.flnm,'3D State Transition ...
        Movie',ind(ia,:)));
end
aviobj = close(aviobj);
function show_trans3d_Callback(hObject, eventdata, handles)
if(get(hObject,'Value'))
    set(handles.tpt,'Enable','on');
    set(handles.tpt,'Enable','on');
    load(fullfile(handles.flnm,'comp_colormap.mat'));
    load(fullfile(handles.flnm,'composition.mat'));
    handles.dat = get(handles.uitable,'Data');
    clmp(1,:) = [ 0.5 0.5 0.5];
    comp = [ zeros(size(comp,2)); comp];
    handles.nwclmp = clmp;
    handles.nwcomp = comp;
    if(get(handles.ex_noise,'Value'))
        handles.nwclmp = clmp(¬logical([0;handles.dat(:,6)]),:);
        handles.nwcomp = comp(¬logical([0;handles.dat(:,6)]),:);
    end
    load(fullfile(handles.flnm,'State_Trans_Top.mat'));
    set(handles.tslider,'Max', size(Topo,3));
    set(handles.tslider,'SliderStep', [1/(size(Topo,3)-1) 0.1]);
    handles.state_trans_topo = Topo;
    draw3d_trans(handles.nwcomp, handles.nwclmp,Topo(:,:,1),'auto');
    figure(586);
    handles.pos_cam =  get(gca,'cameraPosition');
```

```
else
    set(handles.tpt,'Enable','off');
    set(handles.tpt,'Enable','off');
end
guidata(hObject, handles);
function ex_noise_Callback(hObject, eventdata, handles)
function mat = remove_noise(mat,nois,opt)
nois = [ 0 ; nois];
if(opt)
    mat(logical(nois),:,:) = [];
    mat(:,logical(nois),:) = [];
end
function sil_incosist_Callback(hObject, eventdata, handles)
load(fullfile(handles.flnm,'NData/FnNormPixelslist_bean.mat'));
msgbox('Silhouette and Linkage inconsistancy Calculations started!!!  kindly ...
    wait until it is finished','Notice','replace');
set(handles.status,'String','Silhouette Displayed.', 'BackgroundColor',[0 0.8 0]);
figure(2398);
rn = randi(size(upixelslist,1),1e4,1);
spxl = upixelslist(rn,:);
sT1 = handles.T1(rn);
[s, h] = silhouette(spxl,sT1);
figure(3345);
k = inconsistent(handles.Z);
plot(k(:,4),'k.');
function clust_trans_Callback(hObject, eventdata, handles)
opt = get(handles.ex_noise,'Value');
handles.dat = get(handles.uitable,'Data');
load(fullfile(handles.flnm,'state_change.mat'));
load(fullfile(handles.flnm,'composition.mat'));
state_chg_prob = remove_noise(statechangeProb_new(state_chg), ...
    handles.dat(:,6),opt);
clust_trans(state_chg_prob,comp,handles.flnm);
```

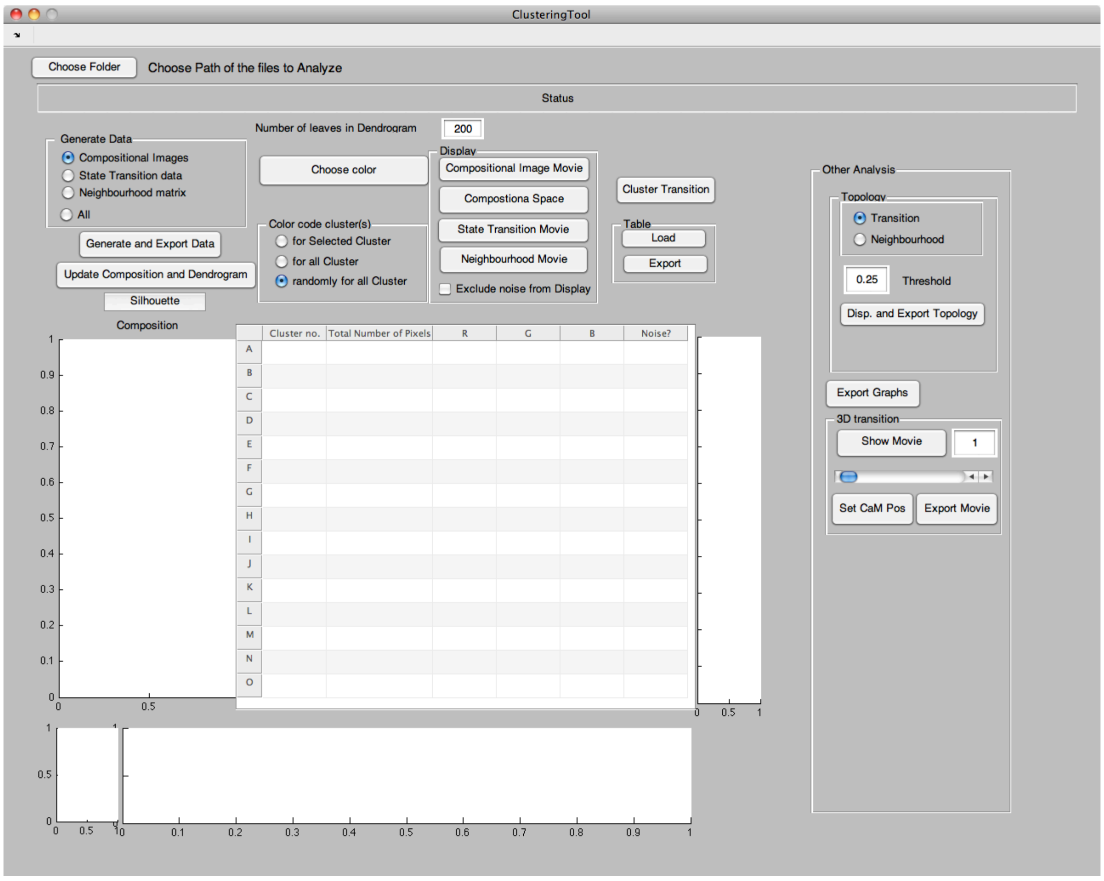


Figure 8.4: Compositional analysis tool -

```
function normpixels4cell(flnm)
load([flnm,'cell.mat']);
thdfn = dir(fullfile(flnm,'Processed_Images/*.mat'));
chn = size(thdfn,1);
save(fullfile(flnm, 'chn.mat'),'chn');
fldnm = [flnm 'Processed_Images/'];
for ia = 1:chn
    chn_thd(ia) = load(fullfile(fldnm, thdfn(ia).name));
end
lst = dir([fldnm '*.tif']);
nn = size(lst,1);
noimg = nn/chn;
imgn = reshape({ lst.name },chn, noimg)';
img = imread( fullfile(fldnm ,  imgn{1,1}));
[m,n]= size(img);
clear img;
pixelslist(1,:) =  zeros(1,chn) -1;
```

```
nmat = zeros(1,chn);
pixels = zeros(m*n,chn);
indexn0 = 1;
disdat = cell2mat({dat(:).framept}');
ind = disdat(1,1); %first frame number
if(exist([flnm 'nucleus.mat'],'file'))
    nuc = load([flnm 'nucleus.mat']);
    nuc_disdat = cell2mat({nuc.dat(:).framept}');
    nuc_ind = disdat(1,1);
end
for ii = 1:noimg
    [mask,k] = fetchmask(dat,disdat,m,n,ind);
    if(k)
        if(exist('nuc','var'))
            nuc_mask = fetchmask(nuc.dat,nuc_disdat,m,n,nuc_ind);
            mask = mask .* double(¬nuc_mask);
            nuc_ind = nuc_ind + 1;
        end
        for jj = 1:chn
            temp_img = double(imread(fullfile(fldnm, imgn{ii,jj})));
            img(:,:,jj) = temp_img;
            thd_mask(:,:,jj) = temp_img > chn_thd(jj).threshold;   %threshold ...
                mask is used to exclude pixels which are
        end
        f_thd_mask = logical(sum(thd_mask,3));
        for jj = 1:chn
            temp_img = img(:,:,jj);
            temp_img(¬f_thd_mask) = 0;
            pixels(:,jj) = double( reshape(temp_img,m*n,1)).*mask;
        end
        indn0 = double(¬ismember(pixels,nmat,'rows')) .* (1:(m*n))';
        indn0 = indn0(¬ismember(indn0,0));
        pixels = pixels(¬ismember(pixels,nmat,'rows'),:);
        ii
        indexn0 = [indexn0 ; indn0 + ((ii-1)* (m*n))];
        pixelslist = [pixelslist;pixels];  % Collect all the pixels
        pixels = zeros(m*n,chn);
        size(pixelslist)
    end %if(k)
    ind = ind +1;
end
indexn0(1) = [];
pixelslist(1,:) = [];
size(pixelslist)
clear pixels;
if(¬exist([flnm 'NData'],'dir'))
    mkdir([flnm 'NData']);
end
display('Collected Pixels');
cpixelslist = colnorm(pixelslist);
display('Column Normalized Pixels');
```

```
fold = max(cpixelslist,[],2);
fldmov = zeros(m,n,noimg);
fldmov(indexn0) = fold;
h = movplayer(fldmov,flnm);
uiwait(h);
fn = fullfile(flnm,'Ampthd.mat');
if(exist(fn,'file'))
    load(fn);
    pixelslist = pixelslist(fold > ampthd,:);
    cpixelslist = colnorm(pixelslist);
    indexn0 = indexn0(fold > ampthd);
    fold = fold(fold > ampthd);
end
save([flnm,'NData/RawPixelsList.mat'],'pixelslist');
save([flnm,'NData/indexn0.mat'], 'indexn0','noimg','m','n','chn');
clear indexn0;
save([flnm 'NData/fold'], 'fold');
clear fold;
rwpixelslist = zeros(size(cpixelslist));
for jj = 1:chn
    rwpixelslist(:,jj) = cpixelslist(:,jj)./sqrt(sum(cpixelslist(:,1:chn).^2,2));
end
display('Pixels Row Normalized');
clear pixelslist;
upixelslist = rwpixelslist;
index = (1:size(upixelslist,1))';
save([flnm 'NData/index'], 'index');
clear index;
freq = ones(size(upixelslist,1),1);
save([flnm 'NData/FnNormPixelslist_bean'], 'upixelslist');
save([flnm 'NData/FreqFnPixelslist'], 'freq');
sprintf('Size of the FnNormalized pixelslist after binning is ...
    %d',size(upixelslist,1))
function cpixelslist = colnorm(pixelslist)
chn = size(pixelslist,2);
mn(1:chn) = mean(pixelslist(:,1:chn));
for jj = chn:-1:1
    cpixelslist(:,jj) = pixelslist(:,jj) / mn(jj);
end
function [mask,k] = fetchmask(dat, disdat, m,n,num)
k = find(disdat(:,1)≤num & disdat(:,2)≥num);
if(k)
    mask = roipoly(m,n, dat(k).polyg(:,1), dat(k).polyg(:,2));
    mask = double(reshape(mask,m*n,1));
else
    mask = 0;
end
```

```
function [comp clustfreq] = cluscompimg(T,clmp,opt,pat,filt)
comp = 0;
```

```
clustfreq = [];
if(opt == 'c' || opt == 'i' || opt == 't' || opt  == 'a' || opt == 'n')
    nclus = size(clmp,1);
    clmp = [0 0 0; clmp];
    colormap(clmp);
        load (fullfile(pat,'NData/index.mat'));
    load (fullfile(pat,'NData/FnNormPixelslist_bean.mat'));
    chn = size(upixelslist,2);
    if(opt == 'c')
        comp = zeros(nclus,chn);
        k = T(index);
        for ia = 1:nclus
            comp(ia,:) = mean(upixelslist(index( k == ia),:));
        end
        save(fullfile(pat,'composition.mat'), 'comp');
    end
    if(opt == 'i' || opt == 't' || opt =='n' || opt  == 'a')
        clear upixelslist;
        load (fullfile(pat,'NData/indexn0.mat'));
        if(¬exist([pat '/ClusCompImg'],'dir'))
            mkdir([pat '/ClusCompImg']);
        end
        fldnm = 'ClusCompImg/'; %Output folder
        ind = (1:noimg) + 1000;
        ind = num2str(ind');
        ind(:,1) = 't';
        img = zeros(m*n*noimg,1);
        img(indexn0) = T(index);
        clear T indexn0 index;
        img = img + 1;
        img = reshape(img,m,n,noimg);
        save(fullfile(pat,'comp_colormap.mat'), 'clmp');
        clustfreq = zeros(nclus,noimg);
        if(exist('filt','var'))
            if(strcmp(filt,'filt'))
                 filtcompimg = filtcompImg(img);
                if(opt == 'i' || opt  == 'a')
                     imwritestack(filtcompimg−1, clmp, fullfile(pat, fldnm, ...
                         'CompositionalImageFilt.tif'));
                end
            end
        end
        if(opt == 'i' || opt  == 'a')
            if(exist(fullfile(pat,fldnm,'CompositionalImage.tif'),'file'))
                delete( fullfile(pat,fldnm,'CompositionalImage.tif'));
            end
            for ii = 1:noimg
                ii
                timg = img(:,:,ii);
                imwrite(timg, clmp, ...
                    fullfile(pat,fldnm,'CompositionalImage.tif') , ...
```

```
                        'writemode', 'append');
                    for ij = 2:(nclus+1)
                        nm= size(find(timg == ij),1);
                        clustfreq(ij−1,ii) = nm;
                    end
                end
                figure(8);
                image(img(:,:,round(ii/2)));
                colormap(clmp);
                axis image off
            end
            if(opt == 't' || opt  == 'a')
                figure(9);
                state_chg = zeros((1+nclus)^2,noimg − 1);
                A = 1:(nclus+1);
                psts = npermutek(A,2);
                ind = 1;
                for ii = 1:(noimg−1)
                    ii
                    img1 = img(:,:,ii);
                    img2 = img(:,:,ii+1);
                    sts = [img1(:),img2(:)];
                    [sts, freq] = uniquencount(sts);
                    for ia = 1:(nclus+1)^2
                        [f,lc] = ismember(psts(ia,:),sts,'rows');
                        if(f)
                            state_chg(ia,ind) = freq(lc);
                        end
                    end
                    ind = ind+1;
                end
                state_chg = reshape(state_chg,nclus+1,nclus+1,noimg−1);
                save(fullfile(pat,'state_change.mat'),'state_chg');
                imagesc(sumstates(state_chg));
            end
        end
        if(opt == 'n' || opt == 'a')
            nb_shifts = [ −1,−1; 0,−1; 1,−1;  1,0; 1,1;  0,1; −1,1; −1,0];
            for ia = 8:−1:1
                shift_r = nb_shifts(ia,1);
                shift_c = nb_shifts(ia,2);
                [t_vec_r1 t_vec_r2] = shiftvec(shift_r,m);
                [t_vec_c1 t_vec_c2] = shiftvec(shift_c,n);
                vec_r1{ia} = t_vec_r1;
                vec_r2{ia} = t_vec_r2;
                vec_c1{ia} = t_vec_c1;
                vec_c2{ia} = t_vec_c2;
            end
            neighb_mat = zeros(nclus,nclus,noimg);
            for ia = 1:noimg
                ia
                simg = img(:,:,ia);  %single image
                nimg = zeros(m,n,8);  %neighbour images with appropriate shifts
                for ib = 1:8
                    nimg(vec_r1{ib},:,ib) = simg(vec_r2{ib},:);
                    nimg(:,vec_c1{ib},ib) = nimg(:,vec_c2{ib},ib);
                end
                for ij = 1:(nclus+1)
                    mask = repmat((simg == ij),[1,1,8]);
                    nbimg = zeros(size(nimg));
                    nbimg(mask) = nimg(mask);
                    for ik = 1:(nclus+1)
                        temp = (nbimg == ik);
                        neighb_mat(ij,ik,ia) = sum(temp(:));
                    end
                end
                figure(46)
                imagesc(neighb_mat(:,:,ia));
            end
            save(fullfile(pat,'Neighbourhood_matrix.mat'),'neighb_mat');
        end
    else
        display('please choose the correct option. Type ''help cluscompimg''');
    end
    function [vec1 vec2] = shiftvec(shift,sz)
    if(shift ≤ 0 )
        vec1 = 1:sz-abs(shift);
        vec2 = (1+ abs(shift)):sz;
    else
        vec1 = (1+abs(shift)):sz;
        vec2 = 1:sz-abs(shift);
    end
```

```
function [hout,T,perm,Z,acluswt] = mydendrogram(Z,varargin)
%The function is a modified version of original MATLAB function dendrogram.m
%In addition to plotting dendrogram, this function labels each node and leaf
%with an ID. Additionally, it exports a new linkage matrix 'Z' and weight of
%all nodes 'acluswt'. 'Z' is required for the code "mycluster.m", that allows
%selection of nodes/clusters based on node ID, from different hierarchy/levels
m = size(Z,1)+1;
if nargin < 2
    p = 30;
end
if nargin == 2
    p = varargin{1};
end
orientation = 't';
horz = false;
color = false;
obslabels = [];
threshold = 0.7 * max(Z(:,3));
```

```
leafOrder = [];
if nargin > 2
    if isnumeric(varargin{1})
        p = varargin{1};
        offset = 1;
    else
        p = 30;
        offset = 0;
    end
    if rem(nargin − offset,2)== 0
        error('stats:dendrogram:BadNumArgs',...
              'Incorrect number of arguments to DENDROGRAM.');
    end
    okargs = {'orientation' 'colorthreshold' 'labels','reorder'};
    for j=(1 + offset):2:nargin−2
        pname = varargin{j};
        pval = varargin{j+1};
        k = strmatch(lower(pname), okargs);
        if isempty(k)
            error('stats:dendrogram:BadParameter',...
                  'Unknown parameter name:  %s.',pname);
        elseif length(k)>1
            error('stats:dendrogram:BadParameter',...
                  'Ambiguous parameter name:  %s.',pname);
        else
            switch(k)
                case 1  % orientation
                    if ¬isempty(pval)
                        if ischar(pval)
                            orientation = lower(pval(1));
                        else
                            orientation = 0;    % bad value
                        end
                    end
                    if ¬ismember(orientation,{'t','b','r','l'})
                        orientation = 't';
                        warning('stats:dendrogram:BadOrientation',...
                            'Unknown orientation specified, using ''top''.');
                    end
                    if ismember(orientation,{'r','l'})
                        horz = true;
                    else
                        horz = false;
                    end
                case 2  % colorthreshold
                    color = true;
                    if ischar(pval)
                        if ¬strncmpi(pval,'default',length(pval))
                            warning('stats:dendrogram:BadThreshold',...
                                'Unknown threshold specified, using default');
                        end
                    end
                    if isnumeric(pval)
                        threshold = pval;
                    end
                case 3 % labels
                    if ischar(pval)
                        pval = cellstr(pval);
                    end
                    if ¬iscellstr(pval)
                        error('stats:dendrogram:BadLabels',...
                            'Value of ''labels'' parameter is invalid');
                    end
                    if ¬isvector(pval) || numel(pval)≠m
                        error('stats:dendrogram:InputSizeMismatch',...
                            'Must supply a label for each observation.');
                    end
                    obslabels = pval(:);
                case 4 % leaf order
                    if ¬isvector(pval) || numel(pval)≠m
                        error('stats:dendrogram:InputSizeMismatch',...
                            'Leaforder is not a valid permutation.');
                    end
                    leafOrder = pval;
            end
        end
    end
end
rZ = Z;
Z = mytransz(Z);
T = (1:m)';
if (m > p) && (p ≠ 0)
    T = myclusternumb2t(rZ,p,Z);
    Z(:,1:2) = sort(Z(:,1:2),2); %restore to original transz
    Y = Z((m−p+1):end,:);          % get the last nodes
    R = unique(Y(:,1:2));
    Rlp = R(R≤p);
    Rgp = R(R>p);
    W(Rlp) = Rlp;                  % use current node number if ≤p
    W(Rgp) = setdiff(1:p, Rlp);    % otherwise get unused numbers ≤p
    W = W(:);
    Y(:,1) = W(Y(:,1));
    Y(:,2) = W(Y(:,2));
    Z = Y;
    m = p; % reset the number of node to be 30 (row number = 29).
else  % my new addition
    Z = transz(Z);
end
A = zeros(4,m−1);
B = A;
n = m;
X = 1:n;
```

