POLITECNICO
DI TORINO

Master's Degree in Biomedical Engineering

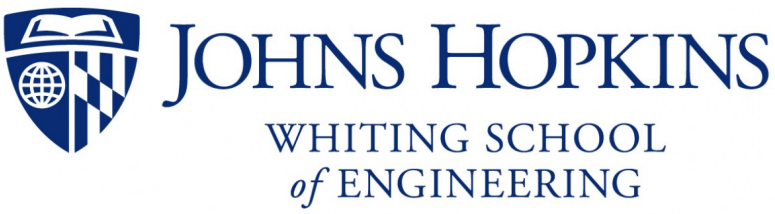
Master's Degree Thesis

Automated Tracking System for Identification of Tagged Mice for Automatic Social Behavior Analysis

Advisor
Danilo Demarchi
Co-Advisor
Ralph Etienne-Cummings

Candidate
Fabio Marcuccio

September 2018



The study presented in this dissertation was entirely carried out at the Computational Sensory-Motor Systems Laboratory of the Electrical and Computer Engineering Department at Johns Hopkins University Whiting School of Engineering, Baltimore, Maryland, USA, under the supervision of professor Ralph Etienne-Cummings, chairman of the department. The project was commissioned by the Pathology Department, division of Neuropathology, of Johns Hopkins University School of Medicine, headed by prof. Alena V. Savonenko, who kindly set up all the necessary to make the experiments feasible. All recordings, tests and experiments were conducted in her laboratory, under her attentive supervision.

I've roamed and rambled, and I followed my footsteps
to the sparkling sands of her diamond deserts;
and all around me, a voice was sounding:
this land was made for you and me.

Woody Guthrie

Acknowledgements

First and foremost, I would like to thank my professor and advisor, Danilo Demarchi. Without him, nothing of this experience would have ever been possible. His kindness and disposition in sharing his international contacts with the students are essential resources for Politecnico di Torino to extend its international network. Thank you for having had faith in me, and for having introduced me to the American scientific research world, that seemed so far and unattainable only up to a year ago.

In addition, I want to thank my American advisor and chairman of the Electrical and Computer Engineering Department at Johns Hopkins University, professor Ralph Etienne-Cummings. Being in Ralph's lab has been the most educational experience of my life. Doing research can be a frustrating and discouraging job, and finding someone who always believes in your person and your skills is not an easy deal. I had the chance to meet not only a boss, but a real leader who shares his knowledge to make his students grow up. That is the best a student can ask, and the best I could ask for.

I want to thank Alena V. Savonenko, professor of pathology at Johns Hopkins University School of Medicine, associated investigator at the Alzheimer's Disease Research Center at Johns Hopkins. She represented an important source of inspiration. Thank you, Alena, for hosting me in your lab, and for all the new ideas you provided, cheering me up during the hard times.

I would like to thank all the ECE Department staff at Johns Hopkins University. A special thanks to Nicole, Melissa, Cora, Makea, Eileen, Debbie and Dana. I'm missing so much your welcome every morning to the department and coming across you for the coffee break in the lounge. Thank you for making me feel at home.

I want to thank Amy and Mark Foster for providing me a desk in their lab. Being in a photonics lab has been extremely interesting and stimulating for my research.

I want to thank my family, for all the strength and support she gave me every single day of this five-year-long travel. Thank you, mum, for your sincere love and compassion, for making me sensitive to all the beauty surrounding me. You taught me the loveliness of traveling through the pages of a book, the life enclosed in every single flower petal, the importance of being together. I carry you inside my heart, as the most important treasure to protect. Thank you, dad, for your great understanding, for being the best advisor of my life. Thanks to you, I know that there is no better satisfaction than working for passion, that working on staying together is better than getting upset to be alone, that the way you love your land is the way she loves you. Thank you, brothers, for having raised me following your passions. There is no better teaching than the love you put in every thing you do. Feeling you close to me means simply feeling safe. Thanks to you all, growing up with you made me realize that life is made of little things wherein happiness resides, the same happiness you give me, as a gift, every single day of my life. I love you all.

I want to thank my extended family, my grandparents, uncles, cousins, sister-in-law and her family. I want to thank my wee nieces, of whom the youngest was born during my stay in the USA. Thank you, Rebecca and Adele, for having brought a new shiny light into my family.

A warm greeting to my uncle, Antimo. Thank you, uncle, for being my jack of all trades. Thanks to you, now, I know that there is always a solution.

I want to spare a special thought to my grandad, who passed away few months ago, and to my uncle, Lino, who has been waiting for longtime my graduation day and who is not able, now, to celebrate it with all of us. Thank you, uncle, for having had faith in me. Your goodness and likableness will be part of me, every step I will take. You will be celebrating with us anyway, wherever you will be.

In the United States, I want to thank all the crew at CSMSL I had the pleasure of working with. In particular, thank you, Adam, John, Bayo, Takeshi, Adam C., Tao, Patrick, Mich, Duncan for all the good time spent together. On Wednesday mornings, I still feel my mind longing for having lunch at Masala Kitchen.

I want to thank all the folks in Fosters' photonics lab. In particular, a big thanks to Milad, Bryan, Jasper, Alexander, Neil, Hongcheng, Michael, Kangmei for having welcomed me more as a friend than a colleague. I will never forget our laughs at happy hours, our beers, and all the great time spent together. Thank you, for having shared with me all your wise knowledge, the most precious gift you could have given.

I want to thank all my friends in doctor Bell's lab. Thank you, Alycen, for all the times you said "abbraccio", I can still hear your voice saying that. Thank you so much, Michelle, for all our never-ending walks through the streets of Hampden, for driving me back and forth to New Jersey, and for all our night talks on your front porch. I will never forget the small house at 3313 Chestnut Avenue with the two pink flamingos in the front yard, and all the rock' n roll we danced at the Cat's Eye Pub on Friday nights. I'm waiting for all of you here, in Italy, to have fun again.

Special thanks go to Milad, Alanna and their families. Thank you, guys, for having being so kind with me. I felt as being part of your family. You gave me the best thanksgiving day of my life as a gift I will carry with me. Thanks, Salar, for having crossed the ocean to be present at my graduation day.

I want to thank Christopher, the greatest bartender in the world. Thank you, Chris, for all your stories that lightened up my darkest nights. That counter came across love, laughs and friendship. I see your smile all the way from here.

I want to thank all my erasmus friends, for having been my companions in one of the most exciting experience of my life. In particular, all the crew of "los pollos hermanos", Gytis, Louis, Elie, Daria, Margherita, Robin, Marta, Marie, Louna, Kyra, Julie, Jedrzej, Henrik, Francesco, Filippo, Christopher, Clara and Serafina for all the great time we had in Budapest. I want to thank Alberto, Veronica, Alberto M. and Enrico, for being my brothers during my time in Hungary. I'm still longing for going back to our nights at Kobuci.

I want to thank Elena, my travel companion, for being a sister during my stay in the US. Thank you, Elena, for all our adventures, for all the miles driven together, and for having followed me through the paradoxes of a land so hard to understand, but so beautiful and fascinating at the same time. You belong to the best memories of my life. My heart is full of gratefulness because of what you did for me.

I want to thank my longtime friends Leonardo, Meryeme and Nico. Thank you, Leo, for being my blood brother along all these years, for all the road trips, the laughs and the adventures we lived together. Thank you for always taking the best of me. There is a long road through all the beauty of this world still waiting for us, and I'm ready to go down every single mile of it, with you as my partner.
Thank you, Mery, for our never-ending friendship. Your love and understanding have been the mainstay of my personal growth.
Thank you, Nico, for being the shoulder I could always rely on.

I want to thank my childhood friend, Marilde, for all the love she has been giving me for all these years. Thank you, Mari, for the delicacy with which you preserve our friendship.

I want to thank my dear friend Sara. Thank you Sara, for the deep emotion you transmit me all the times we meet. Your eyes shine, and my heart gets joyful.

I want to thank my neighbor and childhood friend, Cesare, for all the wisdom shared with me. Thank you, Cesi, I can still hear the engines of our lego cars coming from your basement.

I want to thank my sweetest friend, Flaminia. Thank you, Fla, for sharing beauty. The delicate way you take care of this world encourages me to apply myself to be the change we want to see. I want to thank Matteo and Monica too, for being so lovely with me. I'm sure, one day, we will get our way to the ecovillage.

I want to thank Alessandra, for all the love and strength she provided me in the first years of university. Thank you, Ale, for having had faith in me.

And last but not least, I want to thank all my old friends, university fellows, high school mates, and everyone who contributed, even with a tiny piece, to make me become the person I am now. Big thanks to Mariana, my lovely friend from Argentina. Special thanks to Simone, Stefano, Elisa, Mary, Martina C., Samantha, Alessio, Martina R., Sara F., my longtime partners. Sincere thanks to Ashkan, Vittorio, Luca, Greta, Chiara, Rossana, Alessia, Andrea, Turo, Albi bici, Stefania, Federica M., Federica G., Matteo, Davide, Hassan, and all the beautiful people I met at University. I hope I didn't forget anyone, and please, forgive me if I did.

Thanks to all of you. Nothing would have ever been possible without your support. There is no more reassuring emotion than feeling you beside.
I love you all.

See you on the road,

Fabio

Contents

Chapter 1

Introduction

Next Generation Sequencing (genotyping) has evolved significantly in the last decade, with an unprecedented increase in sequencing speed, decrease in sequencing cost, and overwhelming influence on all branches of health-related science. This progress has led to the development of thousands of genetic models of human diseases, and even more will be developed in the near future featuring more advanced variations of cell-type specificity. In this contest, contemporary bioscience has become Big Data science and it requires a development of computational algorithms and new statistical methods to be handled. In fact, passing from collecting big data sets to extracting knowledge from them is not trivial without adequate tools.

Established research institutions such as Johns Hopkins Medical Institute often conduct large scale behavioral studies on animal specimen, such as mice. As the sample sizes grow, the amount of labor to conduct becomes a limiting factor in the experiment. Classical behavioral testing of disease models is a time-consuming and laborious process that is vulnerable to the effects of human handling, lack of standardization and low reproducibility. State-of-the-art systems still require humans to detect, categorize and classify social behaviors and interactions. Methods that depend so much on humans can't be standardized and reproduced. This condition can not be suitable for a behavioral phenotyping system that aims to achieve some standards that can be reproduced in the same way all over the world. In addition, a non-standardized system brings about huge waste in terms of animal lives. The US government estimated that the number of animals employed in research in 2016 was over 820000 [1]. It is important to specify that this number includes only species covered by the Animal Welfare Act, excluding mice, rats, rabbits and any other kind of rodent that is protected under other kind of regulations. These regulations do not include an obligation in giving the precise number of rodents employed in research each year. Considering EU data, the 93% of animals employed in research do not include animals covered by the US Animal Welfare Act. This means that, if the same data were applied concerning the US, the amount of rodents employed in research would be around 12 millions [1]. This is just a rough estimate that does not

consider many other factors, but it already gives an idea of numbers. This massive use of animals is partially due to the lack of standardization of current behavioral phenotyping systems.

The extensive use of mice in disease modeling brings a need to build a computational system able to analyze and classify their social behaviors during drug testing. Social behaviors and interactions are very important for analyzing the effect of drugs injected in mice. The current state of the art algorithms being applied towards studying animal behavior is heavily underdeveloped. Recent studies are only able to follow the behavior of a single mouse, truly missing the implications of studying social behaviors. Unfortunately, this is not enough to get a good characterization of drugs avoiding human handling.

This study attempts to automate the actual analysis of mice behavior using implementations of modern computer vision techniques to process in real time videos recorded by a camera set on the lid of the cage wherein mice move. A real-time application is required because biological laboratories host thousands of cages placed one on the other; this means that thousands of videos would be recorded at the same time in an hypothetic non-real-time system, and storing all this information would require excessive storage capacity. In addition, the huge amount of cages and their relative position put a constraint in terms of available space and cost. That is why hardware must be small enough to fit the available space, and cheap enough to be reproduced for all cages. Biological laboratories are generally very dark, especially in night hours. Darkness of the working environment is another constraint we had to consider. In fact, light influences mice social behaviors and constitutes another issue that must be solved.

Even though the final aim of this project is the classification and analysis of mice social behaviors, the first obstacle we must overcome is the identification and identity preservation of mice moving within their home cage. State-of-the-art systems generally solve this problem considering one mouse or, at most, two mice moving within their environment. With one mouse only, the solution of the problem is trivial; with two mice, if the algorithm is able to identify the inversion of identities due to an error, it is also able to solve it swapping again the identities. In fact, with two mice, the problem is easy because if identities are swapped, the right ones are obviously the opposite ones. If the two-mice-tracking algorithm is not able to realize that an error is occurring and identities are lost, they can be re-established in a post processing correction, as shown by Stav Braun in her thesis [4]. When we consider three mice moving, things get way more complicated. According to Neuropathology Department of Johns Hopkins School of Medicine, behavioral studies should be done on cages wherein three or four mice move to obtain a good drug characterization. Starting from this statement, our study was focused on tracking and recognition of three mice moving within their home cage. The extension of the study to four mice doesn't represent a big issue as it will be explained in the following chapters.

Our experimental setup consists basically in a camera placed on the lid of the home cage from where videos are recorded. The camera is remotely controlled using Raspberry Pi technology. Four near-infrared LEDs are placed close to the cage to light up the environment and spherical metal tags are bound to mice ears to allow identification. Tracking and identification algorithms have been implemented in MATLAB 2017b using openCV libraries. The first step for social behaviors analysis consists in building a robust tracking algorithm able to track specific landmark points on mice contour. It has been studied that the best landmark points to categorize the most frequent and meaningful social behaviors are head and tail [8]. According to this, our study considers head and tail as main landmark points.

The main issue affecting the tracking system in this kind of environment is occlusion (or merging of segmented blobs) and it takes place at the step of moving objects segmentation. Occlusion happens when two (or more than two) mice are so close between each other to create a unique and indistinguishable big blob instead of two (or more) different ones. During occlusion, tracking landmark points is a big challenge because mice borders are totally undefined, identities are lost and all geometrical parameters are not consistent. According to Neuropathology department of Johns Hopkins School of Medicine, defining social behaviors during occlusion is not strictly important because interactions can be defined by relative positions of landmark points in frames before and immediately after occlusion occur. Strictly important is keeping the identities of mice frame by frame and restoring them in case occlusion happened. That's why we define two different trackers:

- No-Merging Tracker (**NM Tracker**) is run when merging doesn't occur and it is based on simple geometrical calculations of distances;
- Inner-Distance-Shape-Context Tracker (**IDSC Tracker**) is run in frames immediately after merging occurred and it is based on Inner Distance Shape Context, a shape descriptor defined by Ling and Jacobs at the Center for Automation Research and Computer Science Department, University of Maryland [11].

From here on out, when we say "Tracker" we refer to the whole tracker, while it will be specified in case we refer specifically to one of the two. The tracker was tested on videos wherein three mice move within their home cage and show the user an ellipse drawn around each mouse, head and tail position and centroid position. In addition, during merging, it still shows ellipses around mice and tries to keep track of centroid position and to define mice borders using clustering methods. Even though we said that we are not interested in tracking landmark points in frames where merging occurs, keeping track of centroid position and ellipses parameters would be a useful resource for better understanding social behaviors. Results show that in a good percentage of frames we managed to well separate mice during merging but there is a big number of frames where we didn't. This weak result makes the algorithm not

strong enough to be applicable in real systems. Anyway, if we could figure out a way to define what are the frames and relative mice positions that allow the algorithm to work well, we could apply it only when we know that it works. In particular, as it will be clearer at the end of this dissertation, since time is the biggest constraint we must consider, knowing centroids position during merging would allow us to employ an easier identification algorithm instead of using an heavier (but more accurate) one, speeding up the whole system.

Tests made on our tracking algorithm led to excellent results and made us think of applying the tracker not only to trace head and tail, but also left and right ear. Knowing ears position in frames after merging occurred would lead us to re-establish identities after they are lost because of occlusion. In fact, thanks to the available position of ears, metal tags segmentation would be extremely less difficult, as it is described in the following chapters. We extend the features of **IDSC Tracker** to accomplish these functionalities and the results we got in terms of identity detection are excellent. We applied the tracker to videos where only one mouse moves within his home cage and we managed to well segment the metal tag in almost every frame. The price we pay for this good detection is a high computational demand that makes the algorithm not suitable for real-time systems. Nevertheless, the high accuracy and precision achieved make the algorithm exploitable for an off-line analysis of mice shapes, laying the groundwork for a real-time application.

Chapter 2

Background

The surprisingly big expansion of genomics was made possible thanks to the automation of DNA sequencing, outcome of a collaboration between different kinds of specialists, from biologists to engineers. However, while genetics and genomics were expanding, it can not be said that our ability to functionally characterize brain of experimental animals has increased equally [18]. In fact, the incapability of handling this huge amount of data brought a necessity to increase the throughput in terms of development of new computational algorithms and statistical methods. In such a context as promising as disorienting, where thousands of new genetic models have been developed, contemporary bioscience has turned into Big Data Science and no other improvements will be made without adequate developments in computer science, as anticipated in chapter 1. All genetic models require to be studied on laboratory animals and an interpretation key must be found in order to analyze and getting oriented among all the data collected. This interpretation key can be investigated among animals social behaviors because, as Andreas Schaefer and Adam Claridge-Chang says in their article, *"the meaning of a brain is the behavior that it produces"* [18]. Since all behavioral studies and classification are nowadays manually executed by operators, it is clear that this field must increase the automation of behavioral assays. In fact, an automated tracking system for social behavior analysis would extremely improve behavioral test results, greatly decreasing manpower and costs. In addition, the lack of standardization due to human handling would be definitely overcome, allowing an operator-independent system that could be reproduced everywhere in the same way. Furthermore, the total number of animals used in the experiment would be broadly reduced because every animal could be used much more efficiently [20].

In conclusion, social behaviors are an extremely efficient target for automatization, as their classification requires expert-eye scoring from videos that turns into huge time and cost demand. Such a low-throughput system can not be suitable to cope with the improvements made in terms of neural models and to handle the huge amount of data deriving from them. This chapter will deal with the background of

this scene: first, a state-of-the-art overview will be made in order to get oriented among all the systems and devices currently available; then, main social behaviors will be described and the necessity of tracking specific landmark points on mice contours will be explained; in the end, main problems involving the automation of such a complex tracking system will be described and it will be clear why our study focuses so much on mice identification more than behaviors classification. In fact, identity detection and preservation is a key point in social behavior analysis without which no kind of behaviors classifier would be feasible, as already described in chapter 1.

2.1 State of the Art

As has already been said, a project that aims to automatically define social behaviors must firstly deal with mice identity tracking and then with social behaviors classification. Keeping track automatically of identity of multiple mice moving within their home cage is not a trivial issue. Today's typical approaches are generally based on hypothesis that animals are always visible, they never overlap and they don't move too quickly [15]. Other approaches use machine learning techniques and trained networks to identify mice and keep track of their identities. Although heuristcs and machine learning techniques could be useful tools for animals identification, they are often too complicated, making algorithms high demanding in terms of computational power. The most common nowadays systems employed in biological laboratories and the most recent studies are introduced below.

2.1.1 ANY-maze

ANY-maze is the most common system employed in nowadays biological laboratories. *ANY-maze* is today's most advanced video tracking system. It always tracks the center of the body but it can also be set to track head and tail. It can give you information about the area of the body and tracking can be done in every kind of environment in the homecage of the mouse (from here the name *ANY-maze*). The positive properties of this tracking system are the high reliability of the tracking, that can be done even in poor environmental conditions, and the quality of the tracking. In fact, head and tail positions can be tracked frame by frame with excellent results. In addition, the system can efficiently work with a wide range of equipment commonly used in behavioral tests. User-interface employed for initial setup is easy to use, allowing everybody to immediate get confidence with the software. ANY-maze is therefore a very efficient tracking system that can be purchased at a modest price.

The big limitation imposed by *ANY-maze* is the impossibility to execute a multiple tracking. In fact, the system is only able to keep track of one mouse or, at

most, of two mice with different color of the coat. This limitation is due to the incapability of the system to identify mice and maintain their identities, with the immediate consequence not to be able to perform a multiple tracking and, then, social behaviors classification. Here again, the importance of identity preservation is underlined as it represents the essential step for social behaviors analysis.

Another big obstacle the software poses for the automation of social behaviors analysis is the large operator-dependency required to start the tracking. In fact, the user is required to select the area wherein tracking must be run, the number of animals moving within the cage, the color of the coat and many other parameters (e.g. head position). The positive side is that after these short pre-tracking steps, the algorithm works efficiently; the negative one is that even the shortest not automated pre-tracking phase requires an operator to be done, and this is something we can not afford when implementing an automated tracker is the final aim.

Since *ANY-maze* is the most common tracking software for animals used by biological laboratories all over the world, it is clear that state-of-the-art systems are very far from allowing social behaviors classification, since they are not able to track multiple animals simultaneously.

2.1.2 EthoVision XT

EthoVision XT is a cutting-edge video tracking system for animal activity recordings. It is able to track body center, nose and tale for rats and mice, and it can measure the mobility of their body. It's a high-reliable system and it works even in poor lighting conditions. User is required to define some experiment settings, such as the number of arenas in which animals will be tracked (it is possible to select more than one arena), the points to be tracked (center mass, head, tail), the tracking source (video file or live image), and many others. Then, he has to define the detection method, choosing whether or not to use a scan window, adjust the contour settings, adjust sample rate and pixel smoothing of the video, and define the minimum and maximum size of the animals [22] [20].

