DIGITAL ACCESS TO SCHOLARSHIP AT HARVARD
DASH.HARVARD.EDU

HARVARD LIBRARY
Office for Scholarly Communication

Experimental and Computational Tools to Study P53 Dynamics at the Single-Cell Level

Citation
Karhohs, Kyle Wayne. 2015. Experimental and Computational Tools to Study P53 Dynamics at the Single-Cell Level. Doctoral dissertation, Harvard University, Graduate School of Arts & Sciences.

Permanent link
http://nrs.harvard.edu/urn-3:HUL.InstRepos:23845465

Terms of Use
This article was downloaded from Harvard University's DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA

Share Your Story

The Harvard community has made this article openly available.
Please share how this access benefits you. Submit a story .

Accessibility

Experimental and Computational Tools to Study p53 Dynamics at the Single-Cell Level

A DISSERTATION PRESENTED
BY
KYLE WAYNE KARHOHS
TO
THE SYSTEMS BIOLOGY DEPARTMENT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN THE SUBJECT OF
SYSTEMS BIOLOGY

HARVARD UNIVERSITY
CAMBRIDGE, MASSACHUSETTS
JUNE 2015

2015 -Kyle Wayne Karhohs

This work is licensed under the Creative Commons Attribution-NonCommercial 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc/4.0/.

Dissertation Advisor: Professor Lahav Kyle Wayne Karhohs

Experimental and Computational Tools to Study p53 Dynamics at the Single-Cell Level

Abstract

One of the most commonly mutated genes found in cancer is the tumor suppressor p53. p53 is a transcription factor capable of inducing cell-cycle arrest, apoptosis, senescence, and other cellular processes thought to halt the progression of a nascent cancer. As part of a stress signaling pathway, p53 is acutely activated by ionizing radiation and the formation of DNA double-strand breaks. The appearence of this DNA damage causes the concentration of p53 within the nucleus to fluctuate and pulse regularly, which can be observed in single cells using fluorescence time-lapse microscopy. From the time this was first discovered, the connection between these p53 dynamics and p53 function has been speculated upon. A key insight into this connection came from a Lahav Lab publication that demonstrated the act of pulsing, itself, controls p53-dependent transcription and cell fate. The mechanisms and molecular details behind this relationship are now an area of intense study. Another area of high interest is the broader characterization of p53 dynamics in different time-scales, genetic backgrounds, and stresses. These lines of research each depend upon single-cell measurements that are often time consuming, noisy, and yield small sample sizes. The ongoing development of experiemental and computational tools for single-cell biology is needed to overcome these limitations. In the publication referenced earlier, a novel method was created to measure p53 dynamics and gene expression in the same cell. In a seperate study characterizing p53 dynamics over long time-scales, semi-automated tracking software aided in the discovery of new p53 dynamics: sustained elevation of p53 levels that follow a period of pulsing. Population measurements showing similarly elevated p53 levels on the same time-scale are shown to depend on the late induction of the p53-target PIDD.

Contents

List of Figures

List of Tables

This thesis is dedicated to Kayla, my meteor. My hope from the start was that my scientific pursuits would honor her memory.

Additionally, this thesis is dedicated to Sonia, my loving wife, who has been my witness and confidant in graduate school. She has been a wellspring of encouragement in the valleys and a cherished reveler at the summits.

Author List

The following authors contributed to these chapters:

Chapter 1: Kyle W. Karhohs

Chapter 2: Kyle W. Karhohs, Jeremy E. Purvis, Caroline Mock, Eric Batchelor, Alexander Loewer, and Galit Lahav. This work appeared in print in *Science* June 2012 Volume 336, Issue 6087.

Chapter 3: Kyle W. Karhohs, Fabian Degener, Trudy Oliver, and Caroline Mock. Caroline Mock collected and performed the experiments for the population data including qPCR and Western Blot data. She also created a series of PIDD inducible cell lines. Fabian assisted in data collection and experimental design during his 9 month tenure with the lab. Trudy Oliver kindly donated plasmids for the inducible PIDD construct and advised on PIDD-p53 interaction.

Chapter 4: Kyle W. Karhohs and Jose Reyes. The code contained within this thesis is my creation, but it is dependent on code created by Jose Reyes. Jose Reyes and I also formed the vision of the tracking software together.

Chapter 5: Kyle W. Karhohs

Acknowledgments

Family is why I am where I am and whom deserve a great deal of thanks. First, I want to thank my wife Sonia. Our love story is the backdrop to the science (or vice versa). I cannot disentangle my graduate school experience from her. She was also a student, so she honestly empathized with all of my stresses and successes. It has a been a journey and I am so glad Sonia has been there every step of the way.

Most of my other family resides on the West Coast, but has always rooted me on from afar. I am grateful for my parents. I think that my father, Jeff, has ended every phone call I've had with him since I moved to Boston with, "We're all so proud of you, son.“ Hearing words of encouragement like these always lifted me up, especially whenever I was lost in a fog of self-doubt. My mother, Dawn, was always interested in my well-being and made sure I was eating well by sharing recipes and gadgets (s.a. the mini-pie maker). I especially am thankful for her visits to Boston, because these were times I could truly share what I was trying to accomplish; conveying the day-to-day of basic research is difficult to do over the phone. I am grateful for my grandma, Jo, and all of her love and support. I respect no one as much as her and I can only hope that I provided her with some good material when she and her friends talk about their grandchildren. I am grateful for my brother, Reid. His boundless curiousity has always been an inspiration for my own work. Also, his availability to play some online shooters has always been a greatly appreciated source of stress relief. I am also grateful for all my other parents, siblings, aunts, uncles, cousins, and in-laws too: Jayne, Doug, Casey, Kitt, Nicole, Jack, Kelly, Patti, Walt, Jim, Kathy, Debbie, Topher, Katie, Jamie, Brian, Alex, and Greg. I sincerely

apologize for not recognizing each of you with your own acknowledgement, but please know that I am thankful to have each of your in my life.

I would especially like to recognize and thank my Grandpa Len, Uncle Jack, Grandma Theora, and Grandma Sylvia who all passed away during my studies. Each of them had shown me an enormous amount of love and support and I will never forget them.

I am grateful for my in-laws Jeff and Ronda who welcomed me into their family and have also supported my studies. I am grateful for the Nebeker family from Reno and Salt Lake City for their understanding and support. They were unconditionally loving and generous in helping me reach my goals in Boston. I am grateful for Mel and Jared who in addition to being devoted cheerleaders let me use their basement for drafting my Harvard application.

I owe a great debt of gratitude to my lab family. Galit emphasizes and nutures a positive and collaborative lab culture and that culture is embodied by its members. I am grateful for Eric, and his humor, who took me under his wing and introduced me to microscopes and benchwork. I am grateful for Alex, Jared, and Ran who were each paragons of science and always made themselves available for questions. I am grateful for Ketki, Giorgio, and Julia who each set a great example as graduate students and for their commraderie. I am grateful for Jeremy and his time and tutilege; I admired his perseverance and kindness. I am grateful to Andrew for teaching me cloning. I am grateful for Sheng and his deep thinking. I am grateful for Tonia for always bring a smile to work and teaching ChIPSeq. I am grateful to Jacob for showing what hard work can accomplish. I am especially grateful to Jose for being such a good friend and bringing enthusiasm to the computational room everyday. Thanks to everyone for their feedback during group meetings and the daily interactions that helped move my project forward.

A special thanks goes to Galit who is a phenomenal scientist and phenomenal mentor. A special thanks also goes to Caroline. She is extremely resourceful and has answers for every question, including crossword puzzles at lunch. Caroline was of invaluable help to me during my thesis work. She committed a lot of time, energy, and thought to the questions at hand and made numerous and substantial contributions to the experimental work. I cannot thank her enough.

I am grateful for Sam and dissertation committee that includes Tim, John, and Jeremy. Sam was instrumental in making sure I was always on schedule. Tim, John, and Jeremy were always encouraging and always made themselves available during their office hours.

1 Introduction

1.1 Signal Transduction and Systems Biology

An essential quality of the cell is the ability to respond and adapt to changes in its external environment or internal processes. Signal transduction research aims to understand the molecular details concerning the chain of events that lead from a triggering stimulus to a cellular response. Signal transduction is a line of questioning that can be applied to almost any facet of molecular biology. This open-endedness has inspired a search for patterns in signaling pathway architecture and network motifs [1], to provide a more general understanding of the flow of information within a cell [2], which can otherwise appear to be a tangled web of esoteric interactions to the uninitiated. One aim of systems biology is to apply and develop experimental and computational techniques that reveal how the interactions between individual signaling components create the emergent property of cellular decision-making [3]. Such knowledge is valuable in the field of cancer medicine where a quantitative understanding of signaling contributes to the development of next-generation cancer therapies [4].

The importance and complexity of signaling in multicellular organisms is evinced by the orchestration of differentiation, both spatially and temporally, that is studied in developmental biology [5]. Cells are in constant communication with neighboring cells and sample cytokines, metabolites, and hormones found within the extracellular milieu. From a genetic perspective, bioinformatitions discovered there are over 500 kinases encoded in the human genome, or roughly

I have been inspired by the idea that signal transduction can be defined and analyzed in a manner comparable to electronic circuits [6], and signaling pathways are modular units with predictable input-output relationships. This idea resonated with my background in electrical engineering and was the toehold I used to climb into biology research. I've learned this analogy is challenged by the fuzziness of molecular interactions; evolution does not mold these connections for the benefit of human understanding, unlike a circuit designed by an engineer[7]. The combinatorics of molecular interactions and numerous protein states/modifications [8], in addition to stochastics seen within individual cells [9], make it difficult to quantify signaling pathways with exactitude. Fortunately, signaling pathways can still be understood with incomplete information [10] and toy models of pathways can effectively represent complex systems [11].

1.2 Signal Transduction, Cancer, and p53

The study of signaling is also important for human health. Many diseases are the result of out-of-control signaling or a loss of sensitivity to a signal. A primary example is cancer. One of the first oncogenes discovered, Src derived from the Rous Sarcoma Virus, is a tyrosine kinase [12] that produces a strong proliferative signal. Since then many of the hallmarks of cancer have been found to be the result of corrupted signaling [13]. The same genes are found to be mutated, deleted, or amplified over and over again [14], because the disruption of a single key member of a signaling pathway can alter a large number of downstream events.

Genes can be classified as oncogenes or tumor suppressors if their activation or suppression leads to cancer development, respectively. A detailed understanding of the signaling networks and pathways that contain oncogenes and tumor suppressors has led to the development of targeted anti-cancer drugs. For example, the EML4-ALK translocation is repeatedly found in non-small-cell lung cancer leading to oncogenic ALK kinase activity. Lung cancers with this particular mutation can be treated successfully with ALK inhibitors such as ceritinib. However, these cancers usually acquire resistance about a year after the initial treatment [15]. Studying the changes in signaling before and after resistance is acquired can lead to improved combination therapies that slow or prevent the onset of resistance. This strategy led to the discovery that protein-kinase-C was necessary for resistance to ALK inhibitors [15] and demonstrates how detailed knowledge of signaling can improve cancer therapies.

The most commonly mutated tumor suppressor found in cancer is p53 [14]. P53 is a transcription factor that is activated by a wide range of stresses, especially DNA damage [16]. In response to DNA damage, p53 is activated by upstream kinases, which describes part of the DNA damage response (DDR) signaling pathway [17]. In turn a wide range of genes are induced that control cell-cycle arrest, apoptosis, and senescence [18]. One paradox of p53 function is that it activates both pro-survival and pro-death genes, yet a cellular decision can only lead to one of these fates. The confusion between p53 mediated survival and death in response to DNA damage, specifically ionizing radiation treatment, is demonstrated by the radiosensitivity of various organs and tissues in the human body [19]. A similar puzzle is presented by the a mixed response of a population to a singular stimulus, e.g. the fraction of cancer cells undergoing apoptosis in response to ionizing radiation is dose dependent. Invesitgating p53 signaling at the single-cell level will help resolve these paradoxes by being able to measure differences between cells and mapping them to that cell's fate.

1.3 Signal Transduction and Protein Dynamics

Systems biology has been practiced for decades by researchers who did not know it. Schoenheimer first showed how the proteins in a cell are in a dynamic state of constant replication and turnover [20]. This work was performed in the 1930s and made possible through the use of isotopes in a method to chemically label proteins in living cells. However, due to the limitations of methods available at the time to isolate proteins and metabolites these methods could only reasonably probe a handful of compounds in any given experiment. The regulation of protein turnover, modulating protein stability through post-translational modifications, is often a key element of signal transduction. Nearly a century later, Alon was able to survey the stability and turnover of hundreds of proteins in a single study using fluorescent-protein tags [21]. In p53 signaling, the regulation of protein turnover is an essential part of its dynamics [22].

Protein dynamics are an essential aspect of signaling. Advances in the creation of antibodies that target specific signaling proteins made it possible to study signaling dynamics with protein gels. A particularly elegant study of Xenopus oocyte maturation explored the signaling of progesterone stimulus at molecular detail to reveal a hyper-sensitive switch in the MAPK pathway [23]. The technologies that enable the study of dynamics in single-cells were nascent, or unavailable entirely, in the mid-1990s. However, these limitations were circumvented by the fact that the Xenopus oocyte is itself a very large single cell that could be assayed in ways microscopic cells could not. Each cell could be run in its own gel lane. A population, or average, response of MAPK signaling to progesterone was a graded response, yet only single cell studies could reveal whether or not this was due to a gradual rise in signaling across the population or bi-modal response where the population is split between responders and non-responders. The MAPK signaling was shown to be bi-modal with an increasing fraction of the population participating with increasing progesterone. Furthermore, mathematical modeling of the pathway suggested the commitment to maturation, the cell fate switch, resulted from both ultrasensitivity and positive feedback in the MAPK pathway to produce an all-or-none response within individual oocytes. The questions raised in this study[23] continue to be asked today about other signaling pathways in other cellular contexts at the single cell level.

In addition to dynamics governed by switches are oscillations. Perhaps the most obvious oscillation in biology, or periodic rhythm, that everyone can observe is the beating heart. Interestingly, oscillations are also present within individual cells. While the mathematics of oscillations unite these phenomenon across multiple scales and disciplines, the nature of the oscillations can be quite different and unique [24]. In contrast with this variety and complexity, a synthetic oscillator, dubbed the "repressilator", was created within bacteria that demonstrated the relative simplicity needed to establish persistent oscillations within a cell [25]. In light of the minimal requirements needed to establish oscillations it is then not surprising to learn that many natural signaling pathways that contain transcription factors have been shown to exhibit oscillatory behavior.

Protein dynamics of p53 can determine transcription factor function [26], but this is a more general phenomenon. The transcription factor NFKB has similar pulsing dynamics in response to stress that influence the regulation of its gene-targets [27]. The study by Tay et al. demonstrates that the population response of NFKB to TNF hides single-cell behavior and that response is all-or-none similar to the Xenopus oocyte response to progesterone. Like p53, NFKB oscillates after stimulation.

The Crz1 transcription factor, activated by calcium stress, in yeast has temporal dynamics, like p53. However, in contrast to almost oscillatory p53 dynamics Crz1 has more frequent, yet irregular bursts of activity. The frequency of Crz1 bursts is dependent on the concentration of the calcium input. In a study of Crz1 dynamics it was demonstrated that Crz1-target gene expression was frequency modulated by the transcription factor dynamics [28]. The affinity model driven by transcription factor bursting was shown to maintain the relative ratio of target genes over a broad range of induction. The authors speculate that p53 oscillations may function similarly.

Not all pathways may have an oscillatory element to their response, but are instead stimulated by periodic signals. For example, the rising and setting of the sun each day entrains the fluctuatation of circadian rhythm proteins [29]. The function of a pathway can be explored by systematically varying the frequency of the input and studying the response. A study using this technique demonstrated that Hog1 pathway in yeast that responds to osmotic shock in a frequency dependent manner [30]. Using oscillations to study the Hog1 pathway, it was determined that the transcriptional function of Hog1 was not necessary for the initial response to osmotic shock, which takes places on a shorter time-scale, but increases the sensitivity to osmotic shock on a longer time scale due to increased production of glycerol transporters. This separation of protein function between short term and long term responses may be a general feature of transcription factor pathways and is relevant to the p53 response to gamma radiation.

Frequency based analyses have also been applied to the p53 pathway [31]. It was shown that the p53 oscillations, or pulses, in response to ionizing radiation can be reasonably modeled by a negative feedback loop between p53 and mdm2, and a second negative feedback loop between p53 and the upstream kinase ATM. This unique perspective into p53 dynamics supports a model where p53 induction of Wip1 leads to de-phosphorylation of ATM [22]. These two studies are evidence that the function and nature of a signaling pathway can be explored whether a system naturally oscillates or is forced to oscillate.

Heterogeneity is present in all biological systems across many scales and is most profound when something seemingly uniform consists of diverse constituent components. Early in the development of light microscopy an appreciation for heterogeneity was fostered by Robert Hooke who described how the composition of cork, when viewed at high magnification, consists of round, yet irregularly shaped, cells. It is now common knowledge that plants, like the cork oak, and animals are each collections of a large number of cells. The human body has more than 10^{13} cells [32], yet only an estimated 200 distinct cell types [33]. The numerous cells of a given type within a tissue or organ may look the same, but sub-cellular differences in proteins and DNA can make them unique. Most of these differences will not have an impact on the function or behavior of a cell, but sometimes this heterogeneity can have important consequences, especially concerning the development of tumors.

Heterogeneity is exploited by the process of evolution most dramatically on the longest of timescales. Millions of years of selective pressures on subtle differences has contributed to the great diversity and complexity of today's living organisms. On shorter timescales, such as on the order of a human lifetime, heterogeneity and selective pressures can have important implications for an individual's health. The development of cancer relies upon changes and variations that favor proliferation. Evidence for this comes from examples of carcinogens, which are commonly mutagens that increase the frequency of edits and changes in DNA. Exposure to carcinogens such as coal tar or UV radiation alters DNA and can lead to cancer.

2
P53 Dynamics Control Cell Fate

2.1 Introduction

P53 activates the transcription of hundreds of genes [34][35] that regulate cell cycle arrest, senescence, and apoptosis [36]. The balance between arrest and apoptosis is affected by different stimuli, such as DNA damage or oncogene activation, and tissue origin, for example the thymus or colon [16] (Figure 2.1). One of the central mysteries surrounding the function of p53 as a transcription factor is how it is able to differentially

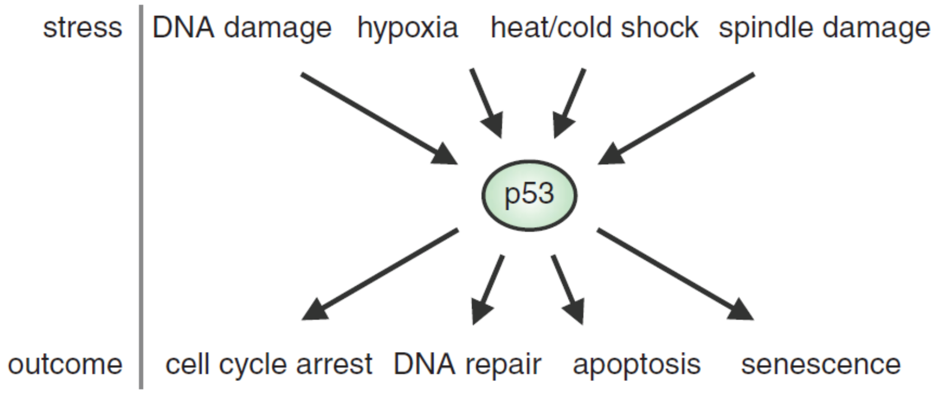


Figure 2.1: p53 is a hub protein that is activated by a wide range of upstream stresses and signals. In turn, p53 acts as a transcription factor that activates genes involved in cell fates such as apoptosis and senescence.

regulate such a large number of genes, members of antagonizing pathways, in a cell-type specific manner. The possibility that p53 dynamics could directly influence target gene regulation was explored in a 2012 study [26] where p53 dynamics were shown to control both gene expression and cell fate. As part of that study a novel experimental technique was developed to measure p53 dynamics and gene expression at single-cell resolution.

2.2 Modes of p53 Regulation

There are many factors that contribute to p53 transcription factor activity. The composition of DNA where p53 binds is a natural place to start. Many genes contain a pair of p53 response elements (p53re) of the sequence pattern RRRCWWGYYY [37]. For example, the gene GADD45 has a p53re of GAACATGTCT. Variations in these sequences and the gap between a pair of p53re affect p53 binding to the DNA and influences the cooperativity of p53 tetramerization during the binding process [38]. Most p53 mutants affect the DNA-binding-domain, disabling transcriptional activity

[39].

P53 protein is also highly modified by post-translational modifications (p53ptm) [40]. P53ptm stabilizes p53 in response to DNA damage [41], leads to degradation through the proteasome [42], and is necessary for tetramerization [43]. These modifications include phosphorylation, acetylation [44], and ubiquitylation. In general, p53ptm regulate the protein-protein interactions that make p53 a hub protein capable of integrating diverse upstream signals [45].

2.3 The Affinity Model and Gene Selectivity

The combinatorics of p53re and p53ptm already represents enormous complexity without even considering other factors such as the chromatin state of a promoter or the relative abundance of each post-translation-modification within the pool of p53 in the cell. Even the 3D configuration of DNA and looping has been shown to influence the potential of p53 binding [46]. Despite all of the variables influencing the propensity of p53 to bind to its target genes the binding behavior can be generally described, and abstracted, as a Hill function that depends on the concentration of the p53 protein within a cell. The parameters of cooperativity and binding constant would all be functions of the sequence information and protein state described above.

A set of equations that represent the p53 binding activity for every target gene is an affinity model. Using an affinity model to describe p53 transcription factor function implies that the concentration of p53 is the most important factor in determining the fate of a cell. Each gene would have its own threshold of activation based upon the p53ptm status and promoter availability. Such a model gives a pleasing explanation to how a cell will determine the path of cell-cycle arrest versus apoptosis: The apoptotic genes would have a higher threshold than arrest genes, so only when p53 protein reach elevated concentrations would apoptosis be activated. In support of this model is a paper that demonstrated P53-driven apoptosis is dependent on highly cooperative binding, whereas cell-cycle arrest is not [47].

2.4 p53 Dynamics and Gene Selectivity

To this point in the chapter everything describing the regulation of p53 as a transcription factor does not explicitly acknowledge any temporal dependency on p53 function. It is easier to ignore dynamics and focus on a single time point or steady state condition given the number of components in the p53 network, let alone the technical challenges required to acquire time-series data. However, when oscillatory p53 dynamics were discovered in response to gamma radiation it was speculated that the function of p53 pulses might be to prevent p53 levels from crossing the apoptosis threshold, while simultaneously arresting the cells [48]. Expanding the affinity model to account for time would make possible the proposed, more sophisticated, gene expression patterns.

In 2012, the connection between transcription factor dynamics and transcription had never been demonstrated in p53. One of the central challenges of this task was separating the influence of p53 dynamics from the other regulators of transcription factor activity that were outlined at the beginning of the chapter. Many variables relating to gene promoters, sequence, and chromatin configuration could be held fixed by using the genetic background, i.e. consistently using the same cell line, and measuring the

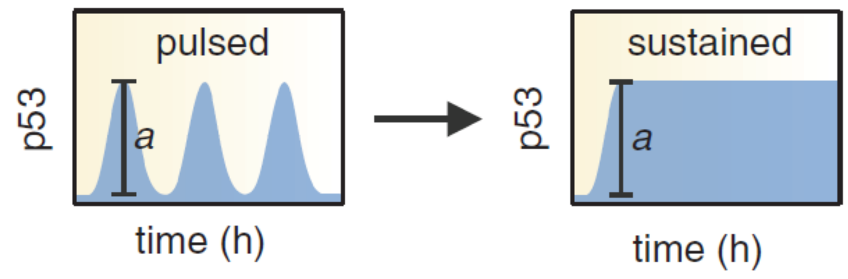


Figure 2.2: Nutlin-3 was used to transform the pulsing behavior stimulated by gamma radition into sustained p53 levels [26].

transcription of several well-characterized p53 targets such as p21 and PML. A greater challenge was with regard to post-translational modifications. P53 pulses follow stimulation by gamma radiation and is phosphorylated and acetylated at residues specific to this stimulus [49], so any perturbations to p53 dynamics would need to happen concurrently with irradiation.

2.5 Nutlin-3 Alters p53 Dynamics

The perturbation of p53 dynamics was enabled by a small molecule inhibitor of p53, Nutlin-3 [50]. We augmented an existing model of p53 dynamics [49] to incorporate the influence of Nutlin-3 on p53 behavior. We used his model to design a protocol of three Nutlin-3 additions to a cell culture and transformed pulsing behavior into sustained behavior over the first 24 hours following gamma radiation (Figure 2.2). One crucial element of the p53 dynamics that to be controlled was the amplitude of the sustained dynamics. This had to remain similar to the amplitude of the pulses, elsewise differences in expression contributed by differential dynamics may become masked by strong expression induced by unusually high p53 levels.

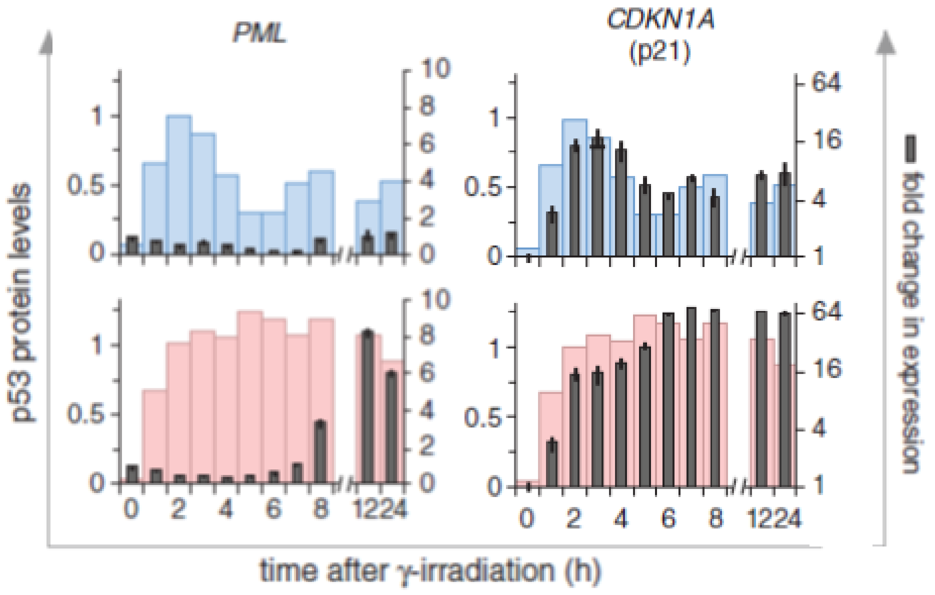


Figure 2.3: The p53-targets PML and p21 each have responses unique to pulsing and sustained p53 dynamics.

The altered dynamics were confirmed via Western blot, and qPCR measurements revealed that several genes had expression patterns that were unique to pulsing and sustained dynamics. In particular, two genes showed dramatically different behavior. Under pulsing conditions, the p21 gene mirrored the pulsatile behavior of the p53 protein, whereas it was induced to levels almost 10-fold higher with sustained dynamics (Figure 2.3). The PML gene, on the other had almost no response over the first 24 hours under pulsing conditions, yet in sustained conditions PML was induced like a delayed switch (Figure 2.3).

2.6 Measuring Sustained p53 Dynamics in Single Cells

The transformation from pulsing to sustained dynamics up to this point had reflected population averages of behavior. Repeating this experiment at the single cell level would provide more quantitative data that could be used to map the relationship between p53 dynamics and gene expression. Furthermore, an additional, distinct approach that revealed the same relationship between dynamics and expression would strengthen the finding.

There were several technical challenges to this experiment. First, the experiment had to be planned around several important time points. The first important time was the moment of irradiation. Every other time point was relative to this moment. The Nutlin-3 protocol consisted of three additions of Nutlin-3 to the cells in gridded-glass-bottom dishes while being imaged on a Nikon TI microscope (Table 2.1). One pitfall came from the necessity to have the media be replaced completely for each addition. Imaging at high magnification is very sensitive to changes in XYZ, so bumping the dishes containing the cells could potentially move the cells out of frame or out of focus. Due to evaporation concerns this also required removing and replacing a glass coverslip on top of each dish. We learned to use steady hands and a delicate touch. After performing the Nutlin-3 protocol we observed the p53 dynamics at the single cell level had the predicted sustained behavior (Figure 2.4).

Time (h post-IR)	Nutlin Conc. (uM)
2.0	0.75
3.5	2.25
5.5	4.00

Table 2.1: The Nutlin-3 protocol consisted of 3 additions of Nutlin-3 at increasing concentrations. The concentration had to be increased to compensate for rising levels of mdm2.

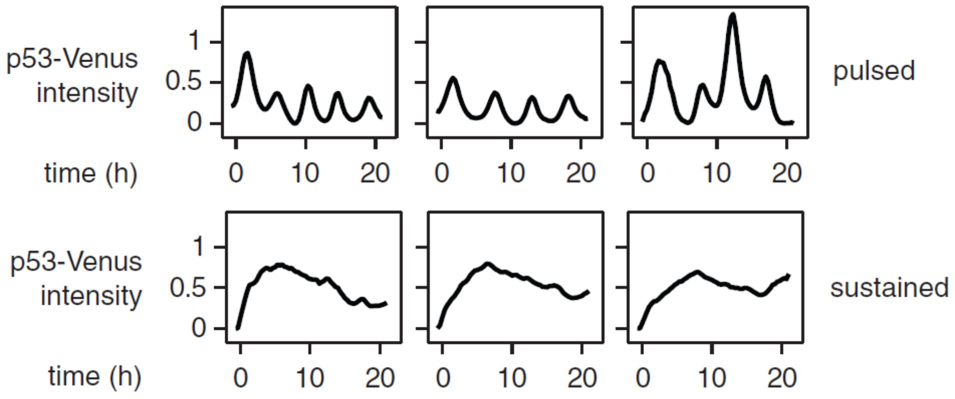


Figure 2.4: Pulsing cells have the period and amplitude expected following activation by ionizing radiation. Sustained cells have similar levels to the peak amplitude of an average p53 pulse. The levels are sustained through-out the duration of the observation.

2.7 Measuring Gene Expression in Single Cells with smFISH

Individual p21 and PML transcripts would be measured in each cell using single-molecule FISH [51], but this required us to fix cells between the acquisition of p53 dynamics and imaging of smFISH. This type of experiment was novel at the time, so we had to develop and troubleshoot the fixation protocol. To begin we tried removing the glass coverslip from the dish using a solvent but after our first attempt we looked for another approach. After the coverslip was removed from the dish it was difficult to find the correct alignment and find the same cells from the movie. Also, as bubbles caught between the slide move-around they can detach cells. It is terribly disappointing when trying to relocate your cells only to find they didn't survive the mounting process. In the end we found performing smFISH, or fixation in general, within the MatTek dish the most expedient approach following a movie, the biggest trade-off being increased use of reagents.

Using MatTek dishes is a matter of scaling reagents. We found that is was sufficient to add 100uL of antibody solution and smFISH probes to the inner well, i.e the cavity created by the hole within the plastic dish and the coverslip underneath. We also found that we did not need to use the same concentrations and could afford to use roughly half the concentration, or roughly the same total reagent when using coverslips, just in a larger volume. When we would do washes we would use 2mL per wash.

The biggest challenge when working with the inner-well of a Mattek dish is preventing total evaporation. In my experience this ruin a sample. Therefore, when adding 100uL volumes I work with 1 dish at a time. I would start by aspirating at the outer edge of the MatTek dish while holding it at a slight angle. This will remove most liquid, but a noticeable amount within the inner well will remain due to surface tension. Often this is too much liquid and adding an additional 100uL would cause the fluid to overflow onto the surface of the plastic dish, resulting in a loss of reagent and inefficient staining. Therefore, this liquid must be removed, too, but without drying out the sample entirely. Do this next part quickly: Load a pipette with 100uL of the probe and set it aside momentarily. Then aspirate the liquid remaining within the inner-well by placing the aspirator-tip near the edge of the inner-well on the plastic surface of the dish. Then, slowly, move the tip closer and closer to the edge of the inner-well until liquid is being sucked into the aspirator. The center of the inner well should never become completely dry. Finally, grab the pipette and carefully add the reagent drop by drop at the edge of the inner-well until it is full again.

When incubating with this volume, evaporation is still an issue. For incubation at 37C we used a tissue culture incubator, because of its high humidity. If you are using 4C we would add sponges saturated with water to a closed container that also contains the MatTek dishes in order to increase the humidity.

After the sample is prepared there are further challenges when imaging. It is not unusual to have high background signal. This was especially troublesome when mRNA counts are low, because it wasn't always clear if the image was full of background or the sample was devoid of signal. Unfortunately, the only solution seemed to be trial and error testing different exposures. If this is an issue the first thing to do is image a sample that has not been treated with smFISH to establish a baseline for the background noise. The exposure lengths to acquire smFISH signal are much longer than for fluorescence time-lapse microscopy. If at first it looks like no signal try increasing the exposure to times >1000ms. Seeing the smFISH foci for the first time was a eureka moment (Figure 2.5).

There is a balancing act between exposure length and the rate of photobleaching. We found photobleaching was the number one concern in our imaging. We would use an enzymatic solution, glucose oxidase, to actively capture free radicals created by the imaging process. We could tell immediately when the enzyme failed, because the sample would be entirely bleached just a few slices into the z-stack acquisition. We would use 2mL of imaging media inside the MatTek dish. Since 2mL is >> the imaging media used for coverslips we would create our own media in lab as a cost saving measure. We used 80

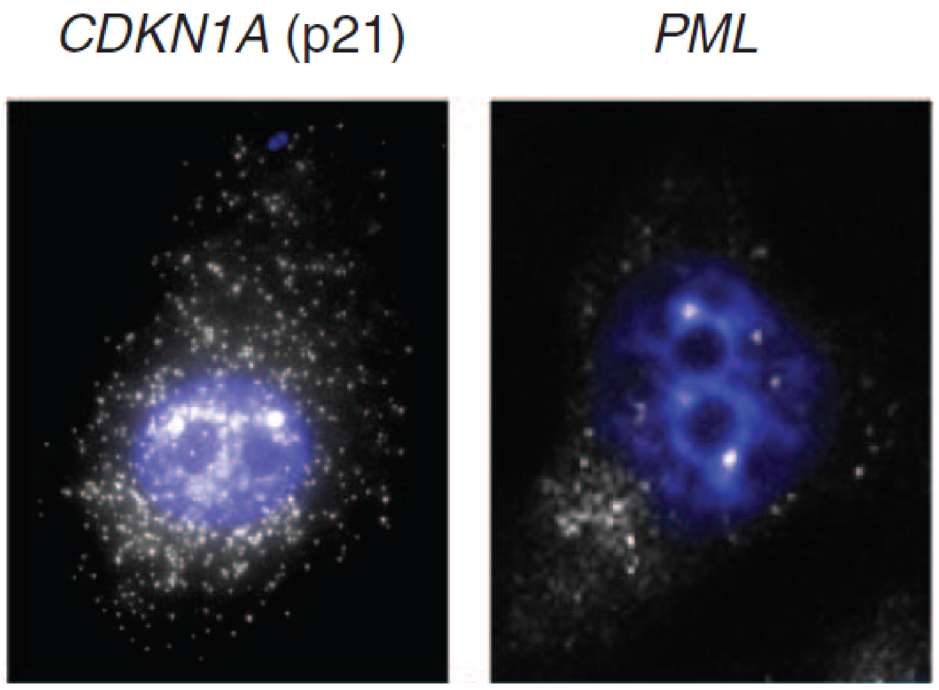


Figure 2.5: p21 and PML mRNA are labeled with smFISH probes. The nucleus is stained with DAPI and is shown in blue. The p21 transcript is more numerous than the PMl transcript.

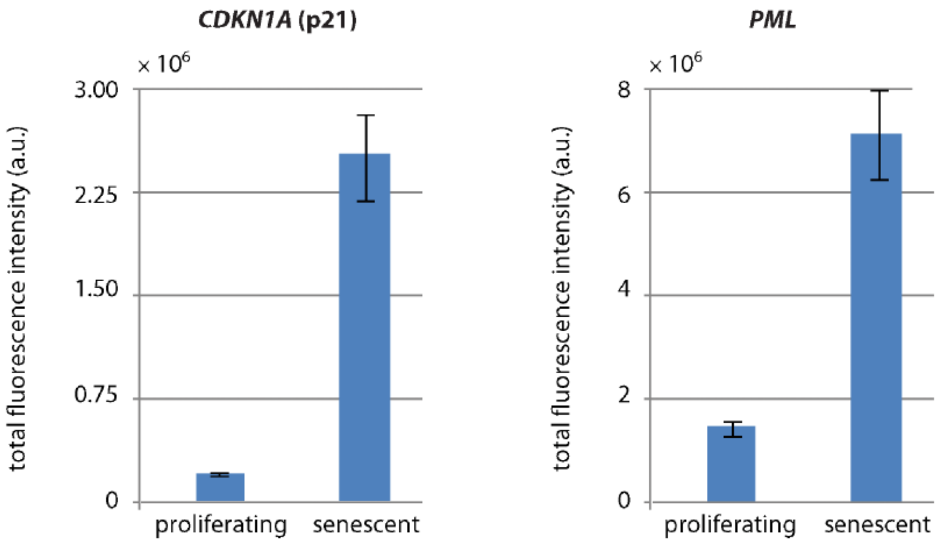


Figure 2.6: The first attempt to quantify single-cell gene expression was to take the mean intensity of the maximum projection of smFISH data. When comparing this signal between untreated and irradiated cells the difference between the two was significant.

When using a new smFISH probe set, use a positive control in an extra sample. We found p21 gives excellent results and was a useful positive control. There are often dozens of copies of p21 within a cell that is still cycling and hundreds to thousands when arrested. Not all probes I attempted to use worked, so having a positive control could be helpful in establishing a protocol.

2.8 Quantification of the smFISH Data for PML and p21

We found that probing mRNA that are present in large quantities can be quantified at lower magnifications by taking the average intensity within the cell (Figure 2.6). This was especially true for the p21 probe. This was important for us as an expedient, because at the time we did not have the means to count the foci in an automated fashion.