```
Y = zeros(n,1);
r = Y;
W = zeros(size(Z));
W(1,:) = Z(1,:);
nsw = zeros(n,1); rsw = nsw;
nsw(Z(1,1:2)) = 1; rsw(1) = 1;
k = 2; s = 2;
while (k < n)
    i = s;
    while rsw(i) || ¬any(nsw(Z(i,1:2)))
        if rsw(i) && i == s
            s = s+1;
        end
        i = i+1;
    end
    W(k,:) = Z(i,:);
    nsw(Z(i,1:2)) = 1;
    rsw(i) = 1;
    if s == i
        s = s+1;
    end
    k = k+1;
end
g = 1;
for k = 1:m−1 % initialize X
    i = W(k,1);
    if ¬r(i),
        X(i) = g;
        g = g+1;
        r(i) = 1;
    end
    i = W(k,2);
    if ¬r(i),
        X(i) = g;
        g = g+1;
        r(i) = 1;
    end
end
if ¬isempty(leafOrder)
   [dummy, X] = sort(leafOrder); %#ok
end
[u,perm]=sort(X);   %#ok perm is the third output value
label = num2str(perm');
if ¬isempty(obslabels)
   label = cellstr(label);
   singletons = find(histc(T,1:m)==1);
   for j=1:length(singletons)
      sj = singletons(j);
      label(perm==sj) = obslabels(T==sj);
   end
end
```

```
theGroups = 1;
groups = 0;
cmap = [0 0 1];
if color
    groups = sum(Z(:,3)< threshold);
    if groups > 1 && groups < (m-1)
        theGroups = zeros(m-1,1);
        numColors = 0;
        for count = groups:-1:1
            if (theGroups(count) == 0)
                P = zeros(m-1,1);
                P(count) = 1;
                P = colorcluster(Z,P,Z(count,1),count);
                P = colorcluster(Z,P,Z(count,2),count);
                numColors = numColors + 1;
                theGroups(logical(P)) = numColors;
            end
        end
        cmap = hsv(numColors);
        cmap(end+1,:) = [0 0 0];
    else
        groups = 1;
    end
end
if  isempty(get(0,'CurrentFigure')) || ishold
    figure('Position', [50, 50, 800, 500]);
else
    newplot;
end
col = zeros(m-1,3);
h = zeros(m-1,1);
for n = 1:(m-1)
    i = Z(n,1); j = Z(n,2); w = Z(n,3);
    A(:,n) = [X(i) X(i) X(j) X(j)]';
    B(:,n) = [Y(i) w w Y(j)]';
    X(i) = (X(i)+X(j))/2; Y(i)  = w;
    if n ≤ groups
        col(n,:) = cmap(theGroups(n),:);
    else
        col(n,:) = cmap(end,:);
    end
end
ymin = min(Z(:,3));
ymax = max(Z(:,3));
margin = (ymax - ymin) ⋆ 0.05;
n = size(label,1);
if(¬horz)
    for count = 1:(m-1)
        h(count) = line(A(:,count),B(:,count),'color',col(count,:));
    end
    lims = [0 m+1 max(0,ymin-margin) (ymax+margin)];
    set(gca, 'Xlim', [.5 ,(n +.5)], 'XTick', 1:n, 'XTickLabel', label, ...
        'Box', 'off');
    mask = logical([0 0 1 1]);
    if strcmp(orientation,'b')
        set(gca,'XAxisLocation','top','Ydir','reverse');
    end
else
    for count = 1:(m−1)
        h(count) = line(B(:,count),A(:,count),'color',col(count,:));
    end
    lims = [max(0,ymin−margin) (ymax+margin) 0 m+1 ];
    set(gca, 'Ylim', [.5 ,(n +.5)], 'YTick', 1:n, 'YTickLabel', label, ...
        'Box', 'off');
    mask = logical([1 1 0 0]);
    if strcmp(orientation, 'l')
        set(gca,'YAxisLocation','right','Xdir','reverse');
    end
end
if margin==0
    if ymax≠0
        lims(mask) = ymax * [0 1.25];
    else
        lims(mask) = [0 1];
    end
end
axis(lims);
if nargout>0
    hout = h;
end
sz = size(perm,2);
zsz = sz − 1;
clssz = sz*2  − 1;
posx = zeros(1,clssz);
[¬,posx(1:sz),¬] = intersect(perm,1:sz);
for ia = 1:zsz
k = Z(ia,1:2);
Z(ia,1:2) = Z(ia,1:2) + clssz;
Z(Z(:,1) == k(1),1) =  sz + ia;
Z(Z(:,2) == k(1),2) =  sz + ia;
posx(sz+ia) = sum(posx(k))/2;
end
Z(:,1:2) = Z(:,1:2) − clssz;
cluswt = ones(1,clssz);
acluswt = ones(1,clssz);
for ia = sz:−1:1,
cluswt(ia) = sum (T == ia) ;
acluswt(ia) = sum((T == ia));
end
for ia= 1:zsz
    cluswt(sz+ia) = cluswt( Z(ia,1)) + cluswt( Z(ia,2));
    acluswt(sz+ia) = acluswt( Z(ia,1)) + acluswt( Z(ia,2));
```

```
end
for ia = zsz:−1:1
     hold on;
     text( posx(sz + ia), Z(ia,3),   sprintf('%d\n(%d)',sz ...
         +ia,acluswt(sz+ia)),'color','r', 'HorizontalAlignment', ...
         'center','VerticalAlignment','top');
end
function T = clusternum(X, T, c, k, m, d)
d = d+1;
n = m; flag = 0;
while n > 1
    n = n−1;
    if X(n,1) == k % node k is not a leave, it has subtrees
        T = clusternum(X, T, c, k, n,d); % trace back left subtree
        T = clusternum(X, T, c, X(n,2), n,d);
        flag = 1; break;
    end
end
if flag == 0 && d ≠ 1 % row m is leaf node.
    T(X(m,1)) = c;
    T(X(m,2)) = c;
end
function T = colorcluster(X, T, k, m)
n = m;
while n > 1
    n = n−1;
    if X(n,1) == k % node k is not a leave, it has subtrees
        T = colorcluster(X, T, k, n); % trace back left subtree
        T = colorcluster(X, T, X(n,2), n);
        break;
    end
end
T(m) = 1;
function Z = transz(Z)
m = size(Z,1)+1;
for i = 1:(m−1)
    if Z(i,1) > m
        Z(i,1) = traceback(Z,Z(i,1));
    end
    if Z(i,2) > m
        Z(i,2) = traceback(Z,Z(i,2));
    end
    if Z(i,1) > Z(i,2)
        Z(i,1:2) = Z(i,[2 1]);
    end
end
function Z = mytransz(Z)
m = size(Z,1)+1;
for i = 1:(m−1)
    if Z(i,1) > m
        Z(i,1) = traceback(Z,Z(i,1));
    end
    if Z(i,2) > m
        Z(i,2) = traceback(Z,Z(i,2));
    end
end
function a = traceback(Z,b)
m = size(Z,1)+1;
if Z(b-m,1) > m
    a = traceback(Z,Z(b-m,1));
else
    a = Z(b-m,1);
end
if Z(b-m,2) > m
    c = traceback(Z,Z(b-m,2));
else
    c = Z(b-m,2);
end
a = min(a,c);
```

```
function T = myclusternumb2t(Z,p,cZ)
sz = size(Z,1)+1;
pr = zeros(2*sz-1,1);
for ia = 1:(sz-1)
   pr(Z(ia,1),1) = sz+ia;
   pr(Z(ia,2),1) = sz+ia;
end
T = zeros(2*sz-1,1);
subZ = Z(end-p+2:end,[1,2]);
subcZ = cZ(end-p+2:end,[1,2]);
[lst idx] = unique(subZ);
lst = lst(1:p);
idx = idx(1:p);
clst = subcZ(idx);
[clst idx] = unique(clst);
lst = lst(idx);
Llp = clst(clst<=p); %clst  <p
Lgp = clst(clst>p);
    W(Llp) = Llp;                 % use current node number if <=p
    W(Lgp) = setdiff(1:p, Llp);   % otherwise get unused numbers <=p
    W = W(:);
colr = W(clst(1:p));
for ia = 1:p
    cp = lst(ia);
    subtr = traceall(cp,pr,sz,Z);
    T(subtr) = colr(ia);
end
T = T(1:sz);
function subtr = traceall(cp,pr,sz,Z)
subtr = false(2*sz-1,1);
fncp =pr(cp);
```

```
while(1)
    subtr(cp) = true;
    if(cp>sz)
        ncp = Z(cp−sz,1);
        if(subtr(ncp) == true)
            ncp = Z(cp−sz,2);
            if(subtr(ncp)== true)
                ncp =pr(cp);
            end
        end
        cp = ncp;
    else
        cp =pr(cp);
    end
    if(cp == fncp)
        break;
    end
end
```

```
function T = mycluster(clist,z,T,H,clmp)
nc =  size(clist,2);
sz = size(z,1) +1 ;
for ia = 1:size(H,1)
    set(H(ia),'color',[0 0 0],'LineWidth',1);
end
for ia = 1:nc
    cl = clist(ia);
    if(cl)
    while( sum(cl> (sz)))
        tmp = cl(cl ≤ sz);
        cl = cl(cl > sz);
        set(H(cl − sz),'color',[clmp(ia,:)],'LineWidth',2);
        cl = z(cl − sz,1:2);
        cl = cl(:)';
        cl = [tmp, cl];
    end
    for ib = cl
    T(T == ib) = sz + ia;
    end
    end
end
T = T − sz;
```

```
function comp_barplot(comp,clmp)
tcomp = comp';
tcomp = tcomp(:);
figure(323209);
h_bar = bar(tcomp);
```

```
xlim([−1 size(tcomp,1)]+1);
set(gca,'xtick',[]);
bar_child=get(h_bar,'Children');
set(bar_child,'CDataMapping','direct');
sz = size(comp,1);
pidx(1,:) = 1:sz;
idx = repmat(pidx, size(comp,2),1);
set(bar_child, 'CData',idx(:));
colormap(clmp);
```

```
function [link cpos] = hircluster(in)
load(fullfile(in,'NData/FnNormPixelslist_bean.mat'));
load(fullfile(in,'NData/FreqFnPixelslist.mat'));
link = mylinkage(upixelslist,'ward','euclidean',freq);
cpos = [];
```

```
function [out, cord] = sph_wardcluster(in, threads, weights,ispla)
if ¬exist('threads','var')
    try
        threads = feature('numCores');
    catch
        threads = 2;
    end
end
savebin('tmp.bin', in);
if exist('weights','var')
    savebin('weights.bin', weights,'w');
end
opt = 'sph ';
if exist('ispla','var')
    if(isequal(ispla,'pla'))
    opt = 'pla ';
    end
end
if nargout == 1
    if ¬exist('weights','var')
        tic; system(['./omp_ward_' opt num2str(threads) ' tmp.bin tmp.out']); ...
            t = toc;
    else
        tic; system(['./omp_ward_' opt num2str(threads) ' tmp.bin tmp.out −w ...
            weights.bin']); t = toc;
    end
elseif nargout == 2
    if  ¬exist('weights','var')
        tic; system(['./omp_ward_' opt num2str(threads) ' tmp.bin tmp.out −c ...
            tmp_coords.out']); t =  toc;
    else
```

```
        tic; system(['./omp_ward_' opt num2str(threads) ' tmp.bin tmp.out -c ...
            tmp_coords.out -w weights.bin']); t =  toc;
    end
    cord = loadCoord('tmp_coords.out');
end
out = loadLinkage('tmp.out');
```

```
function state_chg2 = statechangeProb(state_chg)
[l,l,n] = size(state_chg);
state_chg2 = zeros(l,l,n);
for ia = 1:n,
    for ib = 1:l,
        for ic = 1:l,
            state_chg2(ic,ib,ia) = state_chg(ic,ib,ia)/ ...
                (sum(state_chg(ic,:,ia)) * sum(state_chg(:,ib,ia)));
        end
    end
end
```

```
function disp_transition(in,path_name,file_name)
in = in*64;
colormap(jet(64));
display('State Transition absolute number heat map');
n = size(in,3);
aviobj = avifile(fullfile(path_name, [file_name '.avi']),'compression','None');
figure(11);
axis image
for ia = 1:n
    image(in(:,:,ia));
    colorbar;
    pause(0.01);
    F = getframe;
    aviobj = addframe(aviobj,F);
end
aviobj = close(aviobj);
```

```
function state_chg2 = statechangeProb(state_chg)
[l,l,n] = size(state_chg);
state_chg2 = zeros(l,l,n);
for ia = 1:n,
    for ib = 1:l,
        for ic = 1:l,
            state_chg2(ic,ib,ia) = state_chg(ic,ib,ia)/ ...
                (sum(state_chg(ic,:,ia)) * sum(state_chg(:,ib,ia)));
        end
    end
```

```
end
```

```
function disp_transition(in,path_name,file_name)
in = in*64;
colormap(jet(64));
display('State Transition absolute number heat map');
n = size(in,3);
aviobj = avifile(fullfile(path_name, [file_name '.avi']),'compression','None');
figure(11);
axis image
for ia = 1:n
    image(in(:,:,ia));
    colorbar;
    pause(0.01);
    F = getframe;
    aviobj = addframe(aviobj,F);
end
aviobj = close(aviobj);
```

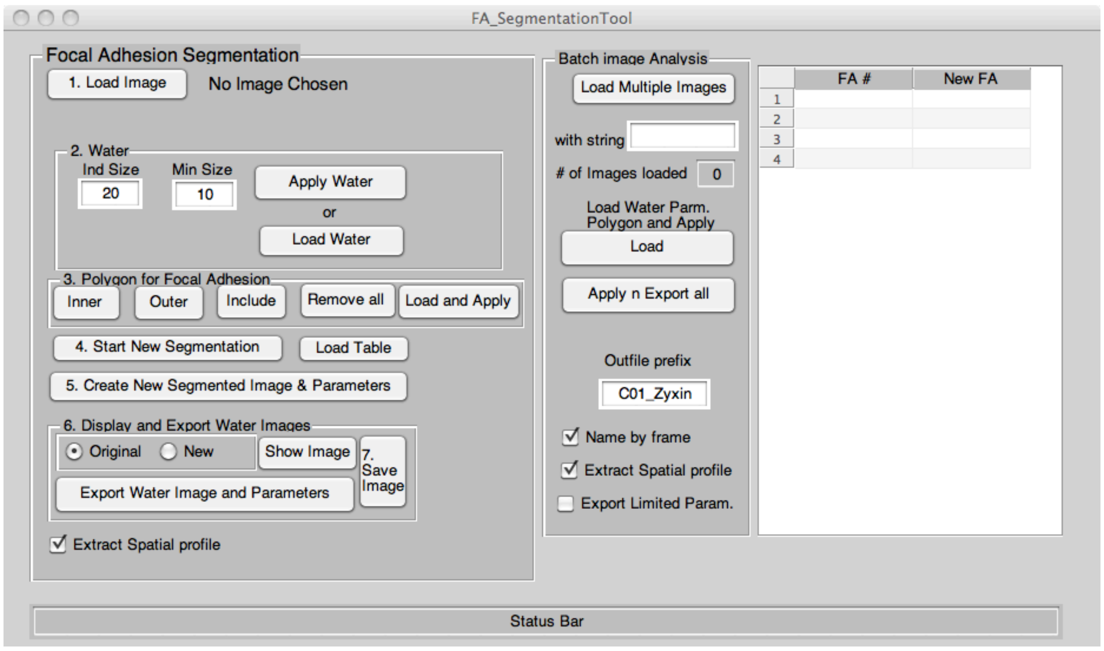
8.3 Segmentation and tracking of adhesion sites in multicolor live cell time-lapse images

8.3.1 Object segmentation tool

- Run Matlab GUI "*FA_SegmentationTool.fig*"
- Click on "load image" and select a processed image
- Type appropriate water parameters Ind. Size and Min Size. Ind. Size: minimum size of an objects (in pixels) required to keep in independent without merging it to the large immediate neighbor. Default value: 20 Min Size: minimum size of the object. Object below this size are excluded after segmentation. Default value: 10
- Click on "apply water" button. Images of segmented images of the cell will open up.
- Enhance the contrast of the colormap of the Matlab figure displaying gray scale image with green dots to visualize objects or adhesion sites
- Draw inner and outer polygons by clicking appropriate buttons. The inner polygon is for removing the unwanted nuclear and peri-nuclear region. The outer

polygon is to include cell area with focal adhesions and to avoid any unwanted region outside the cell.

- Click "Start new segmentation". Segmentation can be manually corrected. To merge two focal adhesion, type ID of both the objects in the same cell in the second column of the table.
- Click "Create new segmented image and parameters" to generate new segmentation based on the correction in the table.
- Click "Export Water Image and Parameters" to export segmented images along with various parameters that includes area, mean and integrated intensity, centre of mass for each segmented objects or adhesion sites.
- Check checkbox "Extract Spatial profile" to extract the intensity profile of the protein along the major axis of the focal adhesion.
- Batch Processing: Using common parameters, time lapse images series of each channel can be processed in batch. Firstly, type the string in the text box to filter the images based on channels.
- Click on "Load multiple images". Images with the given string in the file name will be loaded. The total number of images loaded will be displayed in the textbox.
- Click "Load" and choose the Matlab file already exported after segmentation of single image.
- Type a desired string at "Outfile prefix" text box to add it to the name of output files.
- Click on "Apply and Export all". All the images are segmented and exported. Check "Export Limited Parm" check box if you dont want to edit segmentation in future. This reduces size of the output file and saves space in hard disk.



Ind Size: 20
Min Size: 10
(● Original ○ New)
[✓] Extract Spatial profile
# of Images loaded: 0
Outfile prefix: C01_Zyxin
[✓] Name by frame
[✓] Extract Spatial profile
[ ] Export Limited Param.

Figure 8.5: Segmentation tool -

```
function varargout = FA_SegmentationTool(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @FA_SegmentationTool_OpeningFcn, ...
    'gui_OutputFcn',  @FA_SegmentationTool_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function FA_SegmentationTool_OpeningFcn(hObject, eventdata, handles, varargin)
handles.flnm = '';
handles.flnm2 = '';
tmp = dir('fa_analysistool_log.mat');
if(size(tmp,1))
    load('fa_analysistool_log.mat');
    handles.flnm = tmp{1};
    handles.flnm2 = tmp{2};
end
```