To assure the best result in terms of object detection in any kind of experimental set-up, the system offers three different object detection methods: *gray scaling*, *static subtraction* and *dynamic subtraction*. *Gray scaling* works with a global thresholding filter. In fact, it takes all connecting points above or below a threshold defined in a initial step as a possible object. It is clear that this method, being based on global thresholding, requires an image where pixels belonging to object are in high contrast with background pixels. It works good if, for instance, mice pixels are white while background ones are black or vice-versa. It can not work when contrast is not sufficient to define a specific threshold [22] [21].

Static subtraction works with two different images: a first image of the background without animal and a second image including the animal. The rest of the

algorithm consists in a subtraction between the two images and in the consequent extraction of foreground (moving objects).

Dynamic subtraction works almost as *Static subtraction* detector with the difference that reference image is updated every frame. This detection method allows animals to be consistently traced even when background is changed because of external factors. In fact, changes of environmental conditions such as lighting, can affect background, obstructing tracking. *Dynamic subtraction* detector avoids this kind of situation.

Morphological operations such as erosion and dilation are used to adjust mice contours after segmentation. In fact, noisy background pixels can interfere with mice segmentation, influencing its final result. Erosion and dilation are combined in order to remove protruding pixels and accidental indentations, making the animal contour smoother.

The user is also required to set a minimum and maximum size for objects to be segmented. This function is used to prevent objects other than animals to be tracked.

The interesting advanced function introduced by EthoVision consists in the possibility to extract important parameters relative to nose, tail and center mass. Parameters are listed below [22].

- *Movement:* spatial movement of nose, body center and tail. A speed threshold is required to define what is to be considered as movement and what is not.
- *Distance moved:* distance traveled by nose, tail or center mass between two different frames.
- *Velocity:* distance traveled by nose, tail or center mass per unit time.
- *Direction:* direction of movement of nose, tail or center mass between two different frames;
- *Turn angle:* change in moving direction of nose, tail or center mass.
- *Angular velocity:* change in moving direction of nose, tail or center mass per unit time;
- *In zone:* zone wherein the mouse is standing in the current frame.
- *Elongation:* elongation percentage of animal's body. The function outputs the elongation state that can be *stretched*, *normal* or *contracted*.
- *Mobility:* Percentage of change in the animal's size between two subsequent frames. The function outputs the mobility state that can be *immobile*, *mobile* or *strongly mobile*.

All the parameters that can be extracted frame by frame are used to define ten different behaviors such as *grooming*, *sniffing*, *walking*, *resting*, *rearing*. A particular cage called "PhenoTyper" is used to define animal's social behavior.

EthoVision XT is therefore an extremely performing system in terms of singly-housed mice phenotyping and it reached human level performance. In opposition, it is not able to classify mice behaviors when more than one mouse is housed in the cage [4]. This lack doesn't allow social behaviors identification but single behavioral phenotyping only. In addition, human handling is required to set up the systems, making the algorithm highly operator-dependent. This characteristic is not affordable when an automated tracking system is aimed, as we already said for ANY-maze. That's why this a high-performance tracking system is still not enough for the automation of social behaviors analysis.

2.1.3 idTracker

idTracker is a video tracking software that keeps the correct identity of each individual for the whole video. This new technology is very promising because it tries to overcome the limit of tracking a single individual imposed by other systems currently available. The way idTracker works can be divided in two different steps: in a first step, the algorithm collects a series of images for each animal when animals trajectories are not crossing, and a fingerprint is extracted for each animal; in a second step, idTracker takes the image of a single animal to be identified and extract its fingerprint that is then automatically compared to fingerprints of all other animals until the best matching is found and identification is done. The system requires the user to provide three parameters: number of animals, segmentation threshold and minimum size [16].

The way idTracker works deserves a little more attention because it represents one of the first good results obtained in terms of multiple animals tracking and identification. First and foremost, each frame must be segmented to separate animals from background. Then, a set of individual animal images must be collected in order to build the set of reference images that will be used to extract a fingerprint for each animal. To accomplish this, the algorithm collects a set of images called "fragments" belonging to single animals in frames where all individuals are well separated and segmentation outputs as many blobs as animals. Since fragments are generally too short to collect enough reference images because animals trajectories generally cross very often, the algorithm extracts fragments at different times, only when all animals are separated. Then, another probability-based algorithm is used to accurately define what are the different fragments that belong to the same individual. In fragments where trajectories cross but the animals overlap is not excessive, erosion is executed in order to separate individuals. [17]. Once all reference images are collected, the algorithm takes each blob of each fragment and transforms

it to obtain its intensity and contrast map. Actually, not all blobs are transformed into their corresponding maps, but only blobs of the same individual whose postures are different. Intensity and contrast maps represent animals' fingerprint. Finally, for each frame of the video, the algorithm segments every animal and transforms the corresponding blob into its maps of intensity and contrast. Then, intensity and contrast maps of the current animal are compared with the maps of reference animals and the best matching is found. Identity is then assigned on the base of best matching. All steps are listed below.

1. **Segmentation:** sets of contiguous pixels composing the foreground are segmented and background is extracted. Artifacts due to fluctuations of illumination are eliminated dividing the intensity of each pixels by the average of intensities of all pixels for each frame. Segmentation algorithm works selecting pixels whose normalized intensity is below or above a intensity threshold and larger than a size threshold. Both thresholds are given by the user in a initial step according to the kind of animal to be tracked.

2. **Fragments extraction:** this step finds the parts of the video wherein one individual is moving without crossing any other one. A fragment is a set of blobs of subsequent frames that correspond to the same individual with high probability. The algorithm establishes that two different blobs belong to the same fragment if they overlap with each other and do not overlap with any other blob of the two frames. If this condition is not satisfied, another fragment is initialized [17].

3. **Transformation of images:** segmented images of each fragment, suitable for animal identification, are transformed into their intensity and contrast maps as follow:

   (a) for each image of an animal, intensities of every pair of pixels are taken and the distance between them is calculated. The set of points in the 3D space (i_m, i_n, d) corresponds to the color correlogram of the image.

   (b) the set of points in the 3D space (i_m, i_n, d) is then transformed into a 2D space using the sum of each pair of intensity values instead of their single ones. Each point of the 2D space is then characterized by the sum of intensities of two points of the image and the distance between them $(i_m + i_n, d)$. In this way, 2D-intensity map is extracted.

   (c) the same process is executed characterizing each point with the absolute value of the difference between intensity values of every couple of points in the image $(|i_m - i_n|, d)$. In this way, 2D-contrast map is extracted.

   These maps correspond to animal fingerprint and are invariant to translation and rotation. This feature makes them suitable for a direct comparison with

other maps.

4. **Comparison of images:** image comparisons are executed by differences between maps. According to this, maps are subtracted element-by-element, and the mean of absolute values of differences is taken. In this way, the difference between two images consists of two different numbers:

   Intensity Distance: number obtained calculating the mean difference between the intensity maps of the two images.

   Contrast Distance: number obtained calculating the mean difference between the contrast maps of the two images.

   The result of the comparison can be summarized by a number called *Summed Distance* that is the sum of intensity and contrast distance. The smaller is the summed distance, the higher is the similarity between the two images [17].

5. **Identification:** identification is a consequent step of image comparison. In fact, intensity distance between the image figuring the animal to be identified and all reference images is calculated. Then, the reference image showing the lowest intensity distance is selected and the corresponding animal identity is assigned to the investigated one. Then, the process is repeated using contrast distance. If both the comparisons give the same result, the identity is assigned to the corresponding animal. If the result is different, the identity is taken as ambiguous [17].

6. **Fragment identification:** given the fact that single image identification can make some mistakes due to high similarity between individuals, information about the same individual moving without crossing any other one are collected. In this way it is possible to avoid the errors due to maps comparisons. In fact, if animal trajectory is not crossing with any other one, a sudden change in individual identity can not take place. Though, if maps comparison mistakenly assign a different identity to an individual whose trajectory doesn't cross any other one, the error is immediately solved by a short analysis of the whole fragment [16].

A clear summary of single steps composing the algorithm is shown in figure 2.1 The algorithm has been tested by the developers with eight zebrafish moving within their home aquarium. Results they obtained are surprising, since they managed to keep animal identities even when occlusion events take place. An illustration of different results is shown in figure 2.2.

Even though idTracker looks a great new method for multiple animal tracking, there are some issues that make it not applicable on a large scale. In fact, although it is a free distribution software and it can be downloaded by the official website, it

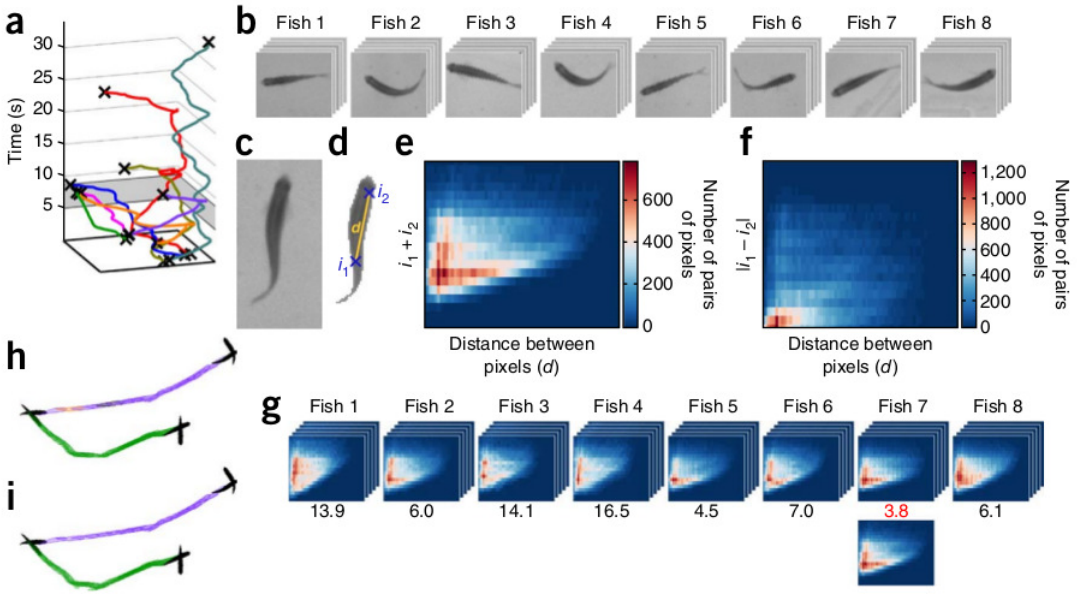


Figure 2.1: Illustration of algorithm working steps. (**a**) Fragments of trajectories of eight zebrafish. Gray portion of the graphics corresponds to period where all animals are well separated and trajectories do not cross. (**b**) Set of reference images collected for each individual. (**c**) Illustration of one single zebrafish. (**d**) Image in (**c**) after segmentation. Two random points on the blob and distance between them are shown. (**e**) Intensity map of segmented animal showing how many points are at a given distance and have a certain sum of intensities. (**f**) Contrast map of segmented animal. (**g**) Illustration of identification of a single individual. Single individual map (at the bottom) is compared with all reference maps. Numbers show the minimum intensity distance between the investigated individual map and all intensity maps of every other individual. The minimum is taken and identity is assigned to the investigated individual. (**h**) Trajectories of two zebrafish moving after two subsequent crossing. Different colors refer to different identities. (**i**) Same as (**h**), but here colors show identification of whole fragments of trajectories. Whole fragment identification corrects the error introduced by map comparisons.

is not easy to run with videos different from the reference ones. In addition, it is very slow to upload the video and to build the set of reference images used for mice identification. In fact, transforming each reference image in its intensity map is a high-computational operation that doesn't allow the system to work in real time. Moreover, information about trajectories, that are kept frame by frame in order to correct identities mistakenly detected by maps comparison, make the algorithm "memory based" with the consequence not to be able to recognize individuals in real time. Nevertheless, the study proposed by Alfonso Pérez-Escudero et al. [16] is a great achievement reached in the past years in terms of multiple animals tracking that was very inspiring for following studies, ours included.

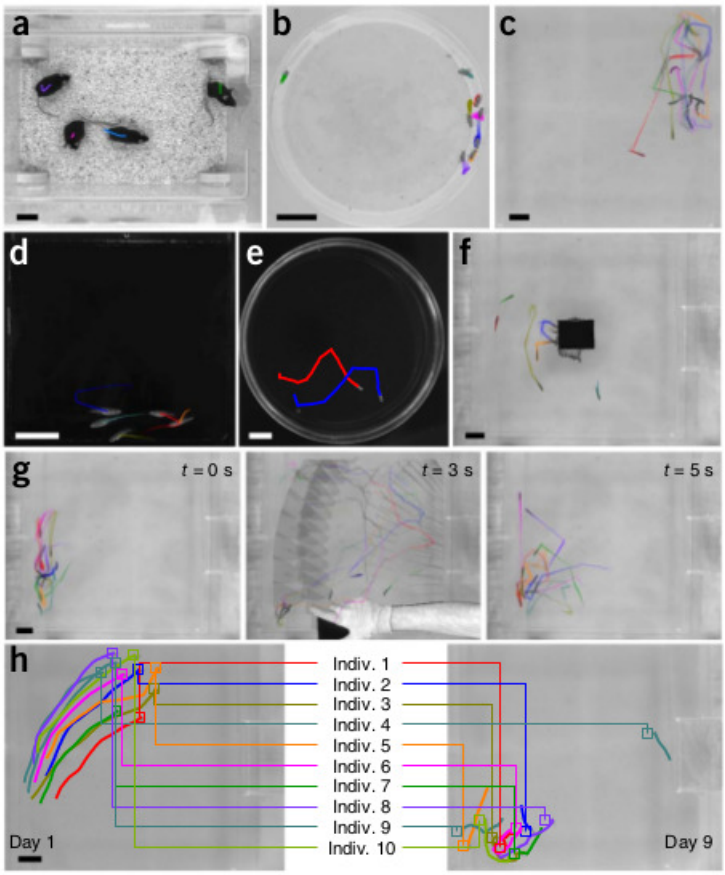


Figure 2.2: Illustration of application of idTracker. (**a-e**) Different frames of videos with animal trajectories traced by idTracker. (**f**) Frame taken by a video where zebrafish move with an object occluding part of the setup. Identities are correctly restored by idTracker. (**g**) Three frames taken from a portion of video where a hand waves under the camera. A frame with the hand waving is shown (center) and trajectories are correctly kept before (left) and after (right) the occlusion. (**h**) Frames of two different videos recorded in different days (day 1 and day 9). The same color refers to the same individual in different days.

2.1.4 RFID-based systems

Radio frequency identification (RFID) is a technology that uses electromagnetic fields to automatically identify and track tags attached to objects [26]. In general, an RFID-based system consists of three main components: one or more RFID transponders, one or more RFID readers and a data-processing unit. RFID trasponders (tags) can be active, passive or bactery-assisted passive:

- **Active tags** have on-board battery and periodically transmit ID signal to RFID readers;

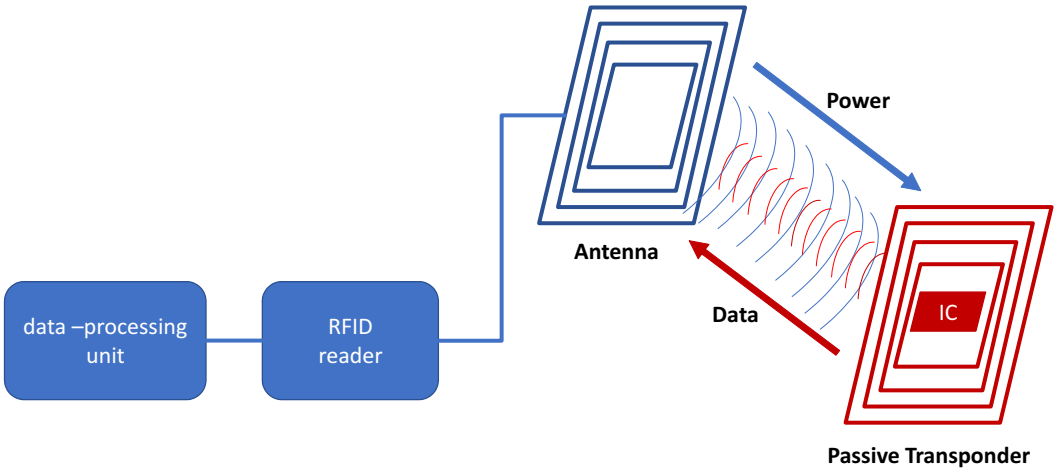


Figure 2.3: RFID technology with passive tag, overview. The RFID reader is powered by the data-processing unit and powers the passive tag sending energy through an antenna. The coiled antenna on the tag receives the energy and powers the chip (IC). The chip sends back radio-frequency wave through the antenna that is then read by the reader and processed by the data-processing unit.

- **Battery-assisted passive tags** have a small on-board battery that is activated only when an RFID reader is in proximity. They are smaller than active tags because of the smaller size of the battery;
- **Passive tags** do not contain a battery and the power is supplied by the reader. They consist in a coiled antenna and an integrated circuit. Coupling happens when radio waves coming from the reader are encountered by the tag. In fact, the antenna, contained in the tag, takes the energy from the reader and send it to the integrated circuit, powering it. The chip then generates a signal back to the radio-frequency system. This kind of phenomenon is called backscattering and signal that is sent back and interpreted by the reader is called backscatter. Passive tags can be classified according to the frequency bandwidth used that can be low, high or ultra-high.

Working principles and main blocks of a generic RFID system with passive transponder can be observed in figure 2.3.

RFID technology can be found in a huge amount of application. It is generally

used to track objects and to manage inventory. RFID is the basis of a lot of technologies we use in our daily life such as contactless payment with credit cards, toll collection, machine readable travel documents, car door lock, item level tagging in retail stores, etc [26]. RFID is also the technology used on chip implanted in pets for following their movements or for storing information regarding the owner, the race, date of birth, etc. In such a huge field of application, RFID technology was exploited also to create new tracking systems for multiple animals. One of the best high-performance ones is described below.

Luca Catarinucci et al. developed a passive, near-field, ultra-high-frequency RFID system for tracking the activity of laboratory animals, and they applied it to mice [5]. The tracking system designed by Luca Catarinucci et al. consists of near-field reader antennas working in ultra-high-frequency bandwidth, more precisely between 860 and 960 MHz. Reader antennas are made by segmented loops and they are placed below the experimental arena. A single reader is able to localize the animal in a squared cell of side 12 cm. In order to reduce artifact due to far field influence, each single antenna generates a magnetic field that is confined as much as possible in the relative cell. Antennas are connected to the reader through a multiplexer able to connect up to 32 channels. Though, 32 is the maximum number of antennas that can be used. Antennas are not powered at the same time but alternatively. The switching time between two antennas is within $200\mu s$. In this way, even in the worst case when all antennas are used at once, the latency between two signal detection will be around $6.4ms$, that is a period of time much shorter than the one required by a mouse to cross a single cell. Software consists in a data acquisition module and a web application. The former is used to acquire raw data coming from the antenna while the latter for data processing and analysis. Processing algorithm is not very articulated and this is a positive feature of RFID technology. In fact, when a single antenna is reading a tag implanted in a mouse, position is saved with no errors. The only problem occurs when more than one antenna detects the same tag because of artifacts or because the mouse is moving between two different cells. To correct this error, the algorithm uses RSSI (Received Signal Strength Indicator) to detect which reader is closer to the tag. Cell that presents a stronger signal detected by the reader is assigned as mouse's position. Once all data are processed by the algorithm, mouse's path is reconstructed. The system was tested on a arena with 12 antennas. As consequence, the whole area of the arena was divided in 12 cells. Results are shown in figure 2.4. Web application allows the user to find out a set of physical and statistical parameters that can be very useful for behavior analysis: the average time spent by each mouse on each cell, the time spent in isolation or locomotion, the time spent by different mice in aggregation on the same cell and many others. In addition, a space-time graph that gives information concerning single mouse movement can be calculated. In this way, overlapping different graphs relative to different mice, it is easier to identify period

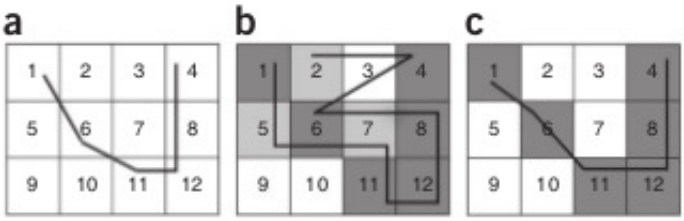


Figure 2.4: RFID system tested on arena with 12 antennas with one mouse moving. Picture was taken from Luca Catarinucci et al. paper [5]. (**a**) Real mouse path across the arena. (**b**) Reconstructed path based on raw data. Dark cells refer to ones rightly included in mouse's path. Light-shaded cells refers to ones mistakenly included in mouse's path. (**c**) Reconstructed path after data processing. Dark cells correspond to ones rightly included in mouse path.

of times where mice were aggregating on the same cell, or many other behaviors.

A validation test was carried in order to define the time spent by a mouse exploring a new object positioned inside the cage. Results were compared with the ones given by a manual scorer. Results showed significant differences between the time recorded by the manual scorer and the one obtained with RFID tracking system [5]. In fact, the RFID algorithm is only able to identify when the mouse is moving closely to the object but it is not able to detect whether the mouse is actually investigating the new object or is just moving around it. This is a skill owned by the manual operator, instead.

Even though RFID technology looks an interesting tool that can be applied in multiple animal tracking system, there is a series of negative characteristics that must be considered:

- RFID passive tags must be implanted in mice. Implantation is a surgical operation that requires a professional operator to be done. In addition, the period of time where each mouse is isolated because of the recovery after the operation could negatively affect mouse behavior. Even the implanted tag itself could be responsible of a changing in mouse behavior. Since tags must be implanted in every mouse, this kind of tracking system requires human handling, time, and costs due to surgical operations.