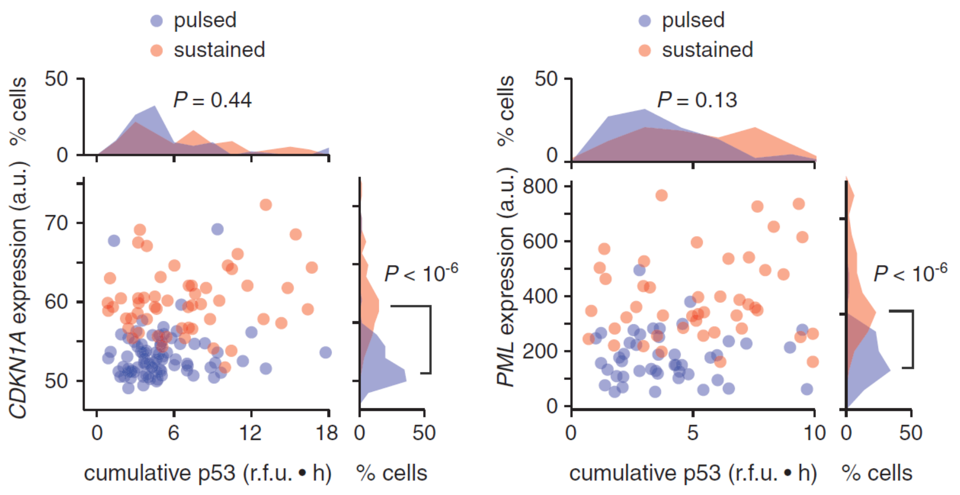


Figure 2.7: Cumulative p53 expression measured from live-cell dynamics was compared to the end point smFISH measurement for p21 and PML. In both instances, the distribution of cumulative p53, seen above the plots, could not be distinguished statistically. On the other hand, the distributions of p21 and PML expression were unique. This suggests that the p53 dynamics, and not just absolute concentration, influences expression.

2.9 p53 Dynamics Control Gene Expression

One challenge common to all studies of protein dynamics is disentangling the effect of changing concentrations from absolute concentrations. Until this study this had never been accomplished experimentally. In order to do so p53 Traces from pulsing and sustained conditions were collapsed into a measure of cumulative expression and compared to the smFISH endpoint measurement. When this dataset is plotted on a two dimensional axes the results were striking (Fig 4).

The simplest model of how dynamics would translate into gene expression would be a linear model, where the cumulative levels of p53 will be proportional to the amount of target gene expression, assuming the degradation rate is fixed. As can be seen in (Figure

2.7), the distributions on top of each plot depict the cumulative measurement of p53 expression and for p21 and PML there is no statistical difference. In contrast, the distribution of gene expression for p21 and PML, seen on the right of each plot, reveals two distinct populations. In both cases, sustained p53 levels lead to more mRNA. This result is inconsistent with the simple linear model, which implies dynamics have a direct influence on gene expression.

2.10 A MATLAB Tool to Quantify smFISH Data

To improve our methods to quantify smFISH data I developed code that would automatically detect foci. I implemented an algorithm from the Danauser lab that was originally developed for imaging actin monomers [52]. In order to implement the algorithm I had to find a 3D hessian matrix of the foci data and there was not built-in MATLAB function for this operation. I accomplished this by sequentially taking the first and then second derivative along each dimension, producing 9 matrices, which when summed together create the Hessian. It was also necessary to determine the size of the point-spread-function based upon the objective used and the physical pixel size of the microscope camera. The resulting script faithfully identified most foci (Fig 5).

2.11 Summary

There are still many questions to be answered about the dynamic behavior of p53. It is still unclear how p53 dynamics regulate the timing and expression levels of its many downstream target genes at a molecular level. There are many layers of p53 regulation

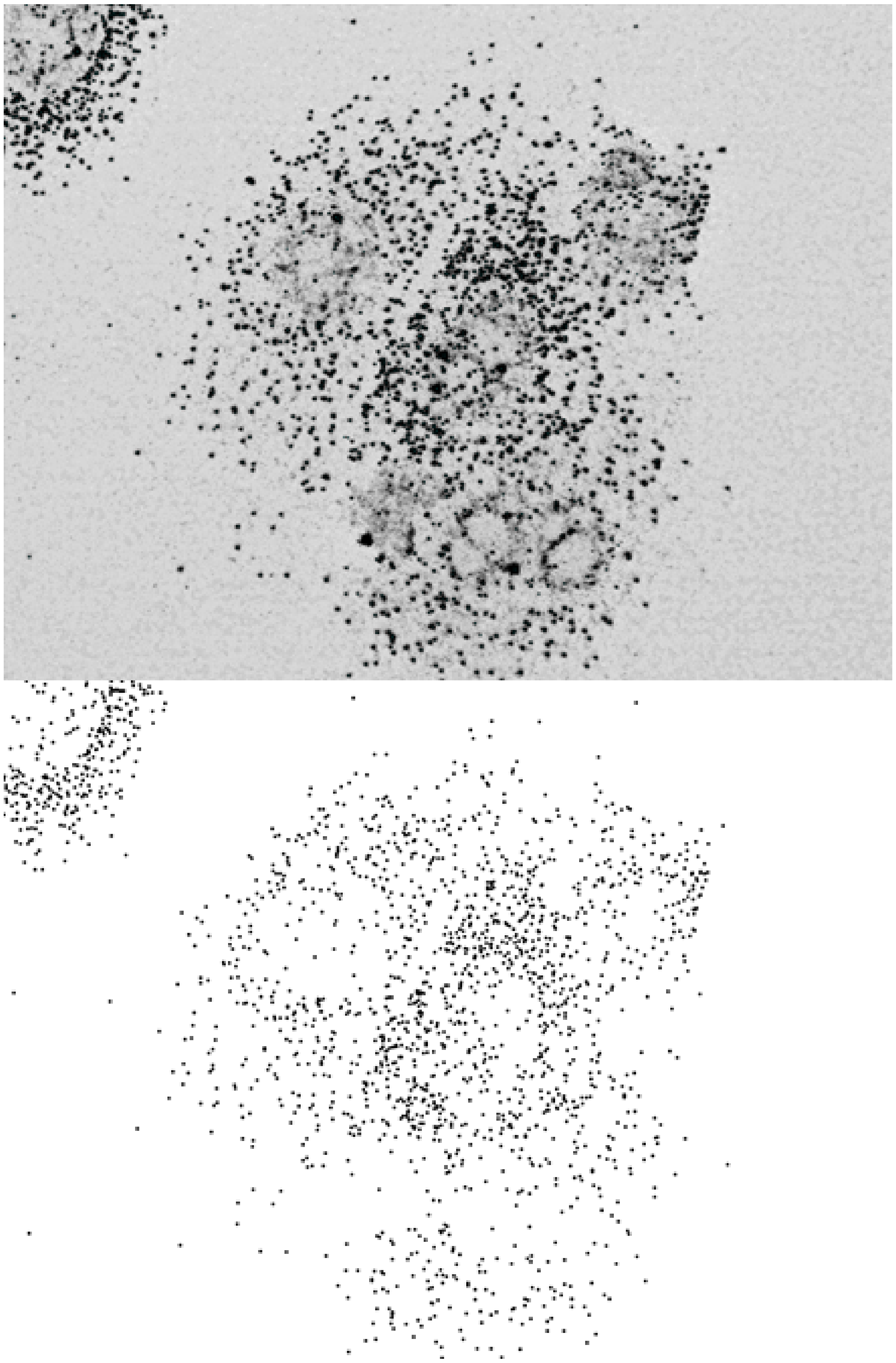


Figure 2.8: p21 FISH probes in an MCF7 cell irradiated with 10Gy. Top image: The raw data from a collection of 30 slices collapsed into a max projection. Bottom image: The mRNA foci identified with a script and then visualized as PSF for demonstration purposes.

that include post-translational modifcations, tetramerization, DNA binding sequences, and transcription co-factor binding. Whether taking a global approach or focusing on the regulation of a single gene, each of these modes of regulation can be explored with respect to p53 dynamics. The downstream target genes that feedback on p53 could also be signficant in translating dynamics since each feedback has the potential to alter p53 signaling behavior. In addtion, recent mouse models demonstrate that p53-dependent cell cycle arrest, apoptosis, and senescence are dispensable for its tumor suppressing functionality. P53 dynamics have primarily been studied in conditions that induce these cell fates, so it would be interesting to study how p53 dynamics exert control over less studied cell fates, e.g. autophagy or ferroptosis.

3 Ionizing Radiation Induced Long-term p53 Dynamics

3.1 Introduction

P53 is activated by the presence of DNA damage and is sensitive to even a handful of breaks [53][54]. One particularly interesting aspect of p53 is its dynamic response to DNA damage(Lev Bar-Or et al. 2000). In response to DSBs p53 levels will pulse with a regular frequency of approximately 5 hours in MCF7 cells [55]. In western blots this

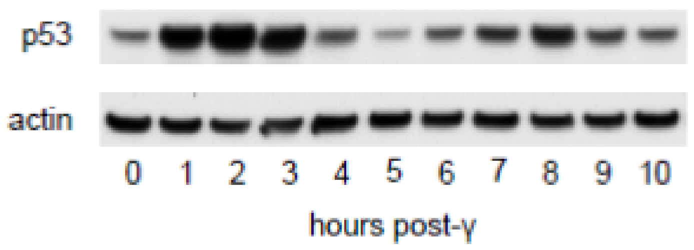


Figure 3.1: p53 dynamics appear as damped oscillations over the first 10 hours following IR exposure.

appears as a damped oscillation as seen in figure 1. There is an initial rise in the p53 stability between 2 and 3 hours after damage and around 8 hours there is another peak of p53 activity. The first 10 hours will be referred to as part of the early response to gamma radiation (Figure 3.1).

This damage is induced by high levels of gamma radiation, 10Gy, which is comparable to the doses delivered by radiosurgery techniques [56]. One of the first genes to be induced by p53 following gamma radiation is the CDK inhibitor p21. P21 is necessary for radiation induced arrest of the cell cycle [57]. P21 activity halts progression through the cell cycle at the G1/S and G2/M transition [58]. P21 has been shown to have a role in preventing chromosomal instability [59], which is often observed in cancer undergoing division while having DSBs can lead to the misseggregated DNA, as some stretches of DNA may not be coupled to a chromosome with a microtubule attached centromere.

3.2 Long-Term p53 Dynamics

The long term response of p53 will refer to the p53 levels in a population on the time scale of days after exposure to gamma radiation (Figure 3.2). In contrast to the short term dynamics the cells have had enough time to repair their damage. It is then a question as whether or not a cell will undergo senescence. In MCF7 cells the primary terminal cell

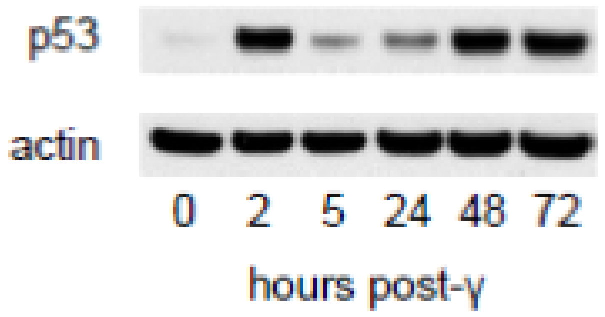


Figure 3.2: Long-term p53 dynamics have elevated levels of p53 that are near peak levels seen at 2 hours. This is at odds with the trajectory of the damped pulses seen at shorter time-scales.

fate in response to gamma radiation is senescence [26]. It is also a question of whether p53 dynamics change over this time frame. At the population scale via Western blot the two hour time point6, representing the peak of the short term response activity is very similar to the long term p53 levels, especially at 48 and 72 time points. This similarity in expression suggests a change in dynamics does occur at some point, because the expectation of the short term dynamics is that the peak p53 levels seen at the population level will decrease as the p53 pulses with individual cells become desynchronized. The 24 hour time point seems to reflect this prediction. However, the 48 and 72 hours are near the 2 hour peak where the stimulating DNA damage has synchronized the DDR (Figure 3.3), suggesting that something unaccounted for is causing p53 levels to rise again.

3.3 There Is a Switch in p53 Dynamics

This unexpected outcome could be explained by several possibilities. The increase in p53 activity could be the result of a population of pulsing cells resynchronizing. However, this seems unlikely as it would imply the 24 and 48 hours were coincidentally taken at similar peak levels. The higher levels of p53 could also be due to an increase in upstream

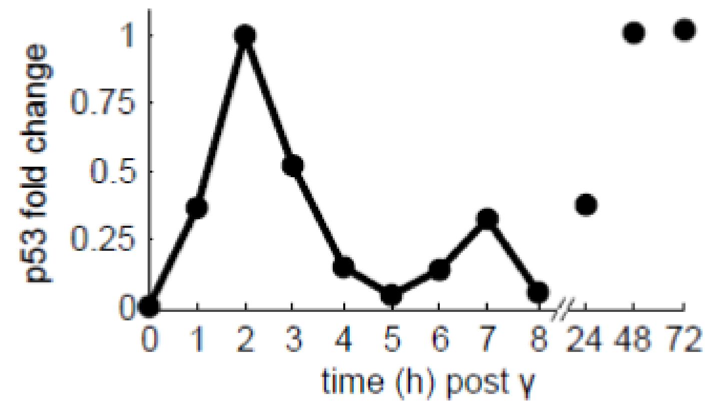


Figure 3.3: Quantification of the Western blot in fig. 2 provides another view of how similar the elevated levels of p53 are the peak 2 hour level.

damage signaling. The initial damage may have pushed the cells into a state where runaway, self-induced DNA damage occurs[60]; or a kinase that is part of upstream the signaling pathway, s.a. ATM or chk2, has its activity sustained even as DNA is repaired. Alternatively, another signaling pathway is activated that stabilizes p53 as part of its signaling, triggered by the DDR, but acting independently thereafter.

The quantification of this Western blot (Figure 3.3) highlights how the long term levels of p53 are significantly higher than the 7 hour peak of the short response. At 10Gy all cells will have a p53 response that persists through the short term.

The short term p53 response is known to be an excitable system [22]. This is evidenced by the 2 hour time point, representing the synchronized first pulse, where across a wide range of doses the amount of p53 is very similar. It has been shown that the first pulse of the p53 response at the single cell level will reach its full amplitude over a wide range of doses and if the DDR signaling through the kinase ATM is abrogated shortly after irradiation [22].

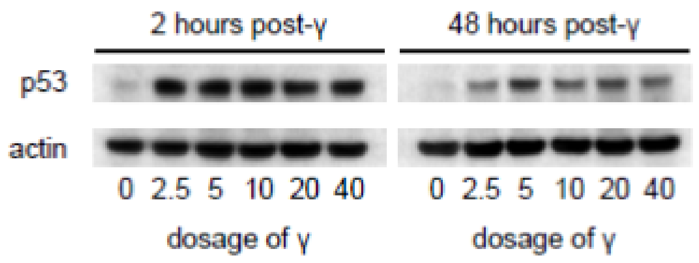


Figure 3.4: The dose response of p53 levels at 2 hours post irradiation demonstrates the excitability of p53. The dose response at 48 hours reveals a similar independence to dose, though the response does appear as strong at 2.5Gy.

3.4 The Does Response Comparison Hints at Positive Feedback

Looking at the dose response of the long term p53 dynamics will be informative of the process leading to its stabilization. Comparing the 48 hour levels to 2 hour levels at the same dose the pattern is very similar with exception to the 2.5Gy dose (Figure 3.4). The dose response is also contradicts the long-term levels of p53 are not as strong as the Western presented earlier.

It has previously been shown that a dose of 10Gy will lead to senescence throughout the population [26]. If the amount of long-term p53 had been more proportional to the dose at the higher doses of 20Gy and 40Gy then perhaps then this would suggest a dependence on the amount of DNA damage a cell incurred. Since the cell fate throughout the population is uniform at doses at or higher than 10Gy the state of senescence may be influencing the p53 behavior. Alternatively, the long-term p53 response could depend upon the state of the cell at the time of damage, when the damage is above the 5Gy threshold. For instance, supposing the cell cycle state at the time of damage is most influential, then, regardless of the specific dose (above a threshold), on average the same fraction of cells will have the same outcome.

The insensitivity to dose might also be explained by a positive feedback. A positive feedback can ensure p53 will reach a new steady state across a wide range of initial conditions. The path to reaching elevated levels may differ, but the ultimate outcome of reaching these elevated levels will always occur.

3.5 MDM2 and p53 are Co-Expressed in the Long-Term

The dynamics of p53 are tightly linked to the ubiquitin ligase mdm2 [61]. Many models that explain p53 pulsing are built around the degradation of p53 being primarily controlled through mdm2 activity [62][49]. The pulsing observed in the short-term dynamics of p53 are echoed by the mdm2 protein (Figure 3.5), pulsing out of phase. As mdm2 levels rise through p53-dependent transcription, p53 levels begin to fall. The system is then reset and further pulses follow, presumably from the persistence of DNA damage that has not been repaired. In contrast to these dynamics, the long-term behavior of mdm2 does not appear to be out of phase with p53 (Figure 3.5). Although it is not as strongly induced, mdm2 levels are elevated relative to their baseline behavior, yet despite this p53 levels remain elevated. This suggests that the relationship between mdm2 and p53 has been altered. The presence of p53 could still be driving mdm2 transcription to levels higher than baseline, but the degradation of p53 through mdm2 has been weakened.

Alternatively, the population of cells could have been divided into two groups: one expressing p53 and another expressing mdm2. The stability of p53 is regulated by upstream signaling by the DDR and through feedback through p53-dependent transcription, especially the transcription of mdm2. In the long-term dynamics, additional components of p53 signaling could be newly translated, even indirectly through p53 as part of a cascade of transcription factors, and alter p53 dynamics. Since we observe elevated levels of p53 in the presence of elevated mdm2 this new component

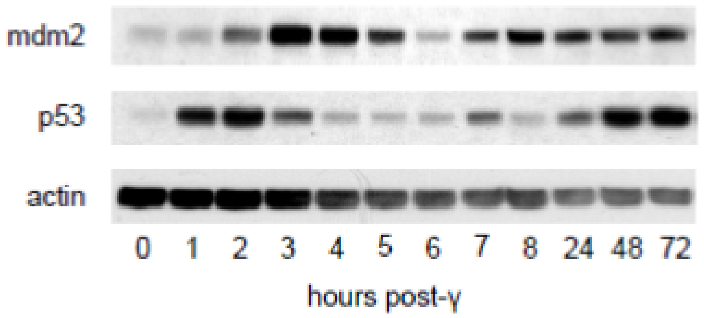


Figure 3.5: MDM2 dynamics lag p53 in the short-term response, yet, similar to p53, it is evelated in the long-term (24, 48, and 72 hours).

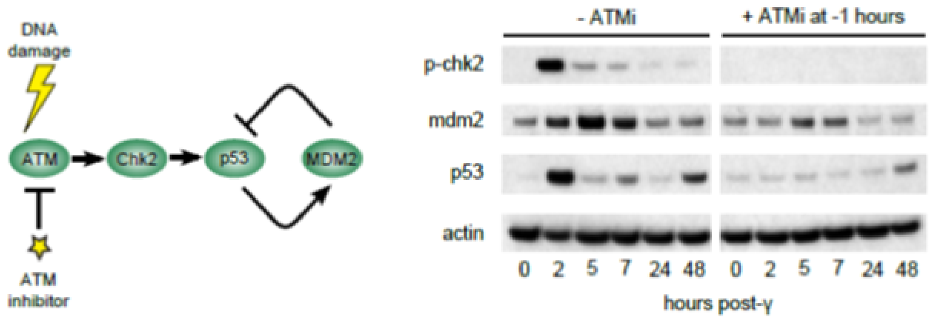


Figure 3.6: ATM kinase activity is stimulated by ionizing radiation. ATM inhibitor (ATMi) negates ATM kinase activity, which is reflected by the phosphorylation of chk2. When the upstream signaling from ATM is abrogated the short-term and long-term response of p53 disappears.

could be part of a positive feedback that appears in the days following irradiation.

3.6 Upstream Signaling Cannot Account of Elevated p53 Levels

Before exploring potential feedbacks, it is important to consider the upstream signaling in further depth. The DDR signaling in response to gamma radiation is primarily channeled through the ATM kinase [63]. A small molecule inhibitor of ATM (ATMi) abrogates the kinase activity [64]. A proxy for kinase activity is the kinase chk2, a direct target of ATM (Figure 3.6).

If cells are treated with ATMi before damage p53 signaling is abolished in the short-term response (Figure 3.6). A rise in p53 activity does appear at the 48 hour time point, but it is a relatively weak response and could be caused events unrelated to the irradiation. For instance, a p53 response has been observed in cells when that are fully confluent in culture dishes. Damage to cells beyond genotoxic stress may be incurred through impaired arrest signaling [65]. The short-term p53 response is necessary for the long-term p53 response. This result favors the possibility of a positive feedback loop whose existence is dependent on p53 transcription in the short-term.

Evidence of a positive feedback loop is further supported by the long-term p53 response when ATMi is added 16 hours after irradiation. Here it is shown that p53 remains elevated at the 48 and 72 hour time points independently of signaling from ATM. It is noteworthy that the levels between the two conditions are very similar, suggesting the cause of the elevated p53 is unperturbed. The idea of a positive feedback fits into the temporal design of this experiment, because the 16 hour window of time before ATMi is added provides a wide window of time for p53-target genes to become transcribed and translated.

3.7 p53-Target Genes PML and PIDD Are Potential Positive Feedbacks

P53 is a very well characterized transcription factor and the literature on p53 has numerous studies that focus on p53 and one its transcriptional targets [34][35]. Two proteins in particular, P53-induced death domain (PIDD) [66] and PML [67](Figure 3.7), have been well characterized and shown to be positive feedbacks, yet these studies did not consider the impact these positive feedbacks might have on the dynamics of p53. Furthermore, these positive feedbacks deserve further investigation, because they are

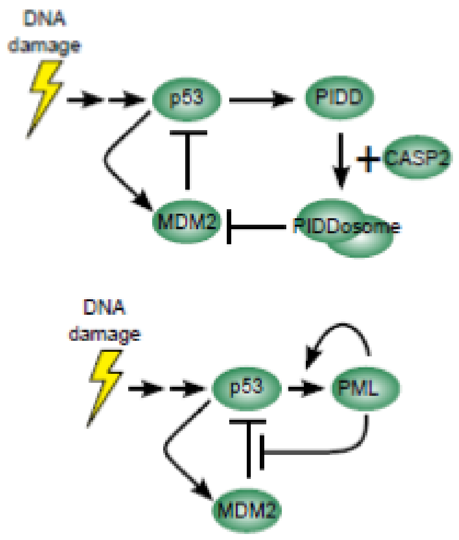


Figure 3.7: PIDD and PML have been identified in the literature to stabilize p53 by interfering with MDM2.

usually described as a necessary component for p53-dependent apoptosis or senescence where cell-fate is the measurement of function.

The PIDD gene was so named, because it was discovered to be transcriptionally induced by p53 in an erythroleukemia cell line with a temperature sensitive p53-mutant [66]. and the PIDD promoter [66]. Sequence analysis of PIDD mRNA revealed a p53 consensus binding site in the 5' UTR and two protein domains: a domain of leucine-rich repeats and a death domain [66]. These domains are associated with protein-protein interactions and were found to be essential to the formation of the PIDDosome protein complex [68]. The PIDDosome is a ring-shaped complex of five PIDD and seven RAIDD proteins [69]. The RAIDD protein has a caspase recruitment domain (CARD) that recruits and activates caspase2 [68], which was discovered long after canonical caspase activation that leads to apoptosis.

Caspase2 is a highly conserved protein whose function is not strongly associated with apoptosis [70]. Structurally it is most similar to caspase8. Caspase2 has been shown to cleave mdm2, and the fragment that is still recognized by common mdm2 antibodies is known as p60 [71]. The cleavage of mdm2 separates the p53-binding domain from the RING ubiquitin-ligase domain, which promotes p53 stability in two ways. First, this prevents mdm2 from tagging p53 for degradation, and, second, the fragment with the p53 binding domain acts as a competitive inhibitor of uncleaved mdm2.

PML forms sub-nuclear structures known as PML bodies [72]. PML bodies are associated with regions of DNA that are being transcribed. Proteins are recruited to PML bodies through the post-translational modification sumoylation. P53 is sumoylated by PML where it may be recruited to PML bodies surrounding p53-target genes [73][74]. PML has also been shown to sumoylate MDM2, but instead of being recruited to DNA it is instead recruited to the nucleolus. In either situation, MDM2 and p53 are sequestered away from each other, which leads to stabilization of p53.

Measuring the expression of PIDD and PML mRNA with qPCR shows that the expression is significantly elevated at the 24 and 48 hour timepoints post-irradiation (Fig 9). This time frame coincides with the elevated p53 levels observed in Western blots. Both genes are expressed during the time frame a positive feedback is expected to occur, so the test the existence of a positive feedback directly siRNA for each gene was applied to MCF7 cells. The siRNA effectively knocked down each gene (Figure 3.8).

The suppression of a positive feedback should lead to less p53 when observed by Western blot. In the case of PML, the long-term dynamics of p53 are relatively unchanged (Figure 3.9). This strongly indicates that PML is not primarily responsible for the elevated levels of p53.

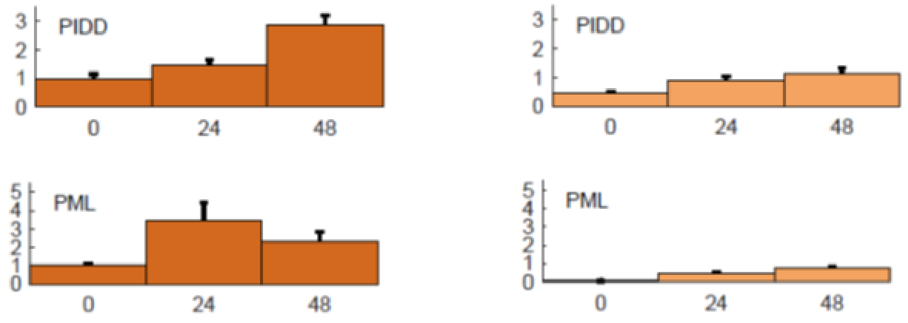


Figure 3.8: The expression profile of both PIDD and PML show increased expression in the long-term time frame of 24 and 48 hours (left panel). SiRNA against PIDD and PML suppress mRNA levels in response to irradiation (right panel).

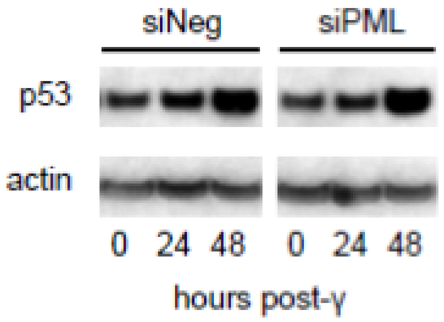


Figure 3.9: Long-term p53 levels are not affected by PML knockdown.

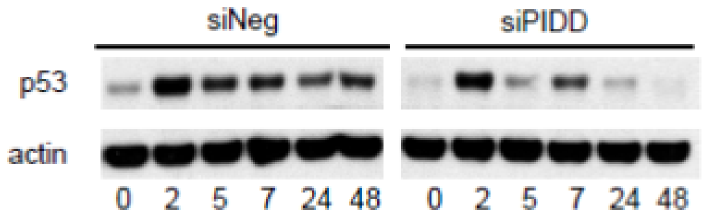


Figure 3.10: The knockdown of PIDD reduces p53 levels at 24 and 48 hours. Notably, the short-term dynamics of p53 are unaffected.

3.8 PIDD Knockdown Destabilizes p53

In contrast with the p53 dynamics with PML knockdown, the long-term levels of p53 are greatly diminished when PIDD is knocked down with siRNA (Figure 3.10). Interestingly, the short-term dynamics of p53 are very similar to each other in both conditions. This implies that PIDD is not required for the initial p53 response following irradiation, which is consistent with the low basal expression of PIDD observed at the time of irradiation. The increased PIDD expression at 48 hours correlates with long-term elevation of p53 and without PIDD p53 is reduced to near basal levels. This result suggests that PIDD may function as a positive feedback that stabilizes p53.

An alternative to using siRNA to disrupt the putative positive feedback between PIDD and p53 is the use of a small molecule inhibitor of caspase2 [75]; caspase2 is activated by PIDD and cleaves mdm2. Caspase2 inhibitor is added to cells 24 hours after radiation in order not to disrupt the short-term response of p53. Similar to the knockdown of PIDD, the addition of casapse2 inhibitor leads to lower long-term levels of p53 (Figure 3.11), providing further evidence that PIDD is part of a positive feedback loop. The effectiveness of the inhibitor can be seen in both p53 and mdm2 bands. In an inhibitor dose dependent manner the amount of mdm2 cleavage product, p60, decreases at the higher concentrations. As the presence of p60 decreases, so does the levels of p53.

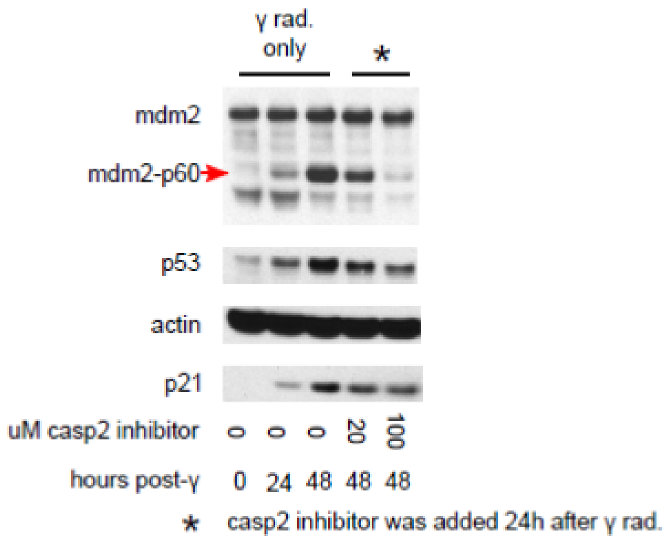


Figure 3.11: Caspase2 inhibitor reduces p53 levels at 48 hours. MDM2-p60, cleaved by activated caspase2, is reduced in a dose dependent manner.

This second method of perturbing the connection between PIDD and p53 further supports the idea of a positive feedback loop.

3.9 Induction of PIDD Increases p53 Levels

It has been shown that disrupting PIDD has led to lower levels of long-term p53. If p53 and PIDD are part of a positive feedback loop then higher levels of PIDD will lead to higher levels of p53. A tet-on inducible Flag-tagged PIDD construct was added to cells to verify this relationship. The inducible PIDD is Flag-tagged, because PIDD antibodies are not reliable. As a negative control GFP protein is induced instead on PIDD. Doxycycline is added 24 after irradiation to activate the inducible promoter. Following irradiation, the long-term levels of p53 are higher with induced PIDD suggesting enhanced stabilization of p53. In the presence of induced GFP there is not a noticeable change in long-term p53 levels (Fig 13). Furthermore, induced PIDD also leads to a greater amount of p60, which

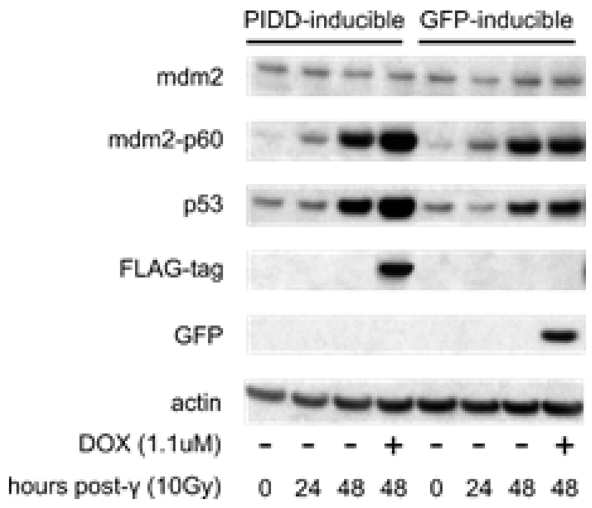

PIDD-inducible
GFP-inducible
mdm2
mdm2-p60
p53
FLAG-tag
GFP
actin
DOX (1.1uM) - - - + - - - +
hours post-γ (10Gy) 0 24 48 48 0 24 48 48

Figure 3.12: A tet-on inducible PIDD system was added to MCF7. Induction of PIDD leads to comparatively higher levels of p53 levels at 48 hours. The negative control of inducible GFP did not affect p53 levels

is consistent with increased caspase2 activity (Figure 3.12).

3.10 Long Term p53 Dynamics at Single-Cell Resolution

Until now, the long-term p53 dynamics and the role of PIDD have been explored using measurements, such as Western blots and qPCR, which represent averages of protein and mRNA within a population of cells. It is important to investigate the long-term p53 dynamics at the single cell level as well, because population measurements can mask underlying heterogeneity [48]. Immunofluorescence (IF) measurements of p53 protein levels reveal that there is heterogeneity in the expression of p53 at 72 hours after irradiation that is not apparent at basal conditions, or at 2 hours after irradiation during the initial pulse of p53 (Figure 3.13).

In addition to the heterogeneity seen in p53, the nuclei of the cells have transformed over the 72 hours following irradiation into a diverse range of sizes (Figure 3.13). At 72 hours some cells appear to have become bi-nucleated cells and other cells contain

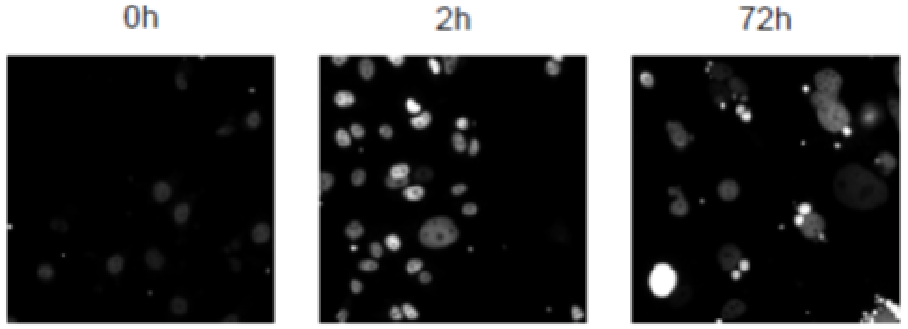


Figure 3.13: Immunofluorescence of irradiated MCF7 cells were treated with anti-p53 antibody. Compared to 0 and 2 hours, the 72 hour timepoint reveals heterogeneity in both p53 signal and nuclear morphology.

micronuclei (MN) [76]. The micronuclei correlate with higher concentrations of p53 seen as bright dots that surround the nuclei.

The diversity in morphology and the appearance of micronuclei complicate the definition of p53 dynamics. During the short-term p53 response the cells are arrested, so there is no division and the nuclei remain whole. Since p53 is primarily confined to the nucleus the p53 dynamics in single cells can be defined as the change in p53 concentration within the nucleus over time. The mean fluorescence intensity of the p53 reporter is proportional to this value. This definition of p53 dynamics in single cells is compatible with the Western blot measurement of p53 dynamics, because when a fixed amount of protein is added to each lane this has an averaging effect; when cells are synchronized the Western blot would represent the average cell.

When a cell is multinucleated or has MN and these different nuclear compartments contain varying levels of p53 it is trickier to define p53 dynamics, because it is unclear how functional p53 is within each nuclear compartment or how differential p53 dynamics affect cellular function or cell fate decision making. This complication can be represented as several ways p53 dynamics can be defined in a time-lapse movie: the average intensity

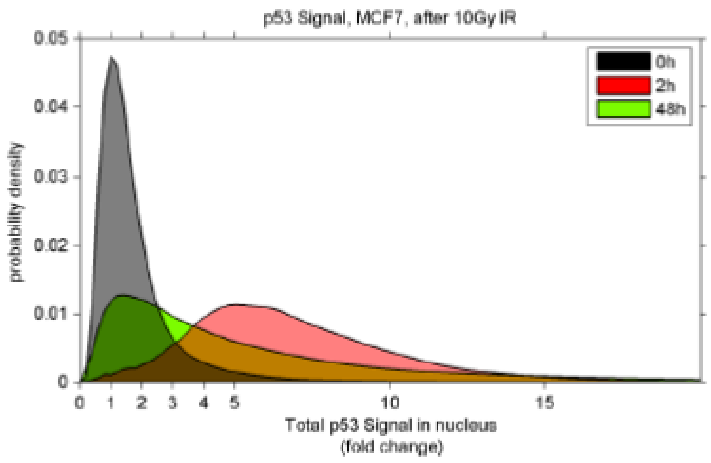


Figure 3.14: Histograms of p53 from immunofluorescence. The 0 and 2 hour time points show the induction of p53 in the whole population. The 48 hour distribution is assymetric and has a long tail that caused by the underlying heterogenity.

of p53 signal across all nuclear compartments, the brightest average intensity of p53 signal in any nuclear compartment at a given time, the p53 signal found in the largest nuclear compartment, or the p53 signal can be tracked for all nuclear compartments and treated independently. From a practical perspective it is easiest to measure the average p53 intensity within the largest nuclear compartment, so an intact nucleus or a nucleus surrounded by micronuclei would be treated the same and micronuclei are ignored.

Immunofluorescence of MCF7 cells following irradiation with 10Gy reveals a distribution of p53 levels with a long tail at 48 hours (Figure 3.14). Relative to the basal distribution of p53, the 2 hour post-irradiation distribution shows the entire population has activated p53 and has an average intensity that is several fold higher than basal conditions. The 48 hour distribution shows that for a large fraction of the population the p53 levels are near basal levels or between 2 hour levels that represent the peak of the first pulse of p53.

The heterogeneity that was observed in the immunofluorescence images translates to the histograms. The 0 and 2 hour distributions are more symmetrical, reflecting the low basal levels of p53 and synchronized first pulse, respectively. The diversity in p53 signal intensity seen at 48 hours is evident in the asymmetric and broad distribution of p53. In contrast to the Western blot data of short-term p53 dynamics, the long-term p53 dynamics will be an average that includes the p53 heterogeneity observed in the nuclei and micronuclei, which adds an additional perspective to the interpretation of these results.

3.11 A Subpopulation of Cells Show p53 Dynamics with Elevated Levels

Interestingly, when single cells are tracked over a 72 hour period using time-lapse microscopy three different long-term behaviors are observed (Figure 3.15). They will be referred to as: return-to-basal, persistent pulsing, and sustained. Return-to-basal dynamics show pulsing immediately after irradiation for approximately 24 hours, throughout the window of time that encompasses short-term dynamics. Persistent pulsing dynamics are those that show p53 oscillations throughout the period of observation, 72 hours. Sustained dynamics are periods of elevated p53 levels that remain elevated for times much longer than the duration of a typical pulse.