```
handles.rbsel2 = 1;
handles.cmp = colormap(jet(512));
handles.cmp = handles.cmp(randperm(512),:);
handles.cmp(1,:)= [ 1 1 1];
handles.figname = 'figure';
addpath('/Users/mrsheriff/Documents/MATLAB/Ruth_related/Water');
handles.output = hObject;
guidata(hObject, handles);
function varargout = FA_SegmentationTool_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function load_img_Callback(hObject, eventdata, handles)
[handles.imgn, handles.flnm] = uigetfile('*.tif', 'Choose a TIFF image for FA ...
    Analysis',handles.flnm);
if(handles.flnm)
    threshold = get_threshold(handles.imgn,handles.flnm);
    handles.img = double(imread( fullfile(handles.flnm ,  handles.imgn)));
    if(threshold)
            handles.img(handles.img ≤ threshold) = 0;
    end
    set(handles.text1,'String', handles.imgn);
    set(handles.statusbar,'String',[' Image Read : ' fullfile(handles.flnm ,  ...
        handles.imgn)]);
    handles.pimg = [];
    handles.bw = ones(size(handles.img));
    guidata(hObject, handles);
end
function edit_is_Callback(hObject, eventdata, handles)
function edit_is_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit_ms_Callback(hObject, eventdata, handles)
function edit_ms_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function app_water_Callback(hObject, eventdata, handles)
set(handles.statusbar,'String', 'Water algorithm being applied on image. ...
    Please wait.....',...
    'BackgroundColor',[0.8 0 0])
pause(0.001);
is = str2num(get(handles.edit_is,'String'));
ms = str2num(get(handles.edit_ms,'String'));
[tpimg,tptch] = water(handles.img,is,ms);
[handles.pimg,handles.ptch] = sortwater(tpimg,tptch);
set(handles.statusbar,'String', 'Focal Adhesions identified and Image ...
    displayed.',...
    'BackgroundColor',[0 0.8 0]);
disp_img(handles.img,handles.pimg,handles.ptch,1,handles.cmp);
```

```
set(handles.text1,'ForegroundColor',[0 0 0.8]);
guidata(hObject, handles);
function show_img_Callback(hObject, eventdata, handles)
if(get(handles.rb_o,'Value'))
    disp_img(handles.img,handles.pimg,handles.ptch,1,handles.cmp);
else
    disp_img(handles.img,handles.npimg,handles.nptch,2,handles.cmp);
end
guidata(hObject, handles);
function poly_Callback(hObject, eventdata, handles)
disp_img(handles.img,handles.pimg,handles.ptch,1,handles.cmp);
switch get(hObject,'Tag')
    case 'in_poly'
        bw = roipoly();
        img  = handles.pimg(bw);
        lst = unique(img);
        lst = setdiff(lst,0);
        handles.ptch(lst,7) = 0;
        handles.bw = handles.bw .* (¬bw);
    case 'out_poly'
        bw = roipoly();
        img  = handles.pimg(¬bw);
        lst = unique(img);
        lst = setdiff(lst,0);
        handles.ptch(lst,7) = 0;
        handles.bw = handles.bw .* bw;
    case 'include_poly'
        bw = roipoly();
        img  = handles.pimg(bw);
        lst = unique(img);
        lst = setdiff(lst,0);
        handles.ptch(lst,7) = 1;
        handles.bw = handles.bw + bw;
    case 'remove_poly'
        handles.ptch(:,7) = 1;
        handles.bw = ones(size(handles.bw));
    case 'load_poly'
        [fname, pname] = uigetfile('*.mat','Choose a file to apply its ...
            polygon',handles.flnm2);
        temp = load(fullfile(pname,fname));
        bw = temp.bw;
        img  = handles.pimg(¬bw);
        lst = unique(img);
        lst = setdiff(lst,0);
        handles.ptch(lst,7) = 0;
        handles.bw = handles.bw .* bw;
end
guidata(hObject, handles);
disp_img(handles.img,handles.pimg,handles.ptch,1,handles.cmp);
function statusbar_Callback(hObject, eventdata, handles)
function statusbar_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function export_water_Callback(hObject, eventdata, handles,fname)
if(¬exist('fname','var'))
[fname,handles.flnm2] = uiputfile('*.mat','Save old and new Water Images and ...
    Parameters',handles.flnm2);
end
if(fname)
    img = handles.img;
    pimg = handles.pimg;
    ptch = handles.ptch;
    npimg = handles.npimg;
    nptch = handles.nptch;
    imgname = handles.imgn;
    bw = handles.bw;
    if(get(handles.spacialprofile1,'Value'))
        FAQ = quantMajorAxis(img,npimg,bw); %MajorAxisQuat
    else
        FAQ = [];
    end
    indsz = str2num(get(handles.edit_is,'String'));
    minsz = str2num(get(handles.edit_ms,'String'));
    table = get(handles.uitable1,'Data');
    if(¬exist('fname','var'))
        save(fullfile(handles.flnm2,fname), 'img', 'pimg', 'ptch', 'indsz', ...
            'minsz', 'npimg', 'nptch', 'table', 'imgname', 'bw','FAQ');
    else
        table = [];
        save(fullfile(handles.flnm2,fname), 'img', 'pimg', 'ptch', 'indsz', ...
            'minsz', 'npimg', 'nptch', 'table', 'imgname','bw','FAQ');
    end
    handles.figname = fname;
    guidata(hObject,handles);
    set(handles.statusbar,'String', 'Normal and newly segmented Water Output ...
        files are Exported.',...
        'BackgroundColor',[0 0.8 0]);
end
function load_water_Callback(hObject, eventdata, handles)
[fname, handles.flnm2] = uigetfile('*.mat', 'Choose the file to load water ...
    image and parameters',handles.flnm2);
if(fname)
    in = load (fullfile(handles.flnm2,fname));
    if(strcmp(handles.imgn,in.imgname))
        handles.pimg = in.pimg;
        handles.ptch = in.ptch;
        set(handles.edit_is,'String', num2str(in.indsz));
        set(handles.edit_ms,'String', num2str(in.minsz));
        set(handles.text1,'ForegroundColor',[0 0 0.8]);
        set(handles.statusbar,'String', 'Water Images and Parameters are ...
            Imported',...
            'BackgroundColor',[0 0.8 0]);
    else
        msgbox(sprintf('Input images are not same. Loaded input file is %s', ...
            in.imgname),'Source Image Conflict','warn','modal');
    end
    guidata(hObject,handles);
end
function create_newseg_Callback(hObject, eventdata, handles)
disdat = get(handles.uitable1,'Data');
nod = find(strcmp(disdat(:,2),'#'))-1; %use # set the end point for the patch ...
    number used
if isempty(nod)
    nod = size(disdat,1);
end
temp = (disdat(1:nod,2));
newa = find_rep(temp);
if(newa)
    msgbox(sprintf('Following numbers are repeated %s. Please correct and ...
        repeat', num2str(newa)),'Focal Adhesion repetation','warn','modal');
    set(handles.statusbar,'String', sprintf('Repeats : %s .',num2str(newa)),...
        'BackgroundColor',[0.8 0 0]);
else
    [npimg,nptch] = newsegmentation(handles.pimg,handles.ptch,disdat(1:nod,2));
    [handles.npimg,handles.nptch] = sortwater(npimg,nptch);
    set(handles.statusbar,'String', 'New Water Image and Parameters generated ...
        from table.',...
        'BackgroundColor',[0 0.8 0]);
    guidata(hObject, handles);
    disp_img(handles.img,handles.npimg,handles.nptch,2,handles.cmp);
end
function newa = find_rep(temp)
coll = [];
for ia = 1:size(temp,1), coll = [coll str2num(temp{ia})];end
sa = sort(coll);
[¬,idx1] = unique(sa,'first');
[¬,idx2] = unique(sa,'last');
newa = setdiff(sa(setdiff(idx1,idx2)),0);
function load_imgw_Callback(hObject, eventdata, handles)
tg = get(hObject,'Tag');
n = str2num( tg(end));
nm = '23';
switch n
    case { 1 , 2}
        [fname, handles.flnm2] = uigetfile('*.mat', 'Choose the file to load ...
            water image and parameters',handles.flnm2);
        if(fname)
            handles.in(n) = load (fullfile(handles.flnm2,fname));
            eval(['set(handles.text' nm(n) ',''String'', [fname '' => '' ...
                handles.in(n).imgname], ''ForegroundColor'',[0 0 0.8]);']);
            set(handles.statusbar,'String', 'Water Images and Parameters are ...
                Imported',...
                'BackgroundColor',[0 0.8 0]);
            handles.imgwn(n) = {fname};
            guidata(hObject,handles);
        end
    case 3
        [handles.imgwn, handles.flnm2] = uigetfile('*.mat', 'Choose a TIFF ...
            image for FA Analysis in the order',handles.flnm2,'MultiSelect','on');
        if(handles.flnm2)
            if(iscell(handles.imgwn) && size(handles.imgwn,2)≥2)
                if(size(handles.imgwn,2) > 2)
                    handles.imgn(3:end) = [];
                end
                for ia = 1:2
                    handles.in(ia) = load ...
                        (fullfile(handles.flnm2,handles.imgwn{ia}));
                    eval(['set(handles.text' nm(ia) ',''String'', ...
                        [handles.imgwn{ia} '' => '' handles.in(ia).imgname], ...
                        ''ForegroundColor'',[0 0 0.8]);']);
                end
                set(handles.statusbar,'String', 'Both Water Images and ...
                    Parameters are Imported',...
                    'BackgroundColor',[0 0.8 0]);
                guidata(hObject, handles);
            else
                warndlg('Please Choose 2 water files in described order. ...
                    Previous selection(s) are ignored','Files Missing','replace');
            end
        end
end
function show_Img2_Callback(hObject, eventdata, handles)
n = handles.rbsel2;
switch n
    case {1,2}
        disp_img(handles.in(n).img,handles.in(n).npimg, handles.in(n).nptch, ...
            n, handles.cmp);
    otherwise
        for ia = 1:2
            disp_img(handles.in(ia).img,handles.in(ia).npimg, ...
                handles.in(ia).nptch, ia, handles.cmp);
        end
end
function disp_export_data_Callback(hObject, eventdata, handles)
dismat = get(handles.uitable1,'Data');
nod = str2num(get(handles.data_until,'String'));
temp1 = (dismat(1:nod,1));
newa1 = find_rep(temp1);
temp2 = (dismat(1:nod,2));
newa2 = find_rep(temp2);
if(¬isempty(newa1)|| ¬isempty(newa2))
    msgbox(sprintf('Following numbers are repeated. Image 1: %s. Image 2:%s ...
        Please correct and repeat', num2str(newa1),num2str(newa2)),'Focal ...
        Adhesion repetation','warn','modal');
    set(handles.statusbar,'String', sprintf('Image 1: %s . Image 2: ...
        %s',num2str(newa1),num2str(newa2)),...
        'BackgroundColor',[0.8 0 0]);
else
    [handles.in1(1).fpimg, handles.in1(1).fptch, handles.in1(2).fpimg, ...
        handles.in1(2).fptch, mtable] ...
        = newseg4matchpair(handles.mtch, handles.in(1).npimg, ...
            handles.in(1).nptch, handles.in(2).npimg, handles.in(2).nptch, ...
            dismat(1:nod,:));
    [¬,name{1},¬,¬] = fileparts(handles.imgwn{1});
    [¬,name{2},¬,¬] = fileparts(handles.imgwn{2});
    if(¬isempty(strfind(name{1},'bait')) && ¬isempty(strfind(name{2},'prey')))
        save(fullfile(handles.flnm2,sprintf('Match_Table_%s_vs_%s', name{1}, ...
            name{2})), 'dismat', 'nod', 'mtable', 'name');
    else
        save(fullfile(handles.flnm2,sprintf('Match_%s_vs_%s',name{1}, ...
            name{2})),'dismat', 'nod', 'mtable', 'name');
    end
    set(handles.statusbar,'String', sprintf(' Focal Adhesions Match Data ...
        Exported'),...
        'BackgroundColor',[0 0.8 0]);
    guidata(hObject,handles);
end
function disp_img(img,pimg,ptch,ind,cmp)
figure(70+ind);
image(pimg+1);
colormap(cmp);
axis image
for ia = 1: size(ptch,1)
    text(round(ptch(ia ,6)),round(ptch( ia,5)), num2str(ptch(ia,1)), ...
        'FontSize',8, 'Color','k');
end
y = ptch(ptch(:,7)==1,5);
x = ptch(ptch(:,7)==1,6);
y1 = ptch(ptch(:,7)==0,5);
x1 = ptch(ptch(:,7)==0,6);
figure(80+ind);
clf;
imagesc(img);
colormap(gray);
axis image
hold on;
plot(x,y,'.','Color','g','MarkerSize',10);
hold on;
plot(x1,y1,'.','Color','r','MarkerSize',10);
function disp_mat(handles,ptch,ind)
disdat(:,1) = ptch(ptch(:,7)==1,1);
set(handles.uitable1,'Data',disdat);
```

```
function start_seg_Callback(hObject, eventdata, handles)
lst = num2str(handles.ptch(handles.ptch(:,7)==1,1));
disdat = cell(size(lst,1),1);
for ia = 1:size(lst,1)
    disdat(ia,1) = {lst(ia,:)};
end
disdat(:,2) = disdat(:,1);
set(handles.ntag1,'String', 'FA #');
set(handles.ntag2,'String', 'New FA');
set(handles.uitable1,'Data',disdat);
if(isfield(handles,'hFig'))
    if(ishandle(handles.hFig))
        close(handles.hFig);
    end
end
handles.hFig = comparison_tool (handles.img,handles.pimg,handles.cmp, ...
    handles.ptch);
guidata(hObject, handles);
function rb2selection_SelectionChangeFcn(hObject, eventdata, handles)
handles.rbsel2 = str2num(get(eventdata.NewValue,'TooltipString')) ;
guidata(hObject, handles);
function figure1_CloseRequestFcn(hObject, eventdata, handles)
tmp{1} = handles.flnm;
tmp{2} = handles.flnm2;
save('fa_analysistool_log.mat','tmp');
delete(hObject);
function sav_img_Callback(hObject, eventdata, handles)
figname = [fullfile(handles.flnm2,handles.figname) '.tiff'];
print('-dtiff', figname,'-f81');
set(handles.statusbar,'String', sprintf(' Original Figure with red/green dots ...
    for FAs is exported'),...
    'BackgroundColor',[0 0.8 0]);
function load_mimages_Callback(hObject, eventdata, handles)
[handles.imgn_lst, handles.flnm] = uigetfile('*.tif', 'Choose a TIFF image for ...
    FA Analysis',handles.flnm,'MultiSelect','on');
if(ischar(handles.imgn_lst))
    set(handles.edit6,'String', '1');
    handles.imgn_lst = {handles.imgn_lst};
else
    set(handles.edit6,'String', num2str(length(handles.imgn_lst)));
end
if(handles.flnm)
    if(¬isempty(get(handles.inpstr,'String')))
        d = dir(fullfile(handles.flnm,['*' get(handles.inpstr,'String') ...
            '*.tif']));
        handles.imgn_lst = {d.name};
        set(handles.edit6,'String', num2str(length(handles.imgn_lst)));
    end
    for ia = size(handles.imgn_lst,2):-1:1
        threshold = get_threshold(handles.imgn_lst{ia});
        img = double(imread( fullfile(handles.flnm ,  handles.imgn_lst{ia})));
```

```
            if(threshold)
            img(img ≤ threshold) = 0;
            end
            handles.img_lst(:,:,ia) = img;
            set(handles.text1,'String', handles.imgn_lst{ia});
            set(handles.statusbar,'String',[' Image Read : ' fullfile(handles.flnm ...
                , handles.imgn_lst{ia})]);
        end
        handles.img = handles.img_lst(:,:,1);
        handles.imgn = handles.imgn_lst{1};
        handles.pimg = [];
        handles.bw = ones(size(handles.img));
        guidata(hObject, handles);
    end
    function edit6_Callback(hObject, eventdata, handles)
    function edit6_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    function load_waternpoly4all_Callback(hObject, eventdata, handles)
    [fname, handles.flnm2] = uigetfile('*.mat', 'Choose the file to load water ...
        image and parameters',handles.flnm2);
    if(fname)
        in = load (fullfile(handles.flnm2,fname));
            set(handles.edit_is,'String', num2str(in.indsz));
            set(handles.edit_ms,'String', num2str(in.minsz));
            set(handles.text1,'ForegroundColor',[0 0 0.8]);
            handles.bw = in.bw;
            set(handles.statusbar,'String', 'Water Parameters and Polygons are ...
                Imported',...
                'BackgroundColor',[0 0.8 0]);
        guidata(hObject,handles);
    end
    function [pimg,ptch] = apply_poly(pimg,ptch,bw)
    img  = pimg(¬bw);
    lst = unique(img);
    lst = setdiff(lst,0);
    ptch(lst,7) = 0;
    function apply_waterparmpoly4all_Callback(hObject, eventdata, handles)
    namp = [get(handles.fprefix,'String') '_'];
    if(get(handles.namebyframe,'Value'))
        namesuf = reshape(sprintf('%04d',1:500),4,[])';
        namesuf(:,1) = 't';
        namesuf = cellstr(namesuf);
    else
        namesuf = {'-02','-04','-05','00','05','10','15','20', '25','30', ...
            'C00','C05','C10','C15','C20', 'C25','C30'};
    end
    for ia = 1: length(handles.imgn_lst)
        handles.img = handles.img_lst(:,:,ia);
        set(handles.text1,'String', handles.imgn_lst{ia});
        app_water_Callback(handles.app_water, eventdata, handles);
        handles = guidata(hObject);
        [handles.pimg,handles.ptch] = ...
            apply_poly(handles.pimg,handles.ptch,handles.bw);
        disp_img(handles.img,handles.pimg,handles.ptch,1,handles.cmp);
        if(get(handles.explim,'Value'))
            handles.npimg = handles.pimg;
            handles.nptch = handles.ptch;
            handles.pimg = [];
            handles.ptch = [];
            fname = [namp namesuf{ia}];
        else
            disp_img(handles.img,handles.pimg,handles.ptch,1,handles.cmp);
            start_seg_Callback(handles.start_seg, eventdata, handles);
            handles = guidata(hObject);
            create_newseg_Callback(handles.create_newseg, eventdata, handles);
            handles = guidata(hObject);
        end
        fname = [namp namesuf{ia}]; %this is used for Ruths
        export_water_Callback(handles.export_water, eventdata, handles,fname);
        handles = guidata(hObject);
        sav_img_Callback(handles.sav_img, eventdata, handles)
    end
    function spacialprofile1_Callback(hObject, eventdata, handles)
    set(handles.spacialprofile,'Value',get(hObject,'Value'));
    function namebyframe_Callback(hObject, eventdata, handles)
    function spacialprofile_Callback(hObject, eventdata, handles)
    set(handles.spacialprofile1,'Value',get(hObject,'Value'));
    function inpstr_Callback(hObject, eventdata, handles)
    function inpstr_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), ...
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
    function explim_Callback(hObject, eventdata, handles)
```

8.3.1.1 Implementation of water algorithm