- Tracking systems that exploit RFID technology can not be very accurate in terms of object position detected. In fact, RFID tags are generally employed in application where high precision is not required. In a 71x44x33 cm cage, as the one we used at Johns Hopkins University School of Medicine, even a single centimeter can be important to analyze social behaviors (an evidence of this problem can be found in the significant error between RFID system and the manual scorer encountered in the previous experiment [5]). This issue finds

its roots in the nature of RFID. In fact, radio frequency tags give information about their position only in presence of the RFID reader. That's why a pixel-resolution tracking system (as the one obtain with computer vision techniques) could be developed only if we could reduce dimensions of receiver antennas to the dimension of a single pixel, that is clearly unfeasible. RFID technology is then very useful to track animals in application in which knowing the exact position of individuals is not relevant and a big uncertainty can be accepted. This is not the case of animals tracking for behavioral phenotyping.

- RFID hardware requires antennas and RFID readers, components that can be very bulky in terms of space. In fact, the available space for setting the hardware is very tiny in biological laboratories, as introduced in chapter 1. This limitation generally can not be satisfied by a RFID technology that is generally pretty hulking.

- The use of radio frequencies could influence mice behavior and distress animals, even though there is no evidence of this statement. Several studies are still on going to scientifically demonstrate that radio frequency waves do not disturb animals and do not have any influence in their behaviors.

In the light of experimental results, RFID techniques are very useful in application where high precision is not required, but present a series of not trivial issues that make these systems not very suitable for animal behavioral phenotyping. Nevertheless, RFID tracking system studies are still on going. In fact, the development of that kind of technology would be very promising in terms of ease of software employed for tracking and in terms of costs. In fact, systems like these do not use any kind of acquisition device such as camera, whose data require strong algorithms to be processed.

2.1.5 Shape-matching-based systems

The kind of environment in which a tracking system is supposed to work in biological laboratories suggests the use of vision-based algorithms. In fact, the placement of a camera on the lid of animals cages represents not a big issue, and videos can be easily recorded. In the last decades, new shape-matching-based methods were born to identify specific kind of shapes among many others. It is not a surprise that these kind of methods were developed only so recently; in fact, the last decade outcomes were impressive in terms of new computer vision techniques development and throughput. These improvements made the implementation of such kind of methods feasible.

The idea which these systems are based on is that a lot of information allowing humans to recognize animals and their parts are encoded in their shape and its components (contour, area, etc.). Remco Veltkamp and Michiel Hagedoorn says that

"matching deals with transforming a pattern, and measuring the resemblance with another pattern using some dissimilarity measure" [24]. According to this, a general shape matching algorithm must be composed at least of a geometrical transformation that turns a pattern into something more countable and standardized, and a dissimilarity measure that makes the output of the transformation comparable by simple arithmetic differences. In this way, a hard problem as comparing two different shapes to make association and find out which shape is more similar to the investigated one would be turned into an easy difference between numbers. According to this idea, new methods have been developed. Among these, shape context descriptor is one of the most relevant in terms of performance. It was introduced for the first time by Belongie, and it was extended by Jacob with the inner-distance shape context [2] [11]. These shape descriptors have been employed in several applications such as letters recognition, human silhouette tracing, inner structure of leaves identification and many others. Among all these, the most interesting one, according to the aim of our study, was developed by Stav Braun for tracking multiple mice [4].

In few words, shape context is a shape descriptor developed to measuring similarity between shapes for object recognition. Shape context gives description of the environment around each point belonging to a shape contour. The shape context at a reference point on a shape contour captures the distribution of the remaining points on the contour relative to it by measuring the euclidean distance between a single point and the remaining ones. In this way, each point is described by its surrounding environment [2]. This is based on the hypothesis that a shape subjected to movement can turn or translate moving its contours but the environment around each point will remain pretty much the same. Once each point is described by the shape descriptor, a point-to-point matching between two different shapes can be executed, and a measure of similarity can be extracted. Jacob extended this idea introducing inner distance and using it instead of euclidean one. This improvement was fundamental to make the descriptor sensitive to complex and articulated shapes [11]. These methods will be accurately described in next chapters, since they represent the core of our study. Braun exploited these descriptors to create a point-to-point matching of mice shapes for their tracking, managing to follow head and tail position frame by frame. In addition, he proposed a solution for solving the problem of occlusion, the artifact that occurs when mice are so close to be segmented under one unique blob. Results obtained were very interesting and laid the groundwork for our study. The reason why Braun's work can not be employed in multiple mice tracking is basically due to the huge computational demand required by shape matching algorithms applied frame by frame. In fact Braun detects head and tail position in each frame using every time inner distance shape context. The result is excellent in terms of accuracy but weak in terms of temporal resolution. Our aim is building a system that can be used in real-time application, as introduced in chapter 1, and this kind of algorithm can not satisfy our need.

In addition, only two mice were taken in account in Braun's study . With two mice, solving the problem of identity preservation is not a big issue because the algorithm could be implemented to be able to realize when identities are swapped. In this case, if the algorithm launches an error, identities are swapped again and are in this way re-established. In addition, a post processing algorithm can be used to correct frames where identification was wrong. With three mice things get way more complicated, and Braun's algorithm must be revisited [4].

Problems relative to Braun's study in tracking multiple mice are resumed below.

- Shape descriptors are applied frame by frame for head and tail tracking. This makes the algorithm extremely slow and exclude any kind of possible real-time application.

- Tracking is tested only on two mice moving within their home cage. Identity preservation and consequent behavioral phenotyping become more difficult in case of tracking more than two mice. According to this, new methods must be developed to keep mice identity over time.

- Identity preservation during occlusions works on mice moving between two consequent frames. Mice at current frame are labeled as m_1 and m_2 while mice at previous frame as p_1 and p_2. Then, the algorithm works as follow: if

$$area(m_1 \cap p_1) + area(m_2 \cap p_2) > area(m_2 \cap p_1) + area(m_1 \cap p_2) \qquad (2.1)$$

  then mouse m_1 corresponds to mouse p_1 at the previous frame and mouse m_2 corresponds to p_2 and vice versa if the result of the inequality is the opposite.

  This method works well only if points belonging to each mouse are known a priori. Braun's study uses shape context descriptor to find out which points belong to each mouse contour as described in section 3.6.3 and 3.6.4 of his thesis [4]. To do that, a set of reference shapes is required to start the matching. Working principle is shown in figure 2.5. Even though the algorithm seems to work very well in contours separation, the extension to multiple mice can not be easily implemented. In fact, with more than two mice, the set of reference shapes would be drastically enlarged (because there are more possible configurations of interactions) with the consequent increase in terms of computational time, a limitation that real-time systems can not afford.

- Occlusion events are treated creating a database of all possible shapes that two mice can assume interacting. Then, when occlusion occurs, point-to-point shape matching is run between the big segmented blob containing the two investigated mice and the reference shapes representing two mice interacting in all possible configurations. The best matching is taken and behavior is assigned according to the database. This method has many limitations such

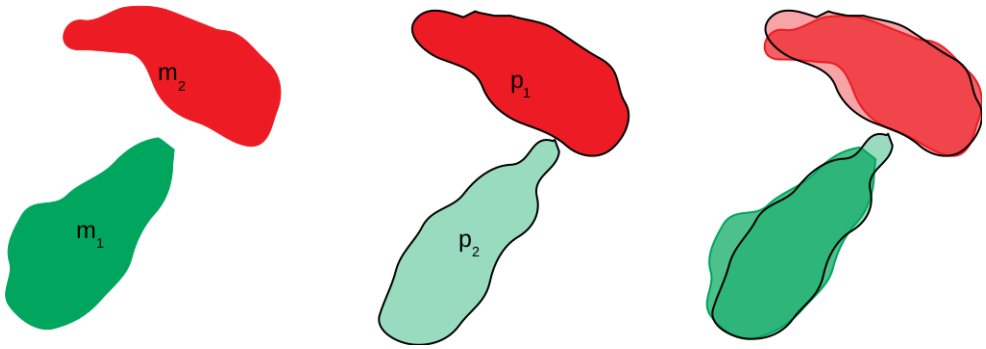


Figure 2.5: Algorithm working principle during occlusion. Left, mice at current frame are represented. Center, mice at previous frame are represented. Right, area overlap is shown. It is clear that mouse $m1$ corresponds to mouse $p2$ and $m2$ to $p1$.

> as an extreme standardization of mice behaviors and a difficult extension to multiple mice tracking.

Even though Braun's algorithm presents some limits in terms of multiple mice tracking (with more than two mice) and consequent expansion to behavioral phenotyping, we considered his system an innovative technique that can be improved to accomplish our needs. For this reason, roots of our studies reside in her work.

2.2 Social Behaviors and their Classification

Social behaviors have been matter of study since the first experiments on animals were carried out. Despite already underlined in the introduction of this chapter, I think that it is important to keep in mind that social behaviors represent the concrete expression of brain functioning. That is why I believe significant to highlight again a quote by Schaefer and Claridge-Chang which enhances the importance of social behaviors as matter of study [18]:

> The meaning of a brain is the behavior that it produces.

According to this thought, our study was entirely directed to social behaviors classification with a focus on one of the first problematics encountered: mice tracking.

Many studies have been done to find out what are the influences of diseases on animal social behaviors, and results collected during the years are extremely significant. Just to cite some of the recent studies, a strong correlation has been found between autism and changes in mice behavior. In fact, individuals subjected to autism disorder showed a significant increase in self-grooming behavior [14]. Another big achievement was given by the analysis of grooming behavior and its utility in studying animals stress, anxiety and depression. In fact, it was discovered that

behavioral analysis of mouse self- and hetero-grooming is a rich source of information for classifying many important mental disorders. In addition, it was demonstrated that mice affected by anxiety and depression show a relevant increase in some specific behaviors such as hypo-activity, aggression and self-aggression [19]. Another important goal was achieved by the department of neuropathology at Johns Hopkins University School of Medicine in studying the influence of environmental enrichment on cognitive deficits in a mouse model of Alzheimer's disease. In their work they found a strong correlation between environmental enrichment and significant improvements of cognitive performance in mice affected by Alzheiemer's disease. In particular, they found out that mice who experienced an environment provided with nesting material, novel objects, exercise wheels and hiding tubes, showed a significant increase in social interactions with a improvement in terms of ability to learn and memorize. In addition, male mice enhanced their aggressive behaviors with others, increasing fights and clashes [10].

All these are good examples that underline again the importance of social behaviors classification. According to the Department of Neuropathology at Johns Hopkins University School of Medicine and to recent studies, there are six main social interactions and three individual behaviors that must be classified [8]. Social interactions are:

1. **Nose-to-body (body sniffing):** the nose of the investigated mouse touches the body of the other or points towards it at a distance that must be less than $0.5cm$.

2. **Nose-to-nose (head sniffing):** the nose of the investigated mouse points the nose of the other at a distance that must be less than $0.5cm$.

3. **Nose-to-genitals (anogenital sniffing):** the nose of the investigated mouse points the genitals of the other at a distance that must be less than $0.5cm$.

4. **Crawling:** the investigated mouse crawls over or under the other one.

5. **Following:** the investigated mouse walks on the same direction of the other one at a distance that must be less than $10cm$.

6. **Stand together:** the investigated mouse is not moving and it is standing in close contact or at a distance of less than $3cm$ from the other mouse. Heads of mice are pointing in different directions.

Individual behaviors are:

1. **Walk alone:** the investigated mouse moves within his home cage without interacting in any way with the other mouse.

2. **Stand alone:** the investigated mouse doesn't move and stands at a minimum distance of $10cm$ from the other ones.

3. **Self-grooming:** the investigated mouse licks its paws and moves his body very fast. From a top view, the body looks shorter and assumes a kind of circular shape. In addition, his centroid oscillates very fast around the same point with very short fluctuations.

An illustration of all social and non-social behaviors is showed in figure 2.6.

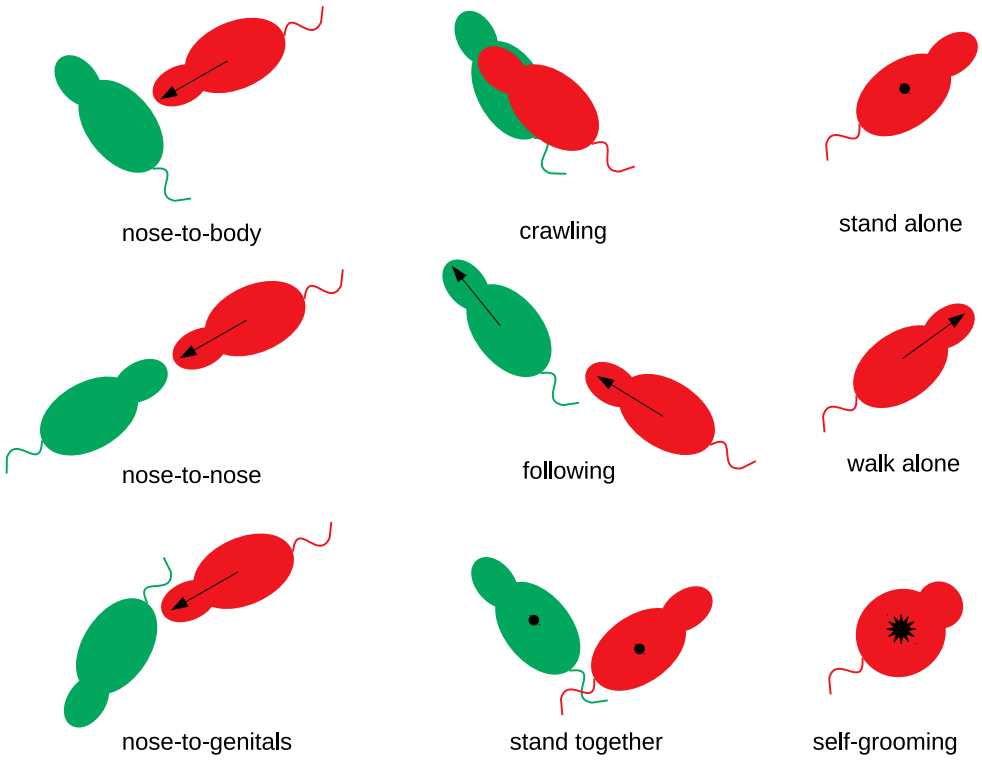


Figure 2.6: Illustration of principal social and non-social behaviors. The first two columns refer to social behaviors, the last column refers to individual behaviors. Red mouse is the investigated one. Arrows indicates direction of movement. Black full circles indicate absence of movement. Self-grooming behavior is identified by a black circle in correspondence of center mass with several tips indicating undefined directions of fast movements.

All these assumptions were found in literature and were enriched by the collaboration between the Electrical and Computer Engineering Department and the Neuropathology Department of Johns Hopkins University [23] [8].

2.3 Landmark Points

According to the department of Neuropathology at Johns Hopkins University School of Medicine, behaviors showed in figure 2.6 are fundamental for social interaction analysis. As a matter of fact, those behaviors are the same that are considered in nowadays laboratories during manual analysis.

Given the fact that a tracking algorithm must be initialized to track specific points, these lasts definition was matter of discussion in the first steps of our study. It was finally decided that head and tail represent the most relevant and useful points to identify social behaviors. We refer to head and tail as *landmark points*, and they are defined as follows:

head: the point positioned on the portion of mouse contour inherent to its head, in the center of the curve that links the ears and traces head edge.

tail: the point positioned on the portion of mouse contour inherent to its back, in correspondence of the point where tail begins.

A stylized illustration of head and tail as landmark points is shown in figure 2.7. Since the algorithm used to find out the exact position of head and tail is pretty heavy (see chapter 4), it is possible to approximate landmark point locations with the endpoints of the major axis of the ellipse that best fits pixels belonging to the mouse. In fact, head and tail are, for the most of the time, very close to major axis endpoints, as visible in figure 2.7. This part will be comprehensively explained in chapter 4.

Actually, our tracker was not implemented only to track head and tail positions, but also left and right ear. Even though the identification algorithm will be comprehensively described in following chapters, it is here introduced to understand why left and right ears are important points for our system. In fact, to solve the problem of mice identification, we decided to bind spherical metal tags to mice ears. In this way, we were able to identify up to four mice at the same time: left ear tag, right ear tag, both ears tags, no tags. The identification algorithm exploits advanced computer vision techniques to segment tags and to find their positions. Once tag locations are found, mice are identified as consequence. Therefore, in the identification algorithm, when we talk about landmark points we refer to four points: head, tail, left and right ear. We define:

left ear : the point positioned on the portion of mouse contour inherent to its left ear, in the center of the curve that traces ear edge.

right ear : the point positioned on the portion of mouse contour inherent to its right ear, in the center of the curve that traces ear edge.

Ears are generally brighter than the rest of the body, therefore, intensity of pixels relative to ears will be higher than surrounding pixels and they won't be segmented with the body. This is traduced with a curve indentation of mouse contour in proximity of the ears. This phenomenon is showed in figure 2.7.

(a) *Head (red) and tail (green)* (b) *Left ear (cyan) and right ear (yellow)*

Figure 2.7: Illustration of landmark points on stylized segmented mouse shape. Ellipse that best fits pixels inherent to mouse is shown in (**a**). Indentation phenomenon due to the different intensity of pixels relative to ears is shown in (**b**)

2.4 Main Issues Facing Multiple Mice Tracking

The first obstacle encountered in the implementation of a multiple mice tracker is occlusion, as already introduced in chapter 1. In fact, when mice move within their home cage, they can interact, crawl on each other, touch each other's body, etc. These interaction are not significant apparently, but they become a big problem in phase of segmentation. In fact, a digital algorithm doesn't make any difference between pixels belonging to different mice, and it segments them all together. The result is a segmentation in which a big unique blob appears as segmented object. That blob contains pixels relative to more than one mouse, and separating pixels belonging to different mice, to get as many blobs as the number of mice interacting, is not trivial. Since merging will be a recursive topic of this dissertation, it is important to give a definition of it. We define *"merging"* the phenomenon that takes place when two (or more) mice are so close between each other to be indistinguishable by common algorithms that segment them as they were a single object. Merging can be of two main different kinds:

- **Partial** when distinguishing an approximate shape of interacting mice is still possible. In this case, a human could still be able to point head and tail locations;
- **Complex** when segmented blob is so undefined and shapeless that distinguishing different mice interacting is not possible.

Partial and complex merging are well illustrated in figure 2.8. The consequences of this recurrent phenomenon affect the performance of tracking. A trained eye would still be able to assign mice identities even in case of merging. The same can not be said for a digital algorithm. In fact, when merging happens, landmark points are completely lost and identities are not available anymore. These issues drastically compromise the correct functioning of the tracker. In addition, since merging happens very often in a small environment such as the home cage, an algorithm that is not able to face the consequences of this phenomenon would never find a real application in biological laboratories. That is the main reason why all state-of-the-art algorithms have never been employed for multiple mice tracking, and that is also the reason why our work was entirely focused in facing this problem. According to this, we propose an algorithm that is able to identify landmark points frame by frame, when merging does not occur, and to restore mice identities in case merging takes place.

2.5 Aim of the Study

Our study aims to implement a tracking system for multiple mice using modern computer vision techniques to process videos recorded by a camera placed on the lid of the cage wherein mice move. Such kind of system would lay the groundwork for automatic social behavior classification, since the main problem affecting the latter is multiple mice tracking, with consequent mice identity preservation. Previous paragraphs showed the most common state-of-the-art systems employed in nowadays biological laboratories and highlighted merits and defects of each one of them. It was then remarked the importance of the automation of social behaviors classification, since it is a time- and money-consuming activity that is affected by lack of reproducibility and standardization. Our system was implemented to face the following conditions:

1. **Video recorded by a camera set on the lid of the cage:**

   cages are small and recording a video that covers all their area (walls included) is not trivial. A camera with a wide field of view is required.

2. **Poor lighting conditions:**

   biological laboratories are generally very dark, night and day. Mice are nocturnal animals and bright environments drastically affect their social behaviors. We propose a solution with NIR LEDs placed close to the cage.

3. **Tiny space for hardware placement:**

   in standard biological laboratories, cages are positioned vertically one on the other, leaving a very tiny space for the data-processing unit. Hardware must be small, with all the limitations concerning it.

4. **Precise landmark points tracking:**

   head and tail must be tracked frame by frame for the whole duration of the video, for a future behavior classification. Left and right ears must be tracked only when required for metal tags identification, or rather, when merging occurs and identities must be restored.

5. **Identity detection and preservation:**

   mice identity must be preserved frame by frame for the whole duration of the video. Identity is lost during merging events, and it is then restored, by tags identification, when mice separate. This means that metal tags must be surgically bound to mice ears before tracking.