A heat map of p53 dynamics from 200 cells (Figure 3.16) summarizes the diversity observed across the population. The dynamics were classified by the consensus of a few lab members using the traces above as a guide to classification. The majority of cells were persistently pulsing and the smallest fraction of cells exhibited sustained p53 dynamics. This result is surprising when compared with the long-term dynamics observed in Western blot. In the Western blot the elevated levels at 48 hours suggested a change in p53 behavior had occurred. However, when the long-term dynamics are observed at the

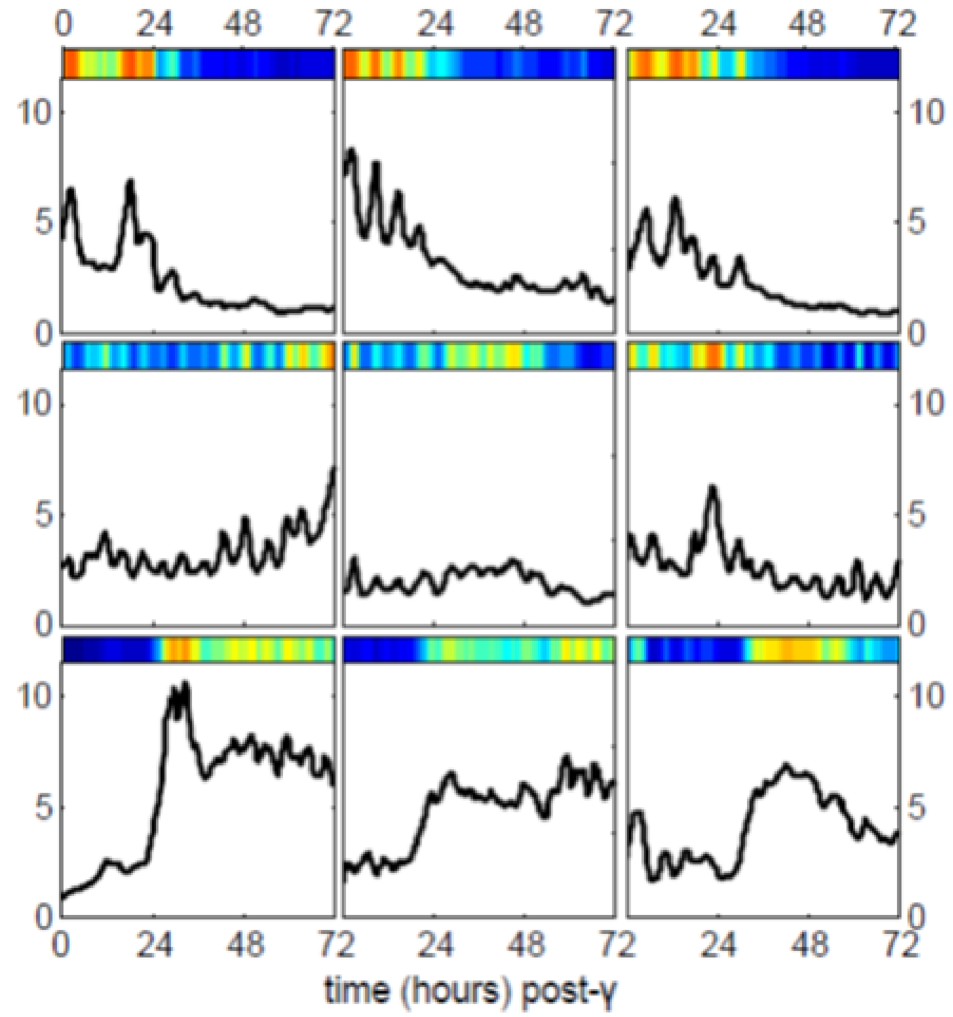


Figure 3.15: Examples of p53 dynamics from individual cells. Three patterns of dynamics were observed. Row 1: return-to-basal. Row 2: persistent pulsing. Row 3: sustained.

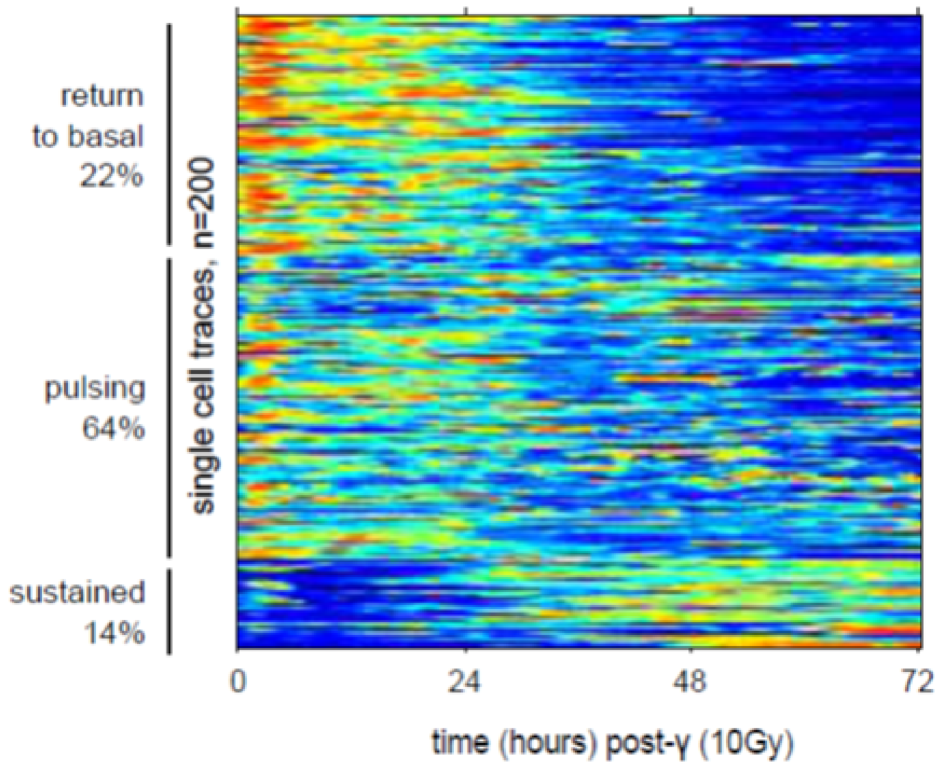


Figure 3.16: A heat map of p53 dynamics from 200 cells shows 3 subpopulations of cells that have different patterns of p53 dynamics. Each trace has been self-normalized.

single cell level the majority are have p53 pulses that are the same as the p53 pulses observed in the short-term. This suggests that the observations made at the population level could be largely influenced by a sub-population of cells that exhibit sustained dynamics. The heterogeneity discovered at the single-cell level shows population measurements mask single-cell behaviors. Our results show a clonal population of cells exposed to a uniform amount of ionizing radiation gives rise to several different p53 dynamics.

3.12 Summary

PIDD has been shown to affect p53 levels in a manner consistent with a positive feedback loop. The repression of PIDD activity, either through siRNA or a small molecule inhibitor, led to lower levels of long-term p53. Additionally, an increase in PIDD activity through a tet-on system led to higher levels of long-term p53. Interestingly, PIDD expression increases over the long-term time scale and does not appear to be required for the short-term p53 response. The behavior of PIDD is evidence that p53 dynamics in the short-term and long-term dynamics are the result of a signaling network that is changing, yet it is not clear if the long-term p53 dynamics have any functional consequence.

P53 is known to influence entry into senescence through the DDR and transcription of p21 [77]. Intriguingly, 10Gy is known to drive the entire population into senescence, yet there are several patterns of p53 dynamics. There are a couple possibilities that could explain why all roads lead to senescence. One possibility is that the decision to enter senescence is made during the short-term response, when p53 dynamics across the population are the most similar. This would imply that the long-term p53 dynamics are not necessary for commitment to senescence. The return-to-basal dynamics supports this idea, because p53 is absent even though these cells are becoming senescent. Another possibility is that the long-term dynamics do matter, but in a contextual manner. This means that there is some variable aspect of a cell that differs across the population, which induces a specific type of p53 behavior as a step towards senescence. This variable could be the state of the cell cycle at the time of damage, or the some complication from DNA damage repair occurs at a low rate and affects only a subset of cells.

Earlier is was shown that the PIDD protein was shown to be part of a positive feedback loop in long-term p53 dynamics, yet the role PIDD remains unclear at the single cell level. The proportion of cells that exhibit sustained dynamics most closely resemble the dynamics observed in Western blot and it suggests that PIDD might be prominently involved in this subpopulation. p53 dynamics should be measured at the single cell level with the knockdown of PIDD and induction of PIDD to explore the connection between PIDD and the p53 stability seen sustained p53 dynamics.

4

Semi-Automated Tracking of p53 Dynamics

4.1 Introduction

There are many technical challenges to quantifying protein dynamics from single-cell fluorescent time-lapse microscopy [78]. One of the challenges facing the long-term observation of p53 dynamics in cancer cells responding to gamma radiation is cell tracking. In the end a custom tracking tool was created to handle the unique image data collected while observing long-term p53 dynamics. The motivations behind this tool are given the perspective when considering the popular open-source software Cell Profiler

[79].

Cell Profiler is developed and maintained by the Carpenter Lab at The Broad Institute. It is widely used and increasing in popularity. Cell Profiler is particularly adept at quantifying images of fixed cells, but has limitations to tracking cells through time. In particular, the algorithm is challenged by the quality of the images and the number of images that make up the movie.

The movies of long-term p53 dynamics are long enough that the performance of the tracking algorithm would fail in two ways. First, the length of the movie was such that many times two cells would pass closely to each other and cause a segmentation error where two cells would be counted as one. When this happened one of the growing tracks would switch cells after the two nuclei could be resolved again, which was unwanted. Second, the software was confused by the act of cellular division. A dividing cell leads to one more additional cell to track and it was ambiguous how these tracks were linked after division. Additionally, the image analysis was complicated by failed mitosis. A failed mitosis or mitotic catastrophe [80] can lead to the formation of micronuclei or a multi-nucleated cell. It is then unclear which nuclear body to track and what constitutes p53 dynamics. The Cell Profiler tracking software was not made to resolve these conflicts.

4.2 The Jaqaman Algorithm

The underlying algorithm used for tracking in Cell Profiler is widely used in other tracking software that were custom made for a given project [81][82][83]. The source algorithm [81] was designed to be flexible. The optimization that leads to a tracking solution depends upon a cost function, which could be altered or expanded to accommodate the unique features of the objects being tracked. Originally, the objects being tracked with sub-diffraction limit receptors. The tracking of cells has additional information that can be incorporated into the cost function such as nuclear area and solidity.

Furthermore, if the cells were imaged at a high enough rate the movement of individual cells was persistent from frame to frame. This information could be incorporated into the cost equation through the use of a Kalman filter. A linear model of movement was found to be satisfactory in predicting cell motion. The Kalman filter can also be applied to the p53 dynamics to aid in tracking in addition to the fluorescent intensity of p53. At any given time there is a wide distribution of p53 intensities, so neighboring cells can be distinguished by their variable intensity. If the imaging frequency is high enough that changes in p53 intensity are relatively slow, then a linear model of p53 dynamics can sufficiently predict the p53 levels from frame to frame. The cost function of the Jaqaman algorithm was expanded to incorporate this information.

In spite of these additional efforts to aid in the tracking errors were still common. To improve the results felt like it would require innovation to the Jaqaman algorithm or a new algorithm altogether. This prospect was beyond my capabilities, so we created a software that would allow for the manual curation of tracking fragments. The cost function was tuned to be sensitive to differences of the nuclei from frame to frame. If a large difference occurred then a track would be terminated and a new track would start. We found that certain events would trigger this termination. For example, when two cells were segmented as one the difference in area would cause both tracks to terminate. When a cell divided the difference in cell shape again caused a termination. This termination included both a proper mitosis and a failed mitosis. The software we made would allow the annotation and editing of these tracks. A division or mitotic catastrophe could be annotated and lineages of division could be created by identifying the mother-daughter relationship between tracks that start and end in division.

4.3 The Image Viewer

Perhaps the most essential component to tracking cells is a window to display the images captured by the microscope and visualize the information that connects cells between frames. The challenges to make an effective viewer include navigating the structure of the original data, creating visuals that communicate tracking, and collecting user feedback to validate and modify the tracks generated algorithmically. The viewer is the primary window of a graphical user interface and will draw the focus of the user for most time.

The simplest instance of the viewer will display the images with no tracking information (Figure 4.1). This can be useful when exploring the data to "get a feel" for the response of a population or to compare the images captured at different positions or times. At first glance this window is very minimal and bare-bones. This approach was inspired by the open source software ImageJ [84] and necessitated by expediency. One of the mantras guiding the creation of the tracking tool was, "don't reinvent the wheel." This meant finding the compromise between functionality and the time required to add a new feature. For the most part during development this meant forgoing adding features that were not directly related to tracking, because the goal was not to recode ImageJ in MATLAB. However, there are basic functionalities that could not be avoided.

4.4 Accessing large data sets using thumbnails

A novel software feature is the quick access to a large collection of images. In contrast to ImageJ, a collection of image files does not need to be imported as a stack to view them sequentially through time. For a single stack of images the time savings is negligible, but the scale of experiments within the Lahav Lab has been increasing in both the number of positions and number of timepoints collected. For example, a movie that consists of 4

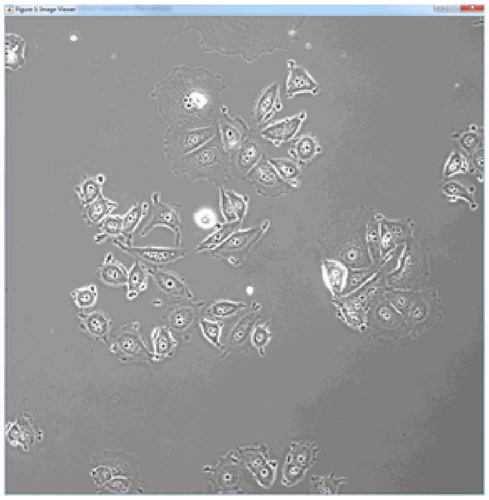


Figure 4.1: The imageviewer displays cells and responds to user-inputs.

channels, 75 positions, and 289 timepoints generates 169GB of TIFF image data. Using ImageJ to view the data would mean importing up to 300 stacks across time, referring to the example data set. Using ImageJ 1.47v with a circa 2010 computer with 3 cores at 2.8GHz and 5400rpm hard drive, the importing an image sequence into a stack requires approximately 45 seconds. This time can be influenced by network speed instead of hard drive speed if the data is stored in the cloud or network drive. To view all the data through ImageJ one must commit to nearly 4 hours of idle, waiting time.

The solution to avoiding this wasted time is to create thumbnails of the captured images (Figure 4.2). The creation of thumbnails does require time for processing, but this time is a single hands-free chunk when the thumbnail script is run, and can be run overnight. The thumbnail images are smaller in size by 2 orders of magnitude by reducing the number of pixels 4 fold, reducing the image depth to 8-bits from 16, and using the PNG lossless image compression format. The 169GB dataset can be converted to 2GB worth of thumbnails. This smaller file size makes it possible to load an image from the hard disk as needed without experiencing a delay that would interfere with

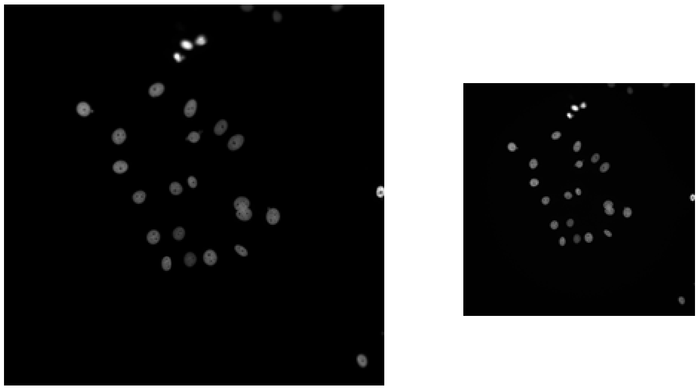


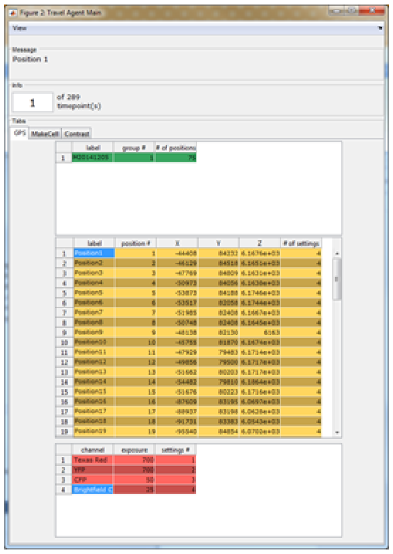
Figure 4.2: The smaller thumbnail greatly reduces the time it takes to read an image from the disk and display it on screen.

semi-automated tracking.

4.5 Navigating large datasets using group, position, and settings labels

The acquisition of images on a microscope can be categorized into a hierarchy that organizes the data by the description of its contents. This hierarchy consists of 3 layers: group, position, and settings. This hierarchy will be referred to as the GPS hereafter. The settings, the first layer, is specified by several imaging parameters, including channel, exposure, z-height, and binning. The position, the second layer, is determined by the (X,Y) location of the stage over the objective. The group, the third layer, organizes the positions. For example, a group of positions can identify each well in a multi-well plate, or different regions within the same plate.

The GPS can be represented graphically by 3 tables. Each row in the group table corresponds to a position table, and each row in the position table corresponds to a settings table. Navigating the set of images collected in time-lapse microscopy



Position 1

1 of 289 timepoint(s)

	label	group #	# of positions
1	M20141209	1	78

	label	position #	X	Y	Z	# of settings
1	Position1	1	-44408	84232	6.1676e+03	4
2	Position2	2	-46129	84518	6.1651e+03	4
3	Position3	3	-47769	84809	6.1631e+03	4
4	Position4	4	-50973	84056	6.1638e+03	4
5	Position5	5	-53873	84188	6.1746e+03	4
6	Position6	6	-53517	82058	6.1744e+03	4
7	Position7	7	-51985	82408	6.1667e+03	4
8	Position8	8	-50748	82408	6.1645e+03	4
9	Position9	9	-48138	82130	6163	4
10	Position10	10	-45755	81870	6.1674e+03	4
11	Position11	11	-47929	79463	6.1714e+03	4
12	Position12	12	-49856	79500	6.1717e+03	4
13	Position13	13	-51662	80203	6.1717e+03	4
14	Position14	14	-54482	79810	6.1864e+03	4
15	Position15	15	-51676	80223	6.1716e+03	4
16	Position16	16	-87609	83195	6.0697e+03	4
17	Position17	17	-89937	83198	6.0628e+03	4
18	Position18	18	-91731	83383	6.0543e+03	4
19	Position19	19	-95540	84854	6.0702e+03	4

	channel	exposure	settings #
1	Texas Red	700	1
2	YFP	700	2
3	CFP	50	3
4	Brightfield C	25	4

Figure 4.3: The GPS gui has a table for each level of the GPS hierarchy.

experiment is greatly simplified by specifying 3 rows in the gui representation of the GPS (Figure 4.3).

4.6 Adjusting the contrast

The default representation of an image on a computer screen often assumes that the range of intensities in an image span the bit-depth of an image. For example, if an image is stored in a 16-bit format, then the lowest value is 0 and the highest value is 65,535. Often, the data collected on a microscope does not span the full range of intensities. This is primarily a result of the strength, or lack thereof, of a fluorophore. A weakly fluorescent protein, compared to a bright Alexa dye, will produce relatively few photons for a given exposure length, which is proportional to the intensity value found in the pixel that corresponds to the location of that protein. The exposure length can be increased to generate a brighter signal, but there is an upper-limit based on photo-toxicity. Another

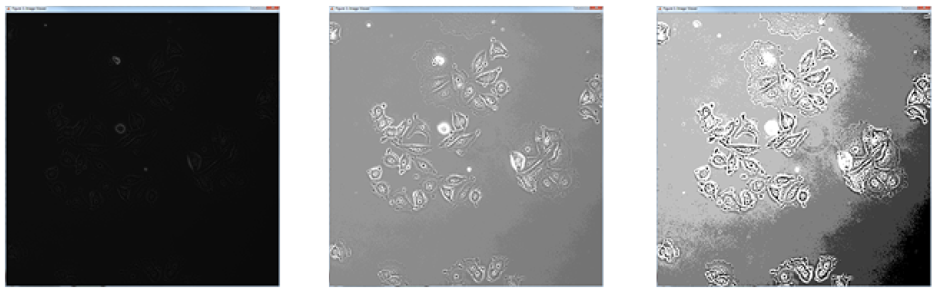


Figure 4.4: Inadequete constrast correction can obscure the information in an image.This series of images show a phase contrast image of MCF7 cells. (Left) The image with contrast spanning the bit-depth of the image, i.e. no correction. (Middle) The image with moderate contrast correction. (Right) The image with heavy correction.

cause of a low maximum-signal-intensity is the dynamic nature of a tagged protein, implying the intensity will vary through time, and not every timepoint will contain high maximum-signal-intensity. In summary, many images consist entirely of intensities much lower than the maximum value of an image format determined by the bit-depth.

When displaying a low intensity image on a computer screen it can initially appear empty or entirely black and the information contained within the image cannot be seen. The range of color that can be displayed by the computer screen does not map effectively to range of intensities in the image. This can be fixed with contrast correction, which changes the lower- and upper-bound of the image range that the computer screen maps into (Figure 4.4). This issue, though prosaic, prevents the proper viewing of an image and, therefore, a gui to change the contrast was added to the tracking software (Figure 4.5).

4.7 Navigating through time

When using stacks to view time-lapse data in ImageJ, it is common to represent time as the variable that is attached to the scroll bar. In other words, the image on display

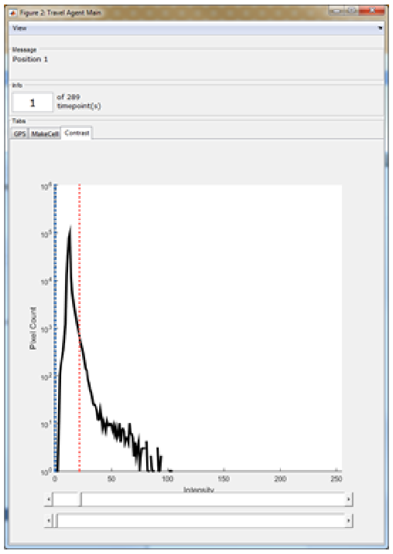
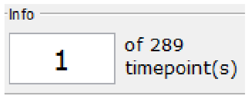


Figure 4.5: The contrast correction gui. The graph shows a histogram of intensities within the image. The red and blue vertical lines represent the lower- and upper-bound of the intensity range being mapped to the computer screen.



Info

1 of 289 timepoint(s)

Figure 4.6: The timepoint text box. Reports on the current timepoint and can be used to jump to other timepoints.

represents a single moment in time and scrolling through the stack will replace this image with another from a different moment in time. In the tracking software time is navigated using the keyboard, in lieu of a scroll bar (ImageJ has this as a redundant feature), because the mouse is typically being used to identify cells within an image. Using the 'period' or 'comma' keys can be used to move forward or backward through time. There is also a text box that can be used to directly enter a timepoint that will then cause the corresponding image to be displayed (Figure 4.6).

4.8 Omitted features

There are convenient features that could be added to the current configuration of the tracking software sometime in the future, but are not essential for the basic function of the tracking software: being able to see the data and navigate a large dataset. The ability to zoom or magnify a region of an image would be helpful when looking at tightly packed cells. Adding pseudo-color to multiple channels and the ability to overlay multiple channels would be helpful for comparing the dynamics of multiple channels within the same cell. Finally, generating a statistics based summary of the dataset derived from intensity histograms could provide an approximation to changes in cell number and changes in dynamic behavior without any single-cell tracking.

4.9 Generating Tracks

Tracking of cells is done using the linear assignment algorithm created by Jaqaman et al.[81]. The original algorithm was used to track sub-diffraction-limit CD36 receptors in macrophages and clathrin-coated pits in BSC1 cells. These objects differ from nuclei in several ways. Nuclei are relatively large in that they typically have a larger area in terms of pixels. This also depends on the magnification of the objective being used, but in general, the CD36 receptors could primarily be described by a single (X,Y) point, even a sub-pixel, and the intensity of the image at that point. In contrast, an MCF7 nucleus under 20x magnification occupies around 1000 pixels. It can be described in terms of a (X,Y) centroid and mean intensity similar to the CD36 receptor, but also by other information such as area, texture of intensity, or number of foci contained within the nucleus. These extra details can be used to track a cell by modifying the entries of the cost matrix.

In the source algorithm the linking cost was defined as the distance squared:

The linking cost can be modified to include more variables or parameters through multiplication. The relative importance of each variable or parameter can affect the cost by weighting each variable with the value of the exponent. In general the linking cost would resemble,

The size of the nucleus can be incorporated into the linking cost. The most straightforward metric to consider is adding the area, in the units of pixels, to the linking cost. However, the distance metric and area share the same unit for distance, which means that they are of approximately equal weight in the following equation.

However, the change in area is not as important as the change in location of nuclei from timepoint to timepoint, so a weight will be added to the area to reduce its relative influence.

4.10 Tracking with cell movement

Cell lines all have mobility to some degree and MCF7 cells are no exception. Without taking into account cell movement the tracking algorithm is at risk of linking two unique cells together, because one cell displace another from one timepoint to the next. Mobility can be incorporated into the linking cost by predicting where a nucleus should be in the next time frame using a Kalman filter. The Kalman filter uses a mathematical model of cell movement and updates the parameters of the model by comparing the actual location of the nucleus with the prediction. The simplest model of cell movement is a linear model. A linear model is an appropriate approximation of complex cell movement as long as the timepoints are relatively frequent. That is to say that for a handful of frames a cell will not appear to change direction or speed.

The movement in the X and Y direction are independent of each other. The process noise is estimated from several tracks that have been manually curated in addition to the a priori estimate error covariance matrix. The following matrices were derived from MCF7 cells imaged with a 20x objective and binning 2:

4.11 Editing Tracks in the image viewer

Once the tracking algorithm processes the dataset they are visualized in the image viewer (Figure 4.7). The lines in the image shows the path of a track through time. The circle on that line shows where the cell is at the current timepoint. There are 3 colors randomly assigned to each track to help distinguish neighboring cells from one another. A text box is adjacent to each circle that contains metadata, such as the ID number of a given track.

The tracking algorithm is currently configured such that a division event, or inconsistent segmentation, will end a track and start a new track at the next time point. The tracks represent the automated part of cell tracking. The manual part of tracking consists of annotating and curating these tracks. Errors in tracking can be fixed by breaking tracks or joining tracks together. Each track can be assigned a cell label. The same cell label can be assigned to multiple tracks. A lineage of mother cells and daughter cells can be created with two mouse clicks that link a mother cell and daughter cell together.

These edits and cell labels can be performed with the help of a gui (Figure 4.8). Keyboard shortcuts exist for joining two tracks, 'n', breaking a track in two, 'b', deleting a track, '-', creating a new cell label, 'c', adding a track to a cell label, 't', and identifying a mother-daughter relationship, 'm'. A table displays the cell label data.

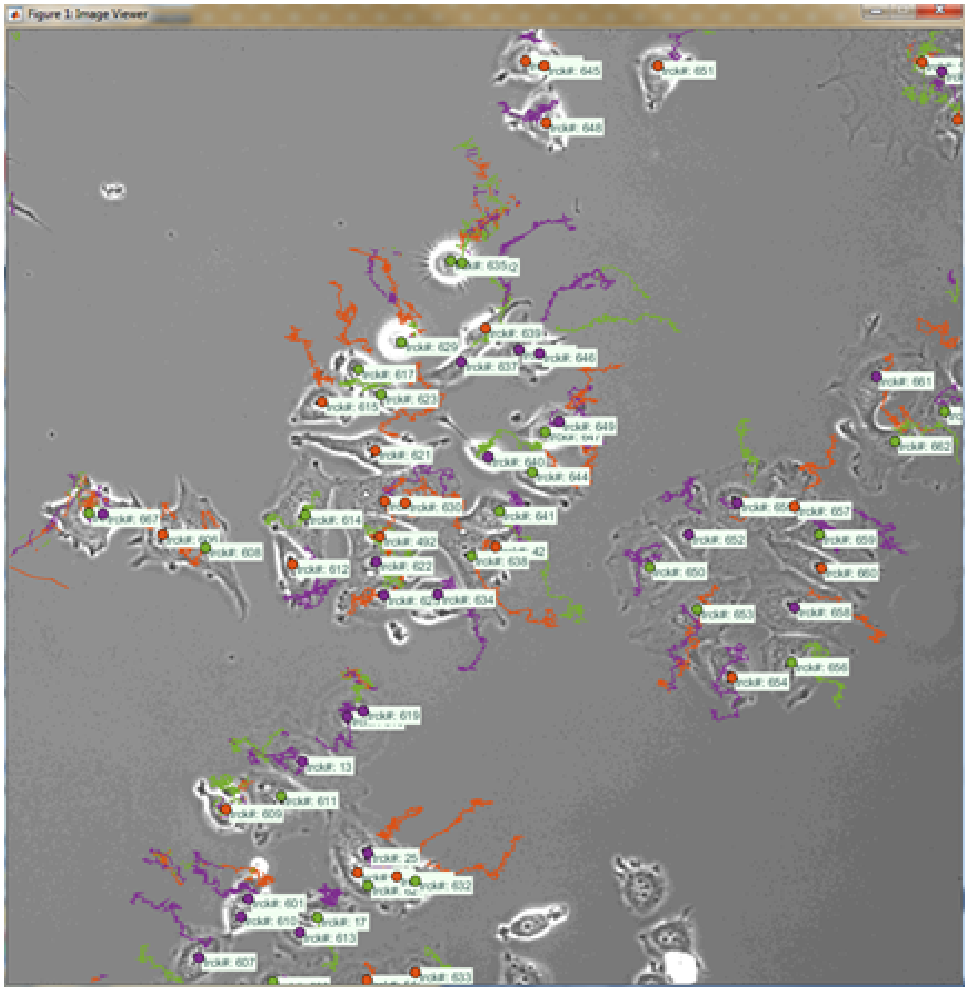


Figure 4.7: The Image viewer with showing tracks output by the tracking algorithm.

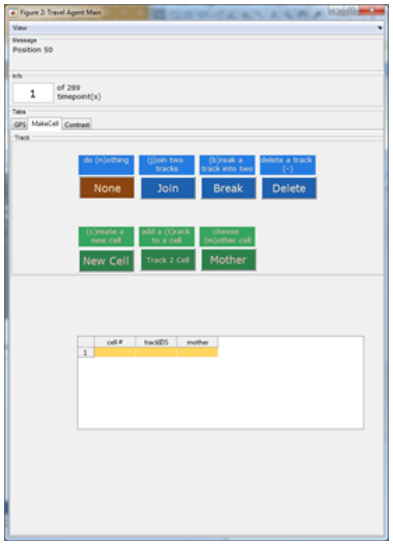


Figure 4.8: The cell metadata control gui. Information about division events is shown in the lineage table.

4.12 Quantifying cell traces

After annotating the tracks and identifying cells within the movie the tracking software is no longer needed. A script that parses the cell label and track data will create a matrix of traces that can be sorted by the GPS. At this point the image data has been quantified and is ready for further analysis. In the data set that has been referenced throughout the description of the tracking software, a collection of cells show p53 pulses in response to irradiation.

4.13 Limitations

The greatest limitation in the tracking software is the need for a nuclear marker in the cell line being analyzed. A good nuclear marker will lead to well segmented images. The better the segmentation the better the tracking, because the tracking algorithm relies upon consistency from frame to frame. Poor segmentation will lead to shorter tracks and more effort will be required during annotation. A good nuclear marker greatly improves the results of the tracking algorithm.

Another limitation is the need to manually identify mother and daughter cells. In the current configuration several steps are required to make this annotation. There is the potential to have this become automated as well. During division the cell changes morphology that is particularly distinct in the phase contrast image. If cell division could be identified by this uniqueness then the daughter cells could be identified at a later timepoint using the Jaqaman merging and splitting algorithm, which is not currently implemented in the tracking software.

5 Conclusion

5.1 Long-term p53 Dynamics Look Like Nutlin-Induced Dynamics

Single-cell analysis of long-term p53 dynamics have shown that a small fraction of cells exhibit sustained p53 dynamics. This observation invites questions about their significance. The sustained, elevated levels of p53 that naturally appear following IR exposure are reminiscent of the sustained dynamics induced in pulsing cells using the small molecule nutlin. The nutlin-sustained p53 dynamics were shown to alter the transcription of p53-target genes, especially PML and p21, changing the timing or strength of induction. It would be interesting to measure the expression of PML and p21 in the cells that exhibit long-term sustained dynamics. Additionally, nutlin-sustained dynamics were shown to increase the probability of entering senescence. It suggests a connection between long-term sustained dynamics and cell fate is possible.

5.2 Do Sustained p53 Dynamics Influence Gene Expression or Cell Fate?

These gene measurements could conceivably be measured at the population level or in single cells. To measure gene expression at the population level would require enrichment of the sustained population and then measuring the protein in a Western blot or the mRNA using qPCR and comparing these levels with cells that were not-sustained. The first challenge is the method to enrich these cells. One solution would use fluorescence-activated cell sorting (FACS) to identify and separate the subpopulation of sustained cells. Irradiated cells could be sorted several days after IR exposure. Some pitfalls towards this approach is the strength of the fluorescent signal and sorting pulsing cells at the peak of activity along with cells with sustained p53 activity. Prior experience attempting to sort fluorescently tagged p53 suggests that the sensitivity of the FACS machine may not be good enough to distinguish p53 levels in the range of activity observed under the microscope, for the tagged p53 is relatively dim.

To overcome the limitations of the p53-reporter brightness and FACS machine sensitivity a reporter gene could be constructed. The ideal reporter gene would distinguish between pulsing cells and sustained cells in a binary fashion. One approach would rely on a fluorescent protein with a long maturation period and high degradation rate. Under pulsing conditions this protein would be degraded before the protein could mature and give off signal. In sustained conditions, strong induction of this protein would lead to concentrations high enough that the average lifetime of the reporter would exceed the maturation time and the cell would become fluorescent. Jacob in the Lahav Lab has demonstrated the feasibility of this kind of reporter. If fluorescent proteins are not bright enough to properly sort the reporter gene could be a cell surface marker that could be targeted with a fluorescent dye before being sorted. Finally, perhaps the best reporter gene would be a tagged version of PIDD. In population measurements PIDD was induced 24 hours after IR exposure. If PIDD is responsible for the sustained behavior then its expression would be exclusive to the sustained dynamics phenotype.

Measuring gene expression at the single-cell level could be accomplished using the same smFISH technique that was used to compare gene expression between cells with nutlin-sustained dynamics and pulsing dynamics. smFISH has an advantage over immunofluorescence, because proteins have an extra layer of regulation that may not correlate with p53 activity. However, smFISH is also more technically challenging, so it is probably worth doing both assays, especially for a gene like p21 since it is a well-studied p53-target. Another challenge would be finding enough long-term sustained cells to make a statistical argument about the gene expression in these cells. At a 10

Perhaps the biggest hurdle would be interpreting the results of the gene expression data. The nutlin-sustained dynamics were synchronized by the proximity to IR exposure, 12 hours, and the 3 additions of nutlin that altered the wild-type pulsing behavior. This meant the cumulative p53 dynamics could be controlled for within an experiment, which was necessary to separate p53-dependent transcription based solely on concentration from transcription dependent on dynamics. The long-term sustained dynamics are more unsynchronized and induced at different timepoints, though generally at least 24 hours after IR exposure. The dynamics before the switch to sustained dynamics could also influence the gene expression measurement and it is not clear how to account for this possibility. Comparing varying degrees of gene expression, for example between p21, may be too difficult to interpret. If the gene expression difference is drastic, i.e. if there are a subset of genes induced only during the sustained dynamics, this would suggest these dynamics have a function role. Measuring expression of the PIDD gene would be a nice start.

The discussion to this point has focused on the possibility that p53 function is different during long-term sustained dynamics when it also possible that p53 function in not relevant. The unwritten assumption has been that the upstream signaling is the result of IR induced DNA damage and any change to p53 behavior would be the result of downstream events. In the days that follow IR exposure a pathway independent of ATM and the DNA-damage response could be activated and influence p53 stability. In the 3 days that follow IR exposure, MCF7 cells have been seen dividing (presumably successfully, but we cannot preclude aberrant chromosome segregation caused by dividing with damage); dividing unsuccessfully leading to the formation of micronuclei, slipping through mitosis and becoming 4N in the G1-phase of the cell-cycle; and dividing and then fusing to become binucleated. Any of these events threaten genomic integrity and could trigger some kind of response that could stabilize p53 or alter p53-target genes in a p53-independent manner. Additionally, the single-cell data presented in chapter 4 were from MCF7 cells damaged with 10 Gy. This level of damage was shown to induce senescence in the entire population of cells, which suggests that the variability in p53 dynamics does not have a functional consequence. However, while the different p53 dynamics may not lead to a unique cell fate, they could affect or reflect the timing of the commitment to senescence.

5.3 Single-Cell Measurements of PIDD

The connection between PIDD and elevated levels of p53 at long time-scales established in population based measurements should be followed up at the single cell level. Long-term p53 levels were noticeably reduced when PIDD was knocked down with siRNA or the PIDDosome-activated caspase2 was inhibited. If PIDD is responsible for the sustained p53 dynamics observed in 10

If the knockdown or induction of PIDD does not affect the fraction of cells with sustained dynamics, then this would call into question how the heterogeneity in dynamics observed in single cells reflects the p53 dynamics seen at the population level. It has been shown that, over the first 12 hours following IR exposure, the damped p53 pulses in Western blots reflects the loss of synchrony between the undamped pulses observed in single cells. The subpopulation of cells that express sustained dynamics mirror the long-term dynamics observed in Western blot and it is possible that this small fraction expresses p53 at such high levels that it dominates what is seen at the population level. It would be curious to discover that PIDD does not affect this fraction, because it would suggest PIDD regulates p53 stability in a more subtle way then the population measurements imply.

5.4 The Importance of Studying p53 Dynamics

The study of p53 dynamics can help elucidate the function of p53 in several fundamental ways. First, it can help enrich and define the signaling pathways that include p53. If one is trying to identify the nodes that belong to the p53 signaling network, this can be experimentally determined by identifying the genes and proteins that regulate p53 dynamics. This approach led to the discovery of the phosphatase Wip1 resetting the upstream ATM signaling between pulses of p53. It is also the approach used to identify and explore the role of PIDD in stabilizing p53 days after IR exposure.