```
function [pimg , ptch] = water(img,ma,ta)
img  = double(img);
m = size(img,1);
n = size(img,2);
fct = 4;  %control this for balance between memonry and speed
ptch = zeros(round(m*n/fct),2);
pp = zeros(round(m*n/fct),ma*50);  %Pixels inded list
img = [ zeros(m,1),img,zeros(m,1)];
img = [zeros(1,n+2);img; zeros(1,n+2)];
img = reshape(img,1,(m+2)*(n+2));
[spxl(:,1), spxl(:,2)] = sort(img,'descend');
spxl = spxl(spxl(:,1) ≠ 0,:);
pimg = zeros(1,(m+2)*(n+2));
pt = 1;
ptch(1,:)  = [pt 1]; %patch
pimg(spxl(1,2)) = pt;
pp(pt,1) = spxl(1,2);
sz = size(spxl,1);
sza = size(spxl,1);
for ia = 2:sz
    if mod(ia,1000)==0
        ia/sza
    end
    ind = spxl(ia,2);
    pimg_n = [pimg(ind−1) pimg(ind+1) pimg(ind−m−2) pimg(ind+m+2)];
    z = pimg_n(pimg_n ≠ 0);
    sz = size(z,2);
    rz = [];
    for ic = 1:(sz−1)
        for ib = (ic + 1):sz
            if(z(ic) == z(ib))
                rz = [rz ib];
            end
        end
    end
    z(rz) = [];
    siz = size(z,1)+ size(z,2) −1; %this is Bcoz of matlabs nature
    switch siz
        case 0
            pt = pt + 1;
            z = pt;
            ptch(pt,:) = [z 1];
        case 1
            ptch(z,2) =  ptch(z,2) + 1;
        otherwise
            flag = 1;
            z = ptch(z,:);
            z = sortrows(z,2);
            z(:,2) = [];
            while(flag == 1)
                if(((ptch(z(1),2) < ma) || (ptch(z(2),2)) < ma))
                    pt = pt + 1;
                    ptch(pt,:) = [pt ptch(z(1),2)+ptch(z(2),2)];
                    p = [ pp(z(1),:) pp(z(2),:)];  %pool the points(linear ...
                        index) under both patches
                    p(p<1)= [];                   %Remove unwanted zeros of pool ...
                        came from the matrix
                    ptch(z(1),2) = 0;
                    ptch(z(2),2) = 0;
                    pimg(p) = pt;
                    sz = size(p,2);
                    for ic = sz:-1:1
                        pp(pt,ic) = p(ic);
                    end
                    z(2) = [];
                    z(1) = pt;
                    if(size(z,1) < 2)
                        flag = 0;
                    end
                else
                    ptch(z(1),2) =  ptch(z(1),2);
                    flag = 0;
                end
            end
            z = z(1);
            ptch(z,2) =  ptch(z,2) + 1;
    end
    pp(z, ptch(z,2)) = ind;
    pimg(ind) = z;
end
ptch(ptch(:,2) == 0,:) = [];
sptch = ptch(ptch(:,2) < ta,:);
pp = pp(1:ptch(end,1),:);
pp(pp<1) = 1;
nd = size(sptch,1);
for ia = 1:nd
    pimg(pp(sptch(ia,1),:)) = 0;
end
ptch = setxor(ptch,sptch,'rows');
nd = size(ptch,1);
for ia = 1:nd
    pimg(pp(ptch(ia,1),:)) = ia;
    sp = pp(ptch(ia,1),:) ;  %sub set of pixels index
    sp = sp(sp ≠ 1);
    ptch(ia,3) = sum(img( sp ));    %Integrated Intensity
    ptch(ia,4) = ptch(ia,3)/ptch(ia,2);
    xx = mod(sp,m+2) -1; % -1 to compensate the borders added
    yy = floor(sp/(m+2)); % + 1 - 1 % ignored but same as above
    ptch(ia,5)  = sum( xx .* img(sp) )/ptch(ia,3);
    ptch(ia,6)  = sum( yy .* img(sp) )/ptch(ia,3);
    ptch(ia,1)= ia;
end
ptch(:,7) = ones(nd,1);
pimg  = reshape(pimg, m+2, n+2);
pimg(1,:) = [];
pimg(m+1,:) = [];
pimg(:,1) = [];
pimg(:,n+1) = [];
end
```

```
function [npimg, nptch]= sortwater(pimg,ptch)
[m n] = size(pimg);
idximg = zeros(m,n);
idximg(1: m*n) =  1:m*n;
bimg = [pimg(:) idximg(:)];
bimg = bimg(bimg(:,1)≠0,:);
ord = sortrows(bimg);
[¬,idx] = unique(ord(:,1),'first');
ord = ord(idx,:);
ord = sortrows(ord,2);
npimg = zeros(size(pimg));
for ia = 1:size(ord,1)
    npimg(pimg == ord(ia,1)) = ia;
end
if(exist('ptch','var'))
    nptch = zeros(size(ptch));
    for ia = 1:size(ord,1)
        nptch(ia,:) = ptch(ord(ia,1),:);
    end
    nptch(:,1) = 1:ia;
else
    nptch = [];
end
```

```
function [npimg,nptch] = newsegmentation(pimg,ptch,table)
nlst = [];
mlst = {};
for ia = 1:size(table,1)
    p = str2num(table{ia});
    if((size(p,2) > 1 ))
        mlst(end+1,1) = {p};    % to be merged
    elseif (p)
        nlst(end+1,1) = p; %chosen normal
    end
end
nptch = ptch(nlst,:);
if(¬isequal(mlst,cell(0)))
    for ia = 1:size(mlst,1)
        p = cell2mat(mlst(ia,1));
        nptch = addpatch_S(ptch,nptch,p);
    end
end
nptch(1:end,1) = 1:size(nptch,1);
npimg = newpimg(pimg,nlst,mlst);
function nptch = addpatch_S(ptch,nptch,p)
n = size(nptch,1)+1;
nptch(n,1) = NaN;
if(p(1) ==0)
    nptch(n,2:7) =0;
```

```
else
nptch(n,2) = sum(ptch(p,2));
nptch(n,3) = sum(ptch(p,3));
nptch(n,4) = nptch(n,3)/nptch(n,2);
nptch(n,5) = sum(ptch(p,3).* ptch(p,5))/nptch(n,3);
nptch(n,6) = sum(ptch(p,3).* ptch(p,6))/nptch(n,3);
nptch(n,7) = 1;
end
function npimg = newpimg(pimg,nlst,mlst)
jj = 1;
npimg = zeros(size(pimg));
    ind = 1;
    for ia = 1:size(nlst,1)
        npimg(pimg == nlst(ia,1)) = ind;
        ind = ind + 1;
    end
    for ib = 1:size(mlst,1)
        p = cell2mat(mlst(ib));
        for ic = 1:size(p,2)
              if(p(ic))
                npimg(pimg == p(ic)) = ind;
              end
        end
        ind = ind + 1;
    end
```

```
function pimg = waterlight(img,ma,ta)
img  = double(img);
m = size(img,1);
n = size(img,2);
ptch = zeros(round(m*n/2),2);
pp = zeros(round(m*n/2),ma*10);  %Pixels inded list
img = [ zeros(m,1),img,zeros(m,1)];
img = [zeros(1,n+2);img; zeros(1,n+2)];
img = reshape(img,1,(m+2)*(n+2));
[spxl(:,1), spxl(:,2)] = sort(img,'descend');
spxl = spxl(spxl(:,1) ≠ 0,:);
pimg = zeros(1,(m+2)*(n+2));
pt = 1;
ptch(1,:)  = [pt 1]; %patch
pimg(spxl(1,2)) = pt;
pp(pt,1) = spxl(1,2);
sz = size(spxl,1);
sza = size(spxl,1);
for ia = 2:sz
    if mod(ia,1000)==0
        ia/sza
    end
    ind = spxl(ia,2);
    pimg_n = [pimg(ind−1) pimg(ind+1) pimg(ind−m−2) pimg(ind+m+2)];
    z = pimg_n(pimg_n ≠ 0);
    sz = size(z,2);
    rz = [];
    for ia = 1:(sz−1)
        for ib = (ia + 1):sz
            if(z(ia) == z(ib))
                rz = [rz ib];
            end
        end
    end
    z(rz) = [];
    siz = size(z,1)+ size(z,2) −1; %this is Bcoz of matlabs nature
    switch siz
        case 0
            pt = pt + 1;
            z = pt;
            ptch(pt,:) = [z 1];
        case 1
            ptch(z,2) =  ptch(z,2) + 1;
        otherwise
            flag = 1;
            z = ptch(z,:);
            z = sortrows(z,2);
            z(:,2) = [];
            while(flag == 1)
                if(((ptch(z(1),2) < ma) || (ptch(z(2),2)) < ma))
                    pt = pt + 1;
                    ptch(pt,:) = [pt ptch(z(1),2)+ptch(z(2),2)];
                    p = [ pp(z(1),:) pp(z(2),:)];  %pool the points(linear ...
                        index) under both patches
                    p(p<1)= [];                 %Remove unwanted zeros of pool ...
                        came from the matrix
                    ptch(z(1),2) = 0;
                    ptch(z(2),2) = 0;
                    pimg(p) = pt;
                    sz = size(p,2);
                    for ia = sz:−1:1
                        pp(pt,ia) = p(ia);
                    end
                    z(2) = [];
                    z(1) = pt;
                    if(size(z,1) < 2)
                        flag = 0;
                    end
                else
                    ptch(z(1),2) =  ptch(z(1),2);
                    flag = 0;
                end
            end
            z = z(1);
            ptch(z,2) =  ptch(z,2) + 1;
    end
    pp(z, ptch(z,2)) = ind;
    pimg(ind) = z;
end
ptch(ptch(:,2) == 0,:) = [];
sptch = ptch(ptch(:,2) < ta,:);
pp(pp<1) = 1;
nd = size(sptch,1);
for ia = 1:nd
    pimg(pp(sptch(ia,1),:)) = 0;
end
ptch = setxor(ptch,sptch,'rows');
nd = size(ptch,1);
for ia = 1:nd
    pimg(pp(ptch(ia,1),:)) = ia;
end
pimg  = reshape(pimg, m+2, n+2);
pimg(1,:) = [];
pimg(m+1,:) = [];
pimg(:,1) = [];
pimg(:,n+1) = [];
```

8.3.2 Object tracking tool

- Run Matlab GUI “GlobalFAMatch.fig”.
- Click on “Load” , a window will popup which allows user to type “strings” to filter the names of the segmented image data from different channels.
- Visualize the segmentation movie by scrolling the time and channel slider.
- Type the ID of the channels (the order of the channel strings typed viz. 1,2,3, or 4) in the textbox “for channels” and click “create common mask” button to create a common segmented image for each time frame by merging multicolor segmentation. To exclude a channel, remove it from the textbox “for channels”
- Click “Load cMask” button. If common mask is already created, skip the previous step and directly click this button.
- To avoid considering any problematic time frames with inaccurate segmentation and common mask, type the number of the time points in the editbox “Except”. To consider all the time points for tracking leave this editbox empty.
- Click “Start matching all”. This step tracks all segmented objects along time and map them with similar color. To visualize this click the radiobutton “Common

Patch Image for Matching".

- Click on "allow matching" . This displays ID over all the tracked objects. The table displays the ID of the matched objects in consecutive time frames.
- First column of the table contains object ID at time t_n and the second column displays the ID of correspondingly matched object at t_{n+1}. To correct, edit the IDs in the table ascending along the time frames.
- "switch images" button allows rapid switching between two frames to visualize and carefully monitor tracking of objects.
- Select radiobutton "All" in the export/load panel and click on "Export matches" to export the pair-wise matching Click "generate big table". This will implement the correction in the tracking if any and generates a final tracked movie of objects.
- Click "Quantify Major Axis" only if required, otherwise skip this step.
- Click "export finalized water" to export the final Matlab output file and color coded segmentation and tracking movie.

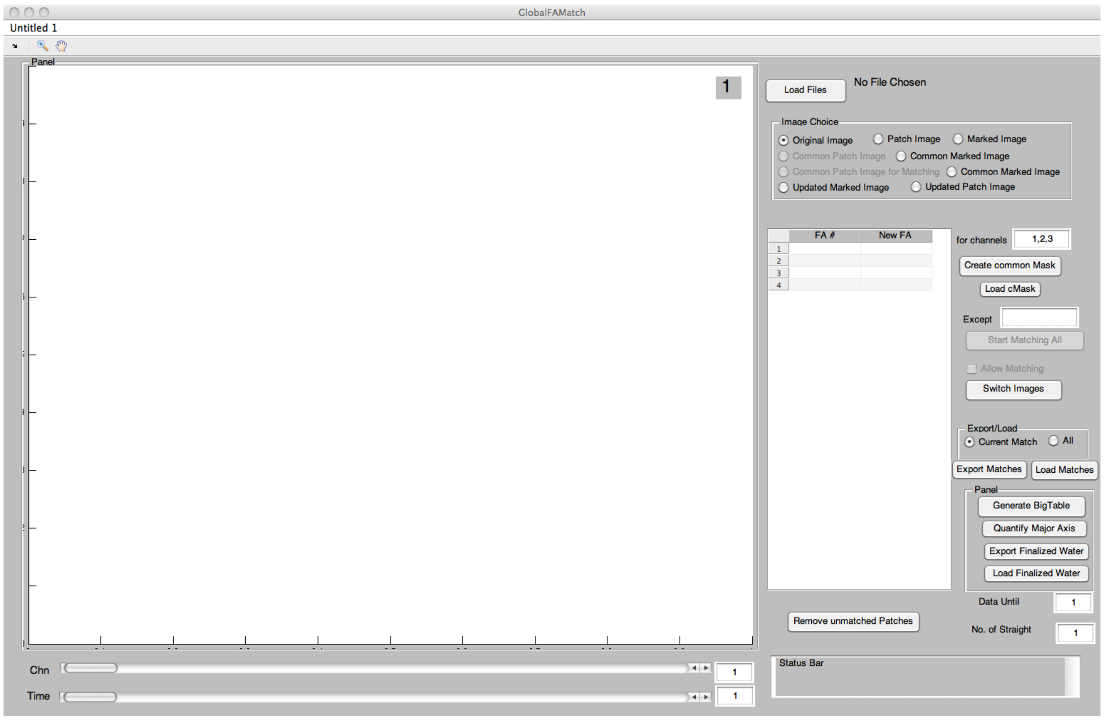


Figure 8.6: Tracking tool -

```matlab
function varargout = GlobalFAMatch(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @GlobalFAMatch_OpeningFcn, ...
    'gui_OutputFcn',  @GlobalFAMatch_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function GlobalFAMatch_OpeningFcn(hObject, eventdata, handles, varargin)
handles.flnm = '';
handles.flnm2 = '';
tmp = dir('matchFAtool_log.mat');
if(size(tmp,1))
    load('matchFAtool_log.mat');
    handles.flnm = tmp{1};
    handles.flnm2 = tmp{2};
end
clmpsz = 1024;
handles.cmp = colormap(jet(clmpsz));
handles.cmp = handles.cmp(randperm(clmpsz),:);
handles.cmp(1,:)= [ 1 1 1];
handles.m1 = [1,1];          % set of channel chosen
handles.n1 = [1,2];          % Set of time points in the chosen channel ...
    respectively
handles.idx.isinterchn = 0; % if the match chosen is inter channel or not. 0 ...
    -> Not, 1 -> Yes
handles.idx.m2 = 1;          % the position of match parameters object, for ...
    either inter-match or inter-time point
handles.idx.t2 = 1;          % the position of time parameters for matches
handles.imchoice = 'oi';     %Choice of image displayed
addpath('/Users/mrsheriff/Documents/MATLAB/Ruth_related/Water');
handles.output = hObject;
guidata(hObject, handles);
function varargout = GlobalFAMatch_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function slider_Callback(hObject, eventdata, handles)
m = get(handles.chnslider,'Value');
n = get(handles.tslider,'Value');
after_slider_text_Callback(hObject,handles,m,n);
handles = guidata(hObject);
guidata(hObject, handles);
function after_slider_text_Callback(hObject,handles,m,n)
m = checkrange(m,handles.nchn);
```