6. **Real-time application:**

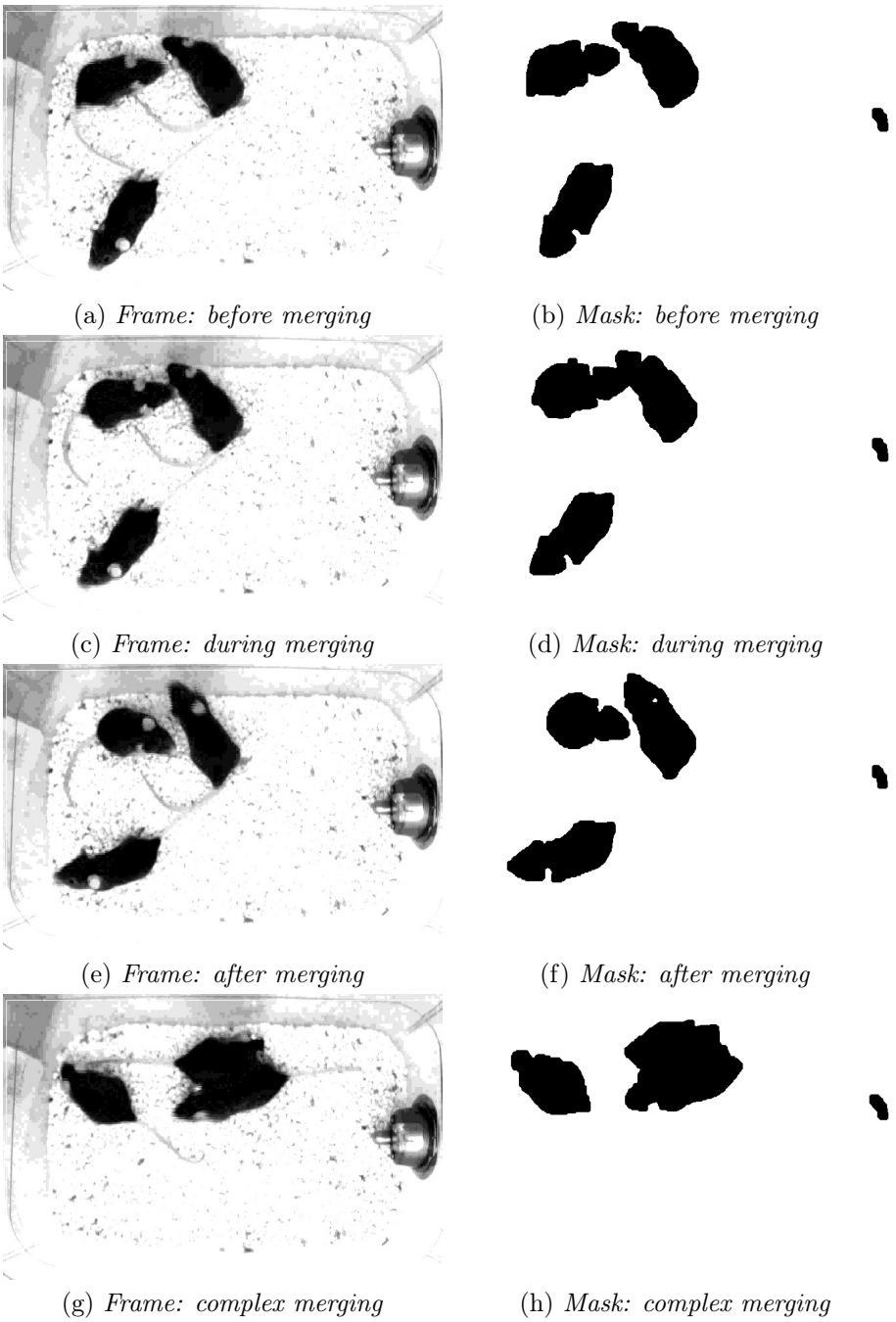
   the algorithm must work in real time. Laboratories host thousands of cages and storing data for all of them is not affordable. Videos must be processed in real time. In this way, no data will be stored and results will be immediately available.

7. **No human intervention:**

   the system must be completely automatic. Human intervention is only requested for binding metal tags to mice ears. But this is beyond the algorithm.

All the listed issues affecting multiple mice tracking were taken in consideration in the implementation of our algorithm. A detailed solution proposal will be described in the following chapters of this dissertation.

So, to sum up, we propose an automated tracking system for identification of tagged mice for a future automatic social behavior analysis. In particular, the system proposed attempts to track head and tail positions frame by frame, when merging does not occur, and left and right ear in case merging occurred, with consequent need for identities to be restored. In this last case, identities are detected through metal tag segmentation and recognition. In frames where merging occurs, the algorithm attempts to follow mice centroid position and separate points belonging to different mice interacting, and this is all it does. As a matter of fact, according to the Neuropathology Department of Johns Hopkins University School of Medicine, behavioral phenotyping during merging is not important, or rather, it does not represent the main feature. In fact, hypothesizing that merging phenomena happen quickly, social interactions can still be classified by the relative position of landmark points immediately before and after occlusion.



(a) *Frame: before merging* (b) *Mask: before merging*

(c) *Frame: during merging* (d) *Mask: during merging*

(e) *Frame: after merging* (f) *Mask: after merging*

(g) *Frame: complex merging* (h) *Mask: complex merging*

Figure 2.8: Illustration of partial and complex merging on frames taken from a video. (**a**) and (**b**) show the instant immediately before merging occurs on the averaged frame and its mask. (**c**) and (**d**) show the instant while merging occurs. In particular, (**d**) shows the partial merging of the two blobs belonging to different mice. (**e**) and (**f**) show the instant immediately after merging occurred on the averaged frame and its mask. (**g**) and (**h**) show a case of complex merging: in this case, segmented blob is completely shapeless and mice borders are not even perceivable.

Chapter 3

Experimental Setup

Before dealing with the development of the algorithm, it is necessary to analyze the experimental setup by treating its individual parts. In fact, many limits listed in the previous chapter concern not only problems related to software, but also to hardware. The tiny space available for placing the data-processing unit, the short distance between moving mice and the camera, the poor lighting conditions of biological laboratories and mice sensitivity to visible spectrum are all important factors that must be considered in hardware design. All the choices made in terms of hardware are described and motivated in this chapter, and a brief description of every component of the system is provided. Experimental setup with all its individual parts is illustrated in a stylized form in figure 3.1 and in the real application in figure 3.4.

3.1 Camera

The first problem we came across in the realization of our system was the choice of the camera. At the beginning, we tried to figure out which was the best technology to use between thermal and NIR cameras. In fact, mice are living creatures and, as all of them, they show a higher temperature in comparison with the surroundings. Thermal videos allowed us to get excellent results in terms of segmentation but videos were recorded with the open cage, without the lid. In fact, the plastic of the cage constituted an obstacle for thermal radiation because it was not transparent to it. Finding a material transparent to thermal radiation that could be employed in all cages was not an easy deal. In addition, thermal cameras are very expensive and finding good tags for mice identification was not easy. That's why our attention was focused on NIR cameras. These lasts are less expensive, do not present any problem of obstruction due to the lid of the cage, allow a good segmentation and make simple metal tags visible. The only difficulty is finding a camera with a wide near-infrared sensitivity spectrum. Merits and defects of both technologies are listed below.

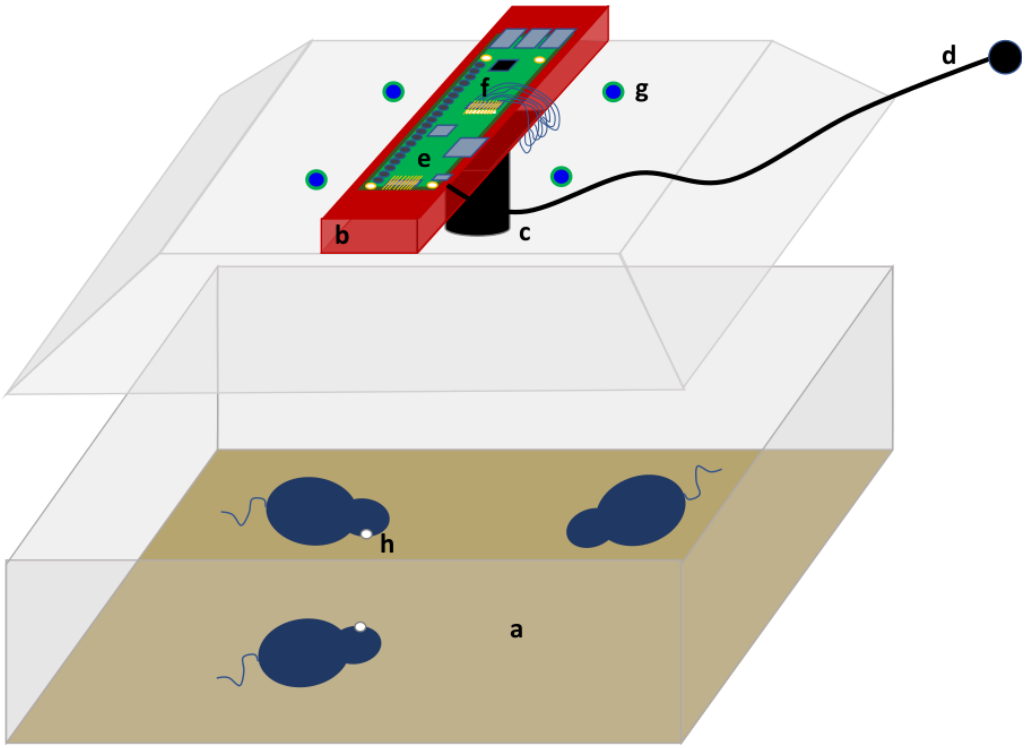


Figure 3.1: Stylized illustration of the experimental setup. (**a**) is the base of the cage of size 28x17.5x13cm. (**b**) represents the support under which the camera is placed and on which the Raspberry Pi is positioned. (**c**) shows the camera module of size 24x24x9mm; power cables of the camera are omitted. (**d**) is the Raspberry Pi usb power cable. (**e**) is a illustration of the Raspberry Pi, where pins relative to the camera module are indicated by (**f**). (**g**) is one of the four LEDs positioned on the lid of the cage. LED's wavelength is 850nm. (**g**) is the metal tag. For the experiments, right tag, left tag and no tag to identify three different mice were used.

- **Thermal:** excellent segmentation, expensive cameras, occlusion due to the lid of the cage, complicated research of good thermal tags;
- **NIR:** good segmentation, cheap cameras, no occlusion due to the lid of the cage, easy metal tags are visible, difficulty in finding a camera with a wide near-infrared sensitivity spectrum.

In addition, as anticipated in the introduction of this chapter, the short distance between the moving mice and the camera module (as visible in figure 3.2) represents a strong constraint in the choice of this last. In fact, standard cage dimensions are generally 28.5x17.5x13cm and the camera must cover the whole area, walls included (because we want to track mice which try to climb walls). Since the camera is placed on the lid of the cage, in a centered position that is about 3cm above the top of the

walls, the field of view of the camera must be necessarily above 156°. In particular: if $b = 3cm$ and $c = 28.5/2 = 14.25cm$, then

$$\alpha = \arctan \frac{c}{b} \simeq 1.36rad = 78°.$$

Since we are interested in 2α,

$$2\alpha = 156° \longrightarrow FOV = 2\alpha \geq 156°.$$

Since there is no camera with $FOV = 156°$ in the market, we employed a camera with $FOV = 160°$. In particular we chose the SainSmart NO IR wide angle camera module compatible with raspberry Pi model A and model B. The absence of IR filter is important because the camera must be sensitive to NIR waves. The short mathematical demonstration of the minimum field of view required for the camera is clearer observing figure 3.2, while figure 3.3 proves that a field of view of 160° is enough to cover the whole area of the cage. The camera features with OV5647 sensor that has a native resolution of 5MP for still images and supports 1080p at 30fps, 720p at 60fps and 640x480p at 60/90fps video recording. In addition, the important feature of this camera is its sensitivity spectrum. In fact, it shows a high sensitivity for long wavelength such as the ones belonging to near-infrared range. This feature is extremely relevant for mice tracking during nocturnal hours. The video acquired by the camera is transmitted frame by frame to the raspberry Pi that elaborates it.

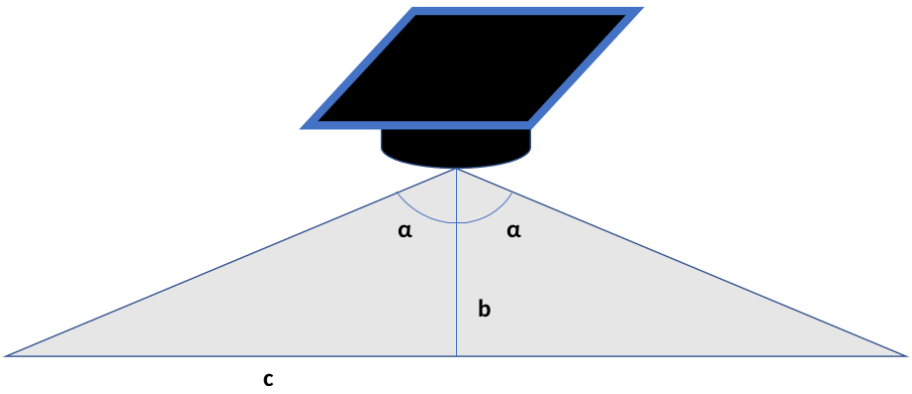


Figure 3.2: Support image to viewing angle problem. The black module indicates the camera. Since $b = 3cm$ (the additional length due to the lid set on the walls of the cage) and $c = 14.25cm$, the minimum viewing angle 2α that allows to cover the whole area is 156°. According to this, a camera with $FOV = 160°$ was employed.

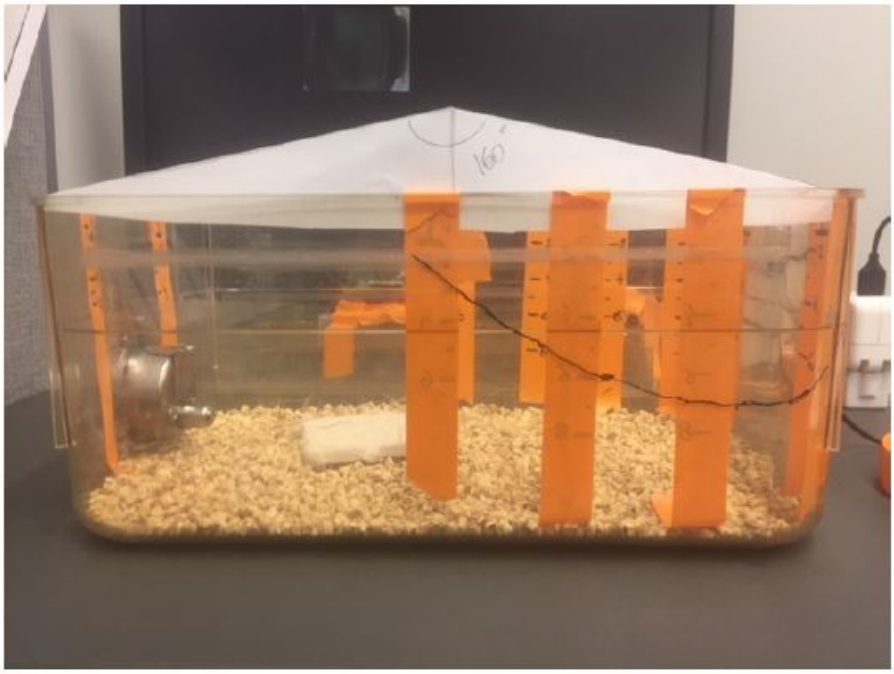


Figure 3.3: Picture of the cage showing the problem relative to the field of view of the camera. The white paper shows that a FOV of 160° is required to cover the whole area of the cage, walls included. The black marker line shows the hidden area in case of employment of a traditional camera.

3.2 Home Cage Setup

All biological laboratories host a big number of cages wherein animals move. There are generally some standards that must be followed. The size of the cage we used for our experiments is 28.5x17.5x13cm and the lid adds about 3cm of height. The camera is placed in the center of the lid under a plastic rectangular support. The camera is so positioned 16cm from the floor of the cage and 3cm from the top of the walls. Over the support that holds the camera, the raspberry Pi, our data-processing unit, is placed. On the lid of the cage, beside the rectangular support, four 850nm LEDs are placed, two for each side of the support. The whole cage, with its individual parts, was put inside a paper box in order to simulate the darkness of biological laboratories. A good stylization of the home cage setup can be observed in figure 3.1 while the real cage, with all the individual parts properly positioned, is shown in figure 3.4.

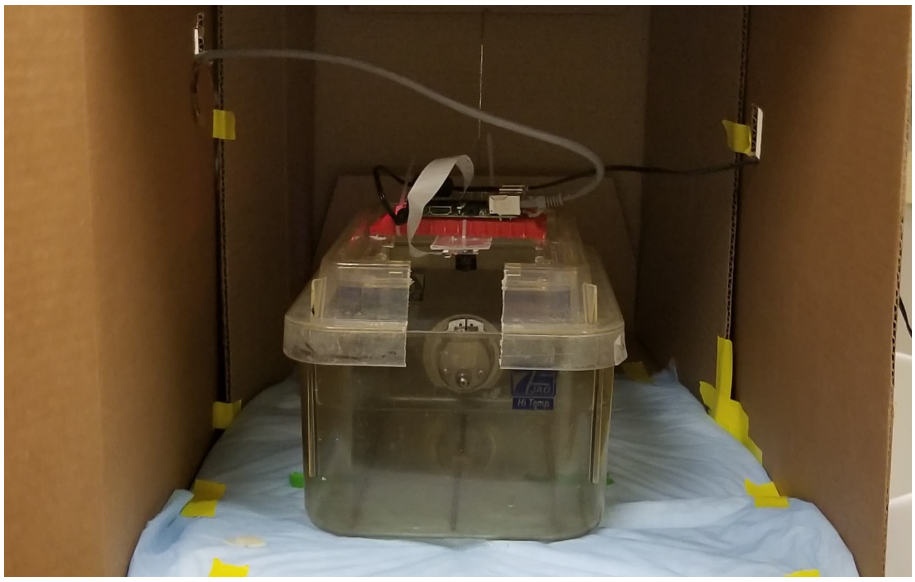


Figure 3.4: Illustration of home cage setup. The picture was taken during an experiment carried out at Johns Hopkins University School of Medicine. On the lid of the cage it is possible to notice the support with the camera module and the raspberry Pi. The whole cage is placed inside a paper box to reproduce the darkness of biological laboratories.

3.3 Lighting Conditions

Lighting condition is an extremely important factor that must be taken in consideration in the realization of a tracker for multiple mice aimed to behavioral phenotyping. In fact, light has a big influence in mice behaviors and does not represent a negligible aspect of the experimental setup. Mice are nocturnal animals and this natural feature must be respected in a biological laboratory. In nocturnal hours, tracking becomes very difficult without proper lighting conditions. To solve this issue, near-infrared LEDs are employed. It was comprehensively demonstrated in the past that mice are not sensitive to infrared-radiation, but they can be partially to the near-infrared. $850nm$ wavelength was chosen as a good compromise. In fact, this wavelength is far enough from the visible spectrum, in order not to influence mice behaviors, but it is not too long to be out-of-range for the camera spectrum sensitivity. In addition, $850nm$ wavelength illumination shows a good reflection on metal tags bound to mice ears. Four LEDs working at this wavelength were employed and placed on the lid, as showed in figure 3.1. This placement allows the light to be spread homogeneously all over the surface of the cage, minimizing unwanted reflection and dark areas.

3.4 Metal Tags

In nowadays biological laboratories, mice have always been tagged with labels or metal tags attached or bound to their ears, in order to recognize them. Though, surgical operations for binding tags to mice ears are executed on a daily basis. According to the Department of Neuropathology at Johns Hopkins University School of Medicine, where these are routine operations, light metal tags bound to ears do not have any influence on mice behaviors. This factor is extremely relevant for our study, and according to that, we decided to employ metal tags to be recognized by our computer-vision algorithm. Tags must be visible under NIR radiation to be recognizable, and must be high-reflective to show a significant difference in terms of pixel intensity in comparison with surrounding pixels. We carried out many experiments in order to figure out what were the tags that showed the best reflection of NIR radiation sent by the LEDs. We tried many different shapes and size, and the biggest reflection was reached by spherical metal tags of $6mm$ diameter. Despite the good results collected with these tags, their dimensions were too big for this application. In fact, mice are very small animals and the ear surface is very limited, making the ear not able to bear big weights. With $6mm$-diameter metal tags, the ear was dilated and showed a clear declivity at the bottom. Such big tags could influence mouse behavior and do not fit our needs. According to that, we had to choose a compromise between high visibility and low weight. The best result was obtained by employing spherical metal tags of $4mm$ diameter. With this tag, the ear looked in a good shape, not deformed, with no influence on mouse behavior. In addition, mice did not look to be annoyed by the presence of the tag, because they did not attempt to scratch it or to take it away. According to these results, $4mm$-spherical-metal tags were employed.

Mouse	Kind of Tag	Tag Position
1	Spherical-aluminum tag ($4mm$ diameter)	Left Ear
2	Spherical-aluminum tag ($4mm$ diameter)	Right Ear
3	No Tag	No Tag
4	Spherical-aluminum tag ($4mm$ diameter)	Both Ears

Table 3.1: Metal tags configuration on different mice.

Chapter 4

Methods

The previous chapters were extremely important to contextualize what is the core of our study. First, an introduction to the state of the art and to social behavior classification was fundamental to understand and to get oriented in such a wide field as behavioral phenotyping. Then, main problems and difficulties affecting multiple mice tracking have been comprehensively discussed in order to make clear the reason behind all the choices made in terms of hardware and software. Concerning the hardware, a detailed description of the whole system and of its individual parts was carried out, and a motivation was given for each one of the decisions taken, driven by the necessity to face specific problematics introduced by the working environment. All we dealt with until now lays the groundwork for software description, matter of this chapter.

Software description can result a bit confusing, since several are the parts it is composed in, and each one of these parts requires some guidelines to be well understood. This is the reason why this chapter is divided in different sections, and each of one of these sections will deal a specific algorithm. A short preview will introduce all the sections, and a brief description will be made in order to always keep the reader focus on the current line. Since the algorithm was implemented for video processing, it is described applied on a single frame but it is obvious that it is repeated cyclically until the end of the video.