Second, it has been shown that information is contained within the dynamics themselves and not just the p53 protein. Dynamics therefore represent another aspect of p53 that can considered a drug target. A clever manipulation of p53 dynamics may prove to be a viable therapy option that could potentially reactivate mutated p53 function, or it could p53-dependent cell death specifically in cancer cells. P53 dynamics should also be considered a potential source of variation that could explain why genotoxic stresses, such as chemotherapy or IR induced DNA damage, can often lead to heterogeneous cell responses at clinically relevant doses.

Finally, p53 dynamics must be studied at the single-cell level. The development and treatment of cancer is highly dependent on heterogeneity within the malignant population to advance towards metastasis, develop resistance to cancer therapies, and aid its overall survival. The sources of this heterogeneity may be detected within measurements of p53 dynamics, because they affect p53 behavior directly or simply by chance when a group of cells with similar dynamics experience different fates. The tools developed to measure and study p53 dynamics at the single-cell level can also be repurposed for the study of other proteins with interesting dynamics.

A
MATLAB Code to Count smFISH Foci

Listings

```
function [] = smfishStackPreprocessing(stackpath,varargin)
%noise reduction by matched filtering with a guassian kernel
%create gaussian filter that approximates 3D PSF of the microscope.
%Distance units are in micrometers.
%------ Set Parameters ------
parameters.objective = 60; %as in 60x
parameters.NA = 1.4; %typical of 60x oil immersion objections
parameters.rindex = 1.51; %typical refractive index of oil
parameters.camerapixellength = 6.45; %Both cameras in the Lahav have pixel dimensions of 6.45 x 6.45
        um.
parameters.zstepsize = 0.3; %User defined with z-stack is obtained
parameters.wavelength = .67; %Cy5 probe wavelength approximately 670 nanometers

%------ Parse varargin ------
%'flatfieldpath' = 'path\2\files'
%'subtractbackground' = true
p = inputParser;
p.addRequired('stackpath', @(x)ischar(x));
p.addParamValue('flatfieldpath', '', @(x)ischar(x));
p.addParamValue('subtractbackground', true, @(x)islogical(x));
p.addParamValue('fluorchan','Cy5',@(x)ischar(x));
p.parse(stackpath, varargin{:});
%------ flatfield correction for each z-slice ------
if ~isempty(p.Results.flatfieldpath)
    flatfieldcorrection(stackpath,p.Results.flatfieldpath)
    tempfoldername=regexp(stackpath,'(?<=\\)[\w ]*','match'); %Prepare to create a new folder to
      place background subtracted stacks
    tempfoldername=[tempfoldername{end},'_ff'];
    stackpath=[stackpath,'\..\',tempfoldername];
end
%------ read the contents of the input folder ------
disp(['working in ', stackpath]); %Sanity Check
dirCon_stack = dir(stackpath);
%------ Create a new folder to hold corrected images ------
tempfoldername=regexp(stackpath,'(?<=\\)[\w ]*','match'); %Prepare to create a new folder to place
      background subtracted stacks
tempfoldername=[tempfoldername{end},'_smFISH'];
smfishstackpath=[stackpath,'\..\',tempfoldername];
mkdir(smfishstackpath);
cd(smfishstackpath)
smfishstackpath=pwd;
```

```
%process only the stacks that contain single molecule FISH
stacknames=importStackNames(dirCon_stack,p.Results.fluorchan);
if isempty(stacknames)
    disp('Did not find any z-stacks of the specified fluorescent channel.')
    return
end
stacknames2=stacknames; %I apologize for the really confusing file naming system.
%This is part of a bug. This file expects the input filename to be of a
%certain format. The following for-loop partially ensures the file names
%are in that format.
for i=1:length(stacknames)
    Name_temp = regexprep(stacknames(i),'\s',''); %Remove all not(alphabetic, numeric, or underscore
      ) characters
    Name_temp = regexprep(Name_temp,'tocamera','','ignorecase'); %remove 'tocamera' if present b/c
      it is not informative
    Name_temp = regexprep(Name_temp,'camera','','ignorecase'); %remove 'camera' if present b/c it is
       not informative
    stacknames2(i) = Name_temp;
end

for bigInd = 1:length(stacknames)
    %----- Load the image file -----
    parameters.stacknametest = [stackpath '\' stacknames{bigInd}];
    [IM,sizeOfImage,hLoG,tempI1,tempI2,hMeanxy,hMeanz,IMMeanIntensity,hGaus,xy,z,pixelRatio] =
      variableInitialization(parameters);
    IM = loadZstack([stackpath '\' stacknames{bigInd}],IM,sizeOfImage);
    dataName = regexprep(stacknames2{bigInd},'(?<=_t)(\w*)(?=\.)','$1_data');
    dataName = regexp(dataName,'.*(?=\.)','match','once');
    save(dataName,'sizeOfImage');
    %----- background subtraction for each z-slice -----
    %I've heard that 3D deconvolution using an iterative blind-maximum-likehood
    %algorithm is very effective at removing out of focus light from each
    %z-slice. I played around with the AutoQuant software package at the NIC.
    %AutoQuant has this deconvolution algorithm and can batch process a whole
    %folder of files. The results were quite nice. However, the deconvolution
    %process is time consuming: it takes about 20 minutes for a 1344x1024x50
    %TIFF file and requires scheduling time at the NIC and using a computer at
    %the NIC for the processing. Perhaps later I can incorporate deconv. into
    %this MATLAB pipeline. Until then, as an alternative, I have found that
    %using the tried and true 'imopen' for background subtraction does a pretty
    %good job at removing out of focus light as long as the structuring element
    %is of an appropriate size (i.e. slightly bigger than a diffraction limited
    %spot).
    if p.Results.subtractbackground
        IM = JaredsBackground(IM);
    end
    %----- enhance diffraction limited spots using the LoG filter -----
    IM = imfilter(IM,hLoG,'symmetric'); %Totally works. sweet!
    %----- calculate the test statistics that will identify legit spots -----
    %%%Test Statistic 1: The 3D curvature. Gives a sense about how much a spot
    %resembles a point source of light. It gives a sense of the spots geometry
    %as opposed to the brightness of the spot.
    curvature = mySobelHessianCurvature(IM,tempI1,tempI2,pixelRatio);
    %A really large negative value indicates geometry like a point source. The
    %numbers produced are often extremely large and it may be a good idea to
    %normalize.
    %I turned a negative into a positive; I want you all to know that.
    curvature = -curvature;
    curvature(curvature<0) = 0;
    %%%Test Statistic 2: The mean brightness of the area. Indeed, we expect the
    %mRNA FISH signal to be brighter than the background. Taking the mean
    %reduces the weight of random peaks due to noise, since noise in these
    %images is of the zero-mean variety.
    %find local maxima
    fociCandidates = imregionalmax(IM,26);
    %Find the mean of a local volume that will capture an entire point source.
    for i=1:sizeOfImage(2)
        tempI1(:,i,:) = imfilter(reshape(IM(:,i,:),[sizeOfImage(1), sizeOfImage(3)]),hMeanz,'
      symmetric'); %z
    end
    for i=1:sizeOfImage(3)
        tempI2(:,:,i) = imfilter(tempI1(:,:,i),hMeanxy,'symmetric'); %y
    end
    for i=1:sizeOfImage(3)
        IMMeanIntensity(:,:,i) = imfilter(tempI2(:,:,i),hMeanxy','symmetric'); %x
    end
    %The final test statistic is the product of test statistic 1 and 2
    spotStat = IMMeanIntensity.*curvature;
```

```
%----- Find a threshold that separates signal from noise -----
spotStat = spotStat.*fociCandidates;
index = find(spotStat);
%clear fociCandidates
if iscolumn(index)
    index = index';
end
spotStat2 = spotStat(index);
IMMeanIntensity2 = IMMeanIntensity(index);
curvature2 = curvature(index);
%The test statistic tends to vary over several orders of magnitude
%therefore it is easier to compare these values in a log scale.
spotStat2 = log(spotStat2);
curvature2 = log(curvature2);
if isrow(spotStat2)
    spotStat2 = spotStat2';
end
if isrow(curvature2)
    curvature2 = curvature2';
end
if isrow(IMMeanIntensity2)
    IMMeanIntensity2 = IMMeanIntensity2';
end
if isrow(index)
    index = index';
end
spotStat3 = [spotStat2,index,curvature2,IMMeanIntensity2];
spotStat3 = sortrows(spotStat3,1);
save(dataName,'spotStat3','-append');
%find a good threshold
[threshold,n,xout,n2,xout2] = triminthresh(spotStat3(:,1)); %#ok<NASGU,ASGLU>
save(dataName,'threshold','n','xout','n2','xout2','-append');
ind = find(spotStat3(:,1)>threshold,1,'first');
foci = zeros(sizeOfImage);
foci(spotStat3(ind:end,2)) = 1;
fociarray = spotStat3(ind:end,2);
save(dataName,'fociarray','-append');
%----- Create the final image with bonafide spots and other aesthetic images -----
%sum projection of foci
sumProj = sum(foci,3);
Name = regexprep(stacknames2{bigInd},'(\w*)(?=\.)','$1_sumProj');
imwrite(uint8(sumProj),[smfishstackpath,'\',Name],'tif','WriteMode','append','Compression','none
  ');
%max project the stamp
stampProj3D = padarray(zeros(sizeOfImage),[xy xy z],'symmetric');
%gaussian stamp (approx. the PSF) on the sum projection of foci
foci2 = padarray(foci,[xy xy z]);
index = find(foci2);
if iscolumn(index)
    index = index';
end
s = size(foci2);
for i=index
    [i2,j2,k2] = ind2sub(s,i);
    stampProj3D(i2-xy:i2+xy,j2-xy:j2+xy,k2-z:k2+z) = stampProj3D(i2-xy:i2+xy,j2-xy:j2+xy,k2-z:k2
  +z) + hGaus;
end
stampProj3D2 = stampProj3D(xy+1:end-xy,xy+1:end-xy,z+1:end-z);
%Save the 3D image
%     Name = regexprep(stacknames2{bigInd},'(?<=_t)(\w*)(?=\.)','$1_stampProj3D');
%     for i = 1:sizeOfImage(3)
%         imwrite(uint8(stampProj(:,:,i)),[smfishstackpath,'\',Name],'tif','WriteMode','append
  ','Compression','none');
%     end
%Project the 3D image into 2D
stampProj = sum(stampProj3D2,3);
Name = regexprep(stacknames2{bigInd},'(\w*)(?=\.)','$1_stampProj');
imwrite(uint8(stampProj),[smfishstackpath,'\',Name],'tif','WriteMode','append','Compression','
  none');

%Create Max Projection of the input image
maxProj = max(IM,[],3);
maxProj = uint16(maxProj);
Name = regexprep(stacknames2{bigInd},'(\w*)(?=\.)','$1_maxProj');
imwrite(uint16(maxProj),[smfishstackpath,'\',Name],'tif','WriteMode','append','Compression','
  none');
%Create Merged Color image
maxProj = bitshift(maxProj, -4);
```

```
    maxProj = uint8(maxProj);
    [s1 s2] = size(maxProj);
    maxProj2 = zeros(s1,s2,3);
    maxProj2(:,:,1) = maxProj;
    maxProj2(:,:,2) = maxProj;
    maxProj2(:,:,3) = maxProj;
    for i = 1:length(fociarray)
        [y2,x2,~] = ind2sub(sizeOfImage,fociarray(i));
        maxProj2(y2,x2,:) = [255 0 0];
    end
    Name = regexprep(stacknames2{bigInd},'(\w*)(?=\.)','$1_ColorMerge');
    imwrite(uint8(maxProj2),[smfishstackpath ,'\',Name],'tif','WriteMode','append','Compression','
      none');
    %3D scatter plot
    %[y2,x2,z2] = ind2sub(s,fociarray);
    %scatter3(x2,y2,z2)
    Name = regexprep(stacknames2{bigInd},'(\w*)(?=\.)','$1_maxProj');
    smfishPlot([dataName '.mat'],smfishstackpath,Name,stacknames2{bigInd});
end
signalCompletionWithEmail();
signalCompletionWithSound();
end

function [tempI1] = mySobelHessianCurvature(I,tempI1,tempI2,pixelRatio)
%This function uses the Sobel filter to approximate the derivatives in a
%gradient. Since the sobel filter is seperable the it can also be
%conveniently extended to find the Hessian.
%The image I is 3D and has coordinates (y,x,z).
h1 = [0.25 0.5 0.25];
h2 = [-0.5 0 0.5];
h2z = h2/pixelRatio;
[sy,sx,sz] = size(I);
Fx = zeros(size(I));
Fy = zeros(size(I));
Fz = zeros(size(I));
Fxx = zeros(size(I));
Fxy = zeros(size(I));
Fxz = zeros(size(I));
Fyy = zeros(size(I));
Fyz = zeros(size(I));
Fzz = zeros(size(I));
%The Sobel separted filters to find the Fx
for i=1:sx
    tempI1(:,i,:) = imfilter(reshape(I(:,i,:),[sy sz]),h1,'symmetric'); %z
end
for i=1:sz
    tempI2(:,:,i) = imfilter(tempI1(:,:,i),h1,'symmetric'); %y
end
for i=1:sz
    Fx(:,:,i) = imfilter(tempI2(:,:,i),h2','symmetric'); %x
end
%The Sobel separted filters to find the Fy
for i=1:sx
    tempI1(:,i,:) = imfilter(reshape(I(:,i,:),[sy sz]),h1,'symmetric'); %z
end
for i=1:sz
    tempI2(:,:,i) = imfilter(tempI1(:,:,i),h2,'symmetric'); %y
end
for i=1:sz
    Fy(:,:,i) = imfilter(tempI2(:,:,i),h1','symmetric'); %x
end
%The Sobel separted filters to find the Fz
for i=1:sx
    tempI1(:,i,:) = imfilter(reshape(I(:,i,:),[sy sz]),h2z,'symmetric'); %z
end
for i=1:sz
    tempI2(:,:,i) = imfilter(tempI1(:,:,i),h1,'symmetric'); %y
end
for i=1:sz
    Fz(:,:,i) = imfilter(tempI2(:,:,i),h1','symmetric'); %x
end
%The Sobel separted filters to find the Fxx
for i=1:sx
    tempI1(:,i,:) = imfilter(reshape(Fx(:,i,:),[sy sz]),h1,'symmetric'); %z
end
for i=1:sz
    tempI2(:,:,i) = imfilter(tempI1(:,:,i),h1,'symmetric'); %y
end
```

```
for i=1:sz
    Fxx(:,:,i) = imfilter(tempI2(:,:,i),h2','symmetric'); %x
end
%The Sobel separted filters to find the Fxy
for i=1:sx
    tempI1(:,i,:) = imfilter(reshape(Fx(:,i,:),[sy sz]),h1,'symmetric'); %z
end
for i=1:sz
    tempI2(:,:,i) = imfilter(tempI1(:,:,i),h2,'symmetric'); %y
end
for i=1:sz
    Fxy(:,:,i) = imfilter(tempI2(:,:,i),h1','symmetric'); %x
end
%The Sobel separted filters to find the Fxz
for i=1:sx
    tempI1(:,i,:) = imfilter(reshape(Fx(:,i,:),[sy sz]),h2z,'symmetric'); %z
end
for i=1:sz
    tempI2(:,:,i) = imfilter(tempI1(:,:,i),h1,'symmetric'); %y
end
for i=1:sz
    Fxz(:,:,i) = imfilter(tempI2(:,:,i),h1','symmetric'); %x
end
%The Sobel separted filters to find the Fyy
for i=1:sx
    tempI1(:,i,:) = imfilter(reshape(Fy(:,i,:),[sy sz]),h1,'symmetric'); %z
end
for i=1:sz
    tempI2(:,:,i) = imfilter(tempI1(:,:,i),h2,'symmetric'); %y
end
for i=1:sz
    Fyy(:,:,i) = imfilter(tempI2(:,:,i),h1','symmetric'); %x
end
%The Sobel separted filters to find the Fyz
for i=1:sx
    tempI1(:,i,:) = imfilter(reshape(Fy(:,i,:),[sy sz]),h2z,'symmetric'); %z
end
for i=1:sz
    tempI2(:,:,i) = imfilter(tempI1(:,:,i),h1,'symmetric'); %y
end
for i=1:sz
    Fyz(:,:,i) = imfilter(tempI2(:,:,i),h1','symmetric'); %x
end
%The Sobel separted filters to find the Fzz
for i=1:sx
    tempI1(:,i,:) = imfilter(reshape(Fz(:,i,:),[sy sz]),h2z,'symmetric'); %z
end
for i=1:sz
    tempI2(:,:,i) = imfilter(tempI1(:,:,i),h1,'symmetric'); %y
end
for i=1:sz
    Fzz(:,:,i) = imfilter(tempI2(:,:,i),h1','symmetric'); %x
end
%Find the curvature matrix
myHessian = zeros(3);
for i = 1:numel(I)
    myHessian(1,1) = Fxx(i);
    myHessian(1,2) = Fxy(i);
    myHessian(1,3) = Fxz(i);
    myHessian(2,1) = Fxy(i);
    myHessian(2,2) = Fyy(i);
    myHessian(2,3) = Fyz(i);
    myHessian(3,1) = Fxz(i);
    myHessian(3,2) = Fyz(i);
    myHessian(3,3) = Fzz(i);
    tempI1(i) = det(myHessian);
end
end

function [threshold,n,xout,n2,xout2] = triminthresh(A)
%Calculate a few rank statistics (assumes A is already sorted)
la = length(A);
q1a = A(round(0.25*la)); %first quartile
q2a = A(round(0.50*la));
q3a = A(round(0.75*la)); %third quartile
myIQRa = q3a-q1a;
myCutoffa = 3*myIQRa+q2a;
%Create the histogram
```

```
[n,xout]=hist(A,100);
%Use the triangle threshold for the initial guess
ind = triangleThreshCore(n);
threshold = xout(ind);
%Look for minimum change in the number of foci or when the change in foci
%is less than 1.
B = A(A>threshold);
[n2,xout2] = hist(B,100);
n2der = smooth(n2);
n2der = conv(n2der,[0.5 0 -0.5],'same'); %the central difference derivative to find the min
for i = 2:length(n2der);
    if (n2der(i-1)<0 && n2der(i)>=0)
        ind = i-1;
        break
    elseif (abs(n2der(i-1))<=1) && (n2(i-1) == 0 || n2(i-1) == 1 || n2(i-1) == 2)
        ind = i-1;
        break
    end
end
threshold = xout2(ind);
logicStepCounter = 1;
while logicStepCounter ~= 0
    switch logicStepCounter
        case 1
            if threshold<myCutoffa
                %Find the peak of the putative signal.
                [~,putativeSignalPeakInd] = max(n2);
                putativeSignalPeak = xout2(putativeSignalPeakInd);
                logicStepCounter = 2;
            else
                break
            end
        case 2
            if (putativeSignalPeak>myCutoffa) && (putativeSignalPeakInd>=2)
                %If the peak is greater than the cutoff than go forward with
                %the threshold search. Look for the min to the left of this peak.
                %It will be the final threshold.
                for i = putativeSignalPeakInd:-1:2;
                    if (n2der(i-1)<0 && n2der(i)>=0)
                        ind = i-1;
                        break
                    elseif (abs(n2der(i-1))<=1) && (n2(i-1) == 0 || n2(i-1) == 1 || n2(i-1) == 2)
                        ind = i-1;
                        break
                    end
                end
                threshold = xout2(ind);
                break
            else
                logicStepCounter = 3;
            end
        case 3
            %Repeat the triangle threshold method
            C = B(B>putativeSignalPeak);
            [n3,xout3] = hist(C,100);
            ind = triangleThreshCore(n3);
            threshold = xout3(ind);
            C = B(B>threshold);
            [n3,xout3] = hist(C,100);
            n3der = smooth(n3);
            n3der = conv(n3der,[0.5 0 -0.5],'same'); %the central difference derivative to find the
    min
            n3der = smooth(n3der);
            for i = 2:length(n3der);
                if (n3der(i-1)<0 && n3der(i)>=0)
                    ind = i-1;
                    break
                elseif (abs(n3der(i-1))<=1) && (n3(i-1) == 0 || n3(i-1) == 1 || n3(i-1) == 2)
                    ind = i-1;
                    break
                end
            end
            threshold = xout3(ind);
            break
        otherwise
            disp('If you see this message something went wrong during threshold calculation.');
            break
    end
```

```
end
end

function [ind2] = triangleThreshCore(n)
%Find the highest peak the histogram
[c,ind]=max(n);
%Assume the long tail is to the right of the peak and envision a line from
%the top of this peak to the end of the histogram.
%The slope of this line, the hypotenuse, is calculated.
x1=0;
y1=c;
x2=length(n)-ind;
y2=n(end);
m=(y2-y1)/(x2-x1); %The slope of the line

%----- Find the greatest distance -----
%We are looking for the greatest distance betweent the histrogram and line
%of the triangle via perpendicular lines
%The slope of all lines perpendicular to the histogram hypotenuse is the
%negative reciprocal
p=-1/m; %The slope is now the negative reciprocal
%We now have two slopes and two points for two lines. We now need to solve
%this two-equation system to find their intersection, which can then be
%used to calculate the distances
iarray=(0:(length(n)-ind));
L=zeros(size(n));
for i=iarray
    intersect=(1/(m-p))*[-p,m;-1,1]*[c;n(i+ind)-p*i];
    %intersect(1)= y coordinate, intersect(2)= x coordinate
    L(i+ind)=sqrt((intersect(2)-i)^2+(intersect(1)-n(i+ind))^2);
end
[~,ind2]=max(L);
end

function [S] = JaredsBackground(S)
resizeMultiplier = 1/2; % Downsampling scale factor makes image processing go faster and smooths
     image
seSize2 = 40; % I find the value of 25 works well with 60x, binning 1, mRNA FISH images
se2 = strel('disk', seSize2*resizeMultiplier); %Structing elements are necessary for using MATLABS
     image processing functions
origSize  = size(S);
for k=1:origSize(3)
    % Rescale image and compute background using closing/opening.
    I     = imresize(S(:,:,k), resizeMultiplier);
    pad   = ceil(seSize2*resizeMultiplier);
    % Pad image with a reflection so that borders don't introduce artifacts
    I     = padarray(I, [pad,pad], 'symmetric', 'both');
    % Perform opening/closing to get background
    I      = imopen(I, se2);     % ignore high-intensity features typical of mRNA spots
    % Remove padding and resize
    I      = floor(imresize(I(pad+1:end-pad, pad+1:end-pad), origSize(1:2)));
    S(:,:,k) = S(:,:,k) - I;
end
S(S<0)=0;
end

function [IM] = loadZstack(path,IM,s)
t = Tiff(path,'r');
if s(3) > 1
    for k=1:s(3)-1
        IM(:,:,k) = double(t.read);
        t.nextDirectory;
    end
end
%one last time without t.nextDirectory
IM (:,:,s(3)) = double(t.read);
t.close;
end

function [Temp] = importStackNames(dirCon_stack,fc)
expr=['.*(?<!thumb.*)_w\d+' fc '.*'];
Temp=cell([1,length(dirCon_stack)]); %Initialize cell array
% ----- Identify the legitimate stacks -----
i=1;
for j=1:length(dirCon_stack)
    Temp2=regexp(dirCon_stack(j).name,expr,'match','once','ignorecase');
    if Temp2
        Temp{i}=Temp2;
        i=i+1;
    end
end
% ------- Remove empty cells -------
Temp(i:end)=[];
% for j=length(Temp):-1:1
%     if isempty(Temp{j})
%         Temp(j)=[];
%     end
% end
end

function [IM,sizeOfImage,hLoG,tempI1,tempI2,hMeanxy,hMeanz,IMMeanIntensity,hGaus,xy,z,pixelRatio] = variableInitialization(p)
%This function was made in a effort to speed things up. I'm not sure it did
%that. It may have just made the code more difficult to read.
info = imfinfo(p.stacknametest,'tif');
sizeOfImage = [info(1).Height, info(1).Width, length(info)];

IM = zeros(sizeOfImage);
tempI1 = zeros(sizeOfImage);
tempI2 = zeros(sizeOfImage);
IMMeanIntensity = zeros(sizeOfImage);
sigmaXYos = 0.21*p.wavelength/p.NA; %lateral st. dev of the gaussian filter in object space
sigmaZos = 0.66*p.wavelength*p.rindex/(p.NA^2) ;%axial st. dev of the gaussian filter in object space
Pxy = p.camerapixellength/p.objective; %lateral pixel size
sigmaXY = sigmaXYos/Pxy; %lateral st. dev of gaussian filter in image space
sigmaZ = sigmaZos/p.zstepsize; %axial st. dev of gaussian filter in image space
xy = round(3*sigmaXY);
z = round(3*sigmaZ);
xyMLV = round(4*sigmaXY);
zMLV = round(4*sigmaZ);
K = 1/((2*pi)^(3/2)*sqrt(sigmaXY^2*sigmaZ)); % log3d coefficient
log3d = @(x,y,z) K*exp(-0.5*(x^2/sigmaXY+y^2/sigmaXY+z^2/sigmaZ))*((x^2-4*sigmaXY)/(4*sigmaXY^2)+(y^2-4*sigmaXY)/(4*sigmaXY^2)+(z^2-4*sigmaZ)/(4*sigmaZ^2));
hLoG = zeros(2*xy+1,2*xy+1,2*z+1);
for i=1:2*xy+1
    for j=1:2*xy+1
        for k=1:2*z+1
            hLoG(i,j,k) = log3d(i-1-xy,j-1-xy,k-1-z); %the 3D filter
        end
    end
end
%tune the filter so that it does not amplify the signal.
temp1 = ones(2*xy+1,2*xy+1,2*z+1);
hLoG=-hLoG; %otherwise the center weight, the largest weight, is negative.
temp2 = sum(sum(sum(temp1.*hLoG)));
hLoG=hLoG/temp2;
K2=1/(xyMLV*xyMLV*zMLV);
hMeanxy=ones(1,xyMLV);
hMeanz=K2*ones(1,zMLV);
%3D Gaussian Filter\Stamp\PSF approximation
mu = [0,0,0]; %zero mean gaussian
SIGMA = [sigmaXY,0,0;0,sigmaXY,0;0,0,sigmaZ];
hGaus = zeros(2*xy+1,2*xy+1,2*z+1);
for i=1:2*xy+1
    for j=1:2*xy+1
        for k=1:2*z+1
            hGaus(i,j,k) = mvnpdf([i-1-xy,j-1-xy,k-1-z],mu,SIGMA); %the 3D filter
        end
    end
end
hGaus=hGaus*(2^5)/(max(max(max(hGaus))));
pixelRatio = sigmaZ/sigmaXY;
end

function [] = signalCompletionWithSound()
global playerkwk
if ~isempty(playerkwk)
    play(playerkwk);
else
    disp('Victory over Data!')
end
end

function [] = signalCompletionWithEmail()
% Send the email
```

```
sendmail('gandalfisarockstar@gmail.com', 'Mail from MATLAB', ...
    'Hi Kyle! Your MATLAB run is complete!')
% sendmail('gandalfisarockstar@gmail.com', 'Mail from MATLAB', ...
%      'Hi Kyle! Your MATLAB run is complete!', ...
%      {'sub_folder/signals.m', 'system.mdl'})
end
```

Listing A.1: smfishStackPreprocessing.m

B

MATLAB Code to Semi-Automated Cell Tracking

Listings

```matlab
%%
% Input: M and N are arrays where each row represents a datapoint and the
% number of dimensions are reflected in the number of columns.
%
%
% Output: D : is an m x n matrix where _m_ is the number of rows in M and
% _n_ is the number of rows in N.
function D = cellularGPSTracking_distanceMatrix(M,N) %#codegen
%%
% analyze inputs
[Mrows,Mcols] = size(M);
[Nrows,Ncols] = size(N);
if Mcols ~= Ncols
    error('cGPSTrackingDistMat:colDisagree','The number of columns in each input array must agree');
end
%%
%
D = zeros(Mrows,Nrows);
for i = 1:Mrows
    for j = 1:Nrows
        D(i,j) = norm(M(i,:)-N(j,:));
    end
end
end
```

Listing B.1: cellularGPSTrackingdistanceMatrix.m

```matlab
%%
% * A = the model described by difference equations
% * B = the control input equation
% * H = the measurement equation
% * I = identity matrix the size of the model
% * K = the Kalman gain%
% * Ppri = a priori estimate error covariance
% * Ppredict = the predicted estimate error covariance
% * Ppost = a posteriori estimate error covariance
% * Q = the measurement noise covariance
% * R = the process noise covariance
% * U = the control input
% * Xpri = the current state
% * Xpredict = the predicted state
% * Xpost = the updated prediction
% * Z = the measured state to be compared to the predicted state
%
% update the measured state, Z
function kf = cellularGPSTracking_Kalman_Correct(kf)
kf.K = kf.Ppredict*transpose(kf.H)/(kf.H*kf.Ppredict*transpose(kf.H) + kf.R); %the division symbol
     is a matrix inverse operation in this case
kf.Xpost = kf.Xpredict + kf.K*(kf.Z - kf.H*kf.Xpredict);
kf.Ppost = (kf.I - kf.K*kf.H)*kf.Ppredict;
end
```

Listing B.2: cellularGPSTrackingKalmanCorrect.m

```matlab
%%
% * A = the model described by difference equations
% * B = the control input equation
% * H = the measurement equation
% * K = the Kalman gain%
% *  Ppri = a priori estimate error covariance
% *  Ppredict = the predicted estimate error covariance
% *  Ppost = a posteriori estimate error covariance
% * Q = the measurement noise covariance
% * R = the process noise covariance
% * U = the control input
% *  Xpri = the current state
% *  Xpredict = the predicted state
% *  Xpost = the updated prediction
% * Z = the measured state to be compared to the predicted state
function kf = cellularGPSTracking_Kalman_Predict(kf)
kf.Xpredict = kf.A*kf.Xpri + kf.B*kf.U;
kf.Ppredict = kf.A*kf.Ppri*transpose(kf.A) + kf.Q;
end
```

Listing B.3: cellularGPSTrackingKalmanPredict.m

```
%%
% * A = the model described by difference equations
% * B = the control input equation
% * H = the measurement equation
% * K = the Kalman gain%
% *  Ppri = a priori estimate error covariance
% *  Ppredict = the predicted estimate error covariance
% *  Ppost = a posteriori estimate error covariance
% * Q = the measurement noise covariance
% * R = the process noise covariance
% * U = the control input
% *  Xpri = the current state
% *  Xpredict = the predicted state
% *  Xpost = the updated prediction
% * Z = the measured state to be compared to the predicted state
function kf = cellularGPSTracking_Kalman_Predict_update(kf)
kf.Xpri = kf.Xpost;
kf.Ppri = kf.Ppost;
end
```

Listing B.4: cellularGPSTrackingKalmanPredictupdate.m

```
function [] = cellularGPSTracking_makeTracks_movementWithIntensity(moviePath)
trackingProfile = loadjson(fullfile(moviePath,'cGPS_trackingProfile.txt'));
smda_database = readtable(fullfile(moviePath,'smda_database.txt'),'Delimiter','\t');
positionNumber = transpose(unique(smda_database.position_number));
for i = 1:length(positionNumber)
    groupNumber(i) = smda_database.group_number(find(smda_database.position_number == positionNumber
      (i),1,'first'));
end
tablePathOut = fullfile(moviePath,'TRACKING_DATA');
if ~isdir(tablePathOut)
    mkdir(tablePathOut);
end
%% find tracks for centroids in each position
%
for i = 1:length(positionNumber)
    %% sort out centroids for each timepoint
    % sorting timepoints in descending order means the tracking will be
    % performed in reverse time.
    cenTablePosition = readtable(fullfile(moviePath,'CENTROID_DATA',sprintf('centroid_measurements_g
      %d_s%d.txt',groupNumber(i),positionNumber(i))),'Delimiter','\t');
    mytime = sort(unique(cenTablePosition.timepoint),'descend');
    centroidCell = cell(size(mytime));
    for j = 1:length(mytime)
        centroidCell{j} = sortrows(cenTablePosition(cenTablePosition.timepoint == mytime(j),:),{'
      centroid_col','centroid_row'},{'ascend','ascend'});
    end
    %% initialize the tracking variables with the first set of centroids
    %
    centroidPrime = centroidCell{1};
    trackCounter = height(centroidPrime)+1;
    trackID = transpose(1:height(centroidPrime));
    trackCostMax = 0;
    centroidPrime.trackID = trackID;
    centroidPrime.trackCost = zeros(height(centroidPrime),1);
    centroidPrime.displacement = zeros(height(centroidPrime),1);
    centroidPrime.speed = zeros(height(centroidPrime),1);
```

```
centroidCell{1} = centroidPrime;
%%% setup the starting conditions for the kalman filter
% The starting conditions are stored within a JSON file. These
% conditions include the model and covariance matrices for process and
% measurement noise, which have been estimated from previous tracking
% data.
kf = trackingProfile.kalmanFilter;
%%% replicate a Kalman filter for every track.
% the variable suffix *M* denotes the data is sourced from the _t-1_
% timepoint. *N* denotes the data is sourced from the _t_ timepoint.
% After a round of tracking the *N* data will become the *M* data.
kfcellM = repmat({kf},height(centroidPrime),1);
for j = 1:size(centroidPrime,1)
    mykf = kfcellM{j};
    mykf.Xpri = [centroidPrime.centroid_col(j);0;centroidPrime.centroid_row(j);0];
    kfcellM{j} = mykf;
end
%% link tracks using the global solution to a cost matrix
% based upon the Jaqaman-Danuser 2008 Nat. Methods paper
for j = 2:length(mytime) %loop 1
    %%% find centroids
    % find the centroids for the _t-1_ and _t_ timepoints
    centroidM = centroidCell{j-1};
    centroidN = centroidCell{j};
    posM = centroidM{:,{'centroid_col','centroid_row'}};
    posN = centroidN{:,{'centroid_col','centroid_row'}};
    masterCentroid = vertcat(centroidCell{1:j-1});
    %%% Kalman filter: linear motion
    % time update, predict
    predictlp1 = zeros(size(posM));
    for k = 1:size(posM,1)
        mykf = kfcellM{k};
        mykf = cellularGPSTracking_Kalman_Predict(mykf);
        predictlp1(k,1) = mykf.Xpredict(1);
        predictlp1(k,2) = mykf.Xpredict(3);
        kfcellM{k} = mykf;
    end
    %distM = cellularGPSTracking_distanceMatrix(posM,posN);
    distM = cellularGPSTracking_distanceMatrix(predictlp1,posN);
    %%% particle specific distance thresholds
    % * track specific movement threshold is 3x the standard deviation of
    % previous links
    % * local density threshold is half the distance to its nearest
    % neighbor
    distM2 = cellularGPSTracking_distanceMatrix(posM,posM);
    for k = 1:size(posM,1)
        displacementlp1 = masterCentroid.displacement(masterCentroid.trackID == trackID(k));
        if length(displacementlp1) > 5
            tsmthresh = mean(displacementlp1) + 2*std(displacementlp1);
        else
            displacementlp1 = masterCentroid.displacement;
            tsmthresh = mean(displacementlp1) + 2*std(displacementlp1);
        end
        distM2row = sort(distM2(k,:));
        ldthresh = 0.5*distM2row(2);
        finalthresh = max([ldthresh,tsmthresh,trackingProfile.distance.movementThresholdMaxMin])
  ;
        distMrow = distM(k,:);
        distMrow(distMrow>finalthresh) = Inf;
        distM(k,:) = distMrow;
    end
    %%% costM11
    %
    costM11 = distM.^2;
    costM11(costM11>trackingProfile.distance.movementThresholdMax^2) = -1;
    %%%
    % this is to initialize the trackCostMax
    if j==2 && any(costM11(:)~=-1)
        trackCostMax = prctile(costM11(costM11~=-1),80);
    end
    %%% costM12
    %
    costM12 = ones(size(posM,1),size(posM,1))*-1;
    for k = 1:size(posM,1) %loop 1 in loop 1
        costM12(k,k) = trackCostMax;
    end
    %%% costM21
    %
```

```
        costM21 = ones(size(posN,1),size(posN,1))*-1;
        for k = 1:size(posN,1)
            costM21(k,k) = trackCostMax;
        end
        %%% costM22
        % The minimum value of the costM11 at the values of the transpose of
        % costM11.
        costM22 = transpose(costM11);
        costM22(costM22 ~= -1) = min([min(costM11(costM11 ~= -1)),min(diag(costM12)),min(diag(
      costM21))]);
        %%% assemble the cost matrix
        %
        costM = [costM11,costM12;costM21,costM22];
        costM(costM == -1) = Inf;
        [ROWSOL,~,~,~] = lapjv(costM);
        %%
        %
        trackID = zeros(size(posN,1),1);
        trackCost = zeros(size(posN,1),1);
        trackDisplacement = zeros(size(posN,1),1);
        trackSpeed = zeros(size(posN,1),1);
        kfcellN = cell(size(posN,1),1);
        distM3 = cellularGPSTracking_distanceMatrix(posM,posN);
        for k = 1:size(posM,1)
            if ROWSOL(k) <= size(posN,1)
                trackID(ROWSOL(k)) = centroidM.trackID(k);
                trackCost(ROWSOL(k)) = costM11(k,ROWSOL(k));
                trackDisplacement(ROWSOL(k)) = distM3(k,ROWSOL(k));
                %%% kalman filter
                % measurement update, correct
                mykf = kfcellM{k};
                mykf.Z = [posN(ROWSOL(k),1);posN(ROWSOL(k),1)-posM(k,1);posN(ROWSOL(k),2);posN(
      ROWSOL(k),2)-posM(k,2)];
                mykf = cellularGPSTracking_Kalman_Correct(mykf);
                mykf = cellularGPSTracking_Kalman_Predict_update(mykf);
                %mykf.Xpri(1) = posN(ROWSOL(k),1);
                %mykf.Xpri(3) = posN(ROWSOL(k),2);
                kfcellN{ROWSOL(k)} = mykf;
                trackSpeed(ROWSOL(k)) = norm([mykf.Xpost(2),mykf.Xpost(4)]);
            end
        end
        if max(trackCost) > trackCostMax
            trackCostMax = max(trackCost);
        end
        for k = transpose(find(trackID == 0))
            trackID(k) = trackCounter;
            trackCounter = trackCounter + 1;
            trackCost(k) = costM21(k,k);
            trackDisplacement(k) = 0;
            %%% kalman filter
            % measurement update, correct
            kf.Xpri = [posN(k,1);0;posN(k,2);0];
            kfcellN{k} = kf;
        end
        centroidN.trackID = trackID;
        centroidN.trackCost = trackCost;
        centroidN.displacement = trackDisplacement;
        centroidN.speed = trackSpeed;
        centroidCell{j} = centroidN;
        kfcellM = kfcellN;
    end
    positionCentroid = vertcat(centroidCell{:});
    positionCentroid2 = positionCentroid(:,{'trackID','timepoint','centroid_row','centroid_col'});
    tablename = sprintf('trackingPosition_%d.txt',positionNumber(i));
    writetable(positionCentroid2,fullfile(tablePathOut,tablename),'Delimiter','\t');
    %% plot data for feedback purposes
    %
    figure;
    hold on
    masterCentroid = vertcat(centroidCell{:});
    trackID = unique(masterCentroid.trackID);
    tracklength = zeros(size(trackID));
    for j = 1:length(trackID) % loop2
        mylogical = masterCentroid.trackID == trackID(j);
        tracklength(j) = sum(mylogical);
        if tracklength(j) == 1
            myrow = masterCentroid.centroid_row(mylogical);
            mycol = masterCentroid.centroid_col(mylogical);
```

```
                mytime = masterCentroid.timepoint(mylogical);
                output = sortrows([mytime,mycol,myrow]);
                plot(output(:,2),output(:,3),'o','Color',[rand rand rand],'LineWidth',1.5);
                continue
            end
            myrow = masterCentroid.centroid_row(mylogical);
            mycol = masterCentroid.centroid_col(mylogical);
            mytime = masterCentroid.timepoint(mylogical);
            output = sortrows([mytime,mycol,myrow]);
            plot(output(:,2),output(:,3),'Color',[rand rand rand],'LineWidth',1.5);
        end
        hold off
        myax = gca;
        set(myax,'ydir','reverse')
        sum(tracklength > 50)
    end

end
```