```
n = checkrange(n,handles.nt);
set(handles.chnslider,'Value',m);
set(handles.tslider,'Value',n);
set(handles.chn_edit,'String',num2str(m));
set(handles.time_edit,'String',num2str(n));
if(get(handles.allow_match,'Value')) %if the allow matching is checked
    update_dismat(hObject,handles);  %to update the displayed table
    handles = guidata(hObject);
end
[handles.m1,handles.n1]=comparison_images(handles.m1, handles.n1,m,n); %find ...
    the images of comparison based on slider position
guidata(hObject, handles);
disp_images(hObject,handles,m,n);
function [m1,n1] = comparison_images(m1,n1,m,n)
sets = sortrows([m1; n1]');
if (¬ismember( [m n] , sets,'rows'))
    if(n≠1)
        m1 = [ m m];
        n1 = [n−1 n];
    elseif(¬ismember(m,m1))
        m1(2) = m;
        if(m ≠1)
            m1(1) = m−1;
        end
        m1 = sort(m1);
        n1(2) = n;
    else
        m1 = [m  m];
        n1 = [n n+1];
    end
end
function [handles,img,clmp,dlst] = updated_patch_Image(handles,m,n)
clmp = handles.cmp;
if(get(handles.allow_match,'Value'))
    if(n>1)                 %choose the right match time point )
        tp = n − 1;         %normally choose timepoint − 1 except for first ...
            one so that the one deleted in previous match can be deleted here too
        to = 2;             %normally choose the img2 in the match except for ...
            first one
    else
        tp = n;
        to = 1;
    end
    tpt = handles.timeptlst(n);
    img = handles.images.cMask_time(tpt).mpimg + 1;
    img(end) = 1024;
    tmp = cellfun(@str2num,handles.dismat(:,n),'UniformOutput',0);
    orglist = [tmp{:}];
    du =  handles.matches.time(tp).data_until;
    tmp = handles.matches.time(tp).upd_dismat(1:du,to);
    tmp = cellfun(@str2num,tmp,'UniformOutput',0);
    upd_lst = [tmp{:}];
    dlst = setdiff(orglist,upd_lst) +1; %list of deleted items (++1 for the ...
        pimg is +1 bcoz of clmp)
    if(dlst)
        clmp(dlst,:) = repmat([1 1 1], length(dlst),1);
        if(n>1)
            du =  handles.matches.time(tp).data_until;
            tmp1 = handles.matches.time(tp).upd_dismat(1:du,1);
            tmp = cellfun(@str2num,tmp1,'UniformOutput',0);
            upd_lst2 = [tmp{:}];
            dlst2 = num2str(setdiff(upd_lst2,upd_lst)');
            for ik = 1:size(dlst2,1)
                handles.matches.time(n).upd_dismat(strcmp(dlst2(ik,:),tmp1),1) ...
                    = {'0'};
            end
        end
    end
else
    img = handles.images.cMask_time(n).mpimg + 1;
    img(end) = 1024;
end
function disp_images(hObject,handles,m,n)
switch handles.imchoice
    case 'oi'
        img = handles.images.ch(m).time(n).img;
        clmp = gray;
        textcolor = 'w';
        gridcolor = 'g';
    case 'pi'
        img = handles.images.ch(m).time(n).npimg + 1;
        clmp = handles.cmp;
        textcolor = 'k';
        gridcolor = 'b';
    case 'mi'
        img = handles.images.ch(m).time(n).npimg;
        outline = imdilate(img,ones(3,3)) > imerode(img,ones(3,3));
        img = handles.images.ch(m).time(n).img;
        img = img ./ max(img(:));
        img = imadjust(img,stretchlim(img),[]);
        textcolor = 'w';
        gridcolor = 'g';
        num = max(img(:))*1.2;
        img(outline) = num;
        clmp = gray;
        clmp(end,:)= [ 1 0 0];
    case 'cp'
        img = handles.images.cMask_time(n).pimg + 1;
        img(end) = 1024;
        clmp = handles.cmp;
        textcolor = 'k';
        gridcolor = 'b';
    case 'cm'                         %cm and mm are exactly same, should check ...
        to minimize in future
        img = handles.images.cMask_time(n).pimg;
        outline = imdilate(img,ones(3,3)) > imerode(img,ones(3,3));
        img = handles.images.ch(m).time(n).img;
        img = img ./ max(img(:));
        img = imadjust(img,stretchlim(img),[]);
        textcolor = 'w';
        gridcolor = 'g';
        num = max(img(:))*1.2;
        img(outline) = num;
        clmp = gray;
        clmp(end,:)= [ 1 0 0];
    case 'mp'
        tpt = handles.timeptlst(n);
        img = handles.images.cMask_time(tpt).mpimg + 1;
        img(end) = 1024;
        clmp = handles.cmp;
        textcolor = 'k';
        gridcolor = 'b';
    case 'mm'
        tpt = handles.timeptlst(n);
        img = handles.images.cMask_time(tpt).mpimg + 1;
        outline = imdilate(img,ones(3,3)) > imerode(img,ones(3,3));
        img = handles.images.ch(m).time(tpt).img;
        textcolor = 'w';
        gridcolor = 'g';
        num = max(img(:))*1.2;
        img(outline) = num;
        clmp = gray;
        clmp(end,:)= [ 0.5 0 0];
    case 'um'
        [handles,img,¬,dlst] = updated_patch_Image(handles,m,n); %clmp not ...
            required here..
        img = img -1;
        img(end) = 0;
        maskimg = im2bw(img,0);
        outline = bwperim(maskimg);
        tpt = handles.timeptlst(n);
        img = handles.images.ch(m).time(tpt).img;
        img = img ./ max(img(:));
        img = imadjust(img,stretchlim(img),[]);
        textcolor = [0 0.5 0];
        gridcolor = 'g';
        num = max(img(:))*1.2;
        img(outline) = num;
        clmp = gray;
        clmp(end,:)= [ 0.5 0 0];
    case 'up'
        textcolor = 'k';
        gridcolor = 'b';
        [handles,img,clmp,dlst] = updated_patch_Image(handles,m,n);
end
axes(handles.axes1);
if(¬isfield(handles,'hSp'))  %initial step to create the scroll image view window
    hIm = imagesc(img);
    colormap(clmp);
    handles.hSp = imscrollpanel(handles.uipanel2,hIm);
    handles.hMagBox = immagbox(handles.uipanel2,hIm);
    pos = get(handles.hMagBox,'Position');
    set(handles.hMagBox,'Position',[0 0 pos(3) pos(4)])
    imoverview(hIm)
    handles.apiI = iptgetapi(handles.hSp);
    axes(handles.axes1);
    axis on;grid minor;
    set(handles.axes1, 'Color', 'none', 'XColor', gridcolor,'YColor', gridcolor);
else
    set(handles.axes1, 'Color', 'none', 'XColor', gridcolor,'YColor', gridcolor);
    handles.apiI.replaceImage(img, 'Colormap', clmp,'PreserveView', 1);
    set(handles.axes1, 'CLimMode', 'auto')
    if(get(handles.allow_match,'Value'))
        if(isfield(handles,'currFA_label'))
            set(handles.currFA_label(:),'Visible','off');    %control handle ...
                for the FA number (text)
        end
        sets = sortrows([handles.m1; handles.n1]');
        n2 = (sets(:,2)==n); %check if the current image is Img1 or Img2 in ...
            the match
        t2 = min(handles.n1);
        tpt = handles.timeptlst(n);
        handles.currFA_label = handles.images.cMask_time(tpt).htext;
        if(get(handles.create_big_table,'Value')==0)
        s_upd_dismat = handles.matches.time(t2).upd_dismat;
        s_data_until = num2str(handles.matches.time(t2).data_until);
        s_nosm = num2str(handles.matches.time(t2).nosm);
        set(handles.uitable1,'Data',s_upd_dismat);
        set(handles.data_until,'String', s_data_until);
        set(handles.no_st_matches,'String', s_nosm);
        end
        if(handles.imchoice(1) == 'u')
            if(dlst)
                if(any(dlst>1))
                    dlst = dlst(dlst>1);
                    handles.currFA_label(dlst−1) = [];
                end
            end
        end
        set(handles.imageorder,'String',num2str(find(n2)));
        set(handles.img1name,'String', ['Chn_' num2str(sets(1,1)) '__t_' ...
            num2str(sets(1,2))]);
        set(handles.img2name,'String', ['Chn_' num2str(sets(2,1)) '__t_' ...
            num2str(sets(2,2))]);
        handles.idx.t2 = t2;
        handles.idx.n2 = n2;
        set(handles.currFA_label(:),'Visible','on');        % control FA ...
            numbers displayed
        set(handles.currFA_label(handles.currFA_label≠0),'Color',textcolor);
    end
end
guidata(hObject, handles);
function load_files_Callback(hObject, eventdata, handles)
prompt = {'Number of Channel:','Channel 1:','Channel 2:','Channel 3:','Channel ...
    4:','Channel 5:'};
dlg_title = 'Name of channels';
num_lines = 1;
def = {'4','C01','C02','C03','C04',''};
chname = inputdlg(prompt,dlg_title,num_lines,def);
if(¬isempty(chname))
    [¬, handles.flnm2] = uigetfile('*.mat', 'Choose the folder containing ...
        Water parameters and Focal Adhesion Match Tables',handles.flnm2);
    if(handles.flnm2)
        pause(0.001);
        handles.nchn = str2double(chname{1});
        for ia = 1:handles.nchn
        path{ia} = fullfile(handles.flnm2, [chname{ia+1} '_*.mat']);
        end
        set(handles.allow_match,'Value',0);
        if(isfield(handles,'currFA_label'))
            set(handles.currFA_label(:),'Visible','off');    %control handle ...
                for the FA number (text)
        end
        for ia = 1:handles.nchn
            handles.d(ia).ds = dir(path{ia});
        end
        handles.nt = size(handles.d(1).ds,1);
        for ia = handles.nt:-1:1
            for ib = 1:handles.nchn
                in = load(fullfile(handles.flnm2,handles.d(ib).ds(ia).name));
                in.htext = [];
                handles.images.ch(ib).time(ia) = in;
            end
        end
        set(handles.tslider,'Max',handles.nt);
        set(handles.tslider,'SliderStep',[1 1]/(handles.nt-1));
        if handles.nchn ≠ 1
            set(handles.chnslider,'Max',handles.nchn);
            set(handles.chnslider,'SliderStep',[1 1]/(handles.nchn-1));
        else
            set(handles.chnslider,'Max',handles.nchn+0.1);
        end
        guidata(hObject,handles);
        set(handles.statusbar,'String', sprintf(' Data has been imported'),...
            'BackgroundColor',[0 0.8 0]);
        set(handles.loadfoldername,'String',handles.flnm2);
        set(handles.allow_match,'Enable','off');
        disp_images(hObject,handles,1,1);
    end
end
function statusbar_Callback(hObject, eventdata, handles)
function figure1_CloseRequestFcn(hObject, eventdata, handles)
tmp{1} = handles.flnm;
tmp{2} = handles.flnm2;
save('matchFAtool_log.mat','tmp');
delete(hObject);
function time_chn_edit_Callback(hObject, eventdata, handles)
m = str2double(get(handles.chn_edit,'String'));
n = str2double(get(handles.time_edit,'String'));
after_slider_text_Callback(hObject,handles,m,n);
handles = guidata(hObject);
guidata(hObject, handles);
function out = checkrange(in,maxv)
out = round(in);
if(out < 1 || out > maxv)
    out = maxv;
end
function update_dismat(hObject,handles)
if(get(handles.create_big_table,'Value') == 0)
handles.matches.time(handles.idx.t2).upd_dismat = get(handles.uitable1,'Data');
handles.matches.time(handles.idx.t2).data_until = ...
    str2double(get(handles.data_until,'String'));
guidata(hObject,handles);
end
function img_choices_SelectionChangeFcn(hObject, eventdata, handles)
l =  get(eventdata.NewValue,'Tag');
handles.imchoice = l(1:2);
m = round(get(handles.chnslider,'Value'));
n = round(get(handles.tslider,'Value'));
disp_images(hObject,handles,m,n);
handles = guidata(hObject);
guidata(hObject, handles);
function create_cMask_Callback(hObject, eventdata, handles)
set(handles.statusbar,'String', ...
    sprintf('Common Mask for Adhesion sites across the channels is being ...
        created!!! Please wait...'),...
    'BackgroundColor',[0.8 0 0]);
pause(0.001);
set(handles.allow_match,'Value',0);
set(handles.start_match,'Enable','Off');
ch4mString = get(handles.channels4mask,'String');
ch4m = str2num(ch4mString);
for ia = 1:handles.nt
    ia
    [pimg ptch] = mergepatch(handles.images,ia,ch4m);
    handles.images.cMask_time(ia).pimg = pimg;
    handles.images.cMask_time(ia).ptch = ptch;
end
cMask_time = handles.images.cMask_time;
save(fullfile(handles.flnm2,'CommonMask.mat'),'cMask_time','ch4mString');
set(handles.start_match,'Enable','On');
set(handles.statusbar,'String', ...
    sprintf('Common Mask for Adhesion sites across the channels is created and ...
        Exported!!!'),...
    'BackgroundColor',[0 0.8 0]);
guidata(hObject, handles);
function load_cMask_Callback(hObject, eventdata, handles)
load(fullfile(handles.flnm2,'CommonMask.mat'));
handles.images.cMask_time = cMask_time;
set(handles.channels4mask,'String',ch4mString);
set(handles.statusbar,'String', sprintf('Common Mask is ...
    Imported!!!'),'BackgroundColor',[0 0.8 0]);
set(handles.cpimg_rb,'Enable','On');
set(handles.start_match,'Enable','On');
guidata(hObject, handles);
function start_match_Callback(hObject, eventdata, handles)
set(handles.statusbar,'String', sprintf('Focal Adhesions pre-Matching ...
    Started!!! Please wait...'),...
    'BackgroundColor',[0.8 0 0]);
pause(0.001);
handles.skiplst = str2num(get(handles.matchskip,'String'));
handles.timeptlst = setdiff(1:handles.nt,handles.skiplst);
handles.nt_c = size(handles.timeptlst,2); %number of timepoints considered for ...
    matching
handles.matches.time = ...
    struct('mtch',[],'nosm',[],'upd_dismat',[],'data_until',[],'mtable',[]);
for ia = 1: handles.nt_c-1 %Start matching for consecutive time points of same ...
    channel
    ia
    bf = handles.timeptlst(ia);
    af = handles.timeptlst(ia+1);
    pimg1 = handles.images.cMask_time(bf).pimg;
    pimg2 = handles.images.cMask_time(af).pimg;
    ptch1 = handles.images.cMask_time(bf).ptch;
    ptch2 = handles.images.cMask_time(af).ptch;
    handles.matches.time(ia) =  sub_start_match_Callback(pimg1,pimg2,ptch1,ptch2);
end
create_big_table_initial(hObject,handles); % or maybe handles = ...
    create_big_table_initial(hObject,handles);
handles = guidata(hObject);
matches = handles.matches;
set(handles.statusbar,'String', sprintf(' FA prematching done for all ...
    images.'),...
    'BackgroundColor',[0 0.8 0]);
set(handles.allow_match,'Enable', 'on');
set(handles.mpatch_rb,'Enable','on');
guidata(hObject, handles);
```

```
function [out] =  sub_start_match_Callback(pimg1,pimg2,ptch1,ptch2)
[out.mtch out.nosm] = matchpatch(pimg1,pimg2,ptch1,ptch2);
out.upd_dismat = {};
out.data_until = [];
out.mtable = [];
function [cMask,ptch] = mergepatch(images,tpt,ch4m)
if(size(ch4m,2) ≠ 1)  %if more than one channel is chosen for mask
    fnimg = zeros(size(images.ch(1).time(tpt).npimg));
    if(isempty(ch4m))
        ch4m = size(images.ch,2):−1:1;
    end
    for ia = ch4m
        img = images.ch(ia).time(tpt).img;
        pimg = images.ch(ia).time(tpt).npimg;
        stats = regionprops(pimg,'PixelIdxList');
        mnimg = zeros(size(img));
        for ib = 1:size(stats,1)
            mnimg(stats(ib).PixelIdxList) = mean(img(stats(ib).PixelIdxList));
        end
        nimg = img./mnimg;
        nimg(isnan(nimg) | nimg == Inf) = 0;
        fnimg = fnimg + nimg;
    end
    for ia = size(images.ch,2):−1:1
        mnsz(ia) = images.ch(ia).time(1).minsz;
        indsz(ia) = images.ch(ia).time(1).indsz;
    end
    [cMask,ptch] = water(fnimg,max(indsz),min(mnsz));
else  %if only one channel is chosen for mask, use the same for common mask
    cMask = images.ch(ch4m).time(tpt).npimg;
    ptch =  images.ch(ch4m).time(tpt).nptch;
end
function create_big_table_initial(hObject,handles)
pftable = genbigindextable_initial(handles.matches);
lpftable = double(¬cellfun(@isempty,pftable));
nofa = size(lpftable,1); %number of focal adhesion
dismat = lpftable .* repmat((1:size(lpftable,1))',1,handles.nt_c);
handles.dismat = {};
for ia = size(lpftable,2):−1:1
    handles.dismat(:,ia) = strtrim(cellstr(num2str(dismat(:,ia))));
end
handles.dismat(2*nofa,end) = {''};
handles.dismat(cellfun(@isempty,handles.dismat)) = {''};
for ia = 1:handles.nt_c
    tpt = handles.timeptlst(ia);
    in = handles.images.cMask_time(tpt);
    [pimg ptch] = finalsegmentation(in.pimg,in.ptch,pftable(:,ia));
    handles.images.cMask_time(tpt).mpimg = pimg;
    handles.images.cMask_time(tpt).mptch = ptch;
end
axes(handles.axes1);
```

```
for ia = handles.nt_c:-1:1
    tpt = handles.timeptlst(ia);
    ptch = handles.images.cMask_time(tpt).mptch;
    for ic = nofa:-1:1
        idx = pftable{ic,ia};
        if(idx)
            handles.images.cMask_time(tpt).htext(ic) = ...
                text(round(ptch(ic,6)),round(ptch(ic,5)), num2str(ic), ...
                'FontSize',12, 'Color','k','Visible','off');
        end
    end
    if(ia < handles.nt_c)
        handles.matches.time(ia).upd_dismat = handles.dismat(:,[ia,ia+1]);
        handles.matches.time(ia).data_until = nofa;
    end
end
guidata(hObject,handles);
function allow_match_Callback(hObject, eventdata, handles)
if(get(handles.allow_match,'Value'))
    m = get(handles.chnslider,'Value');
    n = get(handles.tslider,'Value');
    if(n > handles.nt_c)
        n = 1;
        set(handles.tslider,'Value',1);
        set(handles.time_edit,'String','1');
    end
    [handles.m1,handles.n1]=comparison_images(handles.m1, handles.n1,m,n);
    set(handles.tslider,'Max', handles.nt_c);
    set(handles.tslider,'SliderStep',[1 1]/(handles.nt_c-1));
    disp_images(hObject,handles,m,n);
else
    set(handles.currFA_label(:),'Visible','off');
    set(handles.tslider,'Max',handles.nt);
    set(handles.tslider,'SliderStep',[1 1]/(handles.nt-1));
end
function switch_images_Callback(hObject, eventdata, handles)
if(get(hObject,'Value'))
    k = 1;
else k = 2;
end
sets = sortrows([handles.m1; handles.n1]');
m = sets(k,1);
n = sets(k,2);
set(handles.chn_edit,'String',num2str(m));
set(handles.time_edit,'String',num2str(n));
set(handles.chnslider,'Value',m);
set(handles.tslider,'Value',n);
if(get(handles.allow_match,'Value'))
    update_dismat(hObject,handles);
    handles = guidata(hObject);
    guidata(hObject, handles);
```

```
end
disp_images(hObject,handles,m,n);
function export_matches_Callback(hObject, eventdata, handles)
if get(handles.currentmatch_rb1,'Value')
    sets = sortrows( handles.n1);
    matches = handles.matches.time(handles.idx.t2);     %Check if the matches ...
        are correct for all inter time point matches
    check = check_dismat(matches.upd_dismat,matches.data_until,handles);
    if(check == true)
        out = disp_export_matches(matches, handles.images, handles.d, sets, ...
            handles.flnm2);
        set(handles.statusbar,'String', sprintf('Focal Adhesions Match has ...
            been Exported!!!'),...
            'BackgroundColor',[0 0.8 0]);
    end
else
    set(handles.statusbar,'String', sprintf('Focal Adhesions Match Checking ...
        Started!!! Please wait...'),...
        'BackgroundColor',[0 0.8 0]);
    for ia = (handles.nt_c−1):−1:1
        matches = handles.matches.time(ia);
        check = check_dismat(matches.upd_dismat,matches.data_until,handles);
        if(check == false)
            break;
        end
    end
    set(handles.statusbar,'String', sprintf('Focal Adhesions Matches are ...
        correct!! Now being Exported!!! Please Wait'),...
        'BackgroundColor',[0.8 0 0]);
    if(check == true)
        for ia = (handles.nt_c−1):−1:1
            sets = handles.timeptlst([ ia; ia+1])
            matches = handles.matches.time(ia);
            disp_export_matches(matches,handles.images, handles.d, ...
                sets,handles.flnm2);
        end
        set(handles.statusbar,'String', sprintf('Focal Adhesions Match has ...
            been Exported!!!'),...
            'BackgroundColor',[0 0.8 0]);
    end
end
function out = disp_export_matches(matches,images, dirk, sets,flnm2)
out = [];
nod = matches.data_until;
[¬,name{1}] = fileparts(dirk(1).ds(sets(1)).name);
[¬,name{2}] = fileparts(dirk(1).ds(sets(2)).name);
[¬,idx1] = setdiff(name{1},'bait_');
[¬,idx2] = setdiff(name{2},'bait_');
name{1} = name{1}(idx1);
name{2} = name{2}(idx2);
dismat = matches.upd_dismat;  %need to work from here
```