First and foremost, frame acquisition by the camera and an initial processing are described. Then, an estimation of the background and its utility is presented, and an algorithm implemented to remove static objects in the background is explained. The following step, moving objects segmentation (i.e. mice), is accurately described as it represents the fundamental step for tracking. In fact, the better segmentation result is, the higher is the performance of the final algorithm. Then, the merging detector is presented and the core of the algorithm, the tracker, is comprehensively described in all its parts. In conclusion, tags identification methods are accurately discussed and identity detection is illustrated. All the sections of this chapter are listed below:

1. **Frame Acquisition:** frame is acquired by the camera and a short pre-processing is run to prepare it for the following steps.

2. **Background Estimation and Subtraction:** background (i.e. cage) is estimated by an average of multiple frames and it is subtracted to the current frame in order to extract the foreground.

3. **Static Objects Removal:** static objects, such as feeders, are automatically identified and removed in order not to interfere with segmentation and to compromise the performance of the system.

4. **Merging Detector:** the main issue facing multiple mice tracing, merging, must be detected in order to address the algorithm to the proper following processing steps. In fact, from here on out, the algorithm follows two different branches according to the positive or negative feedback given by the merging detector.

5. **Segmentation:** mice (moving objects) are segmented using a threshold filtering approach. The result of this step strictly influences the performance of the algorithm.

6. **Geometric Information Extraction:** geometric information, such as parameters of ellipses drawn around mice, is extracted. A clustering method will be used in case of merging, while a blob analyzer will be employed in case of no merging.

7. **Tracking Algorithm:** the core of the algorithm is described. Tracking is executed in two different ways according to the feedback of merging detector. In case merging does not occur, *NM Tracker* is employed, while a more complex algorithm, *IDSC Tracker*, is adopted in case of merging, as introduced in chapter 1. Then, ears position detection will be introduced.

8. **Identity Detection and Preservation:** once tracking is executed, identity is detected or kept depending on the feedback of merging detector. In case of merging, tags (so identities) are detected, while they are preserved in case of no merging. Identity preservation is based on centroid distances.

All the different parts are logically illustrated in the flowchart in figure 4.1.

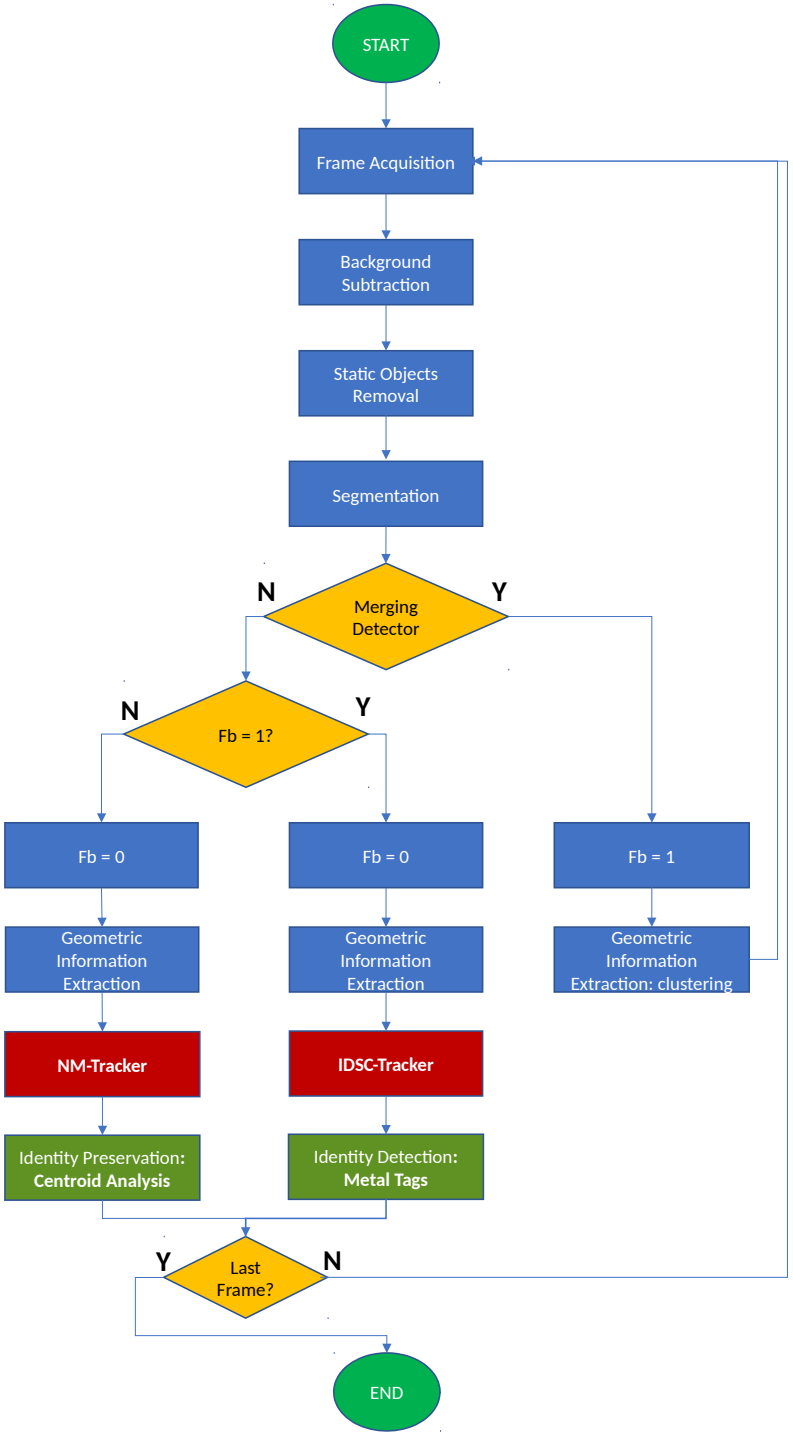


Figure 4.1: Overview of the algorithm: from the original frame to tags identification.

4.1 Frame Acquisition

Frame acquisition represents the first step of the algorithm. Even though the environment can be considered a static object in our study, it shows some fluctuations that are not visible by the human but influence the process of image capture. The result is a difference in terms of pixel intensity that is noticed between different frames. To solve this issue, the image is normalized in the following way. As already said in the introduction of this chapter, it is important to always keep in mind that the algorithm is explained applied to one frame only but it must be thought as applied to the whole video with a cycle ending with the end of the video. First, the frame is captured in RGB color model and must be converted into grayscale model before image processing. Then, the normalization is made dividing every pixel of the image by the mean of the intensities of all pixels of the image as follows:

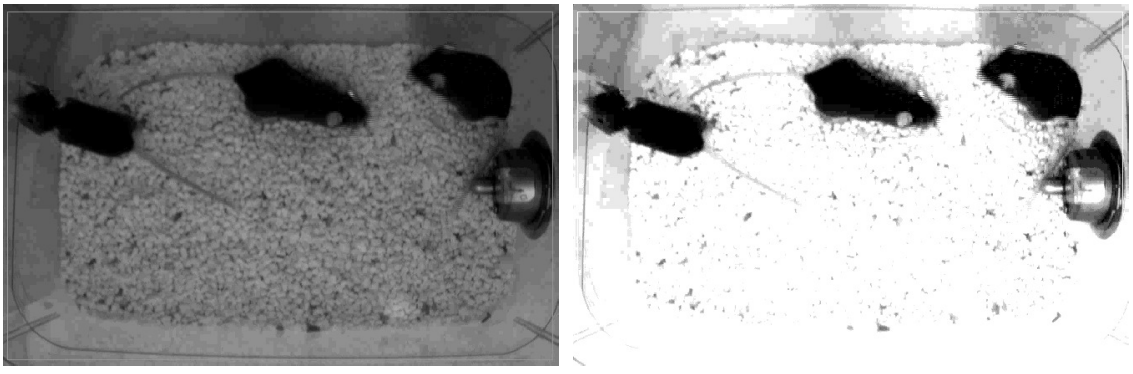
if I is the frame of dimensions mxn, given $I(x, y)$ a generic pixel of image I positioned in (x, y) and $\bar{I}$ the mean of the intensities of all pixels of image I obtained as

$$\bar{I} = \frac{\sum_{x=1}^{m} \sum_{y=1}^{n} I(x, y)}{m \cdot n}$$

then, every pixel of the image is divided as follows,

$$I_{av}(x, y) = \frac{I(x, y)}{\bar{I}}. \tag{4.1}$$

The process of normalization is also called *histogram stretching* because it tries to stretch the distribution of pixel intensities to the normal distribution. In this way, fluctuations are weakened and different frames can be compared more consistently. These operations belong to the part of image processing methods called *image enhancement*. Results of image normalization can be observed in figure 4.2



(a) *Original grayscale frame* (b) *Normalized frame*

Figure 4.2: Illustration of the result of normalization. The original frame is normalized by the average of the intensity of all pixels.

Before continuing with the real processing algorithm, it is important to carry out some other operations of image enhancement. The most important one to improve the result of segmentation is contrast adjustment. To perform this operation, an important value called *luminance* must be extracted. *Luminance* is a photometric measure of the luminous intensity per unit area of light traveling in a specific direction [25]. In digital image processing, luminance is just the average of the intensities of all pixels ($\bar{I}$) that we have already calculated. In contrast adjustment, luminance represents an important parameter. In fact, intensity of pixels with intensity value above the luminance is increased of a given percentage of the intensity distance from the pixel intensity and the luminance, while intensity of pixels under the luminance is decreased of the same percentage. In this way, pixels with intensity value similar to the luminance are not modified, while brighter and darker pixels are modified in a measure that depends on their intensity distance from the luminance. In our study, the contrast of each frame is improved of the 70%. All the operations of image enhancement carried out before the real processing were extremely important to make the frame ready to be correctly analyzed.

4.2 Background Estimation and Subtraction

Another necessary step that must be done before proceeding with the tracking is the subtraction of the static environment from the current frame. In fact, background subtraction would allow to ignore in the processing all the objects belonging to the cage that are useless for tracking (e.g. feeders, sawdust, etc). In addition, even though background can be considered as static in our study, it may undergo a series of modifications due to changes in lighting or to the addition of new static objects to the cage. These modifications can affect segmentation, one of the most important step of the algorithm.

The first thing that comes to the mind when we deal with background subtraction is obviously the trivial solution. In fact, a simple picture of the cage without the mice can be taken and the picture could be subtracted frame by frame with no other calculations. This represents with no doubts the best solution in terms of computational time, but it may be affected by semi-dynamic modifications that can influence the background. In fact, this method would require to change the background picture every time that a small modification is applied to the cage. For example, if a feeder is added to the cage and background picture is not changed, the subtraction would output a result not suitable for segmentation. In fact, no subtraction would be executed on the new feeder since it doesn't exist in the picture representing the background.

According to this, we implemented an algorithm that automatically identify the background, subtracting it frame by frame for foreground extraction. The first seconds of the video are used to achieve the complete background estimation. This

feature doesn't compromise a possible real-time application of the algorithm because after the few seconds required for a correct performance, the algorithm turns into a simple digital subtraction executed frame by frame. It is only important to allow a few seconds at the beginning of the video to let it run. In our study, we utilized always the same cages in the same configurations and we didn't need to repeat the background extraction because we worked under the hypothesis that background can be considered static. The extension to a semi-dynamic background can be easily achieved by running the same algorithm each time that a change is made to the cage. Obviously, for each time the algorithm is called, a few seconds must be allowed to make the algorithm perform well.

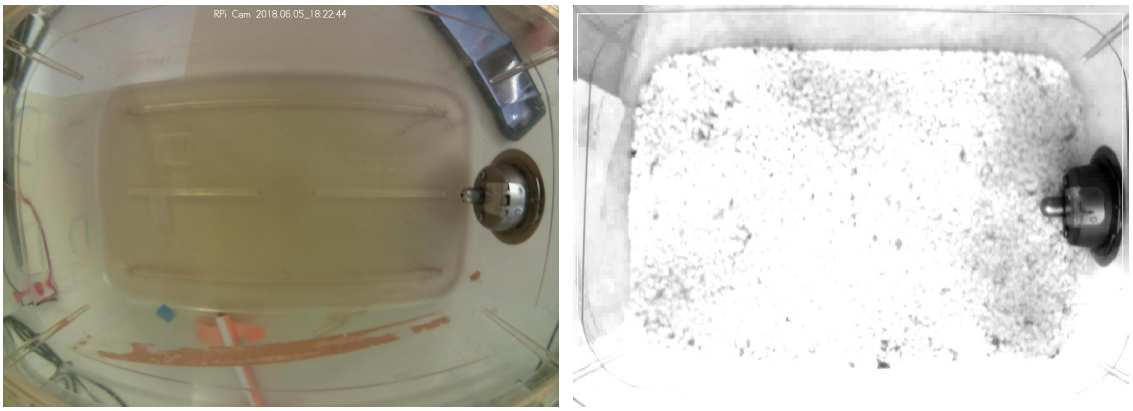
Background estimation is carried out under the hypothesis that mice represent moving objects and the cage a static background. According to this, an averaging of a series of frame is performed in order to filter all the moving objects (i.e. mice). It is clear that the more is the number of averaged frames, the better is the performance of the algorithm, but the longer is the time required for it. A good compromise has been found using few hundreds of frames. In this way we managed to obtain a good estimation of the background that is showed in figure 4.3. The algorithm performs the average summing all frames pixel-by-pixel and dividing them by the total number of frames used for it. If $I_i(x, y)$ is a generic frame and n is the total number of frames that take part to the averaging, then

$$background = \frac{\sum_{i=1}^{n} I_i(x, y)}{n}. \tag{4.2}$$

The result of the averaging is shown in figure 4.3 and it is compared with a real picture of the cage taken in the laboratory of Neuropathology Department at Johns Hopkins University School of Medicine.

4.3 Static Objects Removal

Once background has been estimated and removed from the current frame, foreground is extracted. Since mice are moving, they can interact with static objects that are part of the background. For instance, the most problematic action that mice do is climbing the feeder and standing there for many seconds, trying to climb the wall of the cage. Mice are represented by black pixels, the same as feeders. The difference is that the former belong to the foreground (moving objects), the latter are part of the background (static objects). Therefore, when a mouse is standing on a feeder, background subtraction algorithm doesn't see the mouse, subtracting it with the feeder. The result is a bad segmentation, since only the part of the mouse that is not on the feeder is segmented. The algorithm we implemented attempts to solve this problem and it is divided in the following steps:



(a) *Real picture of the cage* (b) *Background estimated by averaging*

Figure 4.3: Background estimation compared with the original picture of the cage. Darker areas are visible on estimated background. Those areas represent the position assumed by mice for the most of the time. Therefore, those areas are "less filtered" than others, appearing darker.

1. The complement of the normalized frame is obtained. From here on out, mice are whiter and blackground pixels are darker as shown in figure 4.4 **(a) (b)**.

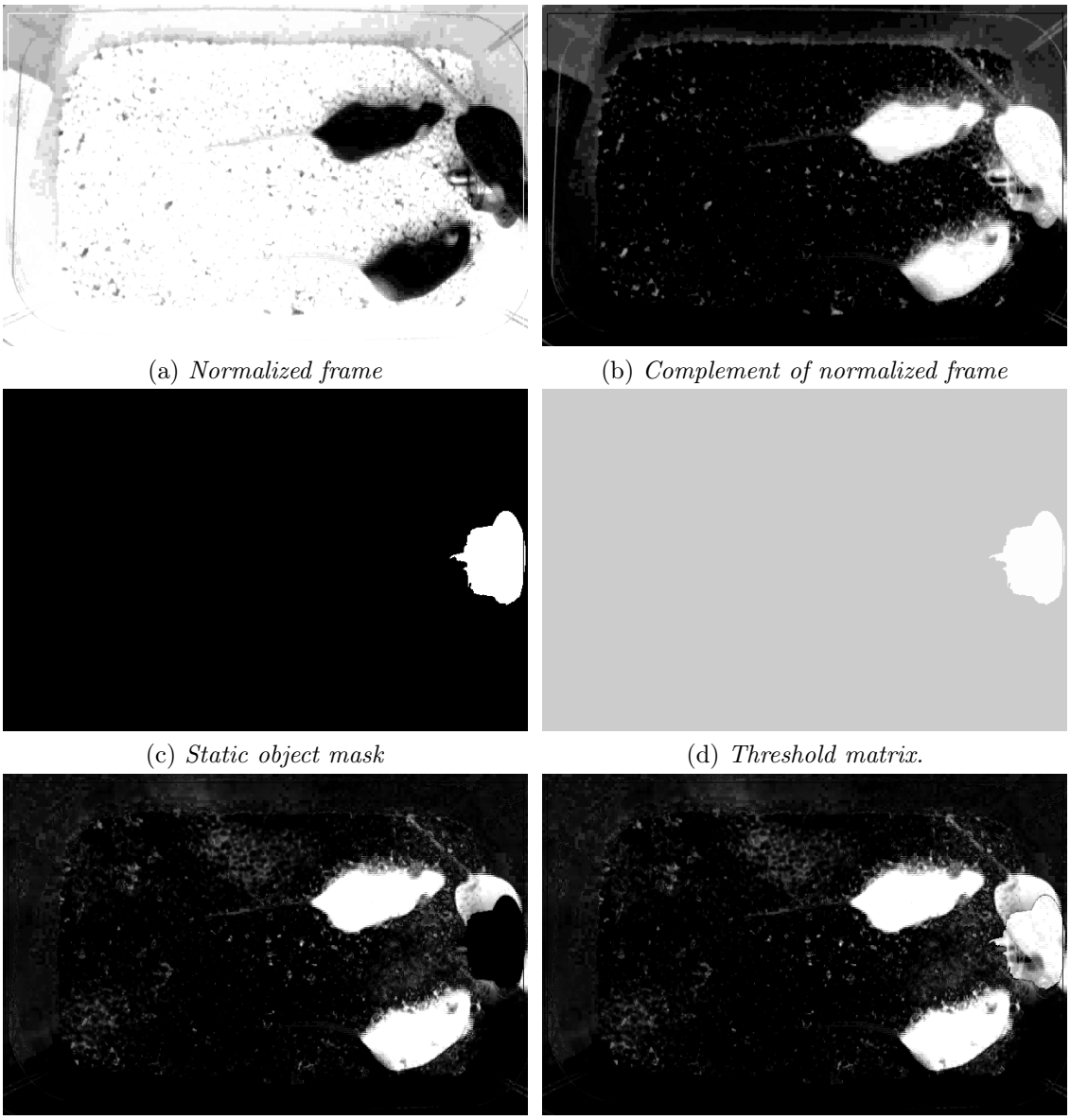
2. The background image is binarized using global thresholding. The threshold chosen for the filter is 0.3. The result is a good segmentation of the feeder (that is darker than the rest).

3. Morphological operations are executed on the feeder in order to segment it in its integrity. The image is then converted into its complement in order to get a mask where the feeder is represented by white pixels and the rest by black ones. Results are shown in figure 4.4 **(c)**.

4. The feeder mask is then converted into a threshold matrix (T), where to pixels belonging to the feeder was assigned a threshold of 0.99, and a threshold of 0.90 was assigned to pixels belonging to the rest. Threshold matrix T is shown in figure 4.4 **(d)** and will be used for segmentation in section 4.4.

5. Background is subtracted by the current frame with the exception of the pixels belonging to the feeder, that are not subtracted. The result is a image where mice are whiter than the background and where there are black pixels instead of pixels belonging to the feeder. Result is shown in figure 4.4 **(e)**.

6. Black pixels belonging to the feeder on the image obtained at the previous step are then replaced by pixels belonging to the feeder on the current normalized frame. In this way, we obtain pixels belonging to the mouse standing on the feeder whiter than the feeder, as visible in figure 4.4 **(f)**.



(a) *Normalized frame*

(b) *Complement of normalized frame*

(c) *Static object mask*

(d) *Threshold matrix.*

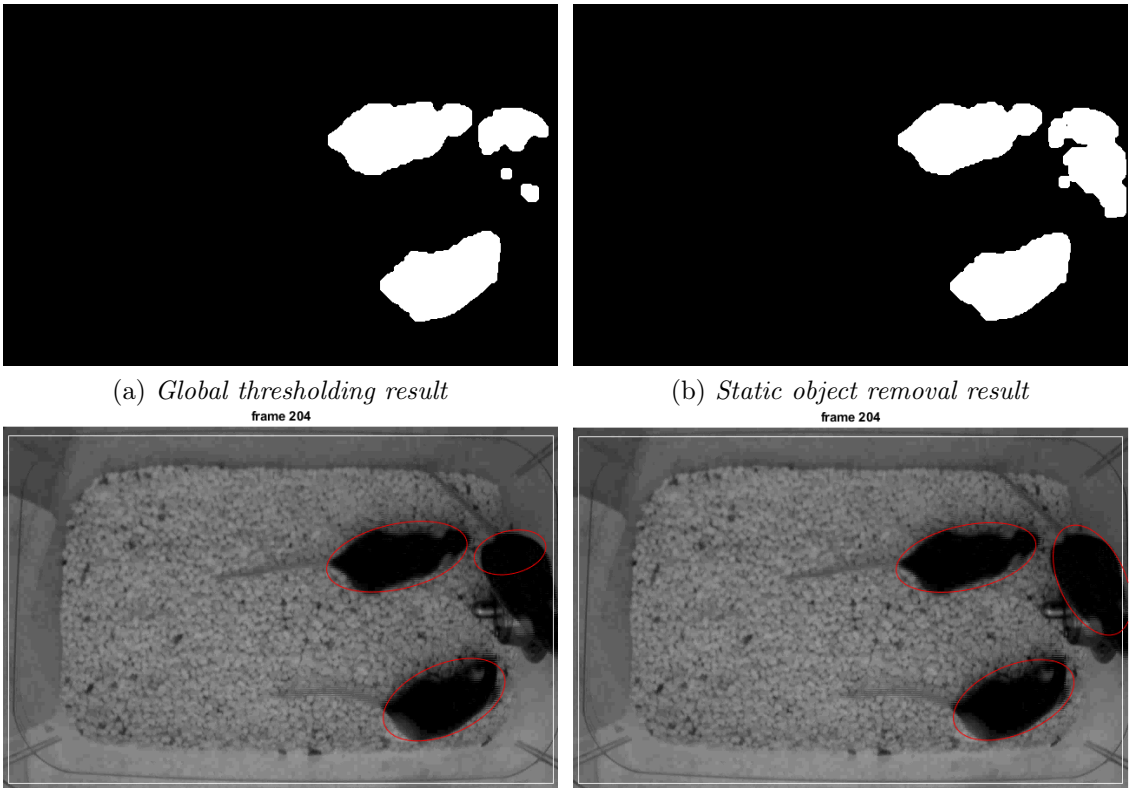
(e) *Subtraction between normalized frame and background without feeder*

(f) *Image ready for segmentation*

Figure 4.4: Illustration of different steps of static object removal. Estimated background showed in figure 4.3 is subtracted to the normalized frame with the exception of pixels belonging to feeder **(e)**. Those pixels are then replaced by pixels belonging to the feeder in the complement of the normalized frame, obtaining an image ready to be segmented **(f)**. In **(d)** is showed the threshold matrix where whiter pixels are 0.99 and darker are 0.90.