Listing B.5: cellularGPSTrackingmakeTracksmovementWithIntensity.m

```
classdef cellularGPSTrackingManual_object < handle
    properties
        %%% DATA
        %
        centroid_measurements
        ity % itinerary
        mcl % makecell
        moviePath
        smda_database
        smda_databaseLogical
        smda_databaseSubset
        track_database
        %%% GUIS
        %
        gui_imageViewer
        gui_control
        %%% INDICES AND POINTERS AND MODES
        % state information about the gui and the information being
        % displayed
        indG = 1;
        indP = 1;
        indS = 1;
        indT = 1;
        indZ = 1;
        pointerGroup = 1;
        pointerPosition = 1;
        pointerSettings = 1;
        indImage = 1;
        makecell_mode = 'none';
    end
%       properties (SetAccess = private)
%       end
%       events
%       end
    methods
        %%
        %
        function obj = cellularGPSTrackingManual_object(moviePath)
            obj.moviePath = moviePath;
            %% Load settings
            %
            obj.smda_database = readtable(fullfile(moviePath,'thumb_database.txt'),'Delimiter','\t')
    ;
            obj.indG = min(obj.smda_database.group_number);
            obj.indP = min(obj.smda_database.position_number);
            obj.indS = min(obj.smda_database.settings_number);

            obj.ity = cellularGPSTrackingManual_object_itinerary;
            obj.ity.import(fullfile(moviePath,'smdaITF.txt'));
            obj.mcl = cellularGPSTrackingManual_object_makecell(moviePath);
            obj.loadTrackData;
            obj.updateFilenameListImage;
            %% Launch gui
            %
            obj.gui_imageViewer = cellularGPSTrackingManual_object_imageViewer(obj);
            obj.gui_control = cellularGPSTrackingManual_object_control(obj);

            obj.gui_imageViewer.loadNewTracks;
            obj.gui_control.tabContrast_axesContrast_ButtonDownFcn;
            obj.gui_control.tabContrast_sliderMax_Callback
        end
        %%
        %
        function initializeImageViewer(obj)
            if(~isempty(obj.gui_imageViewer))
                obj.gui_imageViewer.delete;
            end
            obj.gui_imageViewer = cellularGPSTrackingManual_gui_imageViewer(obj);
            obj.gui_imageViewer.launchImageViewer;
        end
        %%
        %
        function delete(obj)
            delete(obj.gui_imageViewer);
            delete(obj.gui_control);
        end
        %%
        %
        function obj = updateFilenameListImage(obj)
            obj.smda_databaseLogical = obj.smda_database.group_number == obj.indG &...
                obj.smda_database.position_number == obj.indP &...
                obj.smda_database.settings_number == obj.indS;
            mytable = obj.smda_database(obj.smda_databaseLogical,:);
            obj.smda_databaseSubset = sortrows(mytable,{'timepoint'});
        end
        %%
        %
        function obj = loadTrackData(obj)
            numOfPosition = sum(obj.ity.number_position);
            obj.track_database = cell(numOfPosition,1);
            positionInd = horzcat(obj.ity.ind_position{:});
            for i = positionInd
                obj.track_database{i} = readtable(fullfile(obj.moviePath,'TRACKING_DATA',...
                    sprintf('trackingPosition_%d.txt',i)),...
                    'Delimiter','\t');
            end
        end
    end
end
```

Listing B.6: cellularGPSTrackingManualobject.m

```
classdef cellularGPSTrackingManual_object_imageViewer < handle
    %% Properties
    %   ___                        _   _
    %  | _ \_ _ ___ _ __  ___ _ _| |_(_)___ ___
    %  |  _/ '_/ _ \ '_ \/ -_) '_|  _| / -_|_-<
    %  |_| |_| \___/ .__/\___|_|  \__|_\___/__/
    %              |_|
    %
    properties
        tmn; %the cellularGPSTrackingManual_object
        imag3;
        image_width;
        image_height;
        gui_main;

        trackLine
        trackCircle
        trackCenRow
        trackCenCol
        trackCenLogical
        trackCircleSize
        trackColorHighlight2 = [0.301,0.745,0.933];
        trackLineWidthHighlight = 3;
        trackCenLogicalDiff
        trackColor = [0.85,0.325,0.098;0.494,0.184,0.556;0.466,0.674,0.188];
        trackColorHighlight = [0.929,0.694,0.125];
        trackText
        trackTextBackgroundColor = [240 255 240]/255;
        trackTextColor = [47 79 79]/255;
        trackTextFontSize = 9;
        trackTextMargin = 1;
        trackJoinBool = false;

        makecellMotherBool = false;
    end
    %% Methods
    %   __  __     _   _            _
    %  |  \/  |___| |_| |_  ___  __| |___
    %  | |\/| / -_)  _| ' \/ _ \/ _` (_-<
    %  |_|  |_\___|\__|_||_\___/\__,_/__/
    %
    methods
        %% The first method is the constructor
        %    ___             _               _
        %   / __|___ _ _  __| |_ _ _ _  _ __| |_ ___ _ _
        %  | (__/ _ \ ' \(_-<  _| '_| || / _|  _/ _ \ '_|
        %   \___\___/_||_/__/\__|_|  \_,_\__|\__\___/_|
        %
        %
        function obj = cellularGPSTrackingManual_object_imageViewer(tmn)
            %%%
            % parse the input
            q = inputParser;
            addRequired(q, 'tmn', @(x) isa(x,'cellularGPSTrackingManual_object'));
            parse(q,tmn);
            %%
            %
            obj.tmn = q.Results.tmn;
            obj.imag3 = imread(fullfile(tmn.moviePath,'.thumb',tmn.smda_databaseSubset.filename{tmn.
 indImage}));
            obj.image_width = size(obj.imag3,2);
            obj.image_height = size(obj.imag3,1);
            %% Create a gui to enable pausing and stopping
            %    ___ _   _ ___    ___              _   _
            %   / __| | | |_ _|  / __|_ _ ___ __ _| |_(_)___ _ _
            %  | (_ | |_| || |  | (__| '_/ -_) _` |  _| / _ \ ' \
            %   \___|\___/|___|  \___|_| \___\__,_|\__|_\___/_||_|
            %   / _|___ _ _
            %  |  _/ _ \ '_|  __  __      _
            %  |_| \___/_(_) |  \/  |__ _(_)_ _
            %  / _` | || | | | |\/| / _` | | ' \
            %  \__, |\_,_|_|_|_|  |_\__,_|_|_||_|
            %  |___/       |___|
            % Create the figure
            %
            myunits = get(0,'units');
            set(0,'units','pixels');
            Pix_SS = get(0,'screensize');
            set(0,'units','characters');
            Char_SS = get(0,'screensize');
            ppChar = Pix_SS./Char_SS;
            ppChar = ppChar([3,4]);
            set(0,'units',myunits);

            if obj.image_width > obj.image_height
                if obj.image_width/obj.image_height >= Pix_SS(3)/Pix_SS(4)
                    fwidth = 0.9*Pix_SS(3);
                    fheight = fwidth*obj.image_height/obj.image_width;
                else
                    fheight = 0.9*Pix_SS(4);
                    fwidth = fheight*obj.image_width/obj.image_height;
                end
            else
                if obj.image_height/obj.image_width >= Pix_SS(4)/Pix_SS(3)
                    fheight = 0.9*Pix_SS(4);
                    fwidth = fheight*obj.image_width/obj.image_height;
                else
                    fwidth = 0.9*Pix_SS(3);
                    fheight = fwidth*obj.image_height/obj.image_width;

                end
            end

            fwidth = fwidth/ppChar(1);
            fheight = fheight/ppChar(2);

            f = figure('Visible','off','Units','characters','MenuBar','none',...
                'Resize','off','Name','Image Viewer',...
                'Renderer','OpenGL','Position',[(Char_SS(3)-fwidth)/2 (Char_SS(4)-fheight)/2 fwidth
    fheight],...
                'CloseRequestFcn',{@obj.delete},...
                'KeyPressFcn',{@obj.fKeyPressFcn});

            axesImageViewer = axes('Parent',f,...
                'Units','characters',...
                'Position',[0 0 fwidth  fheight],...
                'YDir','reverse',...
                'Visible','on',...
                'XLim',[0.5,obj.image_width+0.5],...
                'YLim',[0.5,obj.image_height+0.5]); %when displaying images the center of the pixels
     are located at the position on the axis. Therefore, the limits must account for the half
    pixel border.

            %% Visuals for Tracks
            %  __   ___               _    _ _
            %  \ \ / (_)____  _ __ _| |___ | | |
            %   \ V /| (_-< || / _` | (_-< |_  _|
            %   _\_/_|_/__/\_,_\__,_|_/__/   |_|
            %  |_   _| _ __ _ __| |__ ___
            %    | || '_/ _` / _| / /(_-<
            %    |_||_| \__,_\__|_\_\/__/
            %
            %% Create an axes to hold these visuals
            % highlighted cell with hover haxesHighlight =
            % axes('Units','characters','DrawMode','fast','color','none',...
            %     'Position',[hx hy hwidth hheight],...
            %     'XLim',[1,master.image_width],'YLim',[1,master.image_height]);
            % cmapHighlight = colormap(haxesImageViewer,jet(16)); %63 matches the number of elements
     in ang
            axesTracks = axes('Parent',f,'Units','characters',...
                'Position',[0 0 fwidth  fheight]);
            axesTracks.NextPlot = 'add';
            axesTracks.Visible = 'off';
            axesTracks.YDir = 'reverse';

            axesText = axes('Parent',f,'Units','characters',...
                'Position',[0 0 fwidth  fheight]);
            axesText.NextPlot = 'add';
            axesText.Visible = 'off';
            axesText.YDir = 'reverse';

            axesCircles = axes('Parent',f,'Units','characters',...
                'Position',[0 0 fwidth  fheight]);
            axesCircles.NextPlot = 'add';
            axesCircles.Visible = 'off';
            axesCircles.YDir = 'reverse';

            obj.trackLine = {};
            obj.trackCircle = {};
            obj.trackText = {};
            obj.trackCircleSize = 11; %must be an odd number

            displayedImage = image('Parent',axesImageViewer,...
                'CData',obj.imag3);
            %% Handles
            %   _  _              _ _
            %  | || |__ _ _ _  __| | |___ ___
            %  | __ / _` | ' \/ _` | / -_|_-<
            %  |_||_\__,_|_||_\__,_|_\___/__/
            %
            % store the uicontrol handles in the figure handles via guidata()
            handles.axesTracks = axesTracks;
            handles.axesCircles = axesCircles;
            handles.axesText = axesText;
            handles.axesImageViewer = axesImageViewer;
            handles.displayedImage = displayedImage;
            obj.gui_main = f;
            guidata(f,handles);
            %% Execute just before the figure becomes visible
            %     _            _     ___ _ _
            %   _ | |_  _ __| |_  | _ ) | |
            %  | || | || (_-<  _| | _ \_  _|
            %  _\__/_\_,_/__/\__| |___/ |_|
            %  \ \ / (_)__(_) |__| |___
            %   \ V /| (_-< | '_ \ / -_)
            %    \_/ |_/__/_|_.__/_\___|
            %
            % The code above organizes and specifies the elements of the figure and
            % gui. The code below may simple store these elements into the handles
            % struct and make the gui visible for the first time. Other commands or
            % functions can also be executed here if certain variables or parameters
            % need to be computed and set.
            obj.updateLimits;
            %%%
            % make the gui visible
            set(f,'Visible','on');
        end
        %% delete
        % for a clean delete make sure the objects that are stored as
        % properties are also deleted.
        function delete(obj,~,~)
            delete(obj.gui_main);
        end
        %%
        %
        function obj = fKeyPressFcn(obj,~,keyInfo)
            switch keyInfo.Key
                case 'period'
                    obj.tmn.indImage = obj.tmn.indImage + 1;
                    if obj.tmn.indImage > height(obj.tmn.smda_databaseSubset)
                        obj.tmn.indImage = height(obj.tmn.smda_databaseSubset);
                        return
                    end
                    handlesControl = guidata(obj.tmn.gui_control.gui_main);
                    handlesControl.infoBk_editTimepoint.String = num2str(obj.tmn.indImage);
                    guidata(obj.tmn.gui_control.gui_main,handlesControl);
                    obj.loop_stepRight;
                case 'comma'
                    obj.tmn.indImage = obj.tmn.indImage - 1;
                    if obj.tmn.indImage < 1
                        obj.tmn.indImage = 1;
                        return
                    end
                    handlesControl = guidata(obj.tmn.gui_control.gui_main);
                    handlesControl.infoBk_editTimepoint.String = num2str(obj.tmn.indImage);
                    guidata(obj.tmn.gui_control.gui_main,handlesControl);
                    obj.loop_stepLeft;
                case 'hyphen'
                    %% delete a track
                    %
                    obj.tmn.makecell_mode = 'delete';
                    handlesControl = guidata(obj.tmn.gui_control.gui_main);
                    handlesControl.tabMakeCell_togglebuttonDelete.Value = 1;
                    obj.tmn.gui_control.tabMakeCell_buttongroup_SelectionChangedFcn;
                    guidata(obj.tmn.gui_control.gui_main,handlesControl);
                case 'rightarrow'

                case 'leftarrow'

                case 'downarrow'

                case 'uparrow'

                case 'backspace'

                case 'd'
                    %% timepoint at end of track
                    %
                    oldIndImage = obj.tmn.indImage;
                    obj.tmn.indImage = find(obj.trackCenLogical(obj.tmn.mcl.pointer_track,:),1,'last
    ');
                    firstInd =  find(obj.trackCenLogical(obj.tmn.mcl.pointer_track,:),1,'first');
                    if oldIndImage >= obj.tmn.indImage
                        obj.tmn.indImage = oldIndImage + 1;
                        if obj.tmn.indImage > height(obj.tmn.smda_databaseSubset)
                            obj.tmn.indImage = height(obj.tmn.smda_databaseSubset);
                            return
                        end
                        handlesControl = guidata(obj.tmn.gui_control.gui_main);
                        handlesControl.infoBk_editTimepoint.String = num2str(obj.tmn.indImage);
                        guidata(obj.tmn.gui_control.gui_main,handlesControl);
                        obj.loop_stepRight;
                    elseif oldIndImage < firstInd
                        obj.tmn.indImage = firstInd;
                        handlesControl = guidata(obj.tmn.gui_control.gui_main);
                        handlesControl.infoBk_editTimepoint.String = num2str(obj.tmn.indImage);
                        guidata(obj.tmn.gui_control.gui_main,handlesControl);
                        obj.loop_stepX;
                    else
                        handlesControl = guidata(obj.tmn.gui_control.gui_main);
                        handlesControl.infoBk_editTimepoint.String = num2str(obj.tmn.indImage);
                        guidata(obj.tmn.gui_control.gui_main,handlesControl);
                        obj.loop_stepX;
                    end
                case 'a'
                    %% timepoint at start of track
                    %
                    oldIndImage = obj.tmn.indImage;
                    obj.tmn.indImage = find(obj.trackCenLogical(obj.tmn.mcl.pointer_track,:),1,'
first');
                    lastInd = find(obj.trackCenLogical(obj.tmn.mcl.pointer_track,:),1,'last');
                    if oldIndImage <= obj.tmn.indImage
                        obj.tmn.indImage = oldIndImage - 1;
                        if obj.tmn.indImage < 1
                            obj.tmn.indImage = 1;
                            return
                        end
                        handlesControl = guidata(obj.tmn.gui_control.gui_main);
                        handlesControl.infoBk_editTimepoint.String = num2str(obj.tmn.indImage);
                        guidata(obj.tmn.gui_control.gui_main,handlesControl);
                        obj.loop_stepLeft;
                    elseif oldIndImage > lastInd
                        obj.tmn.indImage = lastInd;
                        handlesControl = guidata(obj.tmn.gui_control.gui_main);
                        handlesControl.infoBk_editTimepoint.String = num2str(obj.tmn.indImage);
                        guidata(obj.tmn.gui_control.gui_main,handlesControl);
                        obj.loop_stepX;
                    else
                        handlesControl = guidata(obj.tmn.gui_control.gui_main);
                        handlesControl.infoBk_editTimepoint.String = num2str(obj.tmn.indImage);
                        guidata(obj.tmn.gui_control.gui_main,handlesControl);
                        obj.loop_stepX;
                    end
                case 'b'
                    %% break a track into two tracks
                    %
                    obj.tmn.makecell_mode = 'break';
                    handlesControl = guidata(obj.tmn.gui_control.gui_main);
                    handlesControl.tabMakeCell_togglebuttonBreak.Value = 1;
                    obj.tmn.gui_control.tabMakeCell_buttongroup_SelectionChangedFcn;
                    guidata(obj.tmn.gui_control.gui_main,handlesControl);
                case 'c'
                    %% create a new cell
                    %
                    obj.tmn.gui_control.tabMakeCell_pushbuttonNewCell_Callback;
                    obj.tmn.mcl.pointer_makecell3 = obj.tmn.mcl.pointer_makecell;
                case 'j'
                    %% join two tracks
                    %
                    obj.tmn.makecell_mode = 'join';
                    handlesControl = guidata(obj.tmn.gui_control.gui_main);
                    handlesControl.tabMakeCell_togglebuttonJoin.Value = 1;
                    obj.tmn.gui_control.tabMakeCell_buttongroup_SelectionChangedFcn;
                    guidata(obj.tmn.gui_control.gui_main,handlesControl);
                case 'n'
                    %% do nothing
                    %
                    obj.tmn.makecell_mode = 'none';
                    handlesControl = guidata(obj.tmn.gui_control.gui_main);
                    handlesControl.tabMakeCell_togglebuttonNone.Value = 1;
                    obj.tmn.gui_control.tabMakeCell_buttongroup_SelectionChangedFcn;
                    guidata(obj.tmn.gui_control.gui_main,handlesControl);
                case 'm'
                    %% chose mother cell
                    %
                    obj.tmn.makecell_mode = 'mother';
                    handlesControl = guidata(obj.tmn.gui_control.gui_main);
                    handlesControl.tabMakeCell_togglebuttonMother.Value = 1;
                    obj.tmn.gui_control.tabMakeCell_buttongroup_SelectionChangedFcn;
                    guidata(obj.tmn.gui_control.gui_main,handlesControl);
                case 't'
                    %% add a track to a cell
                    %
                    obj.tmn.makecell_mode = 'track 2 cell';
                    handlesControl = guidata(obj.tmn.gui_control.gui_main);
                    handlesControl.tabMakeCell_togglebuttonAddTrack2Cell.Value = 1;
                    obj.tmn.gui_control.tabMakeCell_buttongroup_SelectionChangedFcn;
                    guidata(obj.tmn.gui_control.gui_main,handlesControl);
                case 'escape'
                    %% reset conditional properties
                    %
                    obj.trackJoinBool = false;
                    obj.makecellMotherBool = false;
                    obj.tmn.makecell_mode = 'none';
                    handlesControl = guidata(obj.tmn.gui_control.gui_main);
                    handlesControl.tabMakeCell_togglebuttonNone.Value = 1;
                    obj.tmn.gui_control.tabMakeCell_buttongroup_SelectionChangedFcn;
                    handlesControl.infoBk_textMessage.String = sprintf('Aborted! System is reset.');
                    guidata(obj.tmn.gui_control.gui_main,handlesControl);
            end
        end
        %%
        %
        function obj = updateLimits(obj)
            handles = guidata(obj.gui_main);

            handles.axesTracks.YLim = [1,obj.tmn.ity.imageHeightNoBin/...
                obj.tmn.ity.settings_binning(obj.tmn.indS)];
            handles.axesTracks.XLim = [1,obj.tmn.ity.imageWidthNoBin/...
                obj.tmn.ity.settings_binning(obj.tmn.indS)];

            handles.axesCircles.YLim = [1,obj.tmn.ity.imageHeightNoBin/...
                obj.tmn.ity.settings_binning(obj.tmn.indS)];
            handles.axesCircles.XLim = [1,obj.tmn.ity.imageWidthNoBin/...
                obj.tmn.ity.settings_binning(obj.tmn.indS)];

            handles.axesText.YLim = [1,obj.tmn.ity.imageHeightNoBin/...
                obj.tmn.ity.settings_binning(obj.tmn.indS)];
            handles.axesText.XLim = [1,obj.tmn.ity.imageWidthNoBin/...
                obj.tmn.ity.settings_binning(obj.tmn.indS)];

            guidata(obj.gui_main,handles);
        end
        %%
        %
        function obj = loadNewTracks(obj)
            handles = guidata(obj.gui_main);
            handlesControl = guidata(obj.tmn.gui_control.gui_main);
            handlesControl.infoBk_textMessage.String = sprintf('Loading new tracks...');
            drawnow;
            %%
            % process centroid data
            obj.tmn.mcl.import(obj.tmn.indP);
            obj.tmn.mcl.moviePath = obj.tmn.moviePath;
            mydatabase = obj.tmn.mcl.track_database;
            numOfT = obj.tmn.ity.number_of_timepoints;
            myCenRow = zeros(max(mydatabase.trackID),numOfT);
            myCenCol = zeros(max(mydatabase.trackID),numOfT);
            myCenLogical = false(size(myCenRow));
            handlesControl.infoBk_textMessage.String = sprintf('Tracks identified with\n%d centroids
',height(mydatabase));
            drawnow;
            for v = 1:height(mydatabase)
                mytimepoint = mydatabase.timepoint(v);
                mytrackID = mydatabase.trackID(v);
                myCenRow(mytrackID,mytimepoint) = mydatabase.centroid_row(v);
                myCenCol(mytrackID,mytimepoint) = mydatabase.centroid_col(v);
                myCenLogical(mytrackID,mytimepoint) = true;
            end
            %%%
            % Assignment to the object was required to be after the parfor.
            obj.trackCenRow = myCenRow;
            obj.trackCenCol = myCenCol;
            obj.trackCenLogical = myCenLogical;

            obj.trackCenLogicalDiff = diff(obj.trackCenLogical,1,2);

            %% Recalculate tracks
            % Assumes image size remains the same for this settings
            for i = 1:length(obj.trackCircle)
```

```
                if isa(obj.trackCircle{i},'matlab.graphics.primitive.Rectangle')
                    delete(obj.trackCircle{i});
                end
                if isa(obj.trackLine{i},'matlab.graphics.primitive.Line')
                    delete(obj.trackLine{i});
                end
                if isa(obj.trackText{i},'matlab.graphics.primitive.Text')
                    delete(obj.trackText{i});
                end
            end
            mydatabase1 = obj.tmn.track_database{obj.tmn.indP};
            obj.trackLine = cell(max(mydatabase1.trackID),1);
            obj.trackCircle = cell(max(mydatabase1.trackID),1);
            obj.trackText = cell(max(mydatabase1.trackID),1);
            handlesControl.infoBk_textMessage.String = sprintf('Importing Tracks...');
            drawnow;
            for i = 1:length(obj.trackLine)
                if ~any(obj.trackCenLogical(i,:))
                    continue
                end
                myline = line('Parent',handles.axesTracks);
                myline.Color = obj.trackColor(mod(i,3)+1,:);
                myline.LineWidth = 1;
                myline.YData = obj.trackCenRow(i,obj.trackCenLogical(i,:));
                myline.XData = obj.trackCenCol(i,obj.trackCenLogical(i,:));
                obj.trackLine{i} = myline;
            end
            handlesControl.infoBk_textMessage.String = sprintf('Importing Circles...');
            drawnow;
            for i = 1:length(obj.trackCircle)
                if ~any(obj.trackCenLogical(i,:))
                    continue
                end
                myrec = rectangle('Parent',handles.axesCircles);
                myrec.ButtonDownFcn = @obj.clickLoop;
                myrec.UserData = i;
                myrec.Curvature = [1,1];
                myrec.FaceColor = obj.trackLine{i}.Color;
                myrec.Position = [obj.trackLine{i}.XData(1)-(obj.trackCircleSize-1)/2,obj.trackLine{
      i}.YData(1)-(obj.trackCircleSize-1)/2,obj.trackCircleSize,obj.trackCircleSize];
                mclID = obj.tmn.mcl.track_makecell(i);
                if mclID ~= 0
                    myrec.EdgeColor = obj.trackColorHighlight2;
                    myrec.LineWidth = 2;
                else
                    myrec.EdgeColor = [0,0,0];
                    myrec.LineWidth = 0.5;
                end
                obj.trackCircle{i} = myrec;
            end
            handlesControl.infoBk_textMessage.String = sprintf('Transcribing Text...');
            drawnow;
            for i = 1:length(obj.trackText)
                if ~any(obj.trackCenLogical(i,:))
                    continue
                end
                obj.trackText{i} = text('Parent',handles.axesText);
                obj.updateTrackText(i);
            end
            handlesControl.infoBk_textMessage.String = sprintf('Position %d',obj.tmn.indP);
            drawnow;
            obj.loop_stepX;
            obj.tmn.gui_control.tabGPS_loop;
            obj.tmn.gui_control.tabMakeCell_loop;
            guidata(obj.tmn.gui_control.gui_main,handlesControl);
        end
        %%
        %
        function obj = visualizeTracks(obj)
            handles = guidata(obj.gui_main);
            handlesControl = guidata(obj.tmn.gui_control.gui_main);
            %% Recalculate tracks
            % Assumes image size remains the same for this settings
            cellfun(@delete,obj.trackCircle);
            cellfun(@delete,obj.trackLine);
            obj.trackLine = cell(max(obj.tmn.mcl.track_database.trackID),1);
            obj.trackCircle = cell(max(obj.tmn.mcl.track_database.trackID),1);
            handlesControl.infoBk_textMessage.String = sprintf('Importing Tracks...');
            drawnow;
            for i = 1:length(obj.trackLine)
                if ~obj.tmn.mcl.track_logical(i)
                    continue
                end
                myline = line('Parent',handles.axesTracks);
                myline.Color = obj.trackColor(mod(i,3)+1,:);
                myline.LineWidth = 1;
                myline.YData = obj.trackCenRow(i,obj.trackCenLogical(i,:));
                myline.XData = obj.trackCenCol(i,obj.trackCenLogical(i,:));
                obj.trackLine{i} = myline;
            end
            handlesControl.infoBk_textMessage.String = sprintf('Importing Circles...');
            drawnow;
            for i = 1:length(obj.trackCircle)
                if ~obj.tmn.mcl.track_logical(i)
                    continue
                end
                myrec = rectangle('Parent',handles.axesCircles);
                myrec.ButtonDownFcn = @obj.clickLoop;
                myrec.UserData = i;
                myrec.Curvature = [1,1];
                myrec.FaceColor = obj.trackLine{i}.Color;
                myrec.Position = [obj.trackLine{i}.XData(1)-(obj.trackCircleSize-1)/2,obj.trackLine{
      i}.YData(1)-(obj.trackCircleSize-1)/2,obj.trackCircleSize,obj.trackCircleSize];
                obj.trackCircle{i} = myrec;
            end
            handlesControl.infoBk_textMessage.String = sprintf('Position %d',obj.tmn.indP);
            drawnow;
            guidata(obj.tmn.gui_control.gui_main,handlesControl);
            obj.loop_stepX;
        end
        %%
        %
        function obj = loop_stepX(obj)
            handles = guidata(obj.gui_main);
            obj.imag3 = imread(fullfile(obj.tmn.moviePath,'.thumb',obj.tmn.smda_databaseSubset.
      filename{obj.tmn.indImage}));
            handles.displayedImage.CData = obj.imag3;
            obj.updateLimits;
            guidata(obj.gui_main,handles);

            %%%
            %   _____            _   __   ___
            %  |_   _| _ __ _ __| |__ \ \ / (_)___
            %    | ||  '_/ _` / _| / /  \ V /| (_-<
            %    |_||_| \__,_\__|_\_\   \_/ |_/__/
            %
            if obj.tmn.gui_control.menu_viewTrackBool
                switch obj.tmn.gui_control.menu_viewTime
                    case 'all'
                        trackCircleHalfSize = (obj.trackCircleSize-1)/2;
                        for i = 1:length(obj.trackCircle)
                            if ~obj.tmn.mcl.track_logical(i)
                                continue
                            end
                            obj.trackLine{i}.Visible = 'on';
                            if obj.trackCenLogical(i,obj.tmn.indImage)
                                obj.trackText{i}.Visible = 'on';
                                obj.trackText{i}.Position = [obj.trackCenCol(i,obj.tmn.indImage)+
      trackCircleHalfSize,...
                                    obj.trackCenRow(i,obj.tmn.indImage)+trackCircleHalfSize];
                                obj.trackCircle{i}.Visible = 'on';
                                obj.trackCircle{i}.Position = [obj.trackCenCol(i,obj.tmn.indImage)-
      trackCircleHalfSize,...
                                    obj.trackCenRow(i,obj.tmn.indImage)-trackCircleHalfSize,...
                                    obj.trackCircleSize,obj.trackCircleSize];
                            else
                                obj.trackText{i}.Visible = 'off';
                                obj.trackCircle{i}.Visible = 'off';
                            end
                        end
                    case 'now'
                        trackCircleHalfSize = (obj.trackCircleSize-1)/2;
                        for i = 1:length(obj.trackCircle)
                            if ~obj.tmn.mcl.track_logical(i)
                                continue
                            end
```

```
                        if obj.trackCenLogical(i,obj.tmn.indImage)
                            obj.trackLine{i}.Visible = 'on';
                            obj.trackText{i}.Visible = 'on';
                            obj.trackText{i}.Position = [obj.trackCenCol(i,obj.tmn.indImage)+trackCircleHalfSize,...
                                obj.trackCenRow(i,obj.tmn.indImage)+trackCircleHalfSize];
                            obj.trackCircle{i}.Visible = 'on';
                            obj.trackCircle{i}.Position = [obj.trackCenCol(i,obj.tmn.indImage)-trackCircleHalfSize,...
                                obj.trackCenRow(i,obj.tmn.indImage)-trackCircleHalfSize,...
                                obj.trackCircleSize,obj.trackCircleSize];
                        else
                            obj.trackLine{i}.Visible = 'off';
                            obj.trackText{i}.Visible = 'off';
                            obj.trackCircle{i}.Visible = 'off';
                        end
                    end
            end
        end
    end
    %%
    %
    function obj = loop_stepRight(obj)
        handles = guidata(obj.gui_main);
        obj.imag3 = imread(fullfile(obj.tmn.moviePath,'.thumb',obj.tmn.smda_databaseSubset.filename{obj.tmn.indImage}));
        handles.displayedImage.CData = obj.imag3;
        obj.updateLimits;
        guidata(obj.gui_main,handles);

        %%%
        %  _____            _    __   ___
        % |_   _| _ __ _ __| |__ \ \ / (_)___
        %   | || '_/ _` / _| / /  \ V /| (_-<
        %   |_||_| \__,_\__|_\_\   \_/ |_/__/
        %
        if obj.tmn.gui_control.menu_viewTrackBool
            switch obj.tmn.gui_control.menu_viewTime
                case 'all'
                    trackCircleHalfSize = (obj.trackCircleSize-1)/2;
                    for i = 1:length(obj.trackCircle)
                        if obj.trackCenLogicalDiff(i,obj.tmn.indImage-1) == 0 && ~obj.trackCenLogical(i,obj.tmn.indImage)
                            % do nothing
                        elseif obj.trackCenLogical(i,obj.tmn.indImage) && obj.trackCenLogicalDiff(i,obj.tmn.indImage-1) == 0

                            obj.trackText{i}.Position = [obj.trackCenCol(i,obj.tmn.indImage)+trackCircleHalfSize,...
                                obj.trackCenRow(i,obj.tmn.indImage)+trackCircleHalfSize];
                            obj.trackCircle{i}.Position = [obj.trackCenCol(i,obj.tmn.indImage)-trackCircleHalfSize,...
                                obj.trackCenRow(i,obj.tmn.indImage)-trackCircleHalfSize,...
                                obj.trackCircleSize,obj.trackCircleSize];
                        elseif obj.trackCenLogicalDiff(i,obj.tmn.indImage-1) == -1
                            obj.trackText{i}.Visible = 'off';
                            obj.trackCircle{i}.Visible = 'off';
                        else
                            obj.trackText{i}.Visible = 'on';
                            obj.trackText{i}.Position = [obj.trackCenCol(i,obj.tmn.indImage)+trackCircleHalfSize,...
                                obj.trackCenRow(i,obj.tmn.indImage)+trackCircleHalfSize];
                            obj.trackCircle{i}.Visible = 'on';
                            obj.trackCircle{i}.Position = [obj.trackCenCol(i,obj.tmn.indImage)-trackCircleHalfSize,...
                                obj.trackCenRow(i,obj.tmn.indImage)-trackCircleHalfSize,...
                                obj.trackCircleSize,obj.trackCircleSize];
                        end
                    end
                case 'now'
                    trackCircleHalfSize = (obj.trackCircleSize-1)/2;
                    for i = 1:length(obj.trackCircle)
                        if obj.trackCenLogicalDiff(i,obj.tmn.indImage-1) == 0 && ~obj.trackCenLogical(i,obj.tmn.indImage)
                            % do nothing
                        elseif obj.trackCenLogical(i,obj.tmn.indImage) && obj.trackCenLogicalDiff(i,obj.tmn.indImage-1) == 0
                            obj.trackText{i}.Position = [obj.trackCenCol(i,obj.tmn.indImage)+
trackCircleHalfSize ,...
                                obj.trackCenRow(i,obj.tmn.indImage)+trackCircleHalfSize];
                            obj.trackCircle{i}.Position = [obj.trackCenCol(i,obj.tmn.indImage)−
trackCircleHalfSize ,...
                                obj.trackCenRow(i,obj.tmn.indImage)−trackCircleHalfSize ,...
                                obj.trackCircleSize ,obj.trackCircleSize];
                        elseif obj.trackCenLogicalDiff(i,obj.tmn.indImage−1) == −1
                            obj.trackLine{i}.Visible = 'off';
                            obj.trackText{i}.Visible = 'off';
                            obj.trackCircle{i}.Visible = 'off';
                        else
                            obj.trackLine{i}.Visible = 'on';
                            obj.trackText{i}.Visible = 'on';
                            obj.trackText{i}.Position = [obj.trackCenCol(i,obj.tmn.indImage)+
trackCircleHalfSize ,...
                                obj.trackCenRow(i,obj.tmn.indImage)+trackCircleHalfSize];
                            obj.trackCircle{i}.Visible = 'on';
                            obj.trackCircle{i}.Position = [obj.trackCenCol(i,obj.tmn.indImage)−
trackCircleHalfSize ,...
                                obj.trackCenRow(i,obj.tmn.indImage)−trackCircleHalfSize ,...
                                obj.trackCircleSize ,obj.trackCircleSize];
                        end
                    end
                end
            end
        end
    end
    %%
    %
    function obj = loop_stepLeft(obj)
        handles = guidata(obj.gui_main);
        obj.imag3 = imread(fullfile(obj.tmn.moviePath,'.thumb',obj.tmn.smda_databaseSubset.
filename{obj.tmn.indImage}));
        handles.displayedImage.CData = obj.imag3;
        obj.updateLimits;
        guidata(obj.gui_main,handles);

        %%%
        %   _____               _     __     ___
        %  |_   _| _ __ _ __ __| |__  \ \   / (_)___
        %    | || '_/ _` / _| / /   \ V /| (_-<
        %    |_||_| \__,_\__|_\_\    \_/ |_/__/
        %
        if obj.tmn.gui_control.menu_viewTrackBool
            switch obj.tmn.gui_control.menu_viewTime
                case 'all'
                    trackCircleHalfSize = (obj.trackCircleSize−1)/2;
                    for i = 1:length(obj.trackCircle)
                        if obj.trackCenLogicalDiff(i,obj.tmn.indImage) == 0 && ~obj.
trackCenLogical(i,obj.tmn.indImage)
                            %do nothing
                        elseif obj.trackCenLogical(i,obj.tmn.indImage) && obj.
trackCenLogicalDiff(i,obj.tmn.indImage) == 0
                            obj.trackText{i}.Position = [obj.trackCenCol(i,obj.tmn.indImage)+
trackCircleHalfSize ,...
                                obj.trackCenRow(i,obj.tmn.indImage)+trackCircleHalfSize];
                            obj.trackCircle{i}.Position = [obj.trackCenCol(i,obj.tmn.indImage)−
trackCircleHalfSize ,...
                                obj.trackCenRow(i,obj.tmn.indImage)−trackCircleHalfSize ,...
                                obj.trackCircleSize ,obj.trackCircleSize];
                        elseif obj.trackCenLogicalDiff(i,obj.tmn.indImage) == 1
                            obj.trackText{i}.Visible = 'off';
                            obj.trackCircle{i}.Visible = 'off';
                        else
                            obj.trackText{i}.Visible = 'on';
                            obj.trackText{i}.Position = [obj.trackCenCol(i,obj.tmn.indImage)+
trackCircleHalfSize ,...
                                obj.trackCenRow(i,obj.tmn.indImage)+trackCircleHalfSize];
                            obj.trackCircle{i}.Visible = 'on';
                            obj.trackCircle{i}.Position = [obj.trackCenCol(i,obj.tmn.indImage)−
trackCircleHalfSize ,...
                                obj.trackCenRow(i,obj.tmn.indImage)−trackCircleHalfSize ,...
                                obj.trackCircleSize ,obj.trackCircleSize];
                        end
                    end
                case 'now'
                    trackCircleHalfSize = (obj.trackCircleSize−1)/2;
                    for i = 1:length(obj.trackCircle)
                        if obj.trackCenLogicalDiff(i,obj.tmn.indImage) == 0 && ~obj.
```

```
trackCenLogical(i,obj.tmn.indImage)
                          %do nothing
                      elseif obj.trackCenLogical(i,obj.tmn.indImage) && obj.
trackCenLogicalDiff(i,obj.tmn.indImage) == 0
                          obj.trackText{i}.Position = [obj.trackCenCol(i,obj.tmn.indImage)+
trackCircleHalfSize,...
                              obj.trackCenRow(i,obj.tmn.indImage)+trackCircleHalfSize];
                          obj.trackCircle{i}.Position = [obj.trackCenCol(i,obj.tmn.indImage)-
trackCircleHalfSize,...
                              obj.trackCenRow(i,obj.tmn.indImage)-trackCircleHalfSize,...
                              obj.trackCircleSize,obj.trackCircleSize];
                      elseif obj.trackCenLogicalDiff(i,obj.tmn.indImage) == 1
                          obj.trackLine{i}.Visible = 'off';
                          obj.trackText{i}.Visible = 'off';
                          obj.trackCircle{i}.Visible = 'off';
                      else
                          obj.trackLine{i}.Visible = 'on';
                          obj.trackText{i}.Visible = 'on';
                          obj.trackText{i}.Position = [obj.trackCenCol(i,obj.tmn.indImage)+
trackCircleHalfSize,...
                              obj.trackCenRow(i,obj.tmn.indImage)+trackCircleHalfSize];
                          obj.trackCircle{i}.Visible = 'on';
                          obj.trackCircle{i}.Position = [obj.trackCenCol(i,obj.tmn.indImage)-
trackCircleHalfSize,...
                              obj.trackCenRow(i,obj.tmn.indImage)-trackCircleHalfSize,...
                              obj.trackCircleSize,obj.trackCircleSize];
                      end
                  end
          end
      end
  end
  %%
  %
  function obj = clickLoop(obj,myrec,~)
      %%%
      %    _____               _    __    ___
      %   |_    _| _ __ _ __| |__ \ \ / (_)___
      %     | ||  '_/ _` / _|  / /   \ V /| (_-<
      %     |_||_| \__,_\__|_\_\    \_/ |_/__/
      %
      if obj.tmn.gui_control.menu_viewTrackBool
          %%%
          % if the menu_viewTrackBool is true, then tracks are
          % displayed
          obj.tmn.mcl.pointer_track2 = obj.tmn.mcl.pointer_track;
          obj.tmn.mcl.pointer_track = myrec.UserData;
          obj.tmn.mcl.pointer_makecell2 = obj.tmn.mcl.pointer_makecell;
          obj.tmn.mcl.pointer_makecell = obj.tmn.mcl.track_makecell(obj.tmn.mcl.pointer_track)
;
          %% highlight
          obj.highlightTrack;
          handlesControl = guidata(obj.tmn.gui_control.gui_main);
          %% track edits
          %
          switch obj.tmn.makecell_mode
              case 'none'
                  handlesControl.infoBk_textMessage.String = sprintf('track ID %d\nmakecell ID
 %d',obj.tmn.mcl.pointer_track,obj.tmn.mcl.pointer_makecell);
              case 'join'
                  if obj.trackJoinBool
                      if obj.tmn.mcl.pointer_track2 > obj.tmn.mcl.pointer_track
                          keepTrack = obj.tmn.mcl.pointer_track;
                          replaceTrack = obj.tmn.mcl.pointer_track2;
                      else
                          keepTrack = obj.tmn.mcl.pointer_track2;
                          replaceTrack = obj.tmn.mcl.pointer_track;
                      end
                      obj.tmn.mcl.joinTrack(keepTrack,replaceTrack);
                      obj.trackJoinBool = false;
                      myLogical = ismember(obj.tmn.mcl.track_database.trackID,[keepTrack,
replaceTrack]);
                      myArray = 1:numel(myLogical);
                      myArray = myArray(myLogical);
                      obj.trackCenRow(keepTrack,:) = 0;
                      obj.trackCenCol(keepTrack,:) = 0;
                      obj.trackCenLogical(keepTrack,:) = false;
                      obj.trackCenRow(replaceTrack,:) = 0;
                      obj.trackCenCol(replaceTrack,:) = 0;
```

```
                            obj.trackCenLogical(replaceTrack,:) = false;
                            for v = myArray
                                mytimepoint = obj.tmn.mcl.track_database.timepoint(v);
                                mytrackID = obj.tmn.mcl.track_database.trackID(v);
                                obj.trackCenRow(mytrackID,mytimepoint) = obj.tmn.mcl.track_database.
centroid_row(v);
                                obj.trackCenCol(mytrackID,mytimepoint) = obj.tmn.mcl.track_database.
centroid_col(v);
                                obj.trackCenLogical(mytrackID,mytimepoint) = true;
                            end
                            obj.trackCenLogicalDiff = diff(obj.trackCenLogical,1,2);

                            obj.trackLine{replaceTrack}.Visible = 'off';
                            obj.trackCircle{replaceTrack}.Visible = 'off';
                            obj.trackText{replaceTrack}.Visible = 'off';

                            obj.trackLine{keepTrack}.YData = obj.trackCenRow(keepTrack,obj.
trackCenLogical(keepTrack,:));
                            obj.trackLine{keepTrack}.XData = obj.trackCenCol(keepTrack,obj.
trackCenLogical(keepTrack,:));
                            obj.trackCircle{keepTrack}.Position = [obj.trackLine{keepTrack}.XData(1)
-(obj.trackCircleSize-1)/2,obj.trackLine{keepTrack}.YData(1)-(obj.trackCircleSize-1)/2,obj.
trackCircleSize,obj.trackCircleSize];
                            obj.trackText{keepTrack}.Position = [obj.trackLine{keepTrack}.XData(1)+(
obj.trackCircleSize-1)/2,obj.trackLine{keepTrack}.YData(1)+(obj.trackCircleSize-1)/2];
                            handlesControl.infoBk_textMessage.String = sprintf('Joined track %d with
\ntrack %d.',keepTrack,replaceTrack);
                            %%
                            % return to 'none' mode
                            handlesControl = guidata(obj.tmn.gui_control.gui_main);
                            handlesControl.tabMakeCell_togglebuttonNone.Value = 1;
                            obj.tmn.gui_control.tabMakeCell_buttongroup_SelectionChangedFcn;
                            guidata(obj.tmn.gui_control.gui_main,handlesControl);
                        else
                            handlesControl.infoBk_textMessage.String = sprintf('Join track %d with
...',obj.tmn.mcl.pointer_track);
                            obj.trackJoinBool = true;
                        end
                        obj.tmn.gui_control.tabMakeCell_loop;
                        obj.loop_stepX;
                    case 'break'
                        oldTrack = obj.tmn.mcl.pointer_track;
                        obj.tmn.mcl.breakTrack(obj.tmn.mcl.pointer_track,obj.tmn.indImage);
                        newTrack = obj.tmn.mcl.pointer_track;
                        obj.tmn.mcl.pointer_track = oldTrack;

                        myLogical = ismember(obj.tmn.mcl.track_database.trackID,[oldTrack,newTrack])
;
                        myArray = 1:numel(myLogical);
                        myArray = myArray(myLogical);
                        obj.trackCenRow(oldTrack,:) = 0;
                        obj.trackCenCol(oldTrack,:) = 0;
                        obj.trackCenLogical(oldTrack,:) = false;
                        obj.trackCenRow(newTrack,:) = 0;
                        obj.trackCenCol(newTrack,:) = 0;
                        obj.trackCenLogical(newTrack,:) = false;
                        for v = myArray
                            mytimepoint = obj.tmn.mcl.track_database.timepoint(v);
                            mytrackID = obj.tmn.mcl.track_database.trackID(v);
                            obj.trackCenRow(mytrackID,mytimepoint) = obj.tmn.mcl.track_database.
centroid_row(v);
                            obj.trackCenCol(mytrackID,mytimepoint) = obj.tmn.mcl.track_database.
centroid_col(v);
                            obj.trackCenLogical(mytrackID,mytimepoint) = true;
                        end
                        obj.trackCenLogicalDiff = diff(obj.trackCenLogical,1,2);

                        handles = guidata(obj.gui_main);
                        if newTrack > numel(obj.trackLine)
                            myline = line('Parent',handles.axesTracks);
                            myline.Color = obj.trackColor(mod(newTrack,3)+1,:);
                            myline.LineWidth = 1;
                            myline.YData = obj.trackCenRow(newTrack,obj.trackCenLogical(newTrack,:))
;
                            myline.XData = obj.trackCenCol(newTrack,obj.trackCenLogical(newTrack,:))
;
                            obj.trackLine{newTrack} = myline;
                            myrec = rectangle('Parent',handles.axesCircles);
                            myrec.ButtonDownFcn = @obj.clickLoop;
                            myrec.UserData = newTrack;
                            myrec.Curvature = [1,1];
                            myrec.FaceColor = obj.trackLine{newTrack}.Color;
                            myrec.Position = [obj.trackLine{newTrack}.XData(1)-(obj.trackCircleSize
-1)/2,obj.trackLine{newTrack}.YData(1)-(obj.trackCircleSize-1)/2,obj.trackCircleSize,obj.
trackCircleSize];
                            obj.trackCircle{newTrack} = myrec;

                            obj.trackText{newTrack} = text('Parent',handles.axesText);
                            obj.updateTrackText(newTrack);
                            obj.trackText{newTrack}.Position = [obj.trackLine{newTrack}.XData(1)+(
obj.trackCircleSize-1)/2,obj.trackLine{newTrack}.YData(1)+(obj.trackCircleSize-1)/2];
                        else
                            if isa(obj.trackLine{newTrack},'matlab.graphics.primitive.Line');
                                obj.trackLine{newTrack}.YData = obj.trackCenRow(newTrack,obj.
trackCenLogical(newTrack,:));
                                obj.trackLine{newTrack}.XData = obj.trackCenCol(newTrack,obj.
trackCenLogical(newTrack,:));
                            else
                                myline = line('Parent',handles.axesTracks);
                                myline.Color = obj.trackColor(mod(newTrack,3)+1,:);
                                myline.LineWidth = 1;
                                myline.YData = obj.trackCenRow(newTrack,obj.trackCenLogical(newTrack
,:));
                                myline.XData = obj.trackCenCol(newTrack,obj.trackCenLogical(newTrack
,:));
                                obj.trackLine{newTrack} = myline;
                            end
                            if isa(obj.trackCircle{newTrack},'matlab.graphics.primitive.Rectangle')
                                obj.trackCircle{newTrack}.Position = [obj.trackLine{newTrack}.XData
(1)-(obj.trackCircleSize-1)/2,obj.trackLine{newTrack}.YData(1)-(obj.trackCircleSize-1)/2,obj.
trackCircleSize,obj.trackCircleSize];
                            else
                                myrec = rectangle('Parent',handles.axesCircles);
                                myrec.ButtonDownFcn = @obj.clickLoop;
                                myrec.UserData = newTrack;
                                myrec.Curvature = [1,1];
                                myrec.FaceColor = obj.trackLine{newTrack}.Color;
                                myrec.Position = [obj.trackLine{newTrack}.XData(1)-(obj.
trackCircleSize-1)/2,obj.trackLine{newTrack}.YData(1)-(obj.trackCircleSize-1)/2,obj.
trackCircleSize,obj.trackCircleSize];
                                obj.trackCircle{newTrack} = myrec;
                            end
                            if isa(obj.trackLine{newTrack},'matlab.graphics.primitive.Text');
                                obj.trackText{newTrack}.Position = [obj.trackLine{newTrack}.XData(1)
+(obj.trackCircleSize-1)/2,obj.trackLine{newTrack}.YData(1)+(obj.trackCircleSize-1)/2];

                            else
                                obj.trackText{newTrack} = text('Parent',handles.axesText);
                                obj.trackText{newTrack}.Position = [obj.trackLine{newTrack}.XData(1)
+(obj.trackCircleSize-1)/2,obj.trackLine{newTrack}.YData(1)+(obj.trackCircleSize-1)/2];
                            end
                            obj.updateTrackText(newTrack);
                            obj.trackLine{newTrack}.Visible = 'on';
                            obj.trackCircle{newTrack}.Visible = 'on';
                            obj.trackText{newTrack}.Visible = 'on';
                        end
                        obj.trackLine{oldTrack}.YData = obj.trackCenRow(oldTrack,obj.trackCenLogical
(oldTrack,:));
                        obj.trackLine{oldTrack}.XData = obj.trackCenCol(oldTrack,obj.trackCenLogical
(oldTrack,:));
                        obj.trackCircle{oldTrack}.Position = [obj.trackLine{oldTrack}.XData(1)-(obj.
trackCircleSize-1)/2,obj.trackLine{oldTrack}.YData(1)-(obj.trackCircleSize-1)/2,obj.
trackCircleSize,obj.trackCircleSize];
                        obj.trackText{oldTrack}.Position = [obj.trackLine{oldTrack}.XData(1)+(obj.
trackCircleSize-1)/2,obj.trackLine{oldTrack}.YData(1)+(obj.trackCircleSize-1)/2];
                        obj.tmn.gui_control.tabMakeCell_loop;
                        obj.loop_stepX;
                        %%
                        % return to 'none' mode
                        handlesControl = guidata(obj.tmn.gui_control.gui_main);
                        handlesControl.tabMakeCell_togglebuttonNone.Value = 1;
                        obj.tmn.gui_control.tabMakeCell_buttongroup_SelectionChangedFcn;
                        guidata(obj.tmn.gui_control.gui_main,handlesControl);
                    case 'delete'
                        replaceTrack = obj.tmn.mcl.pointer_track;
```

```
                        obj.tmn.mcl.deleteTrack(replaceTrack);

                        obj.trackCenRow(replaceTrack,:) = 0;
                        obj.trackCenCol(replaceTrack,:) = 0;
                        obj.trackCenLogical(replaceTrack,:) = false;
                        obj.trackCenLogicalDiff = diff(obj.trackCenLogical,1,2);

                        obj.trackLine{replaceTrack}.Visible = 'off';
                        obj.trackCircle{replaceTrack}.Visible = 'off';
                        obj.trackText{replaceTrack}.Visible = 'off';

                        handlesControl.infoBk_textMessage.String = sprintf('Deleted track %d.',
replaceTrack);
                        obj.tmn.gui_control.tabMakeCell_loop;
                        obj.loop_stepX;

                        obj.tmn.gui_control.tabMakeCell_loop;
                        %%
                        % return to 'none' mode
                        handlesControl = guidata(obj.tmn.gui_control.gui_main);
                        handlesControl.tabMakeCell_togglebuttonNone.Value = 1;
                        obj.tmn.gui_control.tabMakeCell_buttongroup_SelectionChangedFcn;
                        guidata(obj.tmn.gui_control.gui_main,handlesControl);
                    case 'mother'
                        if obj.makecellMotherBool
                            obj.makecellMotherBool = false;
                            [mom,dau] = obj.tmn.mcl.identifyMother(obj.tmn.mcl.pointer_makecell2,obj
.tmn.mcl.pointer_makecell);
                            handlesControl.infoBk_textMessage.String = sprintf('Cell %d is the
mother of\ncell %d.',mom,dau);
                            %%
                            % return to 'none' mode
                            handlesControl = guidata(obj.tmn.gui_control.gui_main);
                            handlesControl.tabMakeCell_togglebuttonNone.Value = 1;
                            obj.tmn.gui_control.tabMakeCell_buttongroup_SelectionChangedFcn;
                            obj.updateTrackText;
                            guidata(obj.tmn.gui_control.gui_main,handlesControl);
                        else
                            handlesControl.infoBk_textMessage.String = sprintf('Cell %d will be the
mother of...',obj.tmn.mcl.pointer_makecell);
                            obj.makecellMotherBool = true;
                        end
                        obj.tmn.gui_control.tabMakeCell_loop;
                        obj.loop_stepX;
                    case 'track 2 cell'
                        obj.tmn.mcl.addTrack2Cell(obj.tmn.mcl.pointer_track,obj.tmn.mcl.
pointer_makecell3);
                        obj.tmn.gui_control.tabMakeCell_loop;
                        obj.updateTrackText;
                        obj.highlightTrack;
                        %%
                        % return to 'none' mode
                        handlesControl = guidata(obj.tmn.gui_control.gui_main);
                        handlesControl.tabMakeCell_togglebuttonNone.Value = 1;
                        obj.tmn.gui_control.tabMakeCell_buttongroup_SelectionChangedFcn;
                        guidata(obj.tmn.gui_control.gui_main,handlesControl);
                    otherwise
                        fprintf('trackID %d\n',obj.tmn.mcl.pointer_track);
                end
                guidata(obj.tmn.gui_control.gui_main,handlesControl);
            end
        end
        %%
        %
        function obj = highlightTrack(obj)
            if obj.tmn.mcl.pointer_track2~=obj.tmn.mcl.pointer_track
                myrec = obj.trackCircle{obj.tmn.mcl.pointer_track};
                myrec.FaceColor = obj.trackColorHighlight;

                myrec2 = obj.trackCircle{obj.tmn.mcl.pointer_track2};
                myrec2.FaceColor = obj.trackColor(mod(obj.tmn.mcl.pointer_track2,3)+1,:);

                myline = obj.trackLine{obj.tmn.mcl.pointer_track};
                myline.Color = obj.trackColorHighlight;
                myline.LineWidth = 3;

                myline2 = obj.trackLine{obj.tmn.mcl.pointer_track2};
                myline2.Color = obj.trackColor(mod(obj.tmn.mcl.pointer_track2,3)+1,:);
```

```
                    myline2.LineWidth = 1;
                else
                    myrec = obj.trackCircle{obj.tmn.mcl.pointer_track};
                    myrec.FaceColor = obj.trackColorHighlight;

                    myline = obj.trackLine{obj.tmn.mcl.pointer_track};
                    myline.Color = obj.trackColorHighlight;
                    myline.LineWidth = 3;
                end
                mclID = obj.tmn.mcl.track_makecell(obj.tmn.mcl.pointer_track);
                if mclID ~= 0
                    myrec.EdgeColor = obj.trackColorHighlight2;
                    myrec.LineWidth = 2;
                else
                    myrec.EdgeColor = [0,0,0];
                    myrec.LineWidth = 0.5;
                end
            end
            %%
            %
            function obj = updateTrackText(obj,varargin)
                %%%
                % parse the input
                q = inputParser;
                addRequired(q, 'obj', @(x) isa(x,'cellularGPSTrackingManual_object_imageViewer'));
                addOptional(q, 'trackID',obj.tmn.mcl.pointer_track, @(x)isnumeric(x));
                parse(q,obj,varargin{:});
                trackID = q.Results.trackID;
                obj.trackText{trackID}.Color = obj.trackTextColor;
                obj.trackText{trackID}.BackgroundColor = obj.trackTextBackgroundColor;
                obj.trackText{trackID}.FontSize = obj.trackTextFontSize;
                obj.trackText{trackID}.Margin = obj.trackTextMargin;
                obj.trackText{trackID}.UserData = trackID;
                obj.trackText{trackID}.Position = [obj.trackLine{trackID}.XData(1)+(obj.trackCircleSize
        -1)/2,obj.trackLine{trackID}.YData(1)+(obj.trackCircleSize-1)/2];
                myString = sprintf('trck#: %d',trackID);
                mclID = obj.tmn.mcl.track_makecell(trackID);
                if mclID ~= 0
                    myString = strcat(myString,sprintf('\nmkcl#: %d',mclID));
                    if obj.tmn.mcl.makecell_mother(mclID) ~= 0
                        myString = strcat(myString,sprintf('\nmthr: %d',obj.tmn.mcl.makecell_mother(
        mclID)));
                    end
                    if obj.tmn.mcl.makecell_divisionStart(mclID) ~= 0
                        myString = strcat(myString,sprintf('\ndvSt: %d',obj.tmn.mcl.
        makecell_divisionStart(mclID)));
                    elseif obj.tmn.mcl.makecell_apoptosisStart(mclID) ~= 0
                        myString = strcat(myString,sprintf('\napSt: %d',obj.tmn.mcl.
        makecell_apoptosisStart(mclID)));
                    end
                end
                obj.trackText{trackID}.String = myString;
            end
        end
end
```