```
mtable = cellfun(@str2num,dismat(1:nod,:),'UniformOutput',0);
save(fullfile(flnm2, sprintf('Match_(%03d)_%s_vs_%s', ...
     sets(1),name{1},name{2})), 'dismat', 'nod', 'mtable', 'name');
function check = check_dismat(dismat,nod,handles)
temp1 = (dismat(1:nod,1));
[newa1 idx1] = find_rep(temp1);
temp2 = (dismat(1:nod,2));
[newa2 idx2] = find_rep(temp2);
if(¬isempty(newa1)|| ¬isempty(newa2))
    msgbox(sprintf('Following numbers are repeated. Image 1: %s. Image 2:%s ...
        Please correct and repeat', num2str(newa1'),num2str(newa2')),'Focal ...
        Adhesion repetation','warn','modal');
    set(handles.statusbar,'String', sprintf('Image 1: %s . \nImage 2: ...
        %s',idx1,idx2),...
        'BackgroundColor',[0.8 0 0]);
    check = false ;
else
    check = true ;
end
function [newa, idxs] = find_rep(temp)
tmp = char(temp{:});
sz = size(tmp,1);
for ia = size(tmp,1):-1:1
    tt = str2num(tmp(ia,:));
    coll(ia, 1:size(tt,2)) = tt;
end
sa = sort(coll(:));
[¬,idx1] = unique(sa,'first');
[¬,idx2] = unique(sa,'last');
newa = setdiff(sa(setdiff(idx1,idx2)),0);
idx = {};
if(size(newa,2) == 0)
    newa = [];
end
for ia = size(newa,1):-1:1
    k = mod(find(coll==newa(ia))',sz);
    k(k==0) = sz;
    idx{ia} = [ num2str(newa(ia)) ' (' num2str(k) ') ' ];
end
idxs = char(idx)';
idxs = idxs(:)';
function data_until_Callback(hObject, eventdata, handles)
if(get(handles.allow_match,'Value')) %if the allow matching is checked
    update_dismat(hObject,handles);  %to update the displayed table
    handles = guidata(hObject);
    guidata(hObject,handles);
end
function [upd_dismat, data_until] =copy_loaded_match(in,dirk,sets)
imgn{1} = dirk(sets(1,1)).ds(sets(1,2)).name;
imgn{2} = dirk(sets(2,1)).ds(sets(2,2)).name;
if(strcmp(imgn{1},[in.name{1} '.mat']) && strcmp(imgn{2},[in.name{2} '.mat']))
```

```
    upd_dismat = in.dismat;
    data_until = in.nod;
else
    msgbox(sprintf('Match Table and input water files are not concordant. ...
        Loaded table is for %s and %s', in.name{1},in.name{2}),'Source Image ...
        Conflict','warn','modal');
    error('choose new file');
end
function load_matches_Callback(hObject, eventdata, handles)
if get(handles.currentmatch_rb1,'Value')
    [fname, handles.flnm2] = uigetfile('*.mat', 'Choose the file to load Focal ...
        Adhesion Match Table',handles.flnm2);
    if(fname)
        in = load (fullfile(handles.flnm2,fname));
        sets = sortrows([handles.m1; handles.n1]');
        [upd_dismat, data_until] = copy_loaded_match(in,handles.d,sets);
        if(¬handles.idx.isinterchn)
            handles.matches.ch(handles.idx.m2).time(handles.idx.t2).upd_dismat ...
                = upd_dismat;
            handles.matches.ch(handles.idx.m2).time(handles.idx.t2).data_until ...
                = data_until;
        else
            handles.matches.interchn(handles.idx.m2).time(1).upd_dismat = ...
                upd_dismat;
            handles.matches.interchn(handles.idx.m2).time(1).data_until = ...
                data_until;
        end
        set(handles.uitable1,'Data',upd_dismat);
        guidata(hObject,handles);
    end
    set(handles.statusbar,'String', sprintf('Focal Adhesions Match has been ...
        Imported!!!'),...
        'BackgroundColor',[0 0.8 0]);
else
    handles.flnm2 = uigetdir(handles.flnm2, 'Choose the file to load Focal ...
        Adhesion Match Table');
    if(ischar(handles.flnm2))
        path1 = fullfile(handles.flnm2,'Match_bait*.mat');
        path2 = fullfile(handles.flnm2,'Match_prey*.mat');
        path3 = fullfile(handles.flnm2,'Match_Table_bait*.mat');
        ch(1).t = dir(path1);
        ch(2).t = dir(path2);
        ch(3).t = dir(path3);
        set(handles.statusbar,'String', sprintf('Focal Adhesions Match Data is ...
            being Imported !!! Please wait...'),...
            'BackgroundColor',[0.8 0 0]);
        for ib = 1:handles.nchn
            for ia = 1:(handles.nt_c-1)
                sets = [ib ia;ib ia+1];
                in = load(fullfile(handles.flnm2,ch(ib).t(ia).name));
                [upd_dismat, data_until] = copy_loaded_match(in,handles.d,sets);
                handles.matches.ch(ib).time(ia).upd_dismat = upd_dismat;
                handles.matches.ch(ib).time(ia).data_until = data_until;
            end
        end
        for ib = handles.nchn-1:-1:1
            ia = 1;
            sets = [ib 1;ib+1 1];
            in = load(fullfile(handles.flnm2,ch(3).t(ib).name));
            [upd_dismat, data_until] = copy_loaded_match(in,handles.d,sets);
            handles.matches.interchn(ib).time(ia).upd_dismat = upd_dismat;
            handles.matches.interchn(ib).time(ia).data_until = data_until;
        end
        set(handles.statusbar,'String', sprintf('Focal Adhesions Matches are ...
            Imported !!'),...
            'BackgroundColor',[0 0.8 0]);
        set(handles.uitable1, 'Data', ...
            handles.matches.ch(handles.idx.m2).time(handles.idx.t2).upd_dismat);
        guidata(hObject,handles);
    end
end
function create_big_table_Callback(hObject, eventdata, handles)
[¬, handles.flnm2] = uigetfile('*.mat', 'Choose the folder containing Water ...
    parameters and Focal Adhesion Match Tables',handles.flnm2);
if(handles.flnm2)
    set(handles.statusbar,'String', sprintf(' BigTable is being generated. ...
        Please wait'),...
        'BackgroundColor',[0.8 0 0]);
    pause(0.001);
    set(handles.img1name,'String', 'FA #');
    set(handles.img2name,'String', '');
    handles.ftable = genbigindextable(handles.flnm2);
    nt = size(handles.ftable,2);
    for ia = 1:nt
        tpt = handles.timeptlst(ia);
        in = handles.images.cMask_time(tpt);
        [handles.cMpimg{ia} handles.cMptch{ia}] = ...
            finalsegmentation(in.mpimg,in.mptch,handles.ftable(:,ia));
        tpt = handles.timeptlst(ia);
        for ib = 1:handles.nchn
            handles.ch(ib).fptch{ia} = create_ptch( ...
                handles.images.ch(ib).time(tpt).img, handles.cMpimg{ia});
        end
        ia
    end
    set(handles.uitable1,'Data',(1:size(handles.ftable,1))');
    set(handles.statusbar,'String', sprintf(' BigTable is generated. Now ...
        Choose the Focal Adhesions from the list and Export'),...
        'BackgroundColor',[0 0.8 0]);
    guidata(hObject,handles);
end
function export_Callback(hObject, eventdata, handles)
```

```
[fname, handles.flnm2] = uiputfile('*.mat','Save the final output ...
    file',handles.flnm2);
if(handles.flnm2)
    set(handles.statusbar,'String', sprintf(' BigTable and final Water images ...
        and Parameters are being Exported. Please wait'),...
        'BackgroundColor',[0 0.8 0]);
    nt_sel = handles.nt_c;
    nt_total = handles.nt;
    if(¬isfield(handles,'FAQ'))
        handles.FAQ = [];
        FAQ = [];
    end
    sel = get(handles.uitable1,'Data');
    tpts = handles.timeptlst;
    if(nt_sel ≠ nt_total)
    FAQ = handles.FAQ;
    cMpimg = handles.cMpimg;
    for ib = handles.nchn:−1:1
        ch(ib).fptch = handles.ch(ib).fptch;
    end
    save(fullfile(handles.flnm2,['Sel_' fname]), ...
        'cMpimg','ch','sel','nt_sel','tpts', 'nt_total','FAQ');
    export_finalw_movie(handles.cMpimg, handles.cMptch, nt_sel, ...
        handles.cmp,handles.d(1).ds, fullfile(handles.flnm2,'cMask_sub.avi'));
    clear cMpimg ch FAQ
    end
    sl = handles.skiplst;
    cMpimg(tpts) = handles.cMpimg;
    cMpimg(sl) = {zeros(size(handles.cMpimg{1}))};
    if(¬isempty(handles.FAQ))
        FAQ(tpts) = handles.FAQ;
    end
    for ib = 1:handles.nchn
         ch(ib).fptch(tpts) = handles.ch(ib).fptch;
         ch(ib).fptch(sl) = {zeros(size(handles.ch(ib).fptch{1}))};
    end
    save(fullfile(handles.flnm2,['Total_' fname]), ...
        'cMpimg','ch','sel','nt_sel','tpts', 'nt_total','FAQ');
    export_finalw_movie(cMpimg, handles.cMptch, nt_total, handles.cmp, ...
        handles.d(1).ds, fullfile(handles.flnm2, 'cMask_tot.avi'));
end
set(handles.statusbar,'String', sprintf(' BigTable and final Water images and ...
    Parameters are Exported'),...
    'BackgroundColor',[0 0.8 0]);
function export_finalw_movie(pimg, ptch,nt,cmp,ds,fname)
aviobj = avifile(fname,'compression','None');
psize = get(0,'screensize');
psize(3:4) = psize(3:4)*0.75;
for ia = 1:nt
    h = figure(876);
    set(876,'position',psize);
    image(pimg{ia}+1);
    axis off
    axis image
    colormap(cmp);
    titname = [ 'CMask ' ds(ia).name(6:end)];
    title(['\fontsize{16}', titname])
    for ib = 1: size(ptch{ia},1)
        if(ptch{ia}(ib ,3))
            text(round(ptch{ia}(ib ,6)),round(ptch{ia}( ib,5)), ...
                num2str(ptch{ia}(ib,1)), 'FontSize',10, 'Color','k');
        end
    end
    F = getframe(h);
    aviobj = addframe(aviobj,F);
    print('-f876','-dtiff',fname);
end
aviobj = close(aviobj);
function load_fnwater_Callback(hObject, eventdata, handles)
[fname, handles.flnm2] = uigetfile('*.mat', 'Choose the bigtable final ...
    output',handles.flnm2);
if(fname)
    load(fullfile(handles.flnm2,fname));
    set(handles.statusbar,'String', sprintf(' BigTable and final Water images ...
        and Parameters are being Imported. Please wait'),...
        'BackgroundColor',[0 0.8 0]);
    set(handles.uitable1,'Data','sel');
end
function tslider2_Callback(hObject, eventdata, handles)
n = round(get(hObject,'Value'));
set(handles.timepoint,'String', sprintf('%s and %s', ...
    handles.ds1(n).name,handles.ds2(n).name));
if(get(handles.showprey,'Value'))
    figure(230);
    image(handles.pfpimg{n}+1);
    colormap(handles.cmp);
    title(['\fontsize{16}', handles.ds2(n).name([1:4 , 6,7])])
    for ia = 1: size(handles.pfptch{n},1)
        text(round(handles.pfptch{n}(ia ,6)),round(handles.pfptch{n}( ia,5)), ...
            num2str(handles.pfptch{n}(ia,1)), 'FontSize',8, 'Color','k');
    end
end
if(get(handles.showbait,'Value'))
    figure(240);
    image(handles.bfpimg{n}+1);
    colormap(handles.cmp);
    title({handles.ds1(n).name});
    title(['\fontsize{16}', handles.ds1(n).name([1:4 , 6,7])])
    for ia = 1: size(handles.bfptch{n},1)
        text(round(handles.bfptch{n}(ia ,6)),round(handles.bfptch{n}( ia,5)), ...
            num2str(handles.bfptch{n}(ia,1)), 'FontSize',8, 'Color','k');
    end
```

```
end
function showbait_Callback(hObject, eventdata, handles)
function showprey_Callback(hObject, eventdata, handles)
function no_st_matches_Callback(hObject, eventdata, handles)
function no_st_matches_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function uitable1_CellEditCallback(hObject, eventdata, handles)
if(get(handles.allow_match,'Value') && get(handles.create_big_table,'Value')) ...
    %if the allow matching is checked
    update_dismat(hObject,handles);  %to update the displayed table
    handles = guidata(hObject);
end
guidata(hObject, handles);
function export_matches_CreateFcn(hObject, eventdata, handles)
function channels4mask_Callback(hObject, eventdata, handles)
function channels4mask_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function matchskip_Callback(hObject, eventdata, handles)
function matchskip_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), ...
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function rm_unmatch_patch_Callback(hObject, eventdata, handles)
choice = questdlg('Do you really want to remove the unmatched patches ...
    permanently?', ...
    'Remove Unmatched Patches', ...
    'Yes','No','Yes');
if(strcmp(choice,'Yes'))
    sel = get(handles.uitable1,'Data');
    idx1 = strfind(sel(:,1),'0');
    idx2 = cellfun(@isempty,idx1);
    idx3 = cellfun('prodofsize', sel(:,1))>1;
    sel = sel(idx2 | idx3,:);
    set(handles.uitable1,'Data',sel);
    update_dismat(hObject,handles);
end
function QMA_Callback(hObject, eventdata, handles)
set(handles.statusbar,'String', sprintf(' Major Axis of Focal Adhesions are ...
    being generated. Please wait'),...
        'BackgroundColor',[0.8 0 0]);
for ia = 1:handles.nt_c
        tpt = handles.timeptlst(ia);
        for ib = 1:handles.nchn
                img(:,:,ib) = handles.images.ch(ib).time(tpt).img;
        end
        bw = handles.images.ch(ib).time(tpt).bw;
        handles.FAQ(ia).FAQ = quantMajorAxis(img,handles.cMpimg{ia},bw);
        ia
end
set(handles.statusbar,'String', sprintf(' Major Axis Quantified is generated. ...
    Now Export the fles'),...
        'BackgroundColor',[0 0.8 0]);
guidata(hObject, handles);
```

```
function [mtch, n, npimg1, npimg2] = matchpatch(pimg1, pimg2,ptch1,ptch2)
lst = ptch1(ptch1(:,7)≠1,1);
for ia = 1:size(lst)
    pimg1(pimg1 == lst(ia)) = 0;
end
lst = ptch2(ptch2(:,7)≠1,1);
for ia = 1:size(lst)
    pimg2(pimg2 == lst(ia)) = 0;
end
climg = logical(logical(pimg1).*logical(pimg2));
mimg1 = pimg1(climg);
mimg2 = pimg2(climg);
mtch = [ mimg1(:) , mimg2(:)];
mtch = unique(mtch,'rows');
smtch = mtch;
for ia= 1:2
    [un,ind,¬] = unique(sort(mtch(:,ia)));
    ind = ind − [ 0; ind(1:end−1)];    %ind is the frequency of the element
    un = un(ind >1);  %remove element which are repeated more than once
    [¬,ind]= setdiff(smtch(:,ia),un);  %find index of non−repeated elements
    smtch = smtch(ind,:); %strict match is smatch
end
mtch = smtch;
n = size(mtch,1); %n is the total number of strict matches
percentOverlap = 0.5;
tmtch = [pimg1(:), pimg2(:)];
idx = bsxfun(@and, pimg1(:)>0 , pimg2(:)>0);
tmtch = tmtch(idx,:);
[umtch freq] = uniquencount(tmtch);
[rmtch idx] = setdiff(umtch,mtch,'rows');
if(any(rmtch))
    freq = freq(idx);
    overlap =  [ freq./ ptch1(rmtch(1:end,1),2) , freq./ ptch2(rmtch(1:end,2),2)];
    Bool_overlap = overlap> percentOverlap;
    if(any(Bool_overlap))
        idxoverlap = bsxfun(@or,Bool_overlap(:,1),Bool_overlap(:,2));
        overlapmtch = rmtch(idxoverlap,:);
        omtch = overlapmtch;
        for ia= 1:2
            [un, freq] = uniquencount(overlapmtch(:,ia));
            un = un(freq >5);  %remove element which are repeated more than twice
            for ib = 1:size(un,1)
                omtch(omtch(:,ia)== un(ib),:) = [];
            end
        end
        mtch = [mtch ; omtch];
    end
end
lmtch = [pimg1(:), pimg2(:)];  %lmtch  is lost matches i.e [x 0] or [0 x] patches
lmtch = unique(lmtch,'rows');
nlst1 = setdiff(lmtch(:,1),[mtch(:,1);0]);
nlst2 = setdiff(lmtch(:,2),[mtch(:,2);0]);
rad = 5;  %radius(in pixels) to search for neighboring focal adhesions
x1 = ptch1(nlst1,6);
y1 = ptch1(nlst1,5);
x2 = ptch2(nlst2,6);
y2 = ptch2(nlst2,5);
sz1 = size(x1,1);
sz2 = size(x2,1);
if(¬isempty(x2) && ¬isempty(x1))
    for ia = 1:sz1
        [ind mdist] = mindist(x1(ia),y1(ia),x2,y2);
        lmt1(ia,:) = [ind mdist];
    end
    for ia = 1:sz2
        [ind mdist] = mindist(x2(ia),y2(ia),x1,y1);
        lmt2(ia,:) = [ind mdist];
    end
    nlmtch1 = [];
    nlmtch2 = [];
    for ia = sz1:−1:1
        if(lmt1(ia,2) ≤ lmt2( lmt1(ia,1),2)  && lmt1(ia,2) < rad)
            nlmtch1(ia,:) = [ nlst1(ia,1) , nlst2( lmt1(ia,1),1)];
        else
            nlmtch1(ia,:) = [ nlst1(ia,1) , 0];
        end
    end
    for ia = sz2:−1:1
        if(lmt2(ia,2) ≤ lmt1( lmt2(ia,1),2)  && lmt2(ia,2) < rad)
            nlmtch2(ia,:) = [nlst1( lmt2(ia,1),1) ,  nlst2(ia,1)];
        else
            nlmtch2(ia,:) = [ 0 , nlst2(ia,1) ];
        end
    end
    nlmtch = unique([ nlmtch1; nlmtch2],'rows');
    mtch = [mtch;nlmtch];
end
if nargout == 4
    npimg1 = zeros(size(pimg1));
    npimg2 = zeros(size(pimg2));
    for ia = 1:size(mtch,1)
```

```
        if(mtch(ia,1))
            npimg1(pimg1== mtch(ia,1)) = ia;
        end
        if(mtch(ia,2))
            npimg2(pimg2 == mtch(ia,2)) = ia;
        end
    end
    cmp = colormap(jet(512));
    cmp = cmp(101:512,:);
    cmp = cmp(randperm(412),:);
    cmp(1,:) = [ 1 1 1];
    figure(1253);
    image(pimg1);
    figure(1254)
    image(pimg2);
    figure(1),image(npimg1+1);
    for ia = 1: size(mtch,1)
        if(mtch(ia,1))
            text(round(ptch1( mtch(ia,1) ,6)),round(ptch1( mtch(ia,1),5)), ...
                num2str(ia), 'FontSize',8, 'Color','k');
        end
    end
    colormap(cmp);
    figure(2), image(npimg2+1);
    for ia = 1: size(mtch,1)
        if(mtch(ia,2))
            text(round(ptch2( mtch(ia,2) ,6)),round(ptch2( mtch(ia,2),5)), ...
                num2str(ia), 'FontSize',8, 'Color','k');
        end
    end
    colormap(cmp);
end
[un1,fr] = uniquencount(mtch(:,1));
un1 = un1(fr>1);
un1 = setdiff(un1,0);
[un2,fr] = uniquencount(mtch(:,2));
un2=  un2(fr>1);
un2 = setdiff(un2,0);
tmtch = mtch;
bolrm = false(size(mtch,1),1);
mtch = num2cell(mtch);
mtch(tmtch == 0) = {[]};
for ia = 1:length(un1)
    lst = find(tmtch(:,1) == un1(ia));
    mtch(lst(1),2) = {tmtch(lst,2)'};
    bolrm(lst(2:end)) = true;
end
for ia = 1:length(un2)
    lst = find(tmtch(:,2) == un2(ia));
    mtch(lst(1),1) = {tmtch(lst,1)'};
    bolrm(lst(2:end)) = true;
```