Once the image ready to segment is found, a dynamic thresholding filtering is executed in order to segment only white pixels belonging to mice. Threshold matrix T is used to accomplish filtering. Result is shown in figure 4.5, where it is compared

with the result obtained by a global thresholding filtering without using the static-object-removal algorithm. From a quick gaze to the image it is immediately clear that our algorithm shows a better segmentation.



(a) *Global thresholding result* (b) *Static object removal result*

(c) *Global thresholding result: ellipses* (d) *Static object removal result: ellipses*

Figure 4.5: Comparison between results obtained with a standard global thresholding filtering **(a)** and the ones obtained with static object removal algorithm **(b)**. Results are shown also in terms of segmentation and ellipses around mice. Ellipse drawn in **(d)** fits better pixels relative to mouse placed on the feeder than the ellipse in **(c)**.

4.4 Segmentation

All the processing steps comprehensively discussed in previous sections and chapters, from hardware setup to the image pre-processing, are preparatory to the matter of this chapter: segmentation. In fact, segmentation represents one of the most important steps (if not the most) in tracking algorithms, because its output influences the performance of the whole system. Obtaining a good segmentation is extremely important to well detect landmark points, to discriminate among different mice and to avoid that points belonging to artifacts, or different objects, interfere with the

moving animals we are interested in. There are a lot of artifacts that could affect segmentation in this kind of systems. In fact, moving objects are affected by lighting changes, reflection on the walls, occlusion due to static objects in the cage, occlusion given by interaction of different mice, and by a series of other phenomena that can not always be predicted. That's why preparing a good hardware setup and doing an initial pre-processing to the frame acquired by the camera is strictly important to prevent these unpleasant phenomena.

Once everything is ready and we have the image ready to be segmented available, segmentation is carried out considering a locally adaptive threshold, specified as a matrix of luminance values. Actually, a global thresholding would be sufficient for mice segmentation but we have to consider the problem relative to static objects removal explained in the previous section. In fact, the locally adaptive threshold matrix, employed in segmentation, is the matrix T already found after feeder identification and showed in figure 4.4 **(d)**. Experimental observations demonstrated that a good threshold suitable for mice segmentation in this kind of environment and after pre-processing operations is 0.9. Although this threshold could seem very high, suggesting a lack of sensitivity of the sensor, it allowed us to create a system with results that are a good compromise between specificity and sensitivity. In fact, a threshold of 0.9 is high enough to be specific for the objects we are interested in (other objects present a luminance that is generally way lower), and low enough to be very sensitive for mice segmentation (the luminance of pixels belonging to mice generally does not go below the threshold). The result obtained is excellent. Mice are well segmented for the whole video and artifacts do not generally interfere with them. Sometimes, it can happen that other objects that show similar value of pixels intensity, such as mice reflection on the cage walls, are segmented, constituting objects apart from mice. Digital systems are not smart and can not distinguish between what is a mouse and what is not, considering everything they segment as mice. Fortunately, we can add some rules in order to avoid these kind of phenomena. The rule that helps us in this case is a size rule. Indeed, mice are genetically identical, and their size can change only with the postures they assume. In addition, there is generally no other objects than mice that present their size. This means that a minimum size can be added for their segmentation. According to experimental observation results, a good threshold suitable for avoiding segmentation of objects that are smaller than mice is $2000 pixels$. Obviously, this threshold is suitable for our application with the settings we chose in a preliminary phase. If resolution of the camera is changed, or another camera is employed for recordings, the minimum size threshold must be changed accordingly.

Once mice are segmented, it is important to carry out some morphological operations that allow us to improve the result of segmentation. Indeed, mice generally show some shades increasing from the centroid to the borders. Sometimes, intensity of these shades falls below the segmentation threshold with the consequence not to

be included in the segmented shape. The result is imprecise and must be corrected to restore the original shape of mice. According to this, a series of morphological operations are performed as follows:

1. Segmented structures that are connected to the image border and do not represent mice are suppressed.

2. Image opening is performed to remove objects that are smaller than the structural element used, preserving the background. The structural element employed is a disk with a radius of $3pixels$.

3. Image dilation is performed to enlarge mice pixels. This allows us to improve mice shape and to remove accidental indentations. The structural element employed in this operation is a disk with a radius of $5pixels$.

4. A diagonalization is carried out to eliminate 8-connectivity of the background. For example, the matrix

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

   becomes

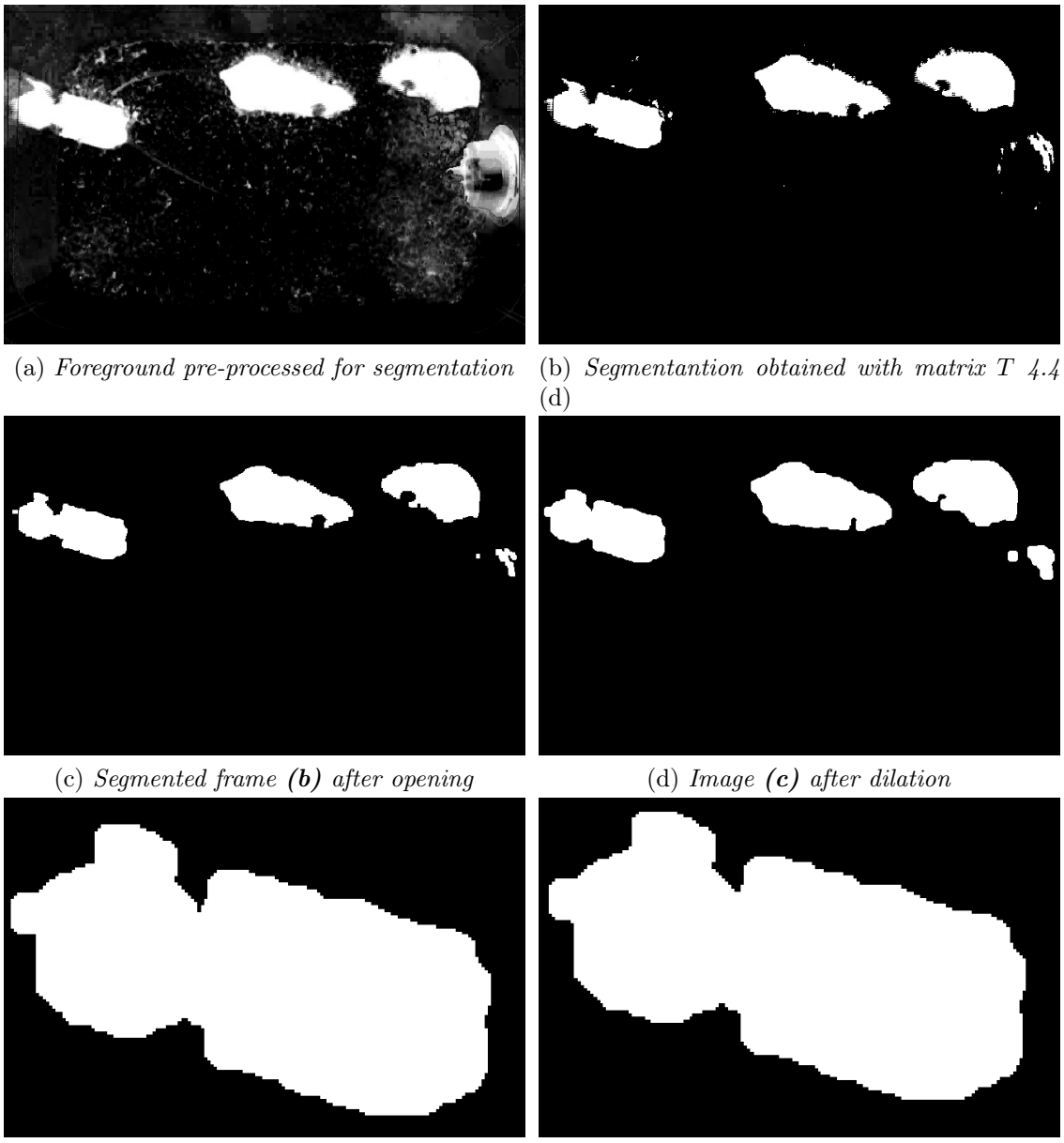
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

   This operation is useful to avoid some problems that can occur in mice contour tracing [12].

5. A majority morphological operation is carried out to homogenize the environment around each pixel. In fact, this operation sets a pixel to 1 if five or more pixels in its 3-by-3 neighborhood are 1s; otherwise, it sets the pixel to 0 [12].

6. The final segmented mask is obtained. Mice are now represented by white pixels on black background in a binary image.

Results of segmentation and the improvements made by morphological operations are shown in figure 4.6. Once morphological operations are performed, the segmented mask is ready to be analyzed by the tracker. Nevertheless, another important step must be carried out to address the algorithm to the following processing.



(a) *Foreground pre-processed for segmentation*

(b) *Segmentantion obtained with matrix T 4.4* (d)

(c) *Segmented frame* ***(b)*** *after opening*

(d) *Image* ***(c)*** *after dilation*

(e) *Zoom on one mouse before diagonalization and majority*

(f) *Zoom on one mouse after diagonalization and majority*

Figure 4.6: Different steps of segmentation process: from pre-processed frame to final segmentation. In figure **(e)** and **(f)** it is possible to see the result of diagonalization on the right ear of the mouse. In figures **(a), (b), (c), (d)** it is possible to notice that some pixels relative to the feeder are segmented on the right part of the image. Those pixels do not represent a big issue because they are not included, since the area they represent is below the threshold.

4.5 Merging Detector

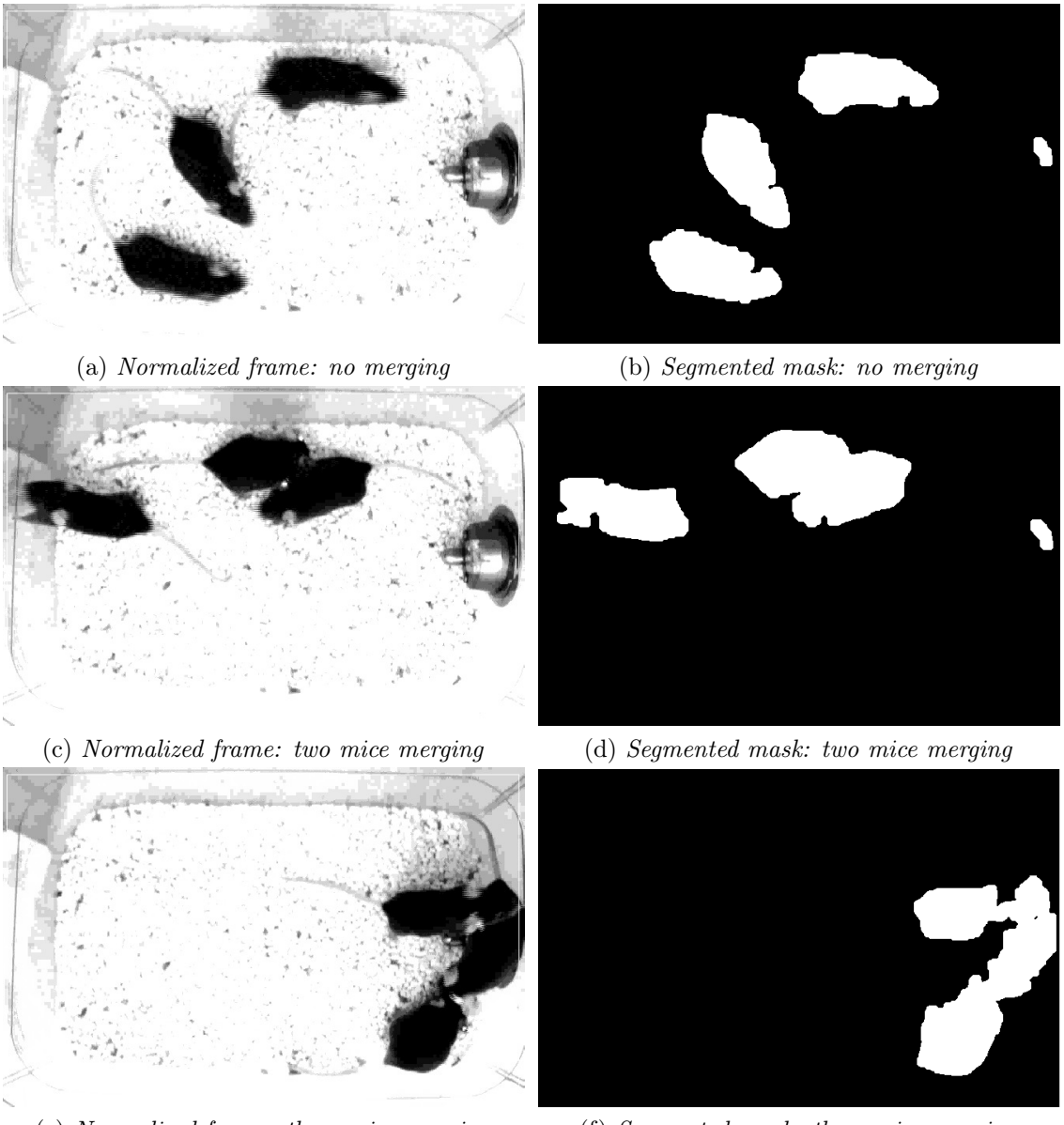
At this point, since we have an image that has just been segmented, it is important to understand whether merging, the main issue affecting multiple mice tracking discussed in chapter 2, is happening or not in the current frame. The reason why it is important is immediately understandable observing the flowchart illustrated in figure 4.1. In fact, it is possible to notice that the flow of the algorithm is split in different branches according to the feedback given by the merging detector. In particular, if merging is occurring in the current frame, the algorithm is limited to trying to separate segmented pixels into the relative number of mice who are merging; otherwise, the algorithm works differently depending on what happened in the previous frame. In fact, if no merging occurred in the previous frame, the tracking problem and identity preservation can be solved easily as explained in following sections; if merging occurred in the previous frames, identities are lost and landmark points too. That's why a stronger algorithm is required for tracking, and identity are re-established by metal tags detection. Therefore, merging detector represents the hinge point of our study. In fact, although its implementation doesn't constitute any particular problem, it deserves a section of this chapter because of its importance.

Merging detector works counting the number of blobs segmented by the segmentation algorithm. Hypothesizing that n_{mice} is the number of mice moving within the home cage, and n_{blobs} is the number of blobs detected by the segmentation algorithm, then:

- if $n_{blobs} > n_{mice} \longrightarrow$ error: segmented blobs are more than number of mice;
- if $n_{blobs} = n_{mice} \longrightarrow$ merging is not occurring: number of segmented blobs equal to number of mice;
- if $n_{blobs} < n_{mice} \longrightarrow$ merging is occurring and the number of mice taking part to merging depends on the number of mice in the cage. In general, if $n_{blobs} = a$, where a is an integer number $1 \leq a \leq n_{mice}$, then the number of mice that are merging (n_{merg}) is given by $n_{merg} = n_{mice} + 1 - a$. For example, if one blob is segmented and three mice are moving, it means that three mice are taking part to merging. In fact, if $a = 1$ and $n_{mice} = 3$, then $n_{merg} = 3 + 1 - 1 = 3$. Knowing how many mice are taking part to merging is important to divide the big blob segmented into the right number of smaller blobs. For example, if three mice are merging, we have to obtain three small blobs from the big one. Therefore, this is a parameter we must know.

Different examples of merging are shown in figure 4.7. Merging detector has been tested on several videos acquired with the experimental setup described in chapter 3 with excellent results. The only noticed errors happened when noise artifacts or mice

specific postures make the segmented blobs so small to be under the size threshold. For instance, if mice are separated but one blob is too small, the algorithm sees two blobs instead of three, and merging detector reports a merging event even if it is not happening.



(a) *Normalized frame: no merging*

(b) *Segmented mask: no merging*

(c) *Normalized frame: two mice merging*

(d) *Segmented mask: two mice merging*

(e) *Normalized frame: three mice merging*

(f) *Segmented mask: three mice merging*

Figure 4.7: Illustration of different episodes of merging. From frames where no merging occur to frames where merging with three mice occurs.

4.6 Geometric Information Extraction

Once segmentation has been performed and mice are represented by white blobs on black background in a binary image, it is necessary to extract geometric information to detect some important parameters for tracking. This is the first step of the processing whose algorithm depends on the feedback of merging detector, as shown in the flowchart in figure 4.1. Indeed, from here on out, every section will be divided in two other subsections. Concerning geometric information extraction, two are the cases we have to deal with:

1. No merging occurring in current frame;
2. Merging occurring in current frame.

In fact, in the first case, the algorithm extracts all the important parameters using MATLAB Blob Analysis block, while k-means clustering is executed to trying to separate mice in the second case.

Before starting to analyze the algorithm, it is important to define what are the geometric parameters we are interested in and why. The aim of our study is mice tracking and identity preservation. According to this, fundamental parameters that can describe each mouse from a digital point of view are the following ones:

- Ellipse around the mouse;
- Centroid position;
- Area of the mouse;
- Perimeter of the mouse;
- Ellipse major axis;
- Ellipse minor axis;
- Ellipse orientation;
- Eccentricity.

The reason behind the choice of these geometrical parameters has its roots in digital-shape description. Indeed, digital algorithms ,not involving machine learning techniques, do not recognize shapes. In fact, they are just able to elaborate simple information way more rapidly than humans, but they can not reproduce human eye. Therefore, we have to employ basic geometric technique in order to obtain a good description of mice shape. In this way, animals shapes, described by their geometrical parameters, become comparable. Comparability of shapes is of crucial importance in these kind of algorithms.

4.6.1 No Merging in Current Frame

In case no merging is occurring in the current frame, the problem of ellipse drawing and geometric parameter extraction can be easily solved. In fact, MATLAB R2017b computer vision toolbox makes available a very powerful block called *Blob Analysis*. *Blob Analysis* is used to compute statistics on labeled regions in binary images [13]. If merging is not happening, three different blobs representing three different mice are segmented by the segmentation algorithm. This means that it is possible to use the segmented binary mask as input of Blob Analysis block. The blob analyzer was initialized with the opening of the ports relative to the geometrical parameters we are interested in. According to this, ports relative to centroid, area, perimeter, major axis, minor axis, orientation and eccentricity were set to 1, allowing the system to output the related parameters. Centroid position is given in (row, col) coordinates. Area, perimeter, major axis and minor axis are given in *pixel*, while orientation is given in *radians*. Eccentricity is an important parameter that has no unit of measurement, since it comes from a fraction that involves major and minor axis. Given an ellipse with general function

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

eccentricity (ecc) is defined as follows:

$$ecc = \sqrt{1 - \frac{b^2}{a^2}} \tag{4.3}$$

and gives an idea of the ellipticity of the ellipse. In fact, an eccentricity of 0 refers to an ellipse with major and minor axis with the same length, or rather, a circumference; an eccentricity of 1 refers to an ellipse that degenerates into a segment corresponding to its major axis. Blob Analysis block automatically outputs all the described parameters and draw an ellipse around each mouse. A maximum of three blobs was given according to the number of mice we are studying. This setting makes the algorithm exclude eventually segmented objects that come from artifacts. Results are shown in figure 4.8

4.6.2 Merging in Current Frame

In case of merging occurring in the current frame, *blob analysis* block doesn't help us anymore. In fact, MATLAB blob analyzer is not able to identify different mice under the same blob, and another solution must be found. Since our study does not include tracking while mice are merging, as already said in chapter 1, we do not aim to a perfect separation algorithm. In fact, our study attempts only to maintain some basic geometrical parameters, such as centroid position, while mice are merging.