Listing B.7: cellularGPSTrackingManualobjectimageViewer.m

```
%% The SuperMDAItinerary
% The SuperMDA allows multiple multi-dimensional-acquisitions to be run
% simulataneously. Each group consists of 1 or more positions. Each
% position consists of 1 or more settings.
classdef cellularGPSTrackingManual_object_itinerary < SuperMDAItineraryTimeFixed_object
    %%
    % * channel_names: the names of the channels group in the current
    % session of uManager.
    % * gps: a matrix that contains the groups, positions, and settings
    % information. As the SuperMDA processes through orderVector it will
    % keep track of which index is changing and execute a function based on
    % this change.
    % * orderVector: a vector with the number of rows of the GPS matrix. It
    % contains the sequence of natural numbers from 1 to the number of
    % rows. The SuperMDA will follow the numbers in the orderVector as they
    % increase and the row that contains the current number corresponds to
    % the next row in the GPS to be executed.
    % * filename_prefix: the string that is placed at the front of the
    % image filename.
    % * fundamental_period: the shortest period that images are taken in
    % seconds.
    % * output_directory: The directory where the output images are stored.
    % * group_order: The group_order exists to deal with the issue of
    % pre-allocation. Performance suffers without pre-allocation. Groups
    % are only active if their index exists in the group_order. The
    % |TravelAgent| enforces the numbers within the group_order vector to
    % be sequential (though not necessarily in order).
    properties

    end
    %%
    %
    methods
        %% The constructor method
        % The first argument is always mm
        function obj = cellularGPSTrackingManual_object_itinerary()

        end

    end
    %%
    %
    methods (Static)

    end
end
```

Listing B.8: cellularGPSTrackingManualobjectitinerary.m

```
classdef cellularGPSTrackingManual_object_makecell < handle
    properties
        moviePath
        positionIndex %the position number
        %%% DATA
        %
        makecell_logical = false;
        makecell_order = cell(1,1);
        makecell_ind = cell(1,1);
        makecell_mother = 0;
        makecell_divisionStart = 0;
        makecell_divisionEnd = 0;
        makecell_apoptosisStart = 0;
        makecell_apoptosisEnd = 0;

        track_database
        track_logical
        track_makecell

        pointer_track = 1;
        pointer_track2 = 1;
        pointer_next_track = 1;
        pointer_makecell = 1;
        pointer_makecell2 = 1;
        pointer_makecell3 = 1;
        pointer_next_makecell = 1;
        pointer_timepoint = 1;

        output_connectedTracks
        output_connectedUniqueTracks
        output_tracks
    end
    %     properties (SetAccess = private)
    %     end
    %     events
    %     end
    methods
        %%
        %
        function obj = cellularGPSTrackingManual_object_makecell(moviePath,varargin)
            %%%
            % parse the input
            q = inputParser;
            addRequired(q, 'moviePath', @(x) isdir(x));
            addOptional(q, 'pInd',0, @(x)isnumeric(x));
            parse(q,moviePath,varargin{:});
            obj.positionIndex = q.Results.pInd;
```

```matlab
            obj.moviePath = q.Results.moviePath;
            if ~isdir(fullfile(obj.moviePath,'MAKECELL_DATA'))
                mkdir(fullfile(obj.moviePath,'MAKECELL_DATA'));
            end
            if obj.positionIndex == 0
                % no positionIndex was given
                return
            end
            obj.import;
        end
        %%
        %
        function obj = loadTrackData(obj,varargin)
            %%%
            % parse the input
            q = inputParser;
            addRequired(q, 'obj', @(x) isa(x,'cellularGPSTrackingManual_object_makecell'));
            addOptional(q, 'trackfilename','nofile', @(x)exist(fullfile(obj.moviePath,'TRACKING_DATA
      ',x),'file'));
            parse(q,obj,varargin{:});

            if ~strcmp(q.Results.myfilename,'nofile')
                obj.track_database = readtable(fullfile(obj.moviePath,'TRACKING_DATA',q.Reults.
      trackfilename),'Delimiter','\t');
            elseif ~istable(obj.track_database)
                error('mkcell:notrack','The track_database is not a table');
            end
            %%%
            % identify tracks
            trackID = unique(obj.track_database.trackID);
            obj.track_logical = false(max(trackID),1);
            obj.track_logical(trackID) = true;
            obj.track_makecell = zeros(max(trackID),1);
            obj.find_pointer_next_track;
        end
        %% find_pointer_next_group
        %
        function obj = find_pointer_next_track(obj)
            if any(~obj.track_logical)
                obj.pointer_next_track = find(~obj.track_logical,1,'first');
            else
                obj.pointer_next_track = numel(obj.track_logical) + 1;
            end
        end
        %% find_pointer_next_group
        %
        function obj = find_pointer_next_makecell(obj)
            if any(~obj.makecell_logical)
                obj.pointer_next_makecell = find(~obj.makecell_logical,1,'first');
            else
                obj.pointer_next_makecell = numel(obj.makecell_logical) + 1;
            end
        end
        %% addTrack
        %
        function obj = addTrack2Cell(obj,varargin)
            %%%
            % parse the input
            q = inputParser;
            addRequired(q, 'obj', @(x) isa(x,'cellularGPSTrackingManual_object_makecell'));
            addOptional(q, 'trackID',obj.pointer_track, @(x)isnumeric(x));
            addOptional(q, 'makecellID',obj.pointer_makecell, @(x)isnumeric(x));
            parse(q,obj,varargin{:});

            obj.pointer_track = q.Results.trackID;
            obj.pointer_makecell = q.Results.makecellID;

            if isempty(obj.makecell_ind{obj.pointer_makecell}) || ~ismember(obj.pointer_track,obj.
      makecell_ind{obj.pointer_makecell})
                obj.makecell_ind{obj.pointer_makecell}(end+1) = obj.pointer_track;
                obj.track_makecell(obj.pointer_track) = obj.pointer_makecell;
                obj.makecell_logical(obj.pointer_makecell) = true;
            end
        end
        %% newCell
        %
        function obj = newCell(obj)
            obj.find_pointer_next_makecell;
            obj.pointer_makecell = obj.pointer_next_makecell;
            obj.makecell_logical(obj.pointer_makecell) = true;
            obj.makecell_ind{obj.pointer_makecell} = [];
            obj.makecell_mother(obj.pointer_makecell) = 0;
            obj.makecell_divisionStart(obj.pointer_makecell) = 0;
            obj.makecell_divisionEnd(obj.pointer_makecell) = 0;
            obj.makecell_apoptosisStart(obj.pointer_makecell) = 0;
            obj.makecell_apoptosisEnd(obj.pointer_makecell) = 0;
        end
        %% breakTrack
        %
        function obj = breakTrack(obj,varargin)
            %%%
            % the columns of the track table are
            % * trackID
            % * timepoint
            % * centroid_row
            % * centroid_col
            %%%
            % parse the input
            q = inputParser;
            addRequired(q, 'obj', @(x) isa(x,'cellularGPSTrackingManual_object_makecell'));
            addOptional(q, 'trackID', obj.pointer_track, @(x)isnumeric(x));
            addOptional(q, 'timepoint', obj.pointer_timepoint, @(x)isnumeric(x));
            parse(q,obj,varargin{:});

            obj.pointer_track = q.Results.trackID;
            obj.pointer_timepoint = q.Results.timepoint;

            myLogicalDatabase = obj.track_database.trackID == obj.pointer_track;
            mySubDatabase = obj.track_database(myLogicalDatabase,:);
            myLogicalBefore = mySubDatabase.timepoint < obj.pointer_timepoint;
            if ~any(myLogicalBefore)
                error('makecell:nobreak','Could not break track, because none of the track exists
 before timepoint %d',q.Results.timepoint);
            end
            tableBefore = mySubDatabase(myLogicalBefore,:);
            tableAfter = mySubDatabase(~myLogicalBefore,:);
            obj.find_pointer_next_track;
            tableAfter.trackID(:) = obj.pointer_next_track;
            obj.pointer_track = obj.pointer_next_track; %the pointer now identifies the new track
 number
            obj.track_makecell(obj.pointer_track) = 0;
            obj.track_logical(obj.pointer_next_track) = true;
            obj.find_pointer_next_track;
            tableOld = obj.track_database(~myLogicalDatabase,:);
            obj.track_database = vertcat(tableOld,tableBefore,tableAfter);
        end
        %% joinTrack
        %
        function obj = joinTrack(obj,varargin)
            %%%
            % parse the input
            q = inputParser;
            addRequired(q, 'obj', @(x) isa(x,'cellularGPSTrackingManual_object_makecell'));
            addOptional(q, 'trackID1',obj.pointer_track, @(x)isnumeric(x));
            addOptional(q, 'trackID2',obj.pointer_track2, @(x)isnumeric(x));
            parse(q,obj,varargin{:});

            obj.pointer_track = q.Results.trackID1;
            obj.pointer_track2 = q.Results.trackID2;

            if obj.pointer_track == obj.pointer_track2
                warning('makecell:sametrack','Could not join tracks, because the inputs %d and %d
 represent only a single track.',q.Results.trackID1,q.Results.trackID2);
                return
            end

            existingTracks = 1:numel(obj.track_logical);
            existingTracks = existingTracks(obj.track_logical);

            if ~ismember(obj.pointer_track,existingTracks) || ~ismember(obj.pointer_track2,
 existingTracks)
                error('makecell:badtrack','Could not join tracks, because the inputs %d and %d
 represent only a single track.',q.Results.trackID1,q.Results.trackID2);
            end

            obj.track_database.trackID(obj.track_database.trackID == obj.pointer_track2) = obj.
```

```
pointer_track;

            if obj.track_makecell(obj.pointer_track2) ~= 0
                obj.makecell_mother(obj.makecell_mother == obj.track_makecell(obj.pointer_track2)) =
 obj.track_makecell(obj.pointer_track);
                obj.deleteCell(obj.track_makecell(obj.pointer_track2));
                obj.track_makecell(obj.pointer_track2) = 0;
            end

            obj.track_logical(obj.pointer_track2) = false;
            obj.pointer_track2 = obj.pointer_track;
            obj.find_pointer_next_track;
        end
        %% deleteTrack
        %
        function obj = deleteTrack(obj,varargin)
            %%%
            % parse the input
            q = inputParser;
            addRequired(q, 'obj', @(x) isa(x,'cellularGPSTrackingManual_object_makecell'));
            addOptional(q, 'trackID',obj.pointer_track, @(x)isnumeric(x));
            parse(q,obj,varargin{:});

            obj.pointer_track = q.Results.trackID;
            existingTracks = 1:numel(obj.track_logical);
            existingTracks = existingTracks(obj.track_logical);

            if ~ismember(obj.pointer_track,existingTracks)
                error('makecell:badtrack','Could not delete track, because the input %d is not a
track.',obj.pointer_track);
            end

            if obj.track_makecell(obj.pointer_track) ~= 0
                obj.deleteCell(obj.track_makecell(obj.pointer_track));
                obj.track_makecell(obj.pointer_track) = 0;
            end

            obj.track_logical(obj.pointer_track) = false;
            obj.track_database = obj.track_database(obj.track_database.trackID(:) ~= obj.
pointer_track,:);
            obj.find_pointer_next_track;
        end
        %%
        %
        function obj = deleteCell(obj,varargin)
            %%%
            % parse the input
            q = inputParser;
            addRequired(q, 'obj', @(x) isa(x,'cellularGPSTrackingManual_object_makecell'));
            addOptional(q, 'makecellID',obj.pointer_makecell, @(x)isnumeric(x));
            parse(q,obj,varargin{:});

            makecellID = q.Results.makecellID;
            obj.makecell_logical(makecellID) = false;
            obj.makecell_order{makecellID} = [];
            obj.makecell_ind{makecellID} = [];
            obj.makecell_mother(makecellID) = 0;
            obj.makecell_mother(obj.makecell_mother == makecellID) = 0;
            obj.makecell_divisionStart(makecellID) = 0;
            obj.makecell_divisionEnd(makecellID) = 0;
            obj.makecell_apoptosisStart(makecellID) = 0;
            obj.makecell_apoptosisEnd(makecellID) = 0;

            obj.track_makecell(obj.track_makecell == makecellID) = 0;

            obj.find_pointer_next_makecell;
        end
        %% import
        %
        function obj = import(obj,varargin)
            %%%
            % parse the input
            q = inputParser;
            addRequired(q, 'obj', @(x) isa(x,'cellularGPSTrackingManual_object_makecell'));
            addOptional(q, 'pInd',obj.positionIndex, @(x)isnumeric(x));
            parse(q,obj,varargin{:});
            obj.positionIndex = q.Results.pInd;
            if exist(fullfile(obj.moviePath,'MAKECELL_DATA',sprintf('trackingPosition_%d.txt',obj.
positionIndex)),'file')
                obj.track_database = readtable(fullfile(obj.moviePath,'MAKECELL_DATA',...
                    sprintf('trackingPosition_%d.txt',obj.positionIndex)),...
                    'Delimiter','\t');
            else
                obj.track_database = readtable(fullfile(obj.moviePath,'TRACKING_DATA',...
                    sprintf('trackingPosition_%d.txt',obj.positionIndex)),...
                    'Delimiter','\t');
                obj.track_database = obj.track_database(:,{'trackID','timepoint','centroid_row','
centroid_col'});
            end
            trackID = unique(obj.track_database.trackID);
            obj.track_logical = false(max(trackID),1);
            obj.track_makecell = zeros(max(trackID),1);
            obj.track_logical(trackID) = true;
            obj.find_pointer_next_track;
            if ~exist(fullfile(obj.moviePath,'MAKECELL_DATA',sprintf('makeCellPosition_%d.txt',obj.
positionIndex)),'file')
                warning('makecell:nofile','The makecell file does not exist for position %d.',obj.
positionIndex);
                obj.makecell_logical = false;
                obj.makecell_order = cell(1,1);
                obj.makecell_ind = cell(1,1);
                obj.makecell_mother = 0;
                obj.makecell_divisionStart = 0;
                obj.makecell_divisionEnd = 0;
                obj.makecell_apoptosisStart = 0;
                obj.makecell_apoptosisEnd = 0;
                obj.track_makecell = zeros(size(obj.track_logical));
                obj.pointer_track = 1;
                obj.pointer_track2 = 1;
                obj.pointer_makecell = 1;
                obj.pointer_makecell2 = 1;
                obj.pointer_makecell3 = 1;
                obj.pointer_timepoint = 1;
            else
                %%
                %
                json = fileread(fullfile(obj.moviePath,'MAKECELL_DATA',sprintf('makeCellPosition_%d.
txt',obj.positionIndex)));
                data = parse_json(json);
                data = data{1}; %the data struct comes wrapped in a cell.
                obj.positionIndex = data.positionIndex;
                if iscell(data.makecell_logical)
                    obj.makecell_logical = logical(cell2mat(data.makecell_logical));
                else
                    obj.makecell_logical = logical(data.makecell_logical);
                end
                if iscell(data.makecell_order)
                    obj.makecell_order = cell(length(data.makecell_order),1);
                    for i = 1:length(data.makecell_order)
                        obj.makecell_order{i} = cell2mat(data.makecell_order{i});
                    end
                elseif data.makecell_order == 0
                    obj.makecell_order = {};
                else
                    obj.makecell_order = {data.makecell_order};
                end
                if iscell(data.makecell_ind)
                    obj.makecell_ind = cell(length(data.makecell_ind),1);
                    for i = 1:length(data.makecell_ind)
                        obj.makecell_ind{i} = cell2mat(data.makecell_ind{i});
                    end
                elseif data.makecell_ind == 0
                    obj.makecell_ind = {};
                else
                    obj.makecell_ind = {data.makecell_ind};
                end
                if iscell(data.makecell_mother)
                    obj.makecell_mother = cell2mat(data.makecell_mother);
                else
                    obj.makecell_mother = data.makecell_mother;
                end
                if iscell(data.makecell_divisionStart)
                    obj.makecell_divisionStart = cell2mat(data.makecell_divisionStart);
                else
                    obj.makecell_divisionStart = data.makecell_divisionStart;
                end
```

```
                if iscell(data.makecell_divisionEnd)
                    obj.makecell_divisionEnd = cell2mat(data.makecell_divisionEnd);
                else
                    obj.makecell_divisionEnd = data.makecell_divisionEnd;
                end
                if iscell(data.makecell_apoptosisStart)
                    obj.makecell_apoptosisStart = cell2mat(data.makecell_apoptosisStart);
                else
                    obj.makecell_apoptosisStart = data.makecell_apoptosisStart;
                end
                if iscell(data.makecell_apoptosisEnd)
                    obj.makecell_apoptosisEnd = cell2mat(data.makecell_apoptosisEnd);
                else
                    obj.makecell_apoptosisEnd = data.makecell_apoptosisEnd;
                end
                if iscell(data.track_logical)
                    obj.track_logical = logical(cell2mat(data.track_logical));
                else
                    obj.track_logical = logical(data.track_logical);
                end
                if iscell(data.track_makecell)
                    obj.track_makecell = cell2mat(data.track_makecell);
                else
                    obj.track_makecell = data.track_makecell;
                end
                obj.pointer_track = data.pointer_track;
                obj.pointer_track2 = data.pointer_track2;
                obj.pointer_next_track = data.pointer_next_track;
                obj.pointer_makecell = data.pointer_makecell;
                obj.pointer_makecell2 = data.pointer_makecell2;
                obj.pointer_makecell3 = data.pointer_makecell3;
                obj.pointer_next_makecell = data.pointer_next_makecell;
                obj.pointer_timepoint = data.pointer_timepoint;
            end
            obj.find_pointer_next_makecell;
        end
        %% export
        %
        function obj = export(obj)
            %%%
            % parse the input
            q = inputParser;
            addRequired(q, 'obj', @(x) isa(x,'cellularGPSTrackingManual_object_makecell'));
            parse(q,obj);
            [obj.track_database,~] = sortrows(obj.track_database,{'trackID','timepoint'},{'ascend','
ascend'});
            writetable(obj.track_database,fullfile(obj.moviePath,'MAKECELL_DATA',sprintf('
trackingPosition_%d.txt',obj.positionIndex)),'Delimiter','\t');

            %% convert data into JSON
            %
            jsonStrings = {};
            n = 1;
            %%%
            %
            jsonStrings{n} = micrographIOT_cellStringArray2json('moviePath',strsplit(obj.moviePath,
filesep)); n = n + 1;
            jsonStrings{n} = micrographIOT_array2json('positionIndex',obj.positionIndex); n = n + 1;
            %%%
            %
            jsonStrings{n} = micrographIOT_array2json('makecell_logical',obj.makecell_logical); n =
n + 1;
            jsonStrings{n} = micrographIOT_cellNumericArray2json('makecell_order',obj.makecell_order
); n = n + 1;
            jsonStrings{n} = micrographIOT_cellNumericArray2json('makecell_ind',obj.makecell_ind); n
 = n + 1;
            jsonStrings{n} = micrographIOT_array2json('makecell_mother',obj.makecell_mother); n = n
+ 1;
            jsonStrings{n} = micrographIOT_array2json('makecell_divisionStart',obj.
makecell_divisionStart);  n = n + 1;
            jsonStrings{n} = micrographIOT_array2json('makecell_divisionEnd',obj.
makecell_divisionEnd); n = n + 1;
            jsonStrings{n} = micrographIOT_array2json('makecell_apoptosisStart',obj.
makecell_apoptosisStart);  n = n + 1;
            jsonStrings{n} = micrographIOT_array2json('makecell_apoptosisEnd',obj.
makecell_apoptosisEnd);  n = n + 1;
            %%%
            %
            jsonStrings{n} = micrographIOT_array2json('track_logical',obj.track_logical); n = n + 1;
            jsonStrings{n} = micrographIOT_array2json('track_makecell',obj.track_makecell); n = n +
1;
            %%%
            %
            jsonStrings{n} = micrographIOT_array2json('pointer_track',obj.pointer_track); n = n + 1;
            jsonStrings{n} = micrographIOT_array2json('pointer_track2',obj.pointer_track2); n = n +
1;
            jsonStrings{n} = micrographIOT_array2json('pointer_next_track',obj.pointer_next_track);
n = n + 1;
            jsonStrings{n} = micrographIOT_array2json('pointer_makecell',obj.pointer_makecell); n =
n + 1;
            jsonStrings{n} = micrographIOT_array2json('pointer_makecell2',obj.pointer_makecell2); n
= n + 1;
            jsonStrings{n} = micrographIOT_array2json('pointer_makecell3',obj.pointer_makecell3); n
= n + 1;
            jsonStrings{n} = micrographIOT_array2json('pointer_next_makecell',obj.
pointer_next_makecell); n = n + 1;
            jsonStrings{n} = micrographIOT_array2json('pointer_timepoint',obj.pointer_timepoint);
            %% export the JSON data to a text file
            %
            myjson = micrographIOT_jsonStrings2Object(jsonStrings);
            fid = fopen(fullfile(obj.moviePath,'MAKECELL_DATA',sprintf('makeCellPosition_%d.txt',obj
.positionIndex)),'w');
            if fid == -1
                error('smdaITF:badfile','Cannot open the file, preventing the export of the smdaITF.
');
            end
            fprintf(fid,myjson);
            fclose(fid);
            %%%
            %
            myjson = micrographIOT_autoIndentJson(fullfile(obj.moviePath,'MAKECELL_DATA',sprintf('
makeCellPosition_%d.txt',obj.positionIndex)));
            fid = fopen(fullfile(obj.moviePath,'MAKECELL_DATA',sprintf('makeCellPosition_%d.txt',obj
.positionIndex)),'w');
            if fid == -1
                error('smdaITF:badfile','Cannot open the file, preventing the export of the smdaITF.
');
            end
            fprintf(fid,myjson);
            fclose(fid);
        end
        %%
        %
        function [mom,dau] = identifyMother(obj,varargin)
            %%%
            % parse the input
            q = inputParser;
            addRequired(q, 'obj', @(x) isa(x,'cellularGPSTrackingManual_object_makecell'));
            addOptional(q, 'mom', obj.pointer_makecell, @(x)isnumeric(x));
            addOptional(q, 'dau', obj.pointer_makecell2, @(x)isnumeric(x));
            parse(q,obj,varargin{:});
            obj.pointer_makecell = q.Results.mom;
            obj.pointer_makecell2 = q.Results.dau;

            existingMakecell = 1:numel(obj.makecell_logical);
            existingMakecell = existingMakecell(obj.makecell_logical);

            if ~ismember(obj.pointer_makecell,existingMakecell) || ~ismember(obj.pointer_makecell2,
existingMakecell)
                error('makecell:badmkcl','Could not assign mother cell, because of invalid cell
number.');
            elseif obj.pointer_makecell == obj.pointer_makecell2
                error('makecell:samemkcl','Could not assign mother cell, because the two cell
numbers are the same.');
            end
            mom = obj.pointer_makecell;
            dau = obj.pointer_makecell2;
            obj.makecell_mother(obj.pointer_makecell2) = obj.pointer_makecell;
        end
        %% exportTracesMatrix
        % Several matrices will be created with time represented by
        % columns:
        %
        % * a matrix where each row represents a cell
        % * a matrix where traces are connected along rows according to
        % their mother
        % * a subset of the previous matrix where only unique traces exist
        % along all rows
        function obj = exportTracesMatrix(obj)
            %% find the centroid table for the position
            %
            smda_database = readtable(fullfile(obj.moviePath,'smda_database.txt'),'Delimiter','\t');
            groupNumber = smda_database.group_number(find(smda_database.position_number == obj.
positionIndex,1,'first'));
            cenTable = readtable(fullfile(obj.moviePath,'CENTROID_DATA',sprintf('
centroid_measurements_g%d_s%d',groupNumber,obj.positionIndex)),'Delimiter','\t');
            %% output_connectedTracks
            %
            obj.output_connectedTracks = {};
            track_makecellTables = cell(size(obj.track_makecell));
            myLogical = obj.track_makecell ~= 0;
            myInd = 1:numel(myLogical);
            myInd = myInd(myLogical);
            for i = myInd
                myLogical2 = false(height(cenTable),1);
                tracktable = obj.track_database(obj.track_database.trackID == i,:);
                for j = 1:height(tracktable)
                    myLogical2 = myLogical2 | (cenTable.centroid_col == tracktable.centroid_col(j) &
     ...
                        cenTable.centroid_row == tracktable.centroid_row(j) & cenTable.timepoint ==
tracktable.timepoint(j));
                end
                track_makecellTables{i} = cenTable(myLogical2,:);
            end

            myLogical = obj.makecell_mother == 0 & obj.makecell_logical;
            seedCells = 1:numel(myLogical);
            seedCells = seedCells(myLogical);
            makecell_mother2 = obj.makecell_mother;
            currentCell = seedCells(1);
            tracks = {};
            tracksPointer = 1;
            while ~isempty(seedCells)
                dauSum = sum(makecell_mother2 == currentCell);
                if dauSum == 0
                    if tracksPointer > numel(tracks)
                        tracks{tracksPointer} = currentCell;
                    else
                        %tracks{tracksPointer}(end+1) = currentCell;
                    end
                    tracksPointer = tracksPointer + 1;
                    if tracksPointer > numel(tracks)
                        seedCells(1) = [];
                        if isempty(seedCells)
                            break;
                        else
                            currentCell = seedCells(1);
                        end
                    else
                        currentCell = tracks{tracksPointer}(end);
                        continue;
                    end
                elseif dauSum == 1
                    if tracksPointer > numel(tracks)
                        tracks{tracksPointer} = currentCell;
                    else
                        %tracks{tracksPointer}(end+1) = currentCell;
                    end
                    myInd = find(makecell_mother2 == currentCell);
                    tracks{tracksPointer}(end+1) = myInd;
                    currentCell = myInd;
                else
                    if tracksPointer > numel(tracks)
                        tracks{tracksPointer} = currentCell;
                    else
                        %tracks{tracksPointer}(end+1) = currentCell;
                    end
                    trackTemp = tracks{tracksPointer};
                    myInd = find(makecell_mother2 == currentCell);
                    tracks{tracksPointer}(end+1) = myInd(1);
                    for i = 2:length(myInd)
                        tracks{end+1} = trackTemp; %#ok<*AGROW>
                        tracks{end}(end+1) = myInd(i);
                    end
```

```
                    currentCell = myInd(1);
                end
            end
            for i = 1:length(tracks)
                cellNum = tracks{i}(1);
                trackNum = obj.makecell_ind{cellNum}(1);
                if trackNum == 0
                    continue
                end
                obj.output_connectedTracks{i} = track_makecellTables{trackNum};
                if length(tracks{i}) > 1
                    for j = 2:length(tracks{i})
                        cellNum = tracks{i}(j);
                        trackNum = obj.makecell_ind{cellNum}(1);
                        obj.output_connectedTracks{i} = vertcat(obj.output_connectedTracks{i},
     track_makecellTables{trackNum});
                    end
                end
            end
            emptylogical = cellfun(@isempty,obj.output_connectedTracks);
            if any(emptylogical)
                obj.output_connectedTracks(emptylogical) = [];
            end
            %% output_connectedUniqueTracks
            %

            %% output_tracks
            %

        end
    end
end
```