```
end
mtch(bolrm,:) = [];
function [ind mdist] = mindist(x,y, xv, yv)
dist = sqrt(((x − xv).^2)   +  ((y − yv).^2));
mdist = min(dist);
ind = find(dist == mdist);
```

```
function [k1, k2] = addpatch(mtch,p1,p2,k1,k2,ptch1,ptch2)
n = size(k1,1)+1;
k1(n,1) = NaN;
if(p1(1) ==0)
    k1(n,2:7) =0;
else
k1(n,2) = sum(ptch1(mtch(p1,1),2));
k1(n,3) = sum(ptch1(mtch(p1,1),3));
k1(n,4) = k1(n,3)/k1(n,2);
k1(n,5) = sum(ptch1(mtch(p1,1),3).* ptch1(mtch(p1,1),5))/k1(n,3);
k1(n,6) = sum(ptch1(mtch(p1,1),3).* ptch1(mtch(p1,1),6))/k1(n,3);
k1(n,7) = 1;
end
k2(n,1) = NaN;
if(p2(1) ==0)
    k2(n,2:7) = 0;
else
k2(n,2) = sum(ptch2(mtch(p2,2),2));
k2(n,3) = sum(ptch2(mtch(p2,2),3));
k2(n,4) = k2(n,3)/k2(n,2);
k2(n,5) = sum(ptch2(mtch(p2,2),3).* ptch2(mtch(p2,2),5))/k2(n,3);
k2(n,6) = sum(ptch2(mtch(p2,2),3).* ptch2(mtch(p2,2),6))/k2(n,3);
k2(n,7) = 1;
end
```

```
function [npimg1,nptch1,npimg2,nptch2,mtable] = ...
    newseg4matchpair(mtch,pimg1,ptch1,pimg2,ptch2,table)
mmtch = cell(0,0);
cnmtch = [];
szmtch = size(mtch,1); %size of match
for ia = 1:size(table,1)
    p1 = str2num(table{ia,1}); p2 = str2num(table{ia,2});
    if(isempty(p1))
        p1 = 0;
    end
    if(isempty(p2))
       p2 =0;
    end
    if((size(p1,2) > 1 )|| (size(p2,2) > 1)  || (p1(1)≠ p2(1))   )
        mmtch{end+1,1} = {p1};    %matches to be merged
        mmtch{end,2} = {p2};
    elseif(¬(p1 == 0 && p2 == 0))
        cnmtch(end+1,1) = ia; %chosen normal matches
    end
end
[nptch1, nptch2] = getpar(mtch,cnmtch,ptch1,ptch2); %get parameters for ...
    directly matched patches
mtable = num2cell(mtch(cnmtch,:));
if(¬isequal(mmtch,cell(0,0)))
    for ia = 1:size(mmtch,1)
        p1 = cell2mat(mmtch{ia,1});
        p2 = cell2mat(mmtch{ia,2});
        [nptch1, nptch2] = addpatch(mtch,p1,p2,nptch1,nptch2,ptch1,ptch2);
        p1(p1==0) = [];
        p2(p2==0) = [];
        mtable(end+1,:)= {mtch(p1,1),mtch(p2,2)};    %here is the problem to ...
            correct.
    end
end
nptch1(1:end,1) = 1:size(nptch1,1);
nptch2(1:end,1) = 1:size(nptch2,1); %size of nptch1 and nptch2 is same
npimg1 = newpimg4mtchpair(pimg1,mtch,cnmtch,mmtch,1);
npimg2 = newpimg4mtchpair(pimg2,mtch,cnmtch,mmtch,2);
function npimg = newpimg4mtchpair(pimg,mtch,cnmtch,mmtch,n)
    npimg = zeros(size(pimg));
    ind = 1;
    for ia = 1:size(cnmtch,1)
        num = mtch(cnmtch(ia),n);
        if(num)
            npimg(pimg == num) = ind;
        end
        ind = ind + 1;
    end
    for ib = 1:size(mmtch,1)
        p1 = cell2mat(mmtch{ib,n});
        for ic = 1:size(p1,2)
            if(p1(ic))
                num = mtch(p1(ic),n);
                if(num)
                    npimg(pimg == num) = ind;
                end
            end
        end
        ind = ind + 1;
    end
```

```
function ptch = create_ptch(img,pimg)
tptch = regionprops(pimg,img,'MeanIntensity','Centroid','Area');
vec = 1:size(tptch,1);
nofa = max(pimg(:));
ptch(1:nofa,1) = (1:nofa)';
ptch(vec,2) = [tptch.Area]';
ptch(vec,4) = [tptch.MeanIntensity]';
ptch(:,3) = ptch(:,2).*ptch(:,4);
iFA = size(ptch,1);
[m,n] = size(img);
timg = img .* repmat(1:n ,m,1);
cmX_FA =  cell2mat(arrayfun(@(x) sum(timg(pimg==x))/ptch(x,3), ...
    1:iFA,'UniformOutput', false)');
ptch(:,5) = cmX_FA;
timg = img .* repmat((1:m)',1,n);
cmY_FA =  cell2mat(arrayfun(@(x) sum(timg(pimg==x))/ptch(x,3), ...
    1:iFA,'UniformOutput', false)');
ptch(:,6) = cmY_FA;
ptch(vec,7) = 1;
end
```

```
function  ftable = genbigindextable_initial(matches)
sz = size(matches.time,2);
for ia = sz:-1:1
        t(ia).m = matches.time(ia).mtch;
end
ftable = bigtableindex(t);
```

```
function  ftable = genbigindextable(path)
if nargin < 1
    path1 = fullfile(pwd,'Match_*.mat');
else
    path1 = fullfile(path,'Match_*.mat');
end
ds = dir(path1);
sz = size(ds,1);
for ia = sz:-1:1
        in = load(fullfile(path,ds(ia).name));
        t(ia).m = in.mtable(1:in.nod,:);
end
ftable = bigtableindex(t);
```

```
function [npimg,nptch] = finalsegmentation(pimg,ptch,table)
nlst = [];
mlst = {};
nindex = [];
mindex = [];
for ia = 1:size(table,1)
    p = (table{ia});
    if(sum(isempty(p))  || sum(isnan(p)))
        p =0;
    end
    p = p(:)';
    if((size(p,2) > 1 ))
        mlst(end+1,1) = {p};    % to be merged
        mindex(end+1) = ia;
    elseif(¬(p==0))
        nlst(end+1,1) = p; %chosen normal
        nindex(end+1) = ia;
    end
end
nptch = zeros(size(table,1),size(ptch,2));
nptch(nindex',:) = ptch(nlst,:);
if(¬isequal(mlst,cell(0)))
    for ia = 1:size(mlst,1)
        p = cell2mat(mlst(ia,1));
        nptch = addpatch_S(ptch,nptch,p,mindex(ia));
    end
end
nptch(1:size(table,1),1) = 1:size(table,1);
npimg = newpimg(pimg,nlst,mlst,nindex,mindex);
function nptch = addpatch_S(ptch,nptch,p,n)
nptch(n,1) = n;
if(p(1) ==0)
    nptch(n,2:7) =0;
else
    nptch(n,2) = sum(ptch(p,2));
    nptch(n,3) = sum(ptch(p,3));
    nptch(n,4) = nptch(n,3)/nptch(n,2);
    nptch(n,5) = sum(ptch(p,3).* ptch(p,5))/nptch(n,3);
    nptch(n,6) = sum(ptch(p,3).* ptch(p,6))/nptch(n,3);
    nptch(n,7) = 1;
end
function npimg = newpimg(pimg,nlst,mlst,nindex,mindex)
npimg = zeros(size(pimg));
for ia = 1:size(nlst,1)
    npimg(pimg == nlst(ia,1)) = nindex(ia);
end
for ib = 1:size(mlst,1)
    p = cell2mat(mlst(ib));
    for ic = 1:size(p,2)
        if(p(ic))
            npimg(pimg == p(ic)) = mindex(ib);
        end
    end
end
```

```
function FAQ = quantMajorAxis(img,npimg,bw)
FAQ = regionprops(npimg, 'Orientation', 'MajorAxisLength', ...
    'MinorAxisLength', 'Eccentricity', 'Centroid','PixelList');
figure(94848);
```

```
imagesc(npimg);
hold on;
outlineimg = bwmorph(imfill(bw),'remove');
[outlx outly] = ind2sub(size(outlineimg),find(outlineimg));
nch = size(img,3);
for ia = 1:length(FAQ)
    if(¬isempty(FAQ(ia).PixelList))
        FAQ(ia).coord = getFAmajoraxisCoord(FAQ(ia));
        for ib = 1:nch
            timg = img(:,:,ib);
            FAQ(ia).MAQ(:,ib) =  arrayfun(@(a,b)timg(a,b),FAQ(ia).coord(:,2), ...
                FAQ(ia).coord(:,1));
            FAQ(ia).uMAQ = imresize(FAQ(ia).MAQ,[50,nch]);
        end
        [id,md] = findclosepoint(FAQ(ia).coord,[outlx outly]);
        FAQ(ia). md = md;
        if id == 2
            FAQ(ia).uMAQ = FAQ(ia).uMAQ(end:−1:1,:);
            FAQ(ia).MAQ = FAQ(ia).MAQ(end:−1:1,:);
            FAQ(ia).coord = FAQ(ia).coord(end:−1:1,:);
        end
    end
end
function coord = getFAmajoraxisCoord(s)
x = s.Centroid(1);
y = s.Centroid(2);
theta = 180−s.Orientation;
m = tand(theta);
c = y − m*x;
lx = unique(s.PixelList(:,1));
ly = m * lx + c;
coord1 = [lx,ly];
miny = min(s.PixelList(:,2));
maxy = max(s.PixelList(:,2));
coord1 = coord1(ly≥miny & ly≤maxy,:);
plot(coord1(:,1),coord1(:,2),'r');
hold on;
ly = unique(s.PixelList(:,2));
lx = (ly − c)./m;
if(m == Inf)
    lx = zeros(size(lx))+x;
end
coord2 = [lx,ly];
minx = min(s.PixelList(:,1));
maxx = max(s.PixelList(:,1));
coord2 = coord2(lx≥minx & lx≤maxx,:);
plot(coord2(:,1),coord2(:,2),'g');
hold on;
text(x,y,'FA');
coord = unique(round([coord1;coord2]),'rows');
function [id md] = findclosepoint(coord,outcord)
```

```
coord = coord([1,end],:);
dist = pdist2(coord,outcord);
[md, idx]= min(dist(:));
[id id2] = ind2sub(size(dist),idx);
```

References

Arganda-Carreras, I., Sorzano, C., Marabini, R., Carazo, J., Ortiz-de Solorzano, C., and Kybic, J. (2006). Consistent and Elastic Registration of Histological Sections using Vector-Spline Regularization. *Cancer Imaging*, 4241:85–95. 30

Attwell, S., Mills, J., Troussard, A., Wu, C., and Dedhar, S. (2003). Integration of cell attachment, cytoskeletal localization, and signaling by integrin-linked kinase (ILK), CH-ILKBP, and the tumor suppressor PTEN. *Molecular Biology of the Cell*, 14:4813–4825. 10

Balaban, N. Q., Schwarz, U. S., Riveline, D., Goichberg, P., Tzur, G., Sabanay, I., Mahalu, D., Safran, S., Bershadsky, a., Addadi, L., and Geiger, B. (2001). Force and focal adhesion assembly: a close relationship studied using elastic micropatterned substrates. *Nature Cell Biology*, 3(5):466–72. 11

Banaszynski, L. A., Liu, C. W., and Wandless, T. J. (2005). Characterization of the fkbp rapamycin frb ternary complex. *Journal of the American Chemical Society*, 127(13):4715–4721. 71

Bansal, M., Belcastro, V., Ambesi-Impiombato, A., and di Bernardo, D. (2007). How to infer gene networks from expression profiles. *Molecular Systems Biology*, 3(78):78. 13, 14, 17

Basso, K., Margolin, A. A., Stolovitzky, G., Klein, U., Dalla-Favera, R., and Califano, A. (2005). Reverse engineering of regulatory networks in human B cells. *Nature Genetics*, 37:382–390. 14

Bellis, S. L., Perrotta, J. A., Curtis, M. S., and Turner, C. E. (1997). Adhesion of fibroblasts to fibronectin stimulates both serine and tyrosine phosphorylation of paxillin. *The Biochemical Journal*, 325:375–381. 10

Bertolucci, C. M., Guibao, C. D., and Zheng, J. (2005). Structural features of the focal adhesion kinase-paxillin complex give insight into the dynamics of focal adhesion assembly. *Protein Science*, 14:644–652. 10, 98

Bois, P. R. J., O'Hara, B. P., Nietlispach, D., Kirkpatrick, J., and Izard, T. (2006). The vinculin binding sites of talin and alpha-actinin are sufficient to activate vinculin. *The Journal of Biological Chemistry*, 281:7228–7236. 11

Brown, M. C., Perrotta, J. A., and Turner, C. E. (1996). Identification of LIM3 as the principal determinant of paxillin focal adhesion localization and characterization of a novel motif on paxillin directing vinculin and focal adhesion kinase binding. *The Journal of Cell Biology*, 135:1109–1123. 10

Burridge, K., Turner, C. E., and Romer, L. H. (1992). Tyrosine phosphorylation of paxillin and pp125FAK accompanies cell adhesion to extracellular matrix: a role in cytoskeletal assembly. *The Journal of Cell Biology*, 119:893–903. 10

Carragher, N. O., Levkau, B., Ross, R., and Raines, E. W. (1999). Degraded collagen fragments promote rapid disassembly of smooth muscle focal adhesions that correlates with cleavage of pp125(FAK), paxillin, and talin. *The Journal of Cell Biology*, 147:619–630. 11

Chan, K. T., Bennin, D. a., and Huttenlocher, A. (2010). Regulation of adhesion dynamics by calpain-mediated proteolysis of focal adhesion kinase (FAK). *The Journal of Biological Chemistry*, 285(15):11418–26. 9

Chen, H., Cohen, D. M., Choudhury, D. M., Kioka, N., and Craig, S. W. (2005). Spatial distribution and functional significance of activated vinculin in living cells. *The Journal of Cell Biology*, 169:459–470. 11, 97

Clarke, D. M., Brown, M. C., LaLonde, D. P., and Turner, C. E. (2004). Phosphorylation of actopaxin regulates cell spreading and migration. *The Journal of Cell Biology*, 166:901–912. 10

Cohen, D. M., Kutscher, B., Chen, H., Murphy, D. B., and Craig, S. W. (2006). A conformational switch in vinculin drives formation and dynamics of a talin-vinculin complex at focal adhesions. *The Journal of Biological Chemistry*, 281:16006–16015. 12

Côté, J. F., Turner, C. E., and Tremblay, M. L. (1999). Intact LIM 3 and LIM 4 domains of paxillin are required for the association to a novel polyproline region (Pro 2) of protein-tyrosine phosphatase-PEST. *The Journal of Biological Chemistry*, 274:20550–20560. 10

Cramer, L. P., Siebert, M., and Mitchison, T. J. (1997). Identification of novel graded polarity actin filament bundles in locomoting heart fibroblasts: implications for the generation of motile force. *The Journal of Cell Biology*, 136:1287–1305. 12

Crawford, A. W., Michelsen, J. W., and Beckerle, M. C. (1992). An interaction between zyxin and alpha-actinin. *The Journal of Cell Biology*, 116:1381–1393. 12

Curtis, A. S. (1964). The mechanism of adhesion of cells to glass. A styudy by interference refection microscopy. *The Journal of Cell Biology*, 20:199–215. 2

Deakin, N. O. and Turner, C. E. (2008). Paxillin comes of age. *Journal of Cell Science*, 121(Pt 15):2435–44. 10, 11

Delanoë-Ayari, H., Al Kurdi, R., Vallade, M., Gulino-Debrac, D., and Riveline, D. (2004). Membrane and acto-myosin tension promote clustering of adhesion proteins. *Proceedings of the National Academy of Sciences of the United States of America*, 101:2229–2234. 11

DeMali, K. A. (2004). Vinculin - A dynamic regulator of cell adhesion. *Trends in Biochemical Sciences*, 29(11):565–567. 11

Devallière, J., Chatelais, M., Fitau, J., Gérard, N., Hulin, P., Velazquez, L., Turner, C. E., and Charreau, B. (2012). LNK (SH2B3) is a key regulator of integrin signaling in endothelial cells and targets α-parvin to control cell adhesion and migration. *FASEB Journal*, 26(6):2592–606. 10

di Bernardo, D., Thompson, M. J., Gardner, T. S., Chobot, S. E., Eastwood, E. L., Wojtovich, A. P., Elliott, S. J., Schaus, S. E., and Collins, J. J. (2005). Chemogenomic profiling on a genome-wide scale using reverse-engineered gene networks. *Nature Biotechnology*, 23:377–383. 14

Digman, M. A., Brown, C. M., Horwitz, A. R., Mantulin, W. W., and Gratton, E. (2008). Paxillin Dynamics Measured during Adhesion Assembly and Disassembly by Correlation Spectroscopy. *Biophysical Journal*, 94(7):2819–2831. 11

Dimitrakopoulou, K., Tsimpouris, C., Papadopoulos, G., Pommerenke, C., Wilk, E., Sgarbas, K. N., Schughart, K., and Bezerianos, A. (2011). Dynamic gene network reconstruction from gene expression data in mice after influenza A (H1N1) infection. *Journal of Clinical Bioinformatics*, 1(1):27. 15

Drees, B. E., Andrews, K. M., and Beckerle, M. C. (1999). Molecular dissection of zyxin function reveals its involvement in cell motility. *The Journal of Cell Biology*, 147:1549–1560. 12

Feng, Y., Ngu, H., Alford, S. K., Ward, M., Yin, F., and Longmore, G. D. (2013). -Actinin1 and 4 Tyrosine Phosphorylation Is Critical for Stress Fiber Establishment, Maintenance and Focal Adhesion Maturation. *Experimental Cell Research*, 319(8):1124–35. 6, 12, 13

Frey, B. J. and Dueck, D. (2007). Clustering by Passing Messages Between Data Points. *Science*, 315:972–976. 67

Friedenberger, M., Bode, M., Krusche, A., and Schubert, W. (2007). Fluorescence detection of protein clusters in individual cells and tissue sections by using toponome imaging system: sample preparation and measuring procedures. *Nature Protocols*, 2:2285–2294. 82

Friedman, N. (2004). Inferring cellular networks using probabilistic graphical models. *Science*, 303(5659):799–805. 13, 14, 17

Friedman, N., Linial, M., Nachman, I., and Pe'er, D. (2000). Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, 7:601–620. 14

Garcia, D. (2010). Robust smoothing of gridded data in one and higher dimensions with missing values. *Computational Statistics and Data Analysis*, 54:1167–1178. 33, 128

Gardner, T. S., di Bernardo, D., Lorenz, D., and Collins, J. J. (2003). Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, 301:102–105. 14

Geiger, B., Spatz, J. P., and Bershadsky, A. D. (2009). Environmental sensing through focal adhesions. *Nature Reviews. Molecular Cell Biology*, 10(1):21–33. 2, 6, 99

Giannone, G., Jiang, G., Sutton, D. H., Critchley, D. R., and Sheetz, M. P. (2003). Talin1 is critical for force-dependent reinforcement of initial integrin-cytoskeleton bonds but not tyrosine kinase activation. *The Journal of Cell Biology*, 163:409–419. 11

Guizar-Sicairos, M., Thurman, S. T., and Fienup, J. R. (2008). Efficient subpixel image registration algorithms. *Optics Letters*, 33(2):156–8. 33

Hanein, D. and Horwitz, A. R. (2012). The structure of cell-matrix adhesions: the new frontier. *Current Opinion in Cell Biology*, 24(1):134–40. 2, 4

Heath, J. P. (1983). Behaviour and structure of the leading lamella in moving fibroblasts. I. Occurrence and centripetal movement of arc-shaped microfilament bundles beneath the dorsal cell surface. *Journal of Cell Science*, 60:331–354. 13

Heath, J. P., Dunn, G. A., and Causeway, W. (1978). Cell to substratum contacts of chick f1broblasts and their relation to the microfilament system . a correlated interference-reflexion and high- voltage electron-microscope s study. *Journal of Cell Science*, 212:197–212. 2, 13

Hempel, S., Koseska, A., Kurths, J., and Nikoloski, Z. (2011a). Inner Composition Alignment for Inferring Directed Networks from Short Time Series. *Physical Review Letters*, 107(5):054101. 14, 65

Hempel, S., Koseska, A., Nikoloski, Z., and Kurths, J. (2011b). Unraveling gene regulatory networks from time-resolved gene expression data - a measures comparison study. *BMC Bioinformatics*, 12(1):292. 14, 17, 65

Herreros, L., Rodríguez-Fernandez, J. L., Brown, M. C., Alonso-Lebrero, J. L., Cabañas, C., Sánchez-Madrid, F., Longo, N., Turner, C. E., and Sánchez-Mateos, P. (2000). Paxillin localizes to the lymphocyte microtubule organizing center and associates with the microtubule cytoskeleton. *The Journal of Biological Chemistry*, 275:26436–26440. 10

Hirata, H., Tatsumi, H., and Sokabe, M. (2008). Zyxin emerges as a key player in the mechanotransduction at cell adhesive structures. *Communicative & Integrative Biology*, 1(2):192–5. 12, 97, 101

Hoffman, L. M., Jensen, C. C., Kloeker, S., Wang, C.-L. A., Yoshigi, M., and Beckerle, M. C. (2006). Genetic ablation of zyxin causes Mena/VASP mislocalization, increased motility, and deficits in actin remodeling. *The Journal of Cell Biology*, 172:771–782. 12

Hotulainen, P. and Lappalainen, P. (2006). Stress fibers are generated by two distinct actin assembly mechanisms in motile cells. *The Journal of Cell Biology*, 173:383–394. 13

Humphries, J. D., Wang, P., Streuli, C., Geiger, B., Humphries, M. J., and Ballestrem, C. (2007). Vinculin controls focal adhesion formation by direct interactions with talin and actin. *The Journal of Cell Biology*, 179:1043–1057. 12

Ickstadt, K., Bornkamp, B., Grzegorczyk, M., Wieczorek, J., Sheriff, M. R., Grecco, H. E., and Zamir, E. (2011). Nonparametric Bayesian Networks. In Bernardo, J. M., Bayarri, M. J., Berger, J. O., Dawid, A. P., Heckerman, D., Smith, A. F. M., and West, M., editors, *Bayesian Statistics 9*, number 0315507, pages 283–316, Valencia. Oxford University Press. 15, 82

Ilić, D., Furuta, Y., Kanazawa, S., Takeda, N., Sobue, K., Nakatsuji, N., Nomura, S., Fujimoto, J., Okada, M., and Yamamoto, T. (1995). Reduced cell motility and enhanced focal adhesion contact formation in cells from FAK-deficient mice. *Nature*, 377:539–544. 9

Izaguirre, G., Aguirre, L., Hu, Y. P., Lee, H. Y., Schlaepfer, D. D., Aneskievich, B. J., and Haimovich, B. (2001). The cytoskeletal/non-muscle isoform of alpha-actinin is phosphorylated on its actin-binding domain by the focal adhesion kinase. *The Journal of Biological Chemistry*, 276:28676–28685. 6

Izard, T., Evans, G., Borgon, R. A., Rush, C. L., Bricogne, G., and Bois, P. R. J. (2004). Vinculin activation by talin through helical bundle conversion. *Nature*, 427:171–175. 11

Izard, T. and Vonrhein, C. (2004). Structural basis for amplifying vinculin activation by talin. *The Journal of Biological Chemistry*, 279:27667–27678. 11

Kholodenko, B. N., Kiyatkin, A., Bruggeman, F. J., Sontag, E., Westerhoff, H. V., and Hoek, J. B. (2002). Untangling the wires: a strategy to trace functional interactions in signaling and gene networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99:12841–12846. 15

Krutzik, P. O., Crane, J. M., Clutter, M. R., and Nolan, G. P. (2008). High-content single-cell drug screening with phosphospecific flow cytometry. *Nature chemical biology*, 4:132–142. 82

Lazarides, E., Harbor, C. S., and York, N. (1975). Localization of a Muscle Structural Protein in Nonmuscle Cells. *Cell*, 6(November):289–298. 6

Le Dévédec, S. E., Geverts, B., de Bont, H., Yan, K., Verbeek, F. J., Houtsmuller, A. B., and van de Water, B. (2012). The residence time of focal adhesion kinase (FAK) and paxillin at focal adhesions in renal epithelial cells is determined by adhesion size, strength and life cycle status. *Journal of Cell Science*, 125(Pt 19):4498–506. 9

Legate, K. R., Montañez, E., Kudlacek, O., and Fässler, R. (2006). ILK, PINCH and parvin: the tIPP of integrin signalling. *Nature Reviews. Molecular Cell Biology*, 7(1):20–31. 9, 10

Lek, M. and North, K. N. (2010). Are biological sensors modulated by their structural scaffolds? The role of the structural muscle proteins α-actinin-2 and α-actinin-3 as modulators of biological sensors. *FEBS Letters*, 584:2974–2980. 6

Lele, T. P., Thodeti, C. K., Pendse, J., and Ingber, D. E. (2008). Investigating complexity of protein-protein interactions in focal adhesions. *Biochemical and Biophysical Research Communications*, 369(3):929–34. 24

Li, N., Goodwin, R. L., and Potts, J. D. (2013). Microscopy Microanalysis Zyxin Regulates Cell Migration and Differentiation in EMT during Chicken AV Valve Morphogenesis. *Microscopy and Microanalysis*, 19:842–854. 12

Margolin, A. A., Nemenman, I., Basso, K., Wiggins, C., Stolovitzky, G., Dalla Favera, R., and Califano, A. (2006). ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7 Suppl 1:S7. 14

Mierke, C. T. (2009). The role of vinculin in the regulation of the mechanical properties of cells. *Cell Biochemistry and Biophysics*, 53(3):115–26. 96

Mitra, S. K., Hanson, D. A., and Schlaepfer, D. D. (2005). Focal adhesion kinase: in command and control of cell motility. *Nature Reviews. Molecular Cell Biology*, 6(1):56–68. 9

Naumanen, P., Lappalainen, P., and Hotulainen, P. (2008). Mechanisms of actin stress fibre assembly. *Journal of Microscopy*, 231:446–454. 13

Oakes, P. W., Beckham, Y., Stricker, J., and Gardel, M. L. (2012). Tension is required but not sufficient for focal adhesion maturation without a stress fiber template. *The Journal of Cell Biology*, 196:363–374. 13

Otey, C. A., Pavalko, F. M., and Burridge, K. (1990). An interaction between alpha-actinin and the beta 1 integrin subunit in vitro. *The Journal of Cell Biology*, 111:721–729. 6

Palazzo, A. F., Eng, C. H., Schlaepfer, D. D., Marcantonio, E. E., and Gundersen, G. G. (2004). Localized stabilization of microtubules by integrin- and FAK-facilitated Rho signaling. *Science*, 303(5659):836–9. 9

Parsons, J. T. (2003). Focal adhesion kinase: the first ten years. *Journal of Cell Science*, 116:1409–1416. 9

Parsons, J. T., Horwitz, A. R., and Schwartz, M. A. (2010). Cell adhesion: integrating cytoskeletal dynamics and cellular tension. *Nature Reviews. Molecular Cell Biology*, 11:633–643. 4

Pavalko, F. M. and LaRoche, S. M. (1993). Activation of human neutrophils induces an interaction between the integrin beta 2-subunit (CD18) and the actin binding protein alpha-actinin. *Journal of Immunology*, 151:3795–3807. 6

Pearl, J. (1988). *Probabalistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, llustrated edition. 14

Pellegrin, S. and Mellor, H. (2007). Actin stress fibres. *Journal of Cell Science*, 120:3491–3499. 13

Petit, V., Boyer, B., Lentz, D., Turner, C. E., Thiery, J. P., and Vallés, A. M. (2000). Phosphorylation of tyrosine residues 31 and 118 on paxillin regulates cell migration through an association with CRK in NBT-II cells. *The Journal of Cell Biology*, 148:957–970. 10

Ren, X. D., Kiosses, W. B., Sieg, D. J., Otey, C. a., Schlaepfer, D. D., and Schwartz, M. a. (2000). Focal adhesion kinase suppresses Rho activity to promote focal adhesion turnover. *Journal of Cell Science*, 113:3673–8. 9

Rottner, K., Hall, A., and Small, J. V. (1999). Interplay between Rac and Rho in the control of substrate contact dynamics. *Current Biology*, 9:640–648. 11

Sachs, K., Gifford, D., Jaakkola, T., Sorger, P., and Lauffenburger, D. A. (2002). Bayesian network approach to cell signaling pathway modeling. *Science's STKE : signal transduction knowledge environment*, 2002:pe38. 14

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., and Nolan, G. P. (2005). Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–9. 14, 15, 82

Santos, S. D. M., Verveer, P. J., and Bastiaens, P. I. H. (2007). Growth factor-induced MAPK network topology shapes Erk response determining PC-12 cell fate. *Nature Cell Biology*, 9(3):324–30. 15, 16, 105

Sasagawa, S., Ozaki, Y.-i., Fujita, K., and Kuroda, S. (2005). Prediction and validation of the distinct dynamics of transient and sustained ERK activation. *Nature Cell Biology*, 7(4):365–73. 84, 85

Saunders, R. M., Holt, M. R., Jennings, L., Sutton, D. H., Barsukov, I. L., Bobkov, A., Liddington, R. C., Adamson, E. A., Dunn, G. A., and Critchley, D. R. (2006). Role of vinculin in regulating focal adhesion turnover. *European Journal of Cell Biology*, 85:487–500. 12

Schaller, M. D., Hildebrand, J. D., Shannon, J. D., Fox, J. W., Vines, R. R., and Parsons, J. T. (1994). Autophosphorylation of the focal adhesion kinase, pp125FAK, directs SH2-dependent binding of pp60src. *Molecular and Cellular Biology*, 14:1680–1688. 9

Schiller, H. B. and Fässler, R. (2013). Mechanosensitivity and compositional dynamics of cell-matrix adhesions. *EMBO reports*, 14(6):509–19. 4

Schubert, W., Bonnekoh, B., Pommer, A. J., Philipsen, L., Böckelmann, R., Malykh, Y., Gollnick, H., Friedenberger, M., Bode, M., and Dress, A. W. M. (2006). Analyzing proteome topology and function by automated multidimensional fluorescence microscopy. *Nature Biotechnology*, 24:1270–1278. 82

Sjöblom, B., Salmazo, A., and Djinović-Carugo, K. (2008). Alpha-actinin structure and regulation. *Cellular and Molecular Life Sciences : CMLS*, 65(17):2688–701. 6, 9, 97

Small, J. V., Rottner, K., Kaverina, I., and Anderson, K. I. (1998). Assembling an actin cytoskeleton for cell attachment and movement. *Biochimica et Biophysica Acta - Molecular Cell Research*, 1404:271–281. 13

Spanjaard, E. and de Rooij, J. (2013). Mechanotransduction: vinculin provides stability when tension rises. *Current Biology*, 23(4):159–161. 11, 12

Turner, C. E., Brown, M. C., Perrotta, J. A., Riedy, M. C., Nikolopoulos, S. N., McDonald, A. R., Bagrodia, S., Thomas, S., and Leventhal, P. S. (1999). Paxillin LD4 motif binds PAK and PIX through a novel 95-kD ankyrin repeat, ARF-GAP protein: A role in cytoskeletal remodeling. *The Journal of Cell Biology*, 145:851–863. 10, 104

Turner, C. E., Glenney, J. R., and Burridge, K. (1990). Paxillin: a new vinculin-binding protein present in focal adhesions. *The Journal of Cell Biology*, 111:1059–1068. 10

Turner, C. E. and Miller, J. T. (1994). Primary sequence of paxillin contains putative SH2 and SH3 domain binding motifs and multiple LIM domains: identification of a vinculin and pp125Fak-binding region. *Journal of Cell Science*, 107:1583–1591. 10

von Wichert, G., Haimovich, B., Feng, G.-S., and Sheetz, M. P. (2003). Force-dependent integrin-cytoskeleton linkage formation requires downregulation of focal complex dynamics by Shp2. *The EMBO Journal*, 22:5023–5035. 6

Wachsstock, D. H., Wilkins, J. A., and Lin, S. (1987). Specific interaction of vinculin with alpha-actinin. *Biochemical and Biophysical Research Communications*, 146:554–560. 6

Webb, D. J., Donais, K., Whitmore, L. a., Thomas, S. M., Turner, C. E., Parsons, J. T., and Horwitz, A. F. (2004). FAK-Src signalling through paxillin, ERK and MLCK regulates adhesion disassembly. *Nature Cell Biology*, 6(2):154–61. 9, 11

Wehrle-Haller, B. (2012). Structure and function of focal adhesions. *Current Opinion in Cell Biology*, 24(1):116–24. 2

Weng, Z., Taylor, J. A., Turner, C. E., Brugge, J. S., and Seidel-Dugan, C. (1993). Detection of Src homology 3-binding proteins, including paxillin, in normal and v-Src-transformed Balb/c 3T3 cells. *The Journal of Biological Chemistry*, 268:14956–14963. 10

Xu, W., Coll, J. L., and Adamson, E. D. (1998). Rescue of the mutant phenotype by reexpression of full-length vinculin in null F9 cells; effects on cell locomotion by domain deleted vinculin. *Journal of Cell Science*, 111:1535–1544. 12

Yang, Y., Guo, L., Blattner, S. M., Mundel, P., Kretzler, M., and Wu, C. (2005). Formation and phosphorylation of the PINCH-1-integrin linked kinase-alpha-parvin complex are important for regulation of renal glomerular podocyte adhesion, architecture, and survival. *Journal of the American Society of Nephrology : JASN*, 16:1966–1976. 10

Yu, J., Smith, V. A., Wang, P. P., Hartemink, A. J., and Jarvis, E. D. (2004). Advances to Bayesian network inference for generating causal networks from observational biological data. *Bioinformatics (Oxford, England)*, 20:3594–3603. 14

Zaidel-Bar, R., Ballestrem, C., Kam, Z., and Geiger, B. (2003). Early molecular events in the assembly of matrix adhesions at the leading edge of migrating cells. *Journal of Cell Science*, 116:4605–4613. 12

Zaidel-Bar, R. and Geiger, B. (2010). The switchable integrin adhesome. *Journal of Cell Science*, 123(Pt 9):1385–8. 4

Zaidel-Bar, R., Itzkovitz, S., Ma'ayan, A., Iyengar, R., and Geiger, B. (2007). Functional atlas of the integrin adhesome. *Nature Cell Biology*, 9(8):858–67. 2, 5, 6

Zamir, E. and Bastiaens, P. I. H. (2008). Reverse engineering intracellular biochemical networks. *Nature Chemical Biology*, 4(11):643–647. 15, 71

Zamir, E. and Geiger, B. (2001a). Components of cell-matrix adhesions. *Journal of Cell Science*, 114(Pt 20):3577–9. 2

Zamir, E. and Geiger, B. (2001b). Molecular complexity and dynamics of cell-matrix adhesions. *Journal of Cell Science*, 114(Pt 20):3583–90. 2, 4, 5, 6

Zamir, E. and Geiger, B. (2013). Focal Adhesions and Related Integrin Contacts. In Lennarz, W. and Lane, M., editors, *Encyclopedia of Biological Chemistry*, pages 318–323. Elsevier, Oxford, second edition. 3

Zamir, E., Geiger, B., Cohen, N., Kam, Z., and Katz, B.-Z. (2005). Resolving and classifying haematopoietic bone-marrow cell populations by multi-dimensional analysis of flow-cytometry data. *British Journal of Haematology*, 129:420–431. 82

Zamir, E., Geiger, B., and Kam, Z. (2008). Quantitative multicolor compositional imaging resolves molecular domains in cell-matrix adhesions. *PloS One*, 3(4):e1901. 4, 16, 18, 24, 35, 36, 37, 97, 111

Zamir, E., Katz, B. Z., Aota, S., Yamada, K. M., Geiger, B., and Kam, Z. (1999). Molecular diversity of cell-matrix adhesions. *Journal of Cell Science*, 112:1655–69. 6, 32, 62

Zamir, E., Katz, M., Posen, Y., Erez, N., Yamada, K. M., Katz, B. Z., Lin, S., Lin, D. C., Bershadsky, a., Kam, Z., and Geiger, B. (2000). Dynamics and segregation of cell-matrix adhesions in cultured fibroblasts. *Nature Cell Biology*, 2(4):191–6. 6

Zhang, Z., Lin, S.-Y., Neel, B. G., and Haimovich, B. (2006). Phosphorylated alpha-actinin and protein-tyrosine phosphatase 1B coregulate the disassembly of the focal adhesion kinase x Src complex and promote cell migration. *The Journal of Biological Chemistry*, 281:1746–1754. 6

Zhu, H., Rao, R. S. P., Zeng, T., and Chen, L. (2012). Reconstructing dynamic gene regulatory networks from sample-based transcriptional data. *Nucleic Acids Research*, 40(21):10657–67. 15

Ziegler, W. H., Liddington, R. C., and Critchley, D. R. (2006). The structure and regulation of vinculin. *Trends in Cell Biology*, 16(9):453–60. 11, 12

Zimerman, B., Arnold, M., Ulmer, J., Blümmel, J., Besser, A., Spatz, J. P., and Geiger, B. (2004). Formation of focal adhesion-stress fibre complexes co-ordinated by adhesive and non-adhesive surface domains. *IEE Proceedings. Nanobiotechnology*, 151:62–66. 11, 104

Zou, C., Denby, K. J., and Feng, J. (2009). Granger causality vs. dynamic Bayesian network inference: a comparative study. *BMC Bioinformatics*, 10:122. 15

Zou, M. and Conzen, S. D. (2005). A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data. *Bioinformatics (Oxford, England)*, 21(1):71–79. 15