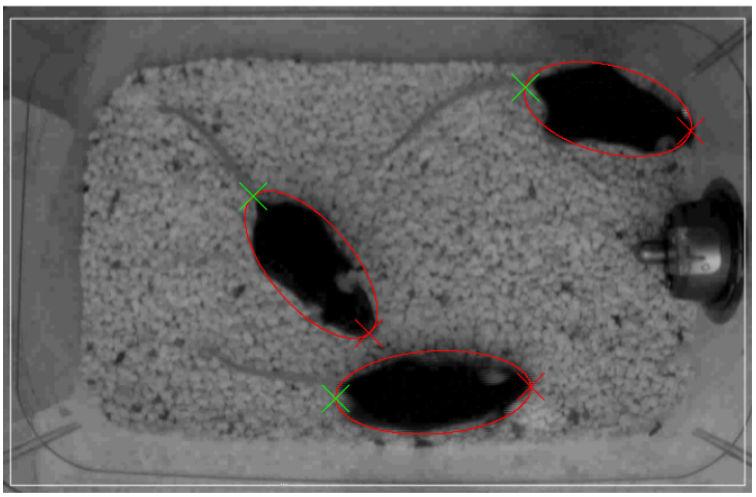


Figure 4.8: Ellipse drawing in case of no merging occurring in current frame. Red cross represents head while green cross represents tail.

According to this, several methods have been employed, from watershed to Gaussian Mixture Models, but no solution has been found to get a good result without big expenses in terms of computational power and time. That is why, supported by the Neuropathology department at Johns Hopkins University School of Medicine, we decided not to deeply analyze mice during merging, but to limit the algorithm to mice separation and detection of their centroid position. This would allow us to keep track of mice identity even during merging, when this last is not too complex for the analysis. First and foremost, it is extremely important to define how many mice are taking part to the merging. Fortunately, our merging detector was implemented to give us this fundamental information. In fact, if three blobs are merging, points belonging to three different mice must be found; but if only two mice are merging, the big blob must be divided only in two mice, since the last one is already segmented apart, and must not be included in clustering. Therefore, knowing how many ellipses we have to find in the same blob is extremely important for the performance of the algorithm.

Once the number of "sub-blobs", with which we have to divide the big blob, is known, the algorithm tries to understand what are the pixels belonging to the different mice using an unsupervised k-means clustering method. K-means algorithm is a simple clustering procedure that attempts to minimize the *intra-cluster variability* and to maximize the *inter-cluster distance* in an iterative cycle. *Intra-cluster variability* is the sum of all distances between each element belonging to a cluster and the cluster centroid. *Inter-cluster distance* is the distance between two cluster centroids. Centroid is generally considered as the mean value of all elements belonging to a cluster. The different steps of the algorithm, in case of merging and not, are presented in the following points:

1. Merging detector reports that merging is happening in the current frame and outputs the number of mice taking part to merging, that represents the number of clusters in which we have to divide the merging blob.

2. The unsupervised k-means clustering is initialized with the centroids of mice obtained in the previous frame. This initialization is feasible with the realistic hypothesis that mice can not move much between two consequent frames. According to this, the same centroid relative to the same mouse in the previous frame must be closer than the others in the current frame.

3. K-means starts working calculating the new centroids of each cluster.

4. K-means reassigns each element to the cluster with the nearest centroid according to some distance measures.

5. If the assignment of points to each cluster is not changed, the algorithm stops, clusters are found and new centroids are saved. Otherwise, it goes back to step 3, calculating new centroids and reassigning each element to the cluster with the nearest centroid.

Once the merging blob is divided in the respective number of clusters, the algorithm that draws the ellipse around each mouse is run. Since ellipse drawing is not so relevant for tracking, it is not described in this chapter. However, knowing its geometric principles could be very interesting for a good analysis of the results. According to this, a short description of the ellipse tracing technique can be found in appendix A. The method used for finding out the best ellipse fitting the single blobs is the one introduced by Radim Halir and Jan Flusser in "Numerically Stable Direct Least Squares Fitting of Ellipses", an extension of Fitzgibbon's study [9] [7]. Even though ellipse-drawing algorithm is not explained here, it is important to justify the weird result obtained with this method. In fact, it is easily found that ellipses drew with this method look smaller than the ones in case of no merging. This feature is due to the way ellipses are calculated. In particular, the solution is biased to smaller ellipses. This is due to the algebraic distance employed to calculate them, which "prefers" points lying inside the ellipse than the ones more external [9]. A block diagram of the algorithm, in case of merging and not, can be found in figure 4.9. K-means and ellipse-drawing results are shown in figure 4.10. In particular, figure 4.10 shows one of the main problem affecting k-means clustering. When mice get close between each other by their side, it is possible that k-means clustering fails. In fact, while mice "slide" side by side, there will be a point in which points belonging to the cluster relative to a mouse becomes closer to the centroid of the other mouse. That point produces an error that can not be solved. If we are trying to preserve mice identity observing centroid position, as it will be explained in the following sections, we fail because there's a point where identities are swapped. This represents the biggest limit of this kind of separation algorithm.

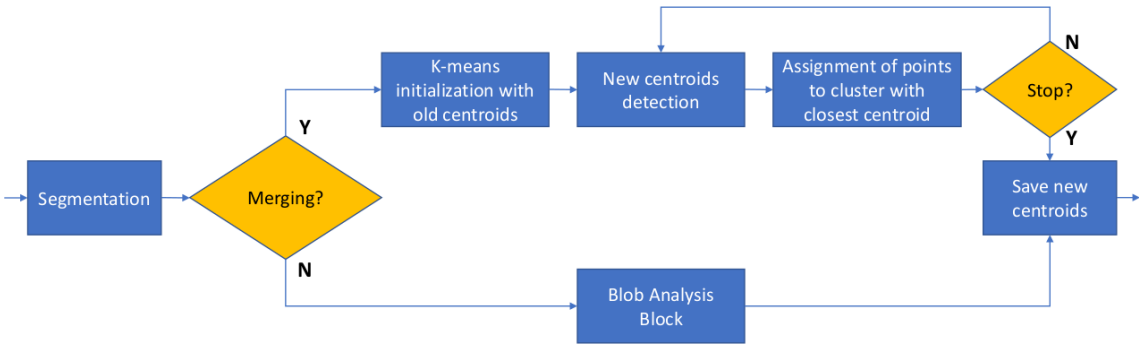
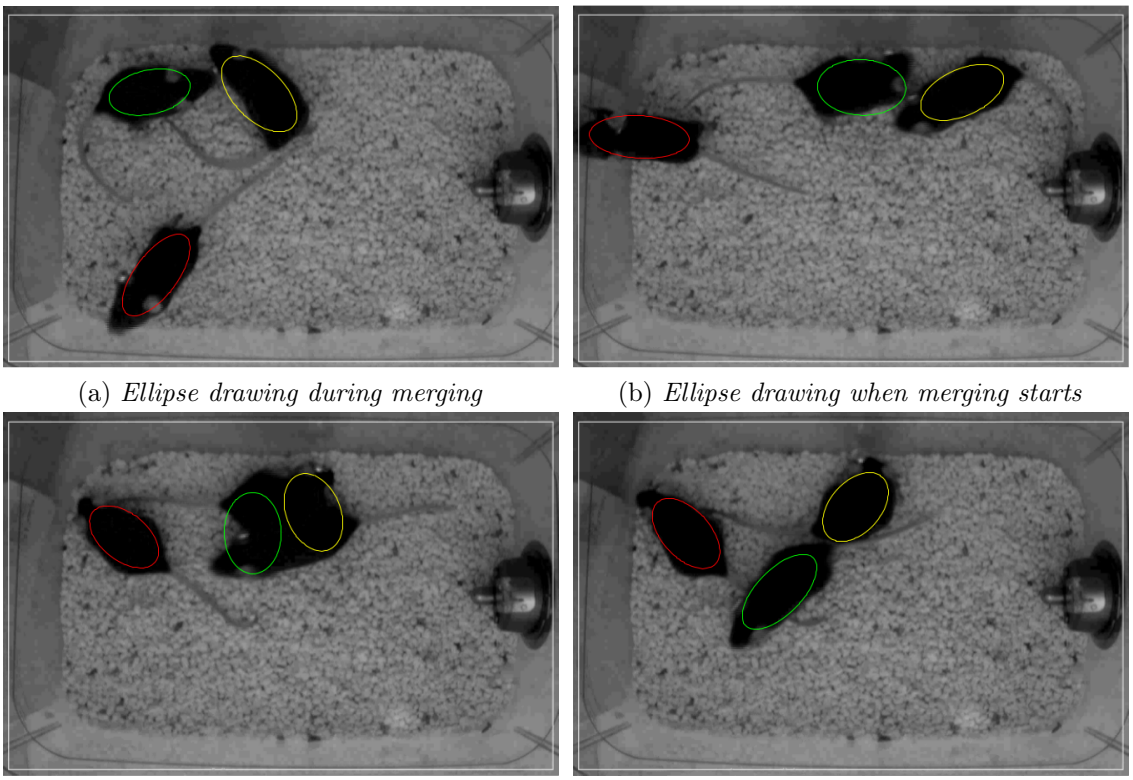


Figure 4.9: Logical illustration of different steps of geometric parameter extraction. The stop condition for k-means cycle is that points assigned to each clusters are the same between consequent iterations.



(a) *Ellipse drawing during merging*

(b) *Ellipse drawing when merging starts*

(c) *Ellipse drawing when merging continues*

(d) *Ellipse drawing when merging ends*

Figure 4.10: Different frames showing ellipse-drawing results. (**b**) (**c**) (**d**) show a typical problem occurring when merging happens from mice sides: ellipses and identities are swapped. In fact, the yellow ellipse in (**b**) slides to the adjacent mouse in (**d**).

4.7 Tracking Algorithm

Now that a generic overview of the algorithm has been given (figure 4.1), and the main steps of the algorithm consisting in segmentation, merging detection and geometric parameter extraction, have been discussed, it is finally time to describe the tracking algorithm and to go deeply into the very heart of our study. As already discussed in the previous chapters, the tracking algorithm, as the last steps described, works differently according to the feedback given by the merging detector. Nevertheless, while geometric information extraction depends on what happens in the current frame only, the tracking algorithm looks at the feedback given by the merging detector not only in the current frame, but in the previous one too. In fact, our algorithm doesn't aim to track landmark points during merging, but when merging doesn't occur. So, looking at the current frame, the tracking algorithm is not required in case of merging. Otherwise, in case of no merging in current frame, the tracking algorithm must know whether merging occurred or not in the previous one. In fact, in the case it occurred, identity and landmark points position are not available anymore, while they are known in case merging did not occur. The block diagram in figure 4.11 shows a very general overview of the tracking algorithm. In particular, it assumes that merging is not occurring in current frame and shows how the two different trackers (NM-tracker and IDSC-Tracker) are called according to what happened in the previous frame.

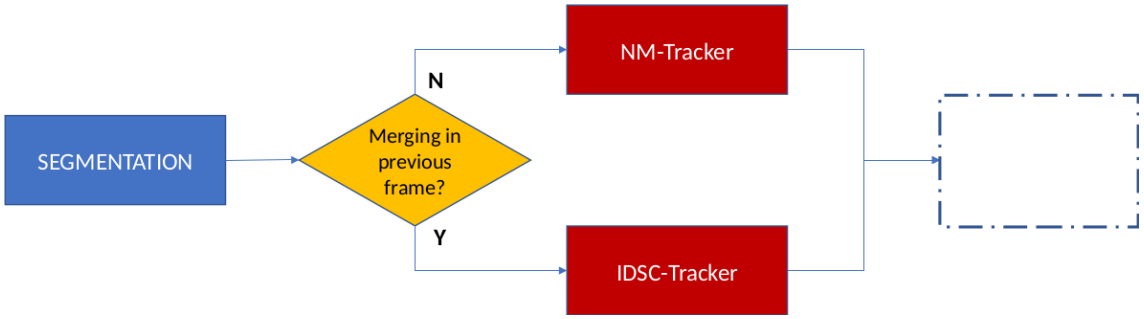


Figure 4.11: Block diagram showing a very general overview of tracking algorithm. It is assumed that merging is not occurring in current frame. The right kind of tracker is selected according to the output of merging detector in the previous frame. If merging occurred, IDSC-Tracker is run. Otherwise, NM-Tracker is employed.

So, assuming that merging is not happening in the current frame, the tracking algorithm works differently depending on whether merging at the previous frame occurred or not. The tracker we implemented consists basically in two different algorithms. In frames where mice are well separated and merging did not occur previously, the tracking is based on basic calculations of euclidean distances; therefore, the algorithm results easy and can run real-time. This tracker is called No-Merging-Tracker and it is abbreviated as **NM-Tracker**. When merging occurred,

and more than one mouse was segmented under the same blob, we lose information about landmark points position, and another stronger algorithm is required to detect them again. Inner Distance Shape Context implemented by Jacob et al. is the shape descriptor we used to build our Inner-Distance-Shape-Context-Tracker, abbreviated as **IDSC-Tracker** [11]. The two trackers will be discussed separately, and their working principles will be comprehensively explained.

4.7.1 NM-Tracker

Assuming that all mice are well separated in the current frame (no merging is occurring), and so they were in the previous one (no merging occurred), then NM-Tracker is run. If no merging occurred, it means that head and tail locations are available from the previous frame. Assuming that a mouse can not move much from one frame to the following one, calculating Euclidean distance between head and tail points in the previous frame and major axis endpoints in the current frame, obtained by geometric information extraction, is the fastest and easiest way to get the new head and tail positions. The endpoint in the current frame that results closer to head position in the previous frame will be labeled as "head", while the other will be labeled as "tail". The block diagram in figure 4.12 shows a logic overview of the tracker, while figure 4.13 represents an illustration of the main *NM-Tracker* working principle based on Euclidean distances. The same method can be applied to centroids to keep mice identity frame by frame when merging does not occur. The centroid of the same mouse can not move much between two subsequent frames; according to this, mouse identity can be kept by taking the shortest distance between centroids in the current frame and each centroid in the previous frame. This method will be explained more in detail in next section.

This algorithm allowed us to get good results in real time with very low computational power request, since the only calculations executed are simple Euclidean distances. This feature is extremely important because our aim is implementing a system that can be employed in real-time applications. According to this, *NM-Tracker* is a very light system that can be easily applied to all frames where merging is not occurring, that represent the majority of frames. The implementation of this tracker aimed to solve the problem of Braun's algorithm. In fact, in his work, only one tracker, based on Jacob's Inner Distance Shape Context, is used for all aims. Since Jacob's algorithm is very heavy, as it will be comprehensible in next subsection, Braun's algorithm is too heavy for real-time processing [**braun:article**] [11]. *NM-Tracker* is a good way to easily track head and tail in the majority of frames, that is when merging doesn't take place, with an extremely light computational demand. The strict constraint is that *NM-Tracker* is applicable only when merging doesn't occur.

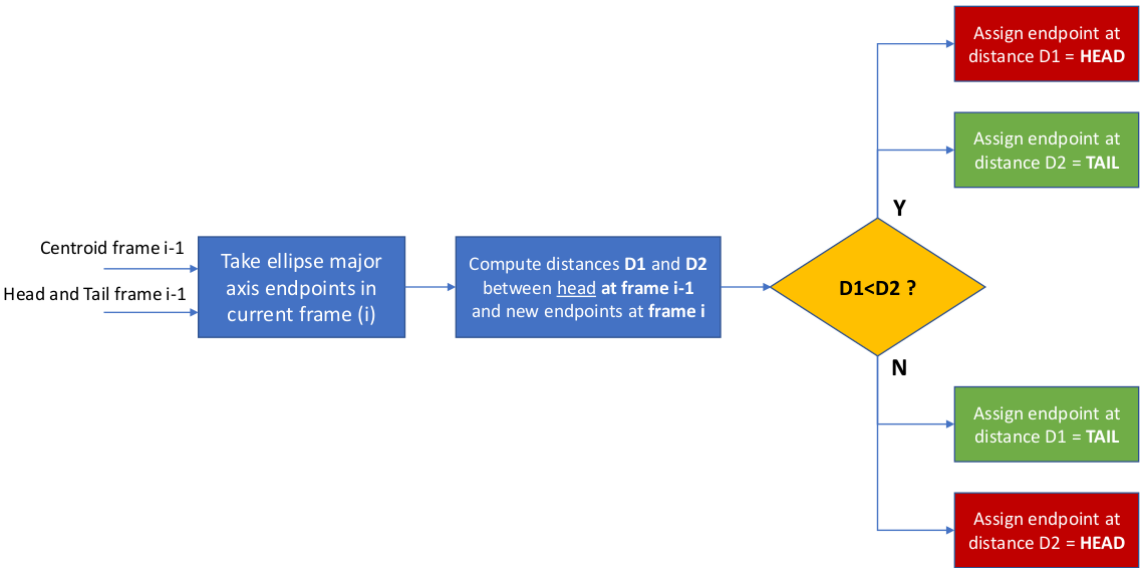


Figure 4.12: Block diagram showing the logical steps of *NM Tracker* algorithm.

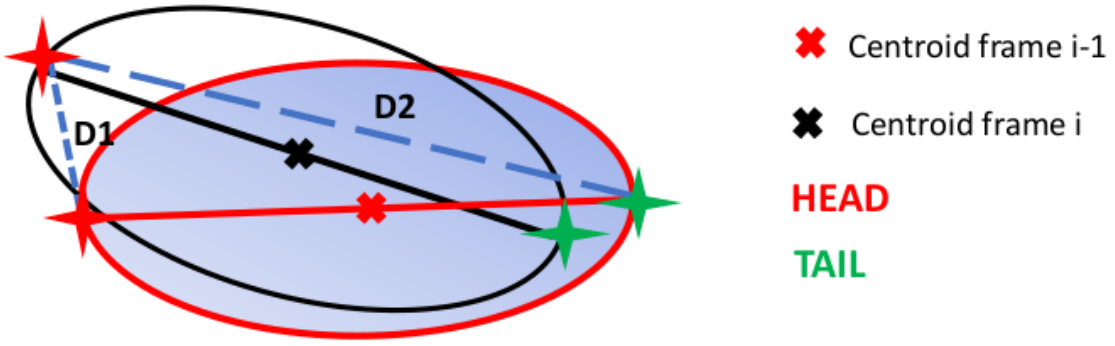


Figure 4.13: NM Tracker working principle: black ellipse represents mouse position in the current frame while red ellipse represents it in the previous frame, when head and tail positions are known. The algorithm calculates Euclidean distances D1 and D2 between old head position and new ellipse endpoints, and take the shortest one to assign new head location. In the illustration, the shortest distance is represented by **D1** and the relative endpoint is labeled as head, while the other as tail.

4.7.2 IDSC-Tracker

While *NM-Tracker* doesn't show any particular difficulty in its functioning and implementation, *IDSC-Tracker* is a more delicate system that deserves a more accurate explanation to well understand how does it work, and what are the working principles it is based on. In fact, assuming that mice are well separated in the current frame and merging occurred in the previous one, head and tail positions are not available anymore, and a stronger algorithm than *NM-Tracker* is required to find head and tail locations. The new tracker we implemented is called *Inner-Distance-Shape-Context Tracker* and it is based on shape context matching to find out the best correspondence between a reference shape of a mouse, and the current shape of the investigated mouse. Specifically, we used an extension of Shape Context called Inner Distance Shape Context, where our tracker takes the name from. The reason why we used this extension is that Inner Distance Shape Context is more suitable for articulated shapes such as mice [11]. Shape Context descriptor was introduced for the first time by Belongie et al. in their study "Shape Matching and Object Recognition Using Shape Contexts" [2]. In their approach, an object is treated as a point set, and the shape of that object can be described essentially by a subset of its points. In other words, a shape is represented by a set of points sampled from the internal or external contours on the object. This means that it is assumed that a shape of an object can be described by a subset of points extracted from its external (or internal) contour. To extract the external contour of our mice, we employed the contour tracing technique described by Fu Chang et al. in their article [6]. This algorithm is the commonest method used for contour extraction, and it is the same employed in MATLAB. According to this, we found not useful to describe the algorithm in this thesis, since a lot of literature is already available for a comprehensive understanding. Much more interesting for our application is deepening the shape matching algorithm introduced by Belongie et al. and successively extended by Jacob et al. [2] [11]. The two following subsections will describe these last two methods, while the last one will discuss our application.

Belongie's Shape Context

Taking in consideration two different mouse shapes, assuming that their external contour has been extracted and assuming that a mouse shape can be described by a subset of points selected from its contour, we aim to find for each point p_i on the first shape, the best matching point q_j on the second shape. Finding the best matching point can not be done by a comparison of the only intensity of the pixels relative to those points. This kind of approach would take to random results, since pixel intensity is a parameter that is too weak to be used alone, hence not suitable for matching. That is why a better algorithm is required. The novel descriptor called *Shape Context* attempts to solve this problem creating a map for each point on the

two contours. This map represents the shape descriptor of the single point and it is then employed for matching.

Considering n points $p_1, p_2, p_3, ..., p_n$ on a shape's contour and looking at the relative Euclidean distance and orientation distribution for each point p_i to the $n-1$ remaining points of the contour, we obtain a rich descriptor of the point p_i. In other words, for each point p_i on the edge of the single shape, a histogram h_i of the relative coordinates of the remaining $n-1$ points is computed as follows:

$$h_i(k) = \#[p_j : j \neq i, p_j - p_i \in bin(k)] \tag{4.4}$$

where the function # must be read as "the number of points" falling in the same bin k. The histogram h_i defines the Shape Context of the point p_i. Before calculating the shape contexts of each point on the contour that belongs to the 2D image space (x, y), points belonging to the contour are transferred to a new logarithmic-polar space. The log-polar space makes the descriptor more sensitive to the positions of nearby points than to the farther ones. Since a good point descriptor must focus on its close environment, the log-polar space is more suitable for this aim. In fact, a point on the contour that is closer to the investigated one must count more than a point located farther. Figure 4.15 shows a log-polar space centered on a generic point on a generic complex shape contour. The process is repeated for all sample points until all shape contexts are calculated.

Once the shape context histogram is built for each point p_i on the first shape, and for each point q_j on the second shape, a cost function $C_{ij} = C(p_i, p_j)$ is defined as follows:

$$C_{ij} \equiv C(p_i, p_j) = \frac{1}{2} \sum_{k=1}^{K} \frac{[h_i(k) - h_j(k)]^2}{h_i(k) + h_j(k)} \tag{4.5}$$

where $h_i(k)$ and $h_j(k)$ denote the K-bin normalized histogram at p_i on the first shape and at p_j on the second one, respectively [2]. The cost function can also include an additional term called *appearance similarity (AS)* at points p_i and p_j. This term depends on the application and can be modified according to robustness requirements. *Appearance Similarity* can be very useful when two shapes deriving from gray-scale images are compared. In that case, for instance, the appearance similarity term could consist in a normalized correlation scores between small gray-scale patches centered at p_i and p_j [2]. Once the cost function C_{ij} is calculated for each pair of points on the two shapes, weighted bipartite matching is used to minimize the total cost of matching defined as follows

$$H(\pi) = \sum_i C(p_i, q_{\pi(i)}) \tag{4.6}$$

The total cost of matching $H(\pi)$ is constrained so that it is forced to be one-to-one. According to this, π refers to a permutation. Figure 4.14 shows all the different steps of shape matching using shape contexts.

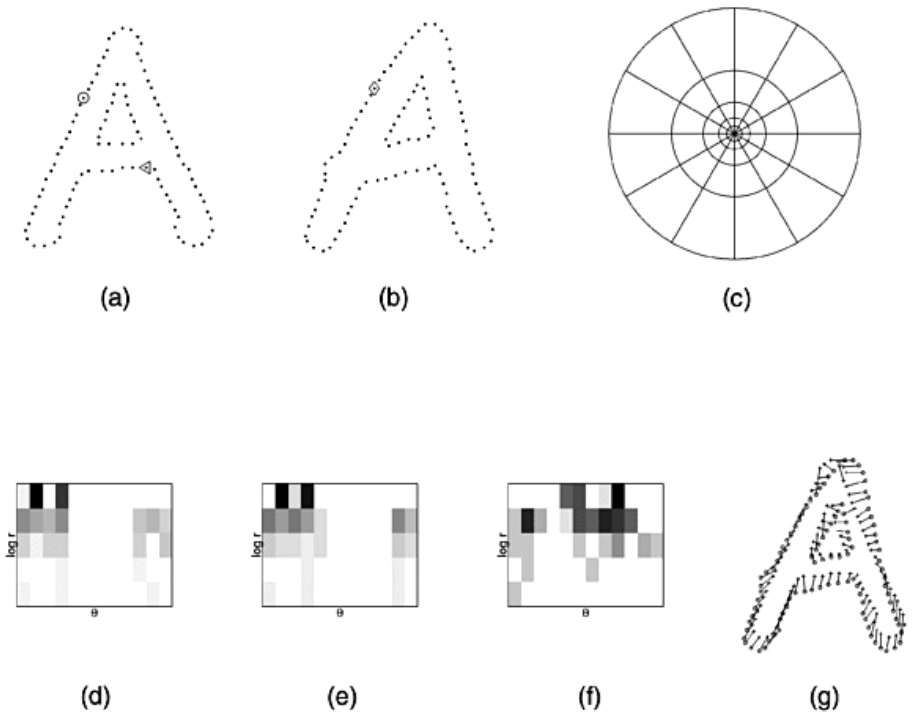
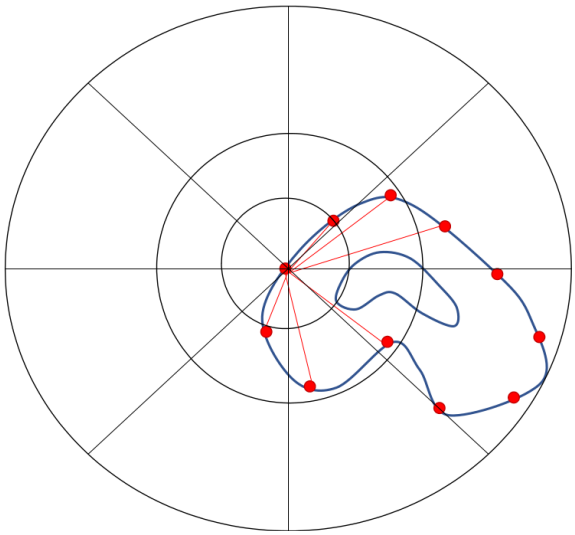


Figure 4.14: Illustration of different steps of shape matching using shape contexts. (**a**) and (**b**) show two different shapes of letter "A". **c** shows the log-polar space built around each point. **d** shows the shape context at the point enclosed in the circle in (**a**) and at the point enclosed in the rhombus in (**b**), respectively. Shapes Contexts look similar because points lie in a similar position in the two shapes. In the histograms, x axis denotes the orientation while y axis denotes log distance bins. (**f**) shows Shape Context at the point enclosed in the triangle in (**b**). Its shape context looks very different from the others, because different is its position. (**g**) shows point-to-point matching between the two shapes, according to the minimization of the cost function.

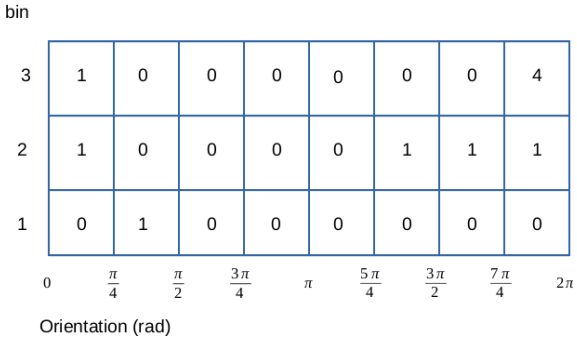
Matching methods are generally required to be invariant under scaling and translation, and they must be robust under small geometrical distortions. Shape Context matching satisfies all these requirements:

- Invariance to translation is intrinsic in Shape Context definition. In fact, all measurements are executed with respect to points on the object, and a log-polar space is built centered on the single investigated point. Complete scale invariance is achieved normalizing all radial distances by the mean distance between the point pairs in the shape.

- Invariance to rotation is demonstrated considering the shape context calculated on a relative frame, treating the tangent vector at each point as the



(a) *log-polar space centered in generic point p*



(b) *related shape context at point p*

Figure 4.15: Illustration of a log-polar space centered around a generic sample point belonging to shape contour with related shape context (**b**). Numbers in (**b**) refers to the amount of points falling in the same bin. The same process is repeated for all sample points on the contour, until all shape contexts are calculated.

positive $x - axis$. The complete demonstration is available in Belongie's article [2].

- Robustness to small geometrical distortion is intrinsic in shape context definition, but it can be improved excluding the estimated outliers from the shape context computation.

Once the best matching is found, a parameter called *Shape Similarity* is extracted combining results given by Shape Context (SC), Appearance Similarity (AS) and Thin-Plate spline geometric estimation [3]. Thin-Plate spline (TPS) is a geometrical method that aims to describe the deformation of a generic surface constrained to a set of points. The origin of the name must be found in the physical analogy involving

the bending of a thin sheet of metal. In fact, when we constrain a sheet of metal to a specific set of points, it adapts its surface to the points, achieving the condition of minimum bending energy. *Bending Energy* (BE) is the important parameter that the transformation outputs. Since our algorithm is based on an extension of this method, and TPS is well known geometric transformation easily found in literature, we decided not to deepen it, leaving the explanation of the method to the curiosity of the reader who can find it in Bookstein's article [3]. Once Shape Context best matching is found, appearance similarity is calculated and bending energy is extracted by Thin-Plate Spline geometrical estimation, Shape Similarity between the two investigated shapes is assigned:

$$ShapeSimilarity = aSC + AS + bBE \tag{4.7}$$

where a and b are coefficients whose value is 1.6 and 0.3, respectively [11]. In this way, Shape Context descriptor is exploited for shape matching.

The main reason why Belongie's Shape Context can not be employed in multiple mice tracking can be found in the intrinsic functioning of the algorithm. In fact, in traditional shape contexts, points are described by the other surrounding points in terms of Euclidean distance. But Euclidean distance represents just the length of the shortest path connecting two different points, and it doesn't take in consideration eventual articulations, complex shapes, internal holes, etc. This represents a big issue in mice tracking, since mice are moving animals with a complex articulated shape that can not be described in terms of Euclidean distance between pairs of contour points. Let's take in consideration the illustration showed in figure 4.16. In this case, while Euclidean distance works well in describing point relative distance in (**a**), it is not even remotely good in describing their relative position in an articulated shape such as (**b**) or (**c**). An other parameter, here introduced as *inner distance*, must be defined to consider articulations.

Jacob's Inner Distance Shape Context

Since the inability of Euclidean distance to consider shape articulations has been already discussed, we just have to introduce the *inner distance*, a new parameter invented by Jacob et al. and employed in their shape matching [11]. The *Inner distance* is defined as the length of the shortest path connecting two points within the shape. The feature of distance of being "within the shape" is of crucial importance for considering articulations in more complex shapes. The *Inner distance* is much more suitable to build a shape descriptor for articulated objects than Euclidean distance. Figure 4.16 shows two shapes where the ability of Inner Distance to consider articulations is illustrated. The *Inner distance* is very similar to the geodesic distance on surfaces. In fact, the geodesic distance is defined as the length of the shortest path connecting two points on the surface. Geodesic surface is invariant

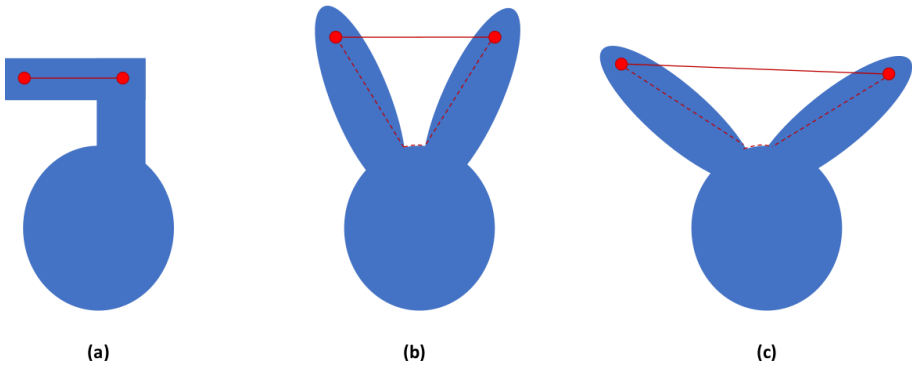


Figure 4.16: Illustration of inability of Euclidean distance to consider articulations. The continuous line refers to Euclidean distance while dashed line to Inner Distance. (**a**) shows a simple shape where Euclidean distance coincides with Inner Distance and it is enough to describe the relative positions of two points. (**b**) and (**c**) show two articulated shapes as if they were taken in two different frames, where Euclidean distance doesn't work well to identify points relative position, and Inner Distance must employed. It is shown that Inner Distance correctly considers articulations.

to surface bending. This invariance is similar to the articulation invariance that *inner distance* aims. When 2D shapes are considered, their surface is represented by their contour. As consequence, the geodesic distance between two points on the contour is the distance between them along the contour itself. Hence, considering 2D shapes, geodesic distance and inner distance lose their similarity, since *inner distance* is represented by the distance between two points within the shape. The easiest way to compute the *inner distance* consists in two different steps:

1. A specific number of sample points is defined on the shape contour. Then, a graph is built with the sample points. For each pair of sample points p_1 and p_2, if the segment connecting them falls entirely in the shape, a line between them is added to the graph with a weight equivalent to the Euclidean distance $||p_1 - p_2||$. In case the segment falls outside the shape, it is discarded and not added to the graph.

2. A shortest path algorithm is applied to the graph in order to detect, between every pair of points p_1 and p_2, what is the shortest set of consequent segments allowing to connect the two points. The shortest path is defined as *inner distance*.

An illustration of the graph built with the sample points is shown in figure 4.17. From inner-distance extraction algorithm, it is clear that there are two important features that are intrinsic in inner distance:

1. It is sensitive to holes and internal contours belonging to the shape, without considering sample points on internal borders, such as hole contours. In fact,

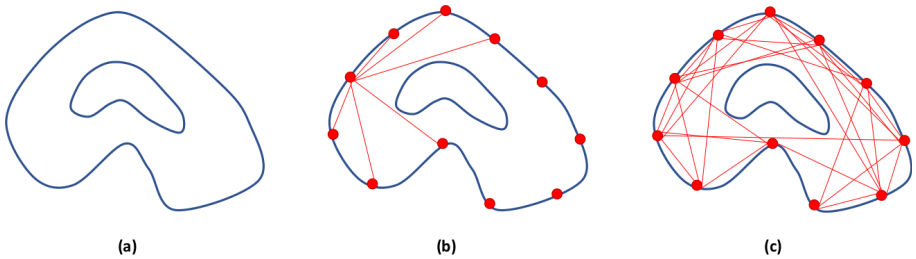


Figure 4.17: Illustration of inner distance calculation algorithm. The first step consists in building a graph with the sample points. Only segments falling entirely within the shape are considered. (**a**) shows a generic complex shape. (**b**) shows the segments connecting a single point to all other sampled points on the contour. Edges that are not traced didn't fall within the shape boundaries. The result (**c**) is a big graph with edges weighted according to the Euclidean distance of segments connecting relative pair of points.

since the segment is traced only if it falls entirely within the object, holes are taken in consideration.

2. It is insensitive to articulation and part structures. In fact, any articulated shape can be divided in smaller rigid part structures connected by junctions. According to this, the shortest path connecting sample points can be divided into segments within each single part.

These features are extremely important when complex shapes, such as mice, are considered. A comprehensive demonstration of inner-distance insensitivity to articulated shapes is available in Jacob's article [11]. Once inner distance is defined, we just have to introduce the Inner-Distance Shape Context. *Inner-Distance Shape Context* follows nearly the same steps as Shape Context implemented by Belongie et al., with the difference that histograms of each point on shape contours are obtaining mapping *inner distance* and *inner angle*. While the former has already been discussed, the latter must still be defined. Given a shape and its boundary, for each sample point p on the boundary and its shortest path $\Gamma(p, q, O)$ to another point q, we define *inner angle* the angle between the contour tangent at p and the direction of $\Gamma(p, q, O)$ at p. This angle is insensitive to articulation and it is used for the orientation bins in the shape context. Once shape contexts of all sample points on the shape contour are extracted using inner distance and inner angle, the following steps are the same than the ones explained for Belongie's shape context. A matching cost function, following χ^2 statistics, is created to find out what is the best matching between each point on the contour of a shape and each point on the contour of another one. Then, a total matching cost is defined and it is minimized using dynamic programming (DP). A penalty τ is considered when a point p_i is left unmatched. Here is a big difference with Belongie's approach. In fact, dynamic programming, with the cost functions defined in equations 4.5 and 4.6, is used instead of a linear

combination of shape context distance, appearance similarity and bending energy. Then, shape similarity is found combining results given by the Inner-Distance Shape Context and Dynamic programming.

$$ShapeSimilarity = IDSC + DP \tag{4.8}$$

It was demonstrated that $IDSC + DP$ framework, besides giving excellent results in terms of shape description, is simpler than $SC + AS + BE$ described for Belongie's Shape Context [11]. Results in terms of quality of IDSC as shape descriptor are shown in figure 4.18.

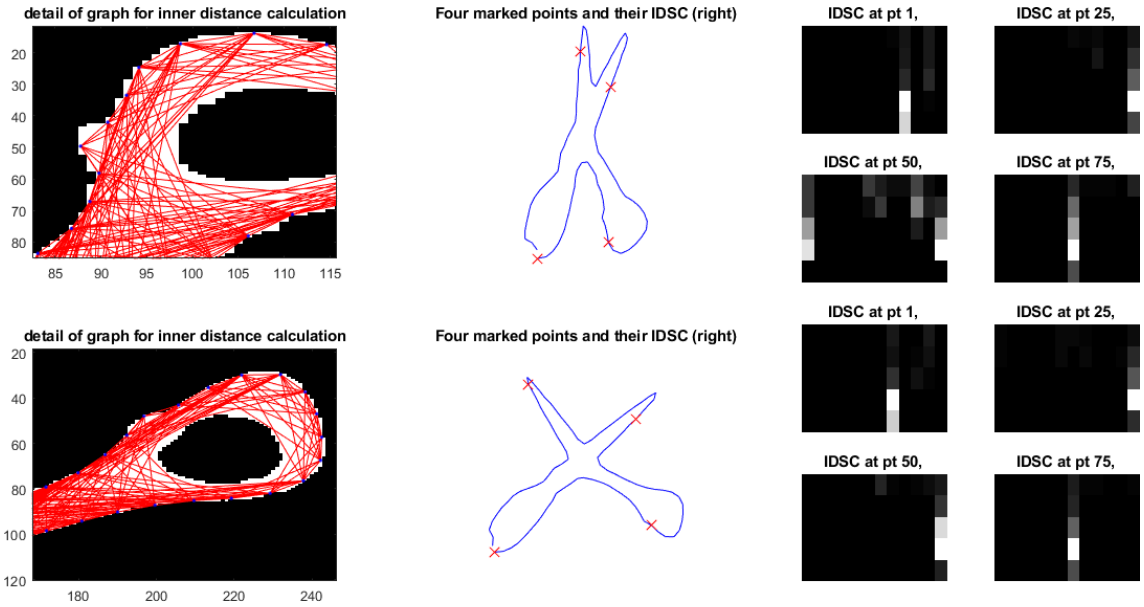


Figure 4.18: Inner-Distance Shape Context applied to scissors. The figure was obtained using the MATLAB code provided by H. Lind and David W. Jacobs [11]. On the left, zoom of the graph built with sample points for inner-distance calculation is shown. In the center, four points are considered on both scissors shape. On the right, IDSC of all four points for each shape are shown. It is immediately clear that points located in the same position in different shapes show the most similar shape context. For instance, shape context at point 25 on the first shape is similar only to shape context at point 25 on the second shape. The same is for all the other three points.

IDSC-Tracker

Now that all the algorithms at the base of our tracking system have been comprehensively explained, our tracker, that is run in case of necessity of restoring landmark points and identities, can be described. We applied inner-distance shape matching method to mice to obtain a point-to-point matching of their contours. We started

using a reference shape of a mouse and using IDSC to match each point of the contour of the reference shape to the correspondent point on the contour of the shape we are interested in. The point-to-point matching through IDSC can be used to track any point belonging to the contour of a mouse we are investigating. This fits exactly our need, since we are interested in finding two landmark points for tracking (head and tail) and two for identity detection (left and right ear). An illustration of IDSC applied to mice is shown in figure 4.19 and the point-to-point matching between a reference shape and a generic mouse shape is visible in figure 4.20.

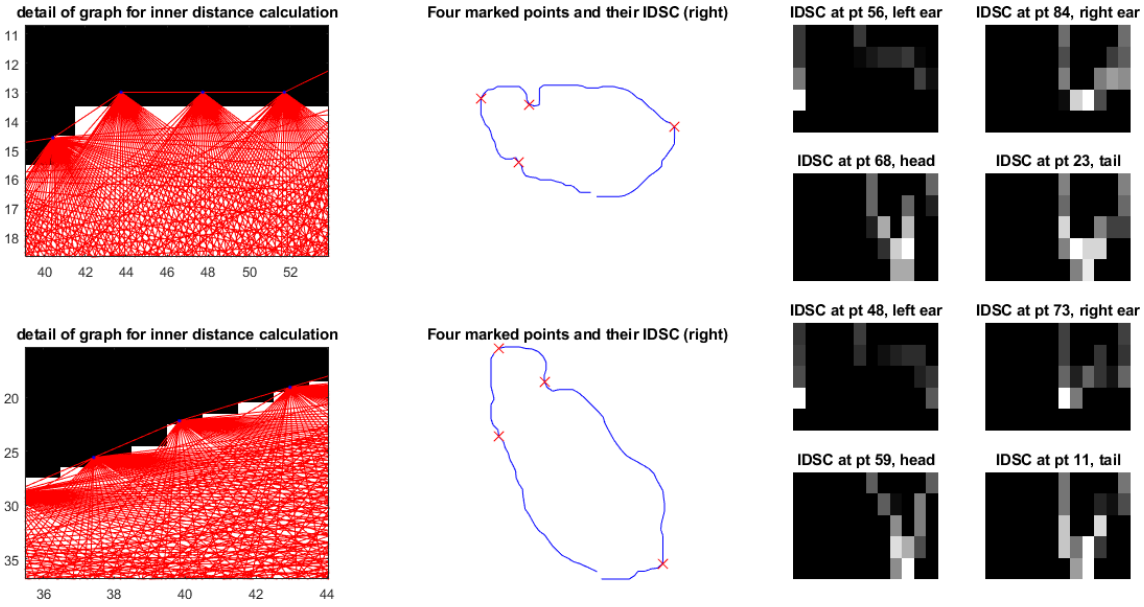


Figure 4.19: IDSC shown for four marked points on two different shapes. The figure was obtained using the MATLAB code provided by H. Lind and David W. Jacobs [11]. A first fast comparison shows that IDSC histograms relative to the same point on two different shapes are very similar while the ones relative to different points show evident differences.

Results achieved in terms of point matching were very promising for our aim. Hence, we decided to employ this technique to implement our *IDSC-Tracker*. Nevertheless, there are two main big problems we had to face:

1. The computational time requested by the algorithm is extremely high. In fact, according to Jacob, a complete matching process between two different shapes where 100 sample points were taken by each shape contour takes about $0.31s$ on a regular Pentium IV 2.8G [11]. With nowadays machines, the computational time can be considerably reduced. However, as explained in previous chapter