Listing B.9: cellularGPSTrackingManualobjectmakecell.m

```
classdef cellularGPSTrackingManual_object_control < handle
    %% Properties
    %  ___                       _   _
    % | _ \_ _ ___ _ __  ___ _ _| |_(_)___ ___
    % |  _/ '_/ _ \ '_ \/ -_) '_|  _| / -_|_-<
    % |_| |_| \___/ .__/\___|_|  \__|_\___/__/
    %             |_|
    %
    properties
        tmn; %the cellularGPSTrackingManual_object
        imag3;
        image_width;
        image_height;
        gui_main;

        contrastHistogram

        %%% menu
        %
        menu_viewTrackBool = true;
        menu_viewTime = 'all';
    end
    %% Methods
    %  __  __     _   _            _
    % |  \/  |___| |_| |_  ___  __| |___
    % | |\/| / -_)  _| ' \/ _ \/ _` (_-<
    % |_|  |_\___|\__|_||_\___/\__,_/__/
    %
    methods
        %% The first method is the constructor
        %   ___             _               _
        %  / __|___ _ _  __| |_ _ _ _  _ __| |_ ___ _ _
        % | (__/ _ \ ' \(_-<  _| '_| || / _|  _/ _ \ '_|
        %  \___\___/_||_/__/\__|_|  \_,_\__|\__\___/_|
        %
        %
        function obj = cellularGPSTrackingManual_object_control(tmn)
            %%%
            % parse the input
            q = inputParser;
            addRequired(q, 'tmn', @(x) isa(x,'cellularGPSTrackingManual_object'));
            parse(q,tmn);
            %%
            %
            obj.tmn = q.Results.tmn;
            obj.imag3 = imread(fullfile(tmn.moviePath,'.thumb',tmn.smda_databaseSubset.filename{tmn.
    indImage}));
            obj.image_width = size(obj.imag3,2);
            obj.image_height = size(obj.imag3,1);
            %% Create a gui to enable pausing and stopping
            %    ___ _   _ ___    ___              _   _
            %   / __| | | |_ _|  / __|_ _ ___ __ _| |_(_)___ _ _
            %  | (_ | |_| || |  | (__| '_/ -_) _` |  _| / _ \ ' \
            %   \___|\___/|___|  \___|_| \___\__,_|\__|_\___/_||_|
            %   / _|___ _ _
            %  |  _/ _ \ '_|  __  __      _
            %  |_| \___/_(_) |  \/  |__ _(_)_ _
            %  / _` | || | | | |\/| / _` | | ' \
            %  \__, |\_,_|_|_|_|  |_\__,_|_|_||_|
            %  |___/       |___|
            % Create the figure
            %
            myunits = get(0,'units');
            set(0,'units','pixels');
            Pix_SS = get(0,'screensize');
            set(0,'units','characters');
            Char_SS = get(0,'screensize');
            ppChar = Pix_SS./Char_SS;
            set(0,'units',myunits);
            fwidth = 136.6; %683/ppChar(3) on a 1920x1080 monitor;
            fheight = 70; %910/ppChar(4) on a 1920x1080 monitor;
            fx = Char_SS(3) - (Char_SS(3)*.1 + fwidth);
            fy = Char_SS(4) - (Char_SS(4)*.1 + fheight);
            f = figure('Visible','off','Units','characters','MenuBar','None','Position',[fx fy
    fwidth fheight],...
                'CloseRequestFcn',{@obj.delete},'Name','Travel Agent Main');
            muView = uimenu(f,'Label','View');
            muViewHT = uimenu(muView,'Label','Hide Tracks',...
                'Callback',@obj.menuViewTracks_Callback);
            muViewTime = uimenu(muView,'Label','Time Window');
            muViewTimeAll = uimenu(muViewTime,'Label','All Time',...
                'Callback',@obj.menuViewTime_Callback,'Checked','on');
            muViewTimeNow = uimenu(muViewTime,'Label','At Present',...
                'Callback',@obj.menuViewTime_Callback);



            textBackgroundColorRegion1 = [37 124 224]/255; %tendoBlueLight
            buttonBackgroundColorRegion1 = [29 97 175]/255; %tendoBlueDark
            textBackgroundColorRegion2 = [56 165 95]/255; %tendoGreenLight
            buttonBackgroundColorRegion2 = [44 129 74]/255; %tendoGreenDark
            textBackgroundColorRegion3 = [255 214 95]/255; %tendoYellowLight
            buttonBackgroundColorRegion3 = [199 164 74]/255; %tendoYellowDark
            textBackgroundColorRegion4 = [255 103 97]/255; %tendoRedLight
            buttonBackgroundColorRegion4 = [199 80 76]/255; %tendoRedDark
            buttonSize = [20 3.0769]; %[100/ppChar(3) 40/ppChar(4)];
            %% Info Brick
            % The section of the gui that contains useful information and messages.
            %   ___      __      ___      _    _
            %  |_ _|_ _ / _|___ | _ )_ _(_)__| |__
            %   | || ' \|  _/ _ \ | _ \ '_| / _| / /
            %  |___|_||_|_| \___/ |___/_| |_\__|_\_\
            %
            infoBk_panelMessage = uipanel('Title','Message','Units','characters','Parent',f,...
                'Position',[0,65,fwidth,5]);
            infoBk_textMessage = uicontrol('Parent',infoBk_panelMessage,'Style','text','Units','
    characters','String','Happy Tracking!',...
                'FontSize',10,'FontName','Verdana','HorizontalAlignment','left',...
                'Position',[1, 0.5, fwidth-2, 3]);
            infoBk_panelInfo = uipanel('Title','Info','Units','characters','Parent',f,...
                'Position',[0,60,fwidth,5]);
            %% timepoint
            %
            infoBk_editTimepoint = uicontrol('Parent',infoBk_panelInfo,'Style','edit','Units','
    characters',...
                'FontSize',14,'FontName','Verdana',...
                'String',num2str(1),...
                'Position',[1, 0.5, 15,2.6923],...
                'Callback',{@obj.infoBk_editTimepoint_Callback});
            infoBk_textTimepoint = uicontrol('Parent',infoBk_panelInfo,'Style','text','Units','characters','String','timepoint',...
                'FontSize',10,'FontName','Verdana','HorizontalAlignment','left',...
                'Position',[17, 0.5, 20, 2.6923]);

            %% Tabs
            %    _____     _
            %   |_   _|_ _| |__  ___
            %     | |/ _` | '_ (_-<
            %     |_|\__,_|_.__/__/
            %
            tab_panel = uipanel('Title','Tabs','Units','characters','Parent',f,...
                'Position',[0,0,fwidth,60]);
            tabgp = uitabgroup(tab_panel,'Units','characters','Position',[0,0,fwidth,58.5]);
            tabGPS = uitab(tabgp,'Title','GPS');
            tabMakeCell = uitab(tabgp,'Title','MakeCell');
            tabContrast = uitab(tabgp,'Title','Contrast');
            %% Contrast Tab: gui
            %    ___         _               _     _____     _
            %   / __|___ _ _| |_ _ _ __ _ __| |_  |_   _|_ _| |__
            %  | (__/ _ \ ' \  _| '_/ _` (_-<  _|   | |/ _` | '_ \
            %   \___\___/_||_\__|_| \__,_/__/\__|   |_|\__,_|_.__/
            %
            %% Create the axes that will show the contrast histogram
            % and the plot that will show the histogram
            hwidth = 104;
            hheight = 40;
            hx = (fwidth-hwidth)/2;
            hy = 10;
            tabContrast_axesContrast = axes('Parent',tabContrast,'Units','characters',...
                'Position',[hx hy hwidth hheight]);
            tabContrast_axesContrast.NextPlot = 'add';
            tabContrast_axesContrast.ButtonDownFcn = @obj.tabContrast_axesContrast_ButtonDownFcn;
            %%% semilogy plot
            %
            obj.tabContrast_findImageHistogram;
            tabContrast_plot = semilogy(tabContrast_axesContrast,(0:255),obj.contrastHistogram,...
                'Color',[0 0 0]/255,...
                'LineWidth',3);
            tabContrast_axesContrast.YScale = 'log';
            tabContrast_axesContrast.XLim = [0,255];
            tabContrast_axesContrast.YLim(1) = 0;
            xlabel('Intensity');
            ylabel('Pixel Count');
            %% Create controls
            %  two slider bars
            hwidth = 112;
            hheight = 2;
            hx = (fwidth-hwidth)/2;
            hy = 5;
            %%% sliderMax
            %
            sliderStep = 1/(256 - 1);
            tabContrast_sliderMax = uicontrol('Parent',tabContrast,'Style','slider','Units','characters',...
                'Min',0,'Max',1,'BackgroundColor',[255 255 255]/255,...
                'Value',1,'SliderStep',[sliderStep sliderStep],'Position',[hx hy hwidth hheight],...
                'Callback',{@obj.tabContrast_sliderMax_Callback});

            hx = (fwidth-hwidth)/2;
            hy = 2;
            %%% sliderMin
            %
            sliderStep = 1/(256 - 1);
            tabContrast_sliderMin= uicontrol('Parent',tabContrast,'Style','slider','Units','characters',...
                'Min',0,'Max',1,'BackgroundColor',[255 255 255]/255,...
                'Value',0,'SliderStep',[sliderStep sliderStep],'Position',[hx hy hwidth hheight],...
                'Callback',{@obj.tabContrast_sliderMin_Callback});
            %% Lines for the min and max contrast levels
            %
            hwidth = 104;
            hheight = 40;
            hx = (fwidth-hwidth)/2;
            hy = 10;
            tabContrast_haxesLine = axes('Parent',tabContrast,'Units','characters',...
                'Position',[hx hy hwidth hheight]);
            tabContrast_haxesLine.NextPlot = 'add';
            tabContrast_haxesLine.Visible = 'off';
            tabContrast_haxesLine.YLim = [0,1];
            tabContrast_haxesLine.XLim = [0,1];
            tabContrast_lineMin = line;
            tabContrast_lineMin.Parent = tabContrast_haxesLine;
            tabContrast_lineMin.Color = [29 97 175]/255;
            tabContrast_lineMin.LineWidth = 3;
            tabContrast_lineMin.LineStyle = ':';
            tabContrast_lineMin.YData = [0,1];
            tabContrast_lineMax = line;
            tabContrast_lineMax.Parent = tabContrast_haxesLine;
            tabContrast_lineMax.Color = [255 103 97]/255;
            tabContrast_lineMax.LineWidth = 3;
            tabContrast_lineMax.LineStyle = ':';
            tabContrast_lineMax.YData = [0,1];

            %% SMDA Tab: gui
            %    ___ ___  ___   _____     _    
            %   / __| _ \/ __| |_   _|_ _| |__ 
            %  | (_ |  _/\__ \   | |/ _` | '_ \
            %   \___|_|  |___/   |_|\__,_|_.__/
            %
            region1 = [0 56.1538]; %[0 730/ppChar(4)]; %180 pixels
            region2 = [0 42.3077]; %[0 550/ppChar(4)]; %180 pixels
            region3 = [0 13.8462]; %[0 180/ppChar(4)]; %370 pixels
            region4 = [0 0]; %180 pixels

            hwidth = 104;
            hx = (fwidth-hwidth)/2;

            %% The group table
            %
            tabGPS_tableGroup = uitable('Parent',tabGPS,'Units','characters',...
                'BackgroundColor',[textBackgroundColorRegion2;buttonBackgroundColorRegion2],...
                'ColumnName',{'label','group #','# of positions'},...
                'ColumnEditable',logical([0,0,0]),...
                'ColumnFormat',{'char','numeric','numeric'},...
                'ColumnWidth',{'auto' 'auto' 'auto'},...
                'FontSize',8,'FontName','Verdana',...
                'CellSelectionCallback',@obj.tabGPS_tableGroup_CellSelectionCallback,...
                'Position',[hx, region2(2)+0.7692, hwidth, 13.0769]);

            %% The position table
            %
            tabGPS_tablePosition = uitable('Parent',tabGPS,'Units','characters',...
                'BackgroundColor',[textBackgroundColorRegion3;buttonBackgroundColorRegion3],...
                'ColumnName',{'label','position #','X','Y','Z','# of settings'},...
                'ColumnEditable',logical([0,0,0,0,0,0]),...
                'ColumnFormat',{'char','numeric','numeric','numeric','numeric','numeric'},...
                'ColumnWidth',{'auto' 'auto' 'auto' 'auto' 'auto' 'auto'},...
                'FontSize',8,'FontName','Verdana',...
                'CellSelectionCallback',@obj.tabGPS_tablePosition_CellSelectionCallback,...
                'Position',[hx, region3(2)+0.7692, hwidth, 28.1538]);
            %% The settings table
            %

            tabGPS_tableSettings = uitable('Parent',tabGPS,'Units','characters',...
                'BackgroundColor',[textBackgroundColorRegion4;buttonBackgroundColorRegion4],...
                'ColumnName',{'channel','exposure','settings #'},...
                'ColumnEditable',logical([0,0,0]),...
                'ColumnFormat',{obj.tmn.ity.channel_names(1),'numeric','numeric'},...
                'ColumnWidth',{'auto' 'auto' 'auto'},...
                'FontSize',8,'FontName','Verdana',...
                'CellSelectionCallback',@obj.tabGPS_tableSettings_CellSelectionCallback,...
                'Position',[hx, region4(2)+0.7692, hwidth, 13.0769]);
            %%
            %   __  __      _        ___     _ _   _____     _    
            %  |  \/  |__ _| |_____ / __|___| | | |_   _|_ _| |__ 
            %  | |\/| / _` | / / -_) (__/ -_) | |   | |/ _` | '_ \
            %  |_|  |_\__,_|_\_\___|\___\___|_|_|   |_|\__,_|_.__/
            %
            textBackgroundColorRegion1 = [37 124 224]/255; %tendoBlueLight
            buttonBackgroundColorRegion1 = [29 97 175]/255; %tendoBlueDark
            textBackgroundColorRegion2 = [56 165 95]/255; %tendoGreenLight
            buttonBackgroundColorRegion2 = [44 129 74]/255; %tendoGreenDark
            textBackgroundColorRegion3 = [255 214 95]/255; %tendoYellowLight
            buttonBackgroundColorRegion3 = [199 164 74]/255; %tendoYellowDark
            textBackgroundColorRegion4 = [255 103 97]/255; %tendoRedLight
            buttonBackgroundColorRegion4 = [199 80 76]/255; %tendoRedDark
            region1 = [0 46]; %[0 730/ppChar(4)]; %180 pixels
            region2 = [0 36]; %[0 550/ppChar(4)]; %180 pixels
            region3 = [0 13.8462]; %[0 180/ppChar(4)]; %370 pixels
            region4 = [0 0]; %180 pixels

            buttonSize = [20 3.0769]; %[100/ppChar(3) 40/ppChar(4)];
            buttongap = 2;
            hx = (fwidth -4*buttonSize(1)-4*buttongap)/2;
            %%
            %
            tabMakeCell_panel = uipanel('Title','Track','Units','characters','Parent',tabMakeCell
    ,...
                'Position',[0,region2(2),fwidth,20]);
            textColor = [255 235 205]/255;


            tabMakeCell_buttongroup = uibuttongroup('Parent',tabMakeCell_panel);
            tabMakeCell_buttongroup.SelectionChangedFcn = @obj.
    tabMakeCell_buttongroup_SelectionChangedFcn;

            tabMakeCell_togglebuttonNone = uicontrol('Parent',tabMakeCell_buttongroup,'Style','
    togglebutton','Units','characters',...
                'FontSize',14,'FontName','Verdana','BackgroundColor',[139  69  19]/255,...
                'String','None',...
                'Position',[hx, 10.5, buttonSize(1),buttonSize(2)],...
                'ForegroundColor',textColor);

            uicontrol('Parent',tabMakeCell_panel,'Style','text','Units','characters','String','do (n
    )othing',...
                'FontSize',10,'FontName','Verdana','BackgroundColor',textBackgroundColorRegion1,...
                'Position',[hx, buttonSize(2)+11, buttonSize(1),2.6923],...
                'ForegroundColor',textColor);

            tabMakeCell_togglebuttonJoin = uicontrol('Parent',tabMakeCell_buttongroup,'Style','
    togglebutton','Units','characters',...
                'FontSize',14,'FontName','Verdana','BackgroundColor',buttonBackgroundColorRegion1
    ,...
                'String','Join',...
                'Position',[hx + buttongap + buttonSize(1), 10.5, buttonSize(1),buttonSize(2)],...
                'ForegroundColor',textColor);

            uicontrol('Parent',tabMakeCell_panel,'Style','text','Units','characters','String','(j)
    oin two tracks',...
                'FontSize',10,'FontName','Verdana','BackgroundColor',textBackgroundColorRegion1,...
                'Position',[hx + buttongap + buttonSize(1),buttonSize(2)+11, buttonSize(1)
    ,2.6923],...
                'ForegroundColor',textColor);

            tabMakeCell_togglebuttonBreak = uicontrol('Parent',tabMakeCell_buttongroup,'Style','
    togglebutton','Units','characters',...
                'FontSize',14,'FontName','Verdana','BackgroundColor',buttonBackgroundColorRegion1
    ,...
                'String','Break',...
                'Position',[hx + buttongap*2 + buttonSize(1)*2,10.5, buttonSize(1),buttonSize(2)
    ],...
                'ForegroundColor',textColor);

            uicontrol('Parent',tabMakeCell_panel,'Style','text','Units','characters','String','(b)
    reak a track into two',...
                'FontSize',10,'FontName','Verdana','BackgroundColor',textBackgroundColorRegion1,...
                'Position',[hx + buttongap*2 + buttonSize(1)*2, buttonSize(2)+11, buttonSize(1)
    ,2.6923],...
                'ForegroundColor',textColor);

            tabMakeCell_togglebuttonDelete = uicontrol('Parent',tabMakeCell_buttongroup,'Style','
    togglebutton','Units','characters',...
                'FontSize',14,'FontName','Verdana','BackgroundColor',buttonBackgroundColorRegion1
    ,...
                'String','Delete',...
                'Position',[hx + buttongap*3 + buttonSize(1)*3,10.5, buttonSize(1),buttonSize(2)
    ],...
                'ForegroundColor',textColor);

            uicontrol('Parent',tabMakeCell_panel,'Style','text','Units','characters','String','
    delete a track (-)',...
                'FontSize',10,'FontName','Verdana','BackgroundColor',textBackgroundColorRegion1,...
                'Position',[hx + buttongap*3 + buttonSize(1)*3, buttonSize(2)+11, buttonSize(1)
    ,2.6923],...
                'ForegroundColor',textColor);
            %%
            %
            buttonSize = [20 3.0769]; %[100/ppChar(3) 40/ppChar(4)];
            buttongap = 2;
            hx = (fwidth-4*buttonSize(1)-4*buttongap)/2;
            %%
            %
            %               tabMakeCell_panelMakeCell = uipanel('Title','MakeCell','Units','characters
    ','Parent',tabMakeCell,...
            %                   'Position',[0,region2(2),fwidth,10]);
            textColor = [255 192 203]/255;


            %               tabMakeCell_buttongroupMakeCell = uibuttongroup('Parent',
    tabMakeCell_panelMakeCell);
            %               tabMakeCell_buttongroupMakeCell.SelectionChangedFcn = @obj.
    tabMakeCell_buttongroupMakeCell_SelectionChangedFcn;
            %
            tabMakeCell_pushbuttonNewCell = uicontrol('Parent',tabMakeCell_panel,'Style','pushbutton
    ','Units','characters',...
                'FontSize',14,'FontName','Verdana','BackgroundColor',buttonBackgroundColorRegion2
    ,...
                'String','New Cell',...
                'Position',[hx, 0.5, buttonSize(1),buttonSize(2)],...
                'ForegroundColor',textColor,...
                'Callback',{@obj.tabMakeCell_pushbuttonNewCell_Callback});

            uicontrol('Parent',tabMakeCell_panel,'Style','text','Units','characters','String','(c)
    reate a new cell',...
                'FontSize',10,'FontName','Verdana','BackgroundColor',textBackgroundColorRegion2,...
                'Position',[hx, buttonSize(2)+1, buttonSize(1),2.6923],...
                'ForegroundColor',textColor);

            tabMakeCell_togglebuttonAddTrack2Cell = uicontrol('Parent',tabMakeCell_buttongroup,'
    Style','togglebutton','Units','characters',...
                'FontSize',10,'FontName','Verdana','BackgroundColor',buttonBackgroundColorRegion2
    ,...
                'String','Track 2 Cell',...
                'Position',[hx + buttongap + buttonSize(1), 0.5, buttonSize(1),buttonSize(2)],...
                'ForegroundColor',textColor);

            uicontrol('Parent',tabMakeCell_panel,'Style','text','Units','characters','String','add a
     (t)rack to a cell',...
                'FontSize',10,'FontName','Verdana','BackgroundColor',textBackgroundColorRegion2,...
                'Position',[hx + buttongap + buttonSize(1),buttonSize(2)+1, buttonSize(1)
    ,2.6923],...
                'ForegroundColor',textColor);

            tabMakeCell_togglebuttonMother = uicontrol('Parent',tabMakeCell_buttongroup,'Style','
    togglebutton','Units','characters',...
                'FontSize',14,'FontName','Verdana','BackgroundColor',buttonBackgroundColorRegion2
    ,...
                'String','Mother',...
                'Position',[hx + buttongap*2 + buttonSize(1)*2,0.5, buttonSize(1),buttonSize(2)],...
                'ForegroundColor',textColor);

            uicontrol('Parent',tabMakeCell_panel,'Style','text','Units','characters','String','
    choose (m)other cell',...
                'FontSize',10,'FontName','Verdana','BackgroundColor',textBackgroundColorRegion2,...
                'Position',[hx + buttongap*2 + buttonSize(1)*2, buttonSize(2)+1, buttonSize(1)
    ,2.6923],...
                'ForegroundColor',textColor);
            %
            %               tabMakeCell_togglebuttonDelete = uicontrol('Parent',
    tabMakeCell_buttongroup,'Style','togglebutton','Units','characters',...
            %                   'FontSize',14,'FontName','Verdana','BackgroundColor',
    buttonBackgroundColorRegion1,...
            %                   'String','Delete',...
            %                   'Position',[hx + buttongap*3 + buttonSize(1)*3,0.5, buttonSize(1),
    buttonSize(2)],...
            %                   'ForegroundColor',textColor);
            %
            %               uicontrol('Parent',tabMakeCell_panel,'Style','text','Units','characters','
    String','delete a track',...
            %                   'FontSize',10,'FontName','Verdana','BackgroundColor',
    textBackgroundColorRegion1,...
```

```
        %                  'Position',[hx + buttongap*3 + buttonSize(1)*3, buttonSize(2)+1,
buttonSize(1),2.6923],...
        %                  'ForegroundColor',textColor);
        %%
        %
        tabMakeCell_table = uitable('Parent',tabMakeCell,'Units','characters',...
            'BackgroundColor',[textBackgroundColorRegion3;buttonBackgroundColorRegion3],...
            'ColumnName',{'cell #','trackIDS','mother'},...
            'ColumnEditable',logical([0,0,0]),...
            'ColumnFormat',{'numeric','char','numeric'},...
            'ColumnWidth',{'auto' 'auto' 'auto'},...
            'FontSize',8,'FontName','Verdana',...
            'CellSelectionCallback',@obj.tabMakeCell_table_CellSelectionCallback,...
            'Position',[hx, region3(2)+0.7692, hwidth, 13.0769]);
        %% Handles
        %   _  _              _  _
        %  | || |__ _ _ _  __| | |___ ___
        %  | __ / _` | ' \/ _` | / -_|_-<
        %  |_||_\__,_|_||_\__,_|_\___/__/
        %
        % store the uicontrol handles in the figure handles via guidata()
        % store the uicontrol handles in the figure handles via guidata()
        handles.muView = muView;
        handles.muViewHT = muViewHT;
        handles.muViewTime = muViewTime;
        handles.muViewTimeAll = muViewTimeAll;
        handles.muViewTimeNow = muViewTimeNow;

        handles.infoBk_textMessage = infoBk_textMessage;
        handles.infoBk_editTimepoint = infoBk_editTimepoint;
        handles.infoBk_textTimepoint = infoBk_textTimepoint;

        handles.tabgp = tabgp;
        handles.tabGPS = tabGPS;
        handles.tabMakeCell = tabMakeCell;
        handles.tabContrast = tabContrast;

        handles.tabContrast_haxesLine = tabContrast_haxesLine;
        handles.tabContrast_lineMin = tabContrast_lineMin;
        handles.tabContrast_lineMax = tabContrast_lineMax;
        handles.tabContrast_plot = tabContrast_plot;
        handles.tabContrast_axesContrast = tabContrast_axesContrast;
        handles.tabContrast_sliderMax = tabContrast_sliderMax;
        handles.tabContrast_sliderMin = tabContrast_sliderMin;

        handles.tabGPS_tableGroup = tabGPS_tableGroup;
        handles.tabGPS_tablePosition = tabGPS_tablePosition;
        handles.tabGPS_tableSettings = tabGPS_tableSettings;

        handles.tabMakeCell_buttongroup = tabMakeCell_buttongroup;
        handles.tabMakeCell_table = tabMakeCell_table;
        handles.tabMakeCell_togglebuttonNone = tabMakeCell_togglebuttonNone;
        handles.tabMakeCell_togglebuttonJoin = tabMakeCell_togglebuttonJoin;
        handles.tabMakeCell_togglebuttonBreak = tabMakeCell_togglebuttonBreak;
        handles.tabMakeCell_togglebuttonDelete = tabMakeCell_togglebuttonDelete;

        handles.tabMakeCell_pushbuttonNewCell = tabMakeCell_pushbuttonNewCell;
        handles.tabMakeCell_togglebuttonAddTrack2Cell = tabMakeCell_togglebuttonAddTrack2Cell;
        handles.tabMakeCell_togglebuttonMother = tabMakeCell_togglebuttonMother;

        obj.gui_main = f;
        guidata(f,handles);
        %% Execute just before the figure becomes visible
        %       _           _      ___ _ _
        %    _ | |_   _ __| |_   | _ ) | |
        %   | || | || (_-<  _|  | _ \_  _|
        %   _\__/_\_,_/__/\__|  |___/ |_|
        %   \ \ / (_)__(_) |__| |___
        %    \ V /| (_-< | '_ \ / -_)
        %     \_/ |_/__/_|_.__/_\___|
        %
        % The code above organizes and specifies the elements of the figure and
        % gui. The code below may simple store these elements into the handles
        % struct and make the gui visible for the first time. Other commands or
        % functions can also be executed here if certain variables or parameters
        % need to be computed and set.
        obj.tabContrast_axesContrast_ButtonDownFcn;
        obj.tabGPS_loop
```

```
            %%%
            % make the gui visible
            set(f,'Visible','on');
        end
        %% delete
        % for a clean delete make sure the objects that are stored as
        % properties are also deleted.
        function delete(obj,~,~)
            delete(obj.gui_main);
        end
        %% Contrast Tab: callbacks and functions
        %    ___          _                 _      _____       _
        %   / __|___ _ _| |_ _ _ __ _ __| |_  |_   _|_ _| |__
        %  | (__/ _ \ ' \  _| '_/ _` (_-<  _|    | |/ _` | '_ \
        %   \___\___/_||_\__|_| \__,_/__/\__|    |_|\__,_|_.__/
        %
        %%
        %
        function obj = tabContrast_findImageHistogram(obj)
            [obj.contrastHistogram,~] = histcounts(reshape(obj.tmn.gui_imageViewer.imag3,1,[])
,-0.5:1:255.5);
        end
        %%
        %
        function obj = tabContrast_axesContrast_ButtonDownFcn(obj,~,~)
            %%%
            % create the contrast histogram to be displayed in the axes
            handles = guidata(obj.gui_main);
            obj.tabContrast_findImageHistogram;
            handles.tabContrast_plot.YData = obj.contrastHistogram;
            obj.tabContrastLineUpdate;
            guidata(obj.gui_main,handles);
        end
        %%
        %
        function obj = tabContrast_sliderMax_Callback(obj,~,~)
            handles = guidata(obj.gui_main);
            sstep = handles.tabContrast_sliderMax.SliderStep;
            mymax = handles.tabContrast_sliderMax.Value;
            mymin = handles.tabContrast_sliderMin.Value;
            if mymax == 0
                handles.tabContrast_sliderMax.Value = sstep(1);
                handles.tabContrast_sliderMin.Value = 0;
            elseif mymax <= mymin
                handles.tabContrast_sliderMin.Value = mymax-sstep(1);
            end
            obj.tabContrast_newColormapFromContrastHistogram;
            obj.tabContrastLineUpdate;
            guidata(obj.gui_main,handles);
        end
        %%
        %
        function obj = tabContrast_sliderMin_Callback(obj,~,~)
            handles = guidata(obj.gui_main);
            sstep = handles.tabContrast_sliderMax.SliderStep;
            mymax = handles.tabContrast_sliderMax.Value;
            mymin = handles.tabContrast_sliderMin.Value;
            if mymin == 1
                handles.tabContrast_sliderMax.Value = 1;
                handles.tabContrast_sliderMin.Value = 1-sstep(1);
            elseif mymin >= mymax
                handles.tabContrast_sliderMax.Value = mymin+sstep(1);
            end
            obj.tabContrast_newColormapFromContrastHistogram;
            obj.tabContrastLineUpdate;
            guidata(obj.gui_main,handles);
        end
        %%
        %
        function obj = tabContrastLineUpdate(obj)
            handles = guidata(obj.gui_main);
            handles.tabContrast_lineMin.XData = [handles.tabContrast_sliderMin.Value,handles.
tabContrast_sliderMin.Value];
            handles.tabContrast_lineMax.XData = [handles.tabContrast_sliderMax.Value,handles.
tabContrast_sliderMax.Value];
            guidata(obj.gui_main,handles);
        end
        %% newColormapFromContrastHistogram
```

```
    % Assumes image is uint8 0-255.
    function obj = tabContrast_newColormapFromContrastHistogram(obj)
        handles = guidata(obj.gui_main);
        sstep = handles.tabContrast_sliderMin.SliderStep;
        mymin = ceil(handles.tabContrast_sliderMin.Value/sstep(1));
        mymax = ceil(handles.tabContrast_sliderMax.Value/sstep(1));
        cmap = colormap(gray(mymax-mymin+1));
        cmap = vertcat(zeros(mymin,3),cmap,ones(255-mymax,3));
        obj.tmn.gui_imageViewer.gui_main.Colormap = cmap;
    end
    %% GPS Tab: callbacks and functions
    %   ___ ___  ___   _____     _
    %  / __| _ \/ __| |_   _|_ _| |__
    % | (_ |  _/\__ \   | |/ _` | '_ \
    %  \___|_|  |___/   |_|\__,_|_.__/
    %
    %%
    %
    function obj = infoBk_editTimepoint_Callback(obj,~,~)
        handles = guidata(obj.gui_main);
        indImage = str2double(handles.infoBk_editTimepoint.String);
        indImage = round(indImage);
        if indImage < 1
            obj.tmn.indImage = 1;
        elseif indImage > height(obj.tmn.smda_databaseSubset)
            obj.tmn.indImage = height(obj.tmn.smda_databaseSubset);
        else
            obj.tmn.indImage = indImage;
        end
        obj.tmn.gui_imageViewer.loop_stepX;
        handles.infoBk_editTimepoint.String = num2str(obj.tmn.indImage);
        guidata(obj.gui_main,handles);
    end
    %%
    %
    function obj = tabGPS_tableGroup_CellSelectionCallback(obj,~,eventdata)
        %%%
        % The main purpose of this function is to keep the information
        % displayed in the table consistent with the Itinerary object.
        % Changes to the object either through the command line or the gui
        % can affect the information that is displayed in the gui and this
        % function will keep the gui information consistent with the
        % Itinerary information.
        %
        % The pointer of the TravelAgent should always point to a valid
        % group from the the group_order.
        if isempty(eventdata.Indices)
            % if nothing is selected, which triggers after deleting data,
            % make sure the pointer is still valid
            if any(obj.tmn.pointerGroup > obj.tmn.ity.number_group)
                % move pointer to last entry
                obj.tmn.pointerGroup = obj.tmn.ity.number_group;
            end
            return
        else
            obj.tmn.pointerGroup = sort(unique(eventdata.Indices(:,1)));
        end

        myGroupOrder = obj.tmn.ity.order_group;
        gInd = myGroupOrder(obj.tmn.pointerGroup(1));
        if any(obj.tmn.pointerPosition > obj.tmn.ity.number_position(gInd))
            % move pointer to first entry
            obj.tmn.pointerPosition = 1;
        end

        obj.tabGPS_loop;
        %%
        % save changes made to the previous position
        obj.tmn.mcl.export;

        obj.tmn.gui_imageViewer.loadNewTracks;
        obj.tmn.gui_imageViewer.loop_stepX;
    end
    %%
    %
    function obj = tabGPS_tablePosition_CellSelectionCallback(obj,~,eventdata)
        %%%
        % The main purpose of this function is to keep the information
```

```
            % displayed in the table consistent with the Itinerary object.
            % Changes to the object either through the command line or the gui
            % can affect the information that is displayed in the gui and this
            % function will keep the gui information consistent with the
            % Itinerary information.
            %
            % The pointer of the TravelAgent should always point to a valid
            % position from the the position_order in a given group.
            myGroupOrder = obj.tmn.ity.order_group;
            gInd = myGroupOrder(obj.tmn.pointerGroup(1));
            if isempty(eventdata.Indices)
                % if nothing is selected, which triggers after deleting data,
                % make sure the pointer is still valid
                if any(obj.tmn.pointerPosition > obj.tmn.ity.number_position(gInd))
                    % move pointer to last entry
                    obj.tmn.pointerPosition = obj.tmn.ity.number_position(gInd);
                end
                return
            else
                obj.tmn.pointerPosition = sort(unique(eventdata.Indices(:,1)));
            end
            obj.tabGPS_loop;
            %%
            % save changes made to the previous position
            obj.tmn.mcl.export;

            obj.tmn.gui_imageViewer.loadNewTracks;
            obj.tmn.gui_imageViewer.loop_stepX;
        end
        %%
        %
        function obj = tabGPS_tableSettings_CellSelectionCallback(obj,~,eventdata)
            %%%
            % The |Travel Agent| aims to recreate the experience that
            % microscope users expect from a multi-dimensional acquistion tool.
            % Therefore, most of the customizability is masked by the
            % |TravelAgent| to provide a streamlined presentation and simple
            % manipulation of the |Itinerary|. Unlike the group and position
            % tables, which edit the itinerary directly, the settings table
            % will modify the the prototype, which will then be pushed to all
            % positions in a group.
            myGroupOrder = obj.tmn.ity.order_group;
            gInd = myGroupOrder(obj.tmn.pointerGroup(1));
            pInd = obj.tmn.ity.ind_position{gInd};
            pInd = pInd(1);
            if isempty(eventdata.Indices)
                % if nothing is selected, which triggers after deleting data,
                % make sure the pointer is still valid
                if any(obj.tmn.pointerSettings > obj.tmn.ity.number_settings{pInd})
                    % move pointer to last entry
                    obj.tmn.pointerSettings = obj.tmn.ity.number_settings(pInd);
                end
                return
            else
                obj.tmn.pointerSettings = sort(unique(eventdata.Indices(:,1)));
            end
            obj.tabGPS_loop;
            obj.tmn.gui_imageViewer.loop_stepX;
        end
        %%
        %
        function obj = tabGPS_loop(obj)
            handles = guidata(obj.gui_main);

            %% Group Table
            % Show the data in the itinerary |group_order| property
            tableGroupData = cell(obj.tmn.ity.number_group,...
                length(get(handles.tabGPS_tableGroup,'ColumnName')));
            n=0;
            for i = obj.tmn.ity.order_group
                n = n + 1;
                tableGroupData{n,1} = obj.tmn.ity.group_label{i};
                tableGroupData{n,2} = i;
                tableGroupData{n,3} = obj.tmn.ity.number_position(i);
            end
            set(handles.tabGPS_tableGroup,'Data',tableGroupData);
            %% Region 3
            %
            %% Position Table
            % Show the data in the itinerary |position_order| property for a given
            % group
            myGroupOrder = obj.tmn.ity.order_group;
            gInd = myGroupOrder(obj.tmn.pointerGroup(1));
            myPositionOrder = obj.tmn.ity.order_position{gInd};
            tablePositionData = cell(length(myPositionOrder),...
                length(get(handles.tabGPS_tablePosition,'ColumnName')));
            n=0;
            for i = myPositionOrder
                n = n + 1;
                tablePositionData{n,1} = obj.tmn.ity.position_label{i};
                tablePositionData{n,2} = i;
                tablePositionData{n,3} = obj.tmn.ity.position_xyz(i,1);
                tablePositionData{n,4} = obj.tmn.ity.position_xyz(i,2);
                tablePositionData{n,5} = obj.tmn.ity.position_xyz(i,3);
                tablePositionData{n,6} = obj.tmn.ity.number_settings(i);
            end
            set(handles.tabGPS_tablePosition,'Data',tablePositionData);
            %% Region 4
            %
            %% Settings Table
            % Show the prototype_settings
            pInd = obj.tmn.ity.ind_position{gInd};
            pInd = pInd(1);
            mySettingsOrder = obj.tmn.ity.order_settings{pInd};
            tableSettingsData = cell(length(mySettingsOrder),...
                length(get(handles.tabGPS_tableSettings,'ColumnName')));
            n=1;
            for i = mySettingsOrder
                tableSettingsData{n,1} = obj.tmn.ity.channel_names{obj.tmn.ity.settings_channel(i)};
                tableSettingsData{n,2} = obj.tmn.ity.settings_exposure(i);
                tableSettingsData{n,3} = i;
                n = n + 1;
            end
            set(handles.tabGPS_tableSettings,'Data',tableSettingsData);
            %% obj.tmn indices
            %
            myGroupOrder = obj.tmn.ity.order_group;
            obj.tmn.indG = myGroupOrder(obj.tmn.pointerGroup(1));
            myPositionOrder = obj.tmn.ity.ind_position{gInd};
            obj.tmn.indP = myPositionOrder(obj.tmn.pointerPosition(1));
            mySettingsOrder = obj.tmn.ity.ind_settings{pInd};
            obj.tmn.indS = mySettingsOrder(obj.tmn.pointerSettings(1));
            obj.tmn.updateFilenameListImage;
            %%
            %
            handles.infoBk_textTimepoint.String = sprintf('of %d\ntimepoint(s)',height(obj.tmn.
      smda_databaseSubset));
            guidata(obj.gui_main,handles);
        end
        %% MakeCell Tab: callbacks and functions
        %   __  __      _        ___     _ _   _____     _
        %  |  \/  |__ _| |_____ / __|___| | | |_   _|_ _| |__
        %  | |\/| / _` | / / -_) (__/ -_) | |   | |/ _` | '_ \
        %  |_|  |_\__,_|_\_\___|\___\___|_|_|   |_|\__,_|_.__/
        %
        %%
        %
        function obj = tabMakeCell_buttongroup_SelectionChangedFcn(obj,~,~)
            handles = guidata(obj.gui_main);
            activeColor = [139  69  19]/255;
            inactiveColor = [29 97 175]/255;
            activeColor2 = [220 20 60]/255;
            inactiveColor2 = [44 129 74]/255;
            switch lower(handles.tabMakeCell_buttongroup.SelectedObject.String)
                case 'none'
                    obj.tmn.makecell_mode = 'none';
                    handles.tabMakeCell_togglebuttonNone.BackgroundColor = activeColor;
                    handles.tabMakeCell_togglebuttonJoin.BackgroundColor = inactiveColor;
                    handles.tabMakeCell_togglebuttonBreak.BackgroundColor = inactiveColor;
                    handles.tabMakeCell_togglebuttonDelete.BackgroundColor = inactiveColor;
                    handles.tabMakeCell_togglebuttonMother.BackgroundColor = inactiveColor2;
                    handles.tabMakeCell_togglebuttonAddTrack2Cell.BackgroundColor = inactiveColor2;
                case 'join'
                    obj.tmn.makecell_mode = 'join';
                    handles.tabMakeCell_togglebuttonNone.BackgroundColor = inactiveColor;
                    handles.tabMakeCell_togglebuttonJoin.BackgroundColor = activeColor;
```

```
                    handles.tabMakeCell_togglebuttonBreak.BackgroundColor = inactiveColor;
                    handles.tabMakeCell_togglebuttonDelete.BackgroundColor = inactiveColor;
                    handles.tabMakeCell_togglebuttonMother.BackgroundColor = inactiveColor2;
                    handles.tabMakeCell_togglebuttonAddTrack2Cell.BackgroundColor = inactiveColor2;
                case 'break'
                    obj.tmn.makecell_mode = 'break';
                    handles.tabMakeCell_togglebuttonNone.BackgroundColor = inactiveColor;
                    handles.tabMakeCell_togglebuttonJoin.BackgroundColor = inactiveColor;
                    handles.tabMakeCell_togglebuttonBreak.BackgroundColor = activeColor;
                    handles.tabMakeCell_togglebuttonDelete.BackgroundColor = inactiveColor;
                    handles.tabMakeCell_togglebuttonMother.BackgroundColor = inactiveColor2;
                    handles.tabMakeCell_togglebuttonAddTrack2Cell.BackgroundColor = inactiveColor2;
                case 'delete'
                    obj.tmn.makecell_mode = 'delete';
                    handles.tabMakeCell_togglebuttonNone.BackgroundColor = inactiveColor;
                    handles.tabMakeCell_togglebuttonJoin.BackgroundColor = inactiveColor;
                    handles.tabMakeCell_togglebuttonBreak.BackgroundColor = inactiveColor;
                    handles.tabMakeCell_togglebuttonDelete.BackgroundColor = activeColor;
                    handles.tabMakeCell_togglebuttonMother.BackgroundColor = inactiveColor2;
                    handles.tabMakeCell_togglebuttonAddTrack2Cell.BackgroundColor = inactiveColor2;
                case 'mother'
                    obj.tmn.makecell_mode = 'mother';
                    handles.tabMakeCell_togglebuttonNone.BackgroundColor = inactiveColor;
                    handles.tabMakeCell_togglebuttonJoin.BackgroundColor = inactiveColor;
                    handles.tabMakeCell_togglebuttonBreak.BackgroundColor = inactiveColor;
                    handles.tabMakeCell_togglebuttonDelete.BackgroundColor = inactiveColor;
                    handles.tabMakeCell_togglebuttonMother.BackgroundColor = activeColor2;
                    handles.tabMakeCell_togglebuttonAddTrack2Cell.BackgroundColor = inactiveColor2;
                case 'track 2 cell'
                    obj.tmn.makecell_mode = 'track 2 cell';
                    handles.tabMakeCell_togglebuttonNone.BackgroundColor = inactiveColor;
                    handles.tabMakeCell_togglebuttonJoin.BackgroundColor = inactiveColor;
                    handles.tabMakeCell_togglebuttonBreak.BackgroundColor = inactiveColor;
                    handles.tabMakeCell_togglebuttonDelete.BackgroundColor = inactiveColor;
                    handles.tabMakeCell_togglebuttonMother.BackgroundColor = inactiveColor2;
                    handles.tabMakeCell_togglebuttonAddTrack2Cell.BackgroundColor = activeColor2;
            end
            guidata(obj.gui_main,handles);
        end
        %%
        %
        function obj = tabMakeCell_loop(obj)
            handles = guidata(obj.gui_main);
            %% Cell Table
            %
            existingCells = 1:length(obj.tmn.mcl.makecell_logical);
            existingCells = existingCells(obj.tmn.mcl.makecell_logical);
            makeCellData = cell(length(obj.tmn.mcl.makecell_logical),...
                length(handles.tabMakeCell_table.ColumnName));
            n=0;
            for i = existingCells
                n = n + 1;
                makeCellData{n,1} = i;
                makeCellData{n,2} = num2str(obj.tmn.mcl.makecell_ind{i});
                makeCellData{n,3} = obj.tmn.mcl.makecell_mother(i);
            end
            handles.tabMakeCell_table.Data = makeCellData;
        end
        %%
        %
        function obj = tabMakeCell_table_CellSelectionCallback(obj,~,eventdata)
            if isempty(eventdata.Indices)
                % if nothing is selected, which triggers after deleting data,
                % make sure the pointer is still valid
                obj.tmn.mcl.find_pointer_next_makecell;
                return
            else
                handles = guidata(obj.gui_main);
                obj.tmn.mcl.pointer_makecell3 = handles.tabMakeCell_table.Data{eventdata.Indices
(1,1),1};
                if isempty(obj.tmn.mcl.pointer_makecell3)
                    obj.tmn.mcl.pointer_makecell3 = obj.tmn.mcl.pointer_next_makecell;
                end
                if ~isempty(obj.tmn.mcl.makecell_ind{obj.tmn.mcl.pointer_makecell3})
                    obj.tmn.mcl.pointer_track2 = obj.tmn.mcl.pointer_track;
                    obj.tmn.mcl.pointer_track = obj.tmn.mcl.makecell_ind{obj.tmn.mcl.
pointer_makecell3}(1);
                    obj.tmn.gui_imageViewer.highlightTrack;
                end
            end
        end
        %%
        %
        function obj = menuViewTracks_Callback(obj,~,~)
            handles = guidata(obj.gui_main);
            if obj.menu_viewTrackBool
                obj.menu_viewTrackBool = false;
                handles.muViewHT.Label = 'Show Tracks';
                for i = 1:length(obj.tmn.gui_imageViewer.trackCircle)
                    obj.tmn.gui_imageViewer.trackCircle{i}.Visible = 'off';
                    obj.tmn.gui_imageViewer.trackLine{i}.Visible = 'off';
                    obj.tmn.gui_imageViewer.trackText{i}.Visible = 'off';
                end
            else
                obj.menu_viewTrackBool = true;
                handles.muViewHT.Label = 'Hide Tracks';
                obj.tmn.gui_imageViewer.loop_stepX;
            end
            guidata(obj.gui_main,handles);
        end
        %%
        %
        function obj = tabMakeCell_pushbuttonNewCell_Callback(obj,~,~)
            obj.tmn.mcl.newCell;
            obj.tabMakeCell_loop;
        end
        %%
        %
        function obj = tabMakeCell_pushbuttonAddTrack2Cell_Callback(obj,~,~)
            obj.tmn.mcl.addTrack2Cell;
            obj.tabMakeCell_loop;
            obj.tmn.gui_imageViewer.updateTrackText;
        end
        %%
        %
        function obj = menuViewTime_Callback(obj,mymenu,~)
            handles = guidata(obj.gui_main);
            switch lower(mymenu.Label)
                case 'all time'
                    obj.menu_viewTime = 'all';
                    handles.muViewTimeAll.Checked = 'on';
                    handles.muViewTimeNow.Checked = 'off';
                case 'at present'
                    obj.menu_viewTime = 'now';
                    handles.muViewTimeAll.Checked = 'off';
                    handles.muViewTimeNow.Checked = 'on';
            end
            obj.tmn.gui_imageViewer.loop_stepX;
            guidata(obj.gui_main,handles);
        end
    end
end
```

Listing B.10: cellularGPSTrackingManualobjectcontrol.m

References

[1] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon, "Topological generalizations of network motifs," *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics*, vol. 70, p. 031909, Sept. 2004.

[2] S. Krishnaswamy, M. H. Spitzer, M. Mingueneau, S. C. Bendall, O. Litvin, E. Stone, D. Pe'er, and G. P. Nolan, "Conditional density-based analysis of T cell signaling in single-cell data," *Science*, vol. 346, pp. 1250689–1250689, Nov. 2014.

[3] P. Nurse and J. Hayles, "The Cell in an Era of Systems Biology," *Cell*, vol. 144, pp. 850–854, Mar. 2011.

[4] Hits, "The pharmaceutical industry is at a critical juncture," 2015.

[5] D. Jukam, B. Xie, J. Rister, D. Terrell, M. Charlton-Perkins, D. Pistillo, B. Gebelein, C. Desplan, and T. Cook, "Opposite feedbacks in the Hippo pathway for growth control and neural fate," *Science (New York, N.Y.)*, vol. 342, p. 1238016, Oct. 2013.

[6] U. Alon, *An introduction to systems biology: design principles of biological circuits*. No. 10 in Chapman & Hall/CRC mathematical and computational biology series, Boca Raton, FL: Chapman & Hall/CRC, 2007.

[7] Y. Lazebnik, "Can a biologist fix a radio?—Or, what I learned while studying apoptosis," *Cancer Cell*, vol. 2, pp. 179–182, Sept. 2002.

[8] C. F. Lopez, J. L. Muhlich, J. A. Bachman, and P. K. Sorger, "Programming biological models in Python using PySB," *Molecular Systems Biology*, vol. 9, p. 646, 2013.

[9] S. L. Spencer and P. K. Sorger, "Measuring and modeling apoptosis in single cells," *Cell*, vol. 144, pp. 926–939, Mar. 2011.

[10] F. Jacob and J. Monod, "Genetic regulatory mechanisms in the synthesis of proteins," *Journal of Molecular Biology*, vol. 3, pp. 318–356, June 1961.

[11] O. Sandler, S. P. Mizrahi, N. Weiss, O. Agam, I. Simon, and N. Q. Balaban, "Lineage correlations of single cell division time as a probe of cell-cycle dynamics," *Nature*, vol. 519, pp. 468–471, Mar. 2015.

[12] G. S. Martin, “The hunting of the Src,” *Nature Reviews. Molecular Cell Biology*, vol. 2, pp. 467–475, June 2001.

[13] D. Hanahan and R. A. Weinberg, “Hallmarks of cancer: the next generation,” *Cell*, vol. 144, pp. 646–674, Mar. 2011.

[14] M. S. Lawrence, P. Stojanov, C. H. Mermel, J. T. Robinson, L. A. Garraway, T. R. Golub, M. Meyerson, S. B. Gabriel, E. S. Lander, and G. Getz, “Discovery and saturation analysis of cancer genes across 21 tumour types,” *Nature*, vol. 505, pp. 495–501, Jan. 2014.

[15] F. H. Wilson, C. M. Johannessen, F. Piccioni, P. Tamayo, J. W. Kim, E. M. Van Allen, S. M. Corsello, M. Capelletti, A. Calles, M. Butaney, T. Sharifnia, S. B. Gabriel, J. P. Mesirov, W. C. Hahn, J. A. Engelman, M. Meyerson, D. E. Root, P. A. Jänne, and L. A. Garraway, “A Functional Landscape of Resistance to ALK Inhibition in Lung Cancer,” *Cancer Cell*, vol. 27, pp. 397–408, Mar. 2015.

[16] K. H. Vousden and C. Prives, “Blinded by the Light: The Growing Complexity of p53,” *Cell*, vol. 137, pp. 413–431, May 2009.

[17] H. C. Reinhardt and B. Schumacher, “The p53 network: cellular and systemic DNA damage responses in aging and cancer,” *Trends in genetics: TIG*, vol. 28, pp. 128–136, Mar. 2012.

[18] G. S. Chang, X. A. Chen, B. Park, H. S. Rhee, P. Li, K. H. Han, T. Mishra, K. Y. Chan-Salis, Y. Li, R. C. Hardison, Y. Wang, and B. F. Pugh, “A comprehensive and high-resolution genome-wide response of p53 to stress,” *Cell Reports*, vol. 8, pp. 514–527, July 2014.

[19] A. V. Gudkov and E. A. Komarova, “The role of p53 in determining sensitivity to radiotherapy,” *Nature Reviews. Cancer*, vol. 3, pp. 117–129, Feb. 2003.

[20] R. Schoenheimer, D. Rittenberg, G. L. Foster, A. S. Keston, and S. Ratner, “THE APPLICATION OF THE NITROGEN ISOTOPE N15 FOR THE STUDY OF PROTEIN METABOLISM,” *Science (New York, N.Y.)*, vol. 88, pp. 599–600, Dec. 1938.

[21] E. Eden, N. Geva-Zatorsky, I. Issaeva, A. Cohen, E. Dekel, T. Danon, L. Cohen, A. Mayo, and U. Alon, “Proteome Half-Life Dynamics in Living Human Cells,” *Science*, vol. 331, pp. 764–768, Feb. 2011.

[22] E. Batchelor, C. S. Mock, I. Bhan, A. Loewer, and G. Lahav, “Recurrent initiation: a mechanism for triggering p53 pulses in response to DNA damage,” *Molecular Cell*, vol. 30, pp. 277–289, May 2008.

[23] J. E. Ferrell and E. M. Machleder, “The biochemical basis of an all-or-none cell fate switch in Xenopus oocytes,” *Science (New York, N.Y.)*, vol. 280, pp. 895–898, May 1998.

[24] A. Goldbeter, *Biochemical oscillations and cellular rhythms: the molecular bases of periodic and chaotic behaviour*. Cambridge: Cambridge University Press, transferred to digital print ed., 2004.

[25] M. B. Elowitz and S. Leibler, “A synthetic oscillatory network of transcriptional regulators,” *Nature*, vol. 403, pp. 335–338, Jan. 2000.

[26] J. E. Purvis, K. W. Karhohs, C. Mock, E. Batchelor, A. Loewer, and G. Lahav, “p53 Dynamics Control Cell Fate,” *Science*, vol. 336, pp. 1440–1444, June 2012.

[27] S. Tay, J. J. Hughey, T. K. Lee, T. Lipniacki, S. R. Quake, and M. W. Covert, “Single-cell NF-kappaB dynamics reveal digital activation and analogue information processing,” *Nature*, vol. 466, pp. 267–271, July 2010.

[28] L. Cai, C. K. Dalal, and M. B. Elowitz, “Frequency-modulated nuclear localization bursts coordinate gene regulation,” *Nature*, vol. 455, pp. 485–490, Sept. 2008.

[29] R. Cao, C. G. Gkogkas, N. de Zavalia, I. D. Blum, A. Yanagiya, Y. Tsukumo, H. Xu, C. Lee, K.-F. Storch, A. C. Liu, S. Amir, and N. Sonenberg, “Light-regulated translational control of circadian behavior by eIF4e phosphorylation,” *Nature Neuroscience*, Apr. 2015.

[30] J. T. Mettetal, D. Muzzey, C. Gómez-Uribe, and A. van Oudenaarden, “The frequency dependence of osmo-adaptation in Saccharomyces cerevisiae,” *Science (New York, N.Y.)*, vol. 319, pp. 482–484, Jan. 2008.

[31] N. Geva-Zatorsky, E. Dekel, E. Batchelor, G. Lahav, and U. Alon, “Fourier analysis and systems identification of the p53 feedback loop,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 107, pp. 13550–13555, July 2010.

[32] P. Rue and A. Martinez Arias, “Cell dynamics and gene expression control in tissue homeostasis and development,” *Molecular Systems Biology*, vol. 11, pp. 792–792, Feb. 2015.

[33] ScienceNetLinks, “The Cells in Your Body,” 2015.

[34] T. Riley, E. Sontag, P. Chen, and A. Levine, “Transcriptional control of human p53-regulated genes,” *Nature Reviews Molecular Cell Biology*, vol. 9, pp. 402–412, May 2008.

[35] M. A. Allen, Z. Andrysik, V. L. Dengler, H. S. Mellert, A. Guarnieri, J. A. Freeman, K. D. Sullivan, M. D. Galbraith, X. Luo, W. L. Kraus, R. D. Dowell, and J. M. Espinosa, "Global analysis of p53-regulated transcription identifies its direct targets and unexpected regulatory mechanisms," *eLife*, vol. 3, May 2014.

[36] K. H. Vousden and X. Lu, "Live or let die: the cell's response to p53," *Nature Reviews. Cancer*, vol. 2, pp. 594–604, Aug. 2002.

[37] R. Beckerman and C. Prives, "Transcriptional Regulation by P53," *Cold Spring Harbor Perspectives in Biology*, vol. 2, pp. a000935–a000935, Aug. 2010.

[38] R. L. Weinberg, D. B. Veprintsev, M. Bycroft, and A. R. Fersht, "Comparative Binding of p53 to its Promoter and DNA Recognition Elements," *Journal of Molecular Biology*, vol. 348, pp. 589–596, May 2005.

[39] C. J. Brown, S. Lain, C. S. Verma, A. R. Fersht, and D. P. Lane, "Awakening guardian angels: drugging the p53 pathway," *Nature Reviews Cancer*, vol. 9, pp. 862–873, Dec. 2009.

[40] D. W. Meek and C. W. Anderson, "Posttranslational modification of p53: cooperative integrators of function," *Cold Spring Harbor Perspectives in Biology*, vol. 1, p. a000950, Dec. 2009.

[41] S. Y. Shieh, M. Ikeda, Y. Taya, and C. Prives, "DNA damage-induced phosphorylation of p53 alleviates inhibition by MDM2," *Cell*, vol. 91, pp. 325–334, Oct. 1997.

[42] M. S. Rodriguez, J. M. Desterro, S. Lain, D. P. Lane, and R. T. Hay, "Multiple C-terminal lysine residues target p53 for ubiquitin-proteasome-mediated degradation," *Molecular and Cellular Biology*, vol. 20, pp. 8458–8467, Nov. 2000.

[43] G. Gaglia, Y. Guan, J. V. Shah, and G. Lahav, "Activation and control of p53 tetramerization in individual living cells," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 110, pp. 15497–15501, Sept. 2013.

[44] Y. Tang, W. Zhao, Y. Chen, Y. Zhao, and W. Gu, "Acetylation is indispensable for p53 activation," *Cell*, vol. 133, pp. 612–626, May 2008.

[45] J.-P. Kruse and W. Gu, "Modes of p53 regulation," *Cell*, vol. 137, pp. 609–622, May 2009.

[46] T. Zhou, N. Shen, L. Yang, N. Abe, J. Horton, R. S. Mann, H. J. Bussemaker, R. Gordân, and R. Rohs, "Quantitative modeling of transcription factor binding specificities using DNA shape," *Proceedings of the National Academy of Sciences*, vol. 112, pp. 4654–4659, Apr. 2015.

[47] O. Timofeev, K. Schlereth, M. Wanzel, A. Braun, B. Nieswandt, A. Pagenstecher, A. Rosenwald, H.-P. Elsässer, and T. Stiewe, “p53 DNA binding cooperativity is essential for apoptosis and tumor suppression in vivo,” *Cell Reports*, vol. 3, pp. 1512–1525, May 2013.

[48] G. Lahav, N. Rosenfeld, A. Sigal, N. Geva-Zatorsky, A. J. Levine, M. B. Elowitz, and U. Alon, “Dynamics of the p53-Mdm2 feedback loop in individual cells,” *Nature Genetics*, vol. 36, pp. 147–150, Feb. 2004.

[49] E. Batchelor, A. Loewer, C. Mock, and G. Lahav, “Stimulus-dependent dynamics of p53 in single cells,” *Molecular Systems Biology*, vol. 7, p. 488, May 2011.

[50] L. T. Vassilev, B. T. Vu, B. Graves, D. Carvajal, F. Podlaski, Z. Filipovic, N. Kong, U. Kammlott, C. Lukacs, C. Klein, N. Fotouhi, and E. A. Liu, “In vivo activation of the p53 pathway by small-molecule antagonists of MDM2,” *Science (New York, N.Y.)*, vol. 303, pp. 844–848, Feb. 2004.

[51] A. Raj, P. van den Bogaard, S. A. Rifkin, A. van Oudenaarden, and S. Tyagi, “Imaging individual mRNA molecules using multiple singly labeled probes,” *Nature Methods*, vol. 5, pp. 877–879, Oct. 2008.

[52] A. Ponti, P. Vallotton, W. C. Salmon, C. M. Waterman-Storer, and G. Danuser, “Computational analysis of F-actin turnover in cortical actin meshworks using fluorescent speckle microscopy,” *Biophysical Journal*, vol. 84, pp. 3336–3352, May 2003.

[53] L. C. Huang, K. C. Clarkin, and G. M. Wahl, “Sensitivity and selectivity of the DNA damage sensor responsible for activating p53-dependent G1 arrest,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 93, pp. 4827–4832, May 1996.

[54] A. Loewer, K. Karanam, C. Mock, and G. Lahav, “The p53 response in single cells is linearly correlated to the number of DNA breaks without a distinct threshold,” *BMC biology*, vol. 11, p. 114, 2013.

[55] N. Geva-Zatorsky, N. Rosenfeld, S. Itzkovitz, R. Milo, A. Sigal, E. Dekel, T. Yarnitzky, Y. Liron, P. Polak, G. Lahav, and U. Alon, “Oscillations and variability in the p53 system,” *Molecular Systems Biology*, vol. 2, June 2006.

[56] E. J. Moding, M. B. Kastan, and D. G. Kirsch, “Strategies for optimizing the response of cancer and normal tissues to radiation,” *Nature Reviews. Drug Discovery*, vol. 12, pp. 526–542, July 2013.

[57] J. Brugarolas, C. Chandrasekaran, J. I. Gordon, D. Beach, T. Jacks, and G. J. Hannon, "Radiation-induced cell cycle arrest compromised by p21 deficiency," *Nature*, vol. 377, pp. 552–557, Oct. 1995.

[58] T. Abbas and A. Dutta, "p21 in cancer: intricate networks and multiple activities," *Nature Reviews. Cancer*, vol. 9, pp. 400–414, June 2009.

[59] K. C. Shen, H. Heng, Y. Wang, S. Lu, G. Liu, C.-X. Deng, S. C. Brooks, and Y. A. Wang, "ATM and p21 cooperate to suppress aneuploidy and subsequent tumor development," *Cancer Research*, vol. 65, pp. 8747–8753, Oct. 2005.

[60] J. D. Orth, A. Loewer, G. Lahav, and T. J. Mitchison, "Prolonged mitotic arrest triggers partial activation of apoptosis, resulting in DNA damage and p53 induction," *Molecular Biology of the Cell*, vol. 23, pp. 567–576, Feb. 2012.

[61] R. Khosravi, R. Maya, T. Gottlieb, M. Oren, Y. Shiloh, and D. Shkedy, "Rapid ATM-dependent phosphorylation of MDM2 precedes p53 accumulation in response to DNA damage," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 96, pp. 14973–14977, Dec. 1999.

[62] X.-P. Zhang, F. Liu, and W. Wang, "Two-phase dynamics of p53 in the DNA damage response," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 108, pp. 8990–8995, May 2011.

[63] Y. Shiloh and Y. Ziv, "The ATM protein kinase: regulating the cellular response to genotoxic stress, and more," *Nature Reviews. Molecular Cell Biology*, vol. 14, pp. 197–210, Apr. 2013.

[64] I. Hickson, Y. Zhao, C. J. Richardson, S. J. Green, N. M. B. Martin, A. I. Orr, P. M. Reaper, S. P. Jackson, N. J. Curtin, and G. C. M. Smith, "Identification and characterization of a novel and specific inhibitor of the ataxia-telangiectasia mutated kinase ATM," *Cancer Research*, vol. 64, pp. 9152–9159, Dec. 2004.

[65] L. Biddlestone-Thorpe, M. Sajjad, E. Rosenberg, J. M. Beckta, N. C. K. Valerie, M. Tokarz, B. R. Adams, A. F. Wagner, A. Khalil, D. Gilfor, S. E. Golding, S. Deb, D. G. Temesi, A. Lau, M. J. O'Connor, K. S. Choe, L. F. Parada, S. K. Lim, N. D. Mukhopadhyay, and K. Valerie, "ATM kinase inhibition preferentially sensitizes p53-mutant glioma to ionizing radiation," *Clinical Cancer Research: An Official Journal of the American Association for Cancer Research*, vol. 19, pp. 3189–3200, June 2013.

[66] Y. Lin, W. Ma, and S. Benchimol, "Pidd, a new death-domain-containing protein, is induced by p53 and promotes apoptosis," *Nature Genetics*, vol. 26, pp. 122–127, Sept. 2000.

[67] R. Bernardi, P. P. Scaglioni, S. Bergmann, H. F. Horn, K. H. Vousden, and P. P. Pandolfi, "PML regulates p53 stability by sequestering Mdm2 to the nucleolus," *Nature Cell Biology*, vol. 6, pp. 665–672, July 2004.

[68] A. Tinel, "The PIDDosome, a Protein Complex Implicated in Activation of Caspase-2 in Response to Genotoxic Stress," *Science*, vol. 304, pp. 843–846, May 2004.

[69] H. H. Park, E. Logette, S. Raunser, S. Cuenin, T. Walz, J. Tschopp, and H. Wu, "Death domain assembly mechanism revealed by crystal structure of the oligomeric PIDDosome core complex," *Cell*, vol. 128, pp. 533–546, Feb. 2007.

[70] L. Bouchier-Hayes and D. R. Green, "Caspase-2: the orphan caspase," *Cell Death and Differentiation*, vol. 19, pp. 51–57, Jan. 2012.

[71] T. G. Oliver, E. Meylan, G. P. Chang, W. Xue, J. R. Burke, T. J. Humpton, D. Hubbard, A. Bhutkar, and T. Jacks, "Caspase-2-mediated cleavage of Mdm2 creates a p53-induced positive feedback loop," *Molecular Cell*, vol. 43, pp. 57–71, July 2011.

[72] V. Lallemand-Breitenbach and H. de Thé, "PML nuclear bodies," *Cold Spring Harbor Perspectives in Biology*, vol. 2, p. a000661, May 2010.

[73] M. Pearson, R. Carbone, C. Sebastiani, M. Cioce, M. Fagioli, S. Saito, Y. Higashimoto, E. Appella, S. Minucci, P. P. Pandolfi, and P. G. Pelicci, "PML regulates p53 acetylation and premature senescence induced by oncogenic Ras," *Nature*, vol. 406, pp. 207–210, July 2000.

[74] L. Y. Peche, M. Scolz, M. F. Ladelfa, M. Monte, and C. Schneider, "MageA2 restrains cellular senescence by targeting the function of PMLIV/p53 axis at the PML-NBs," *Cell Death and Differentiation*, vol. 19, pp. 926–936, June 2012.

[75] M. Garcia-Calvo, E. P. Peterson, B. Leiting, R. Ruel, D. W. Nicholson, and N. A. Thornberry, "Inhibition of human caspases by peptide-based and macromolecular inhibitors," *The Journal of Biological Chemistry*, vol. 273, pp. 32608–32613, Dec. 1998.

[76] M. T. Hayashi and J. Karlseder, "DNA damage associated with mitosis and cytokinesis failure," *Oncogene*, vol. 32, pp. 4593–4601, Sept. 2013.

[77] T. Kuilman, C. Michaloglou, W. J. Mooi, and D. S. Peeper, "The essence of senescence," *Genes & Development*, vol. 24, pp. 2463–2479, Nov. 2010.

[78] D. Muzzey and A. van Oudenaarden, "Quantitative time-lapse fluorescence microscopy in single cells," *Annual Review of Cell and Developmental Biology*, vol. 25, pp. 301–327, 2009.

[79] T. R. Jones, I. H. Kang, D. B. Wheeler, R. A. Lindquist, A. Papallo, D. M. Sabatini, P. Golland, and A. E. Carpenter, "CellProfiler Analyst: data exploration and analysis software for complex image-based screens," *BMC bioinformatics*, vol. 9, p. 482, 2008.

[80] M. Castedo, J.-L. Perfettini, T. Roumier, K. Andreau, R. Medema, and G. Kroemer, "Cell death by mitotic catastrophe: a molecular definition," *Oncogene*, vol. 23, pp. 2825–2837, Apr. 2004.

[81] K. Jaqaman, D. Loerke, M. Mettlen, H. Kuwata, S. Grinstein, S. L. Schmid, and G. Danuser, "Robust single-particle tracking in live-cell time-lapse sequences," *Nature Methods*, vol. 5, pp. 695–702, Aug. 2008.

[82] "TrackMate," 2015.

[83] J. Bieler, R. Cannavo, K. Gustafson, C. Gobet, D. Gatfield, and F. Naef, "Robust synchronization of coupled circadian and cell cycle oscillators in single mammalian cells," *Molecular Systems Biology*, vol. 10, p. 739, 2014.

[84] C. A. Schneider, W. S. Rasband, and K. W. Eliceiri, "NIH Image to ImageJ: 25 years of image analysis," *Nature Methods*, vol. 9, pp. 671–675, June 2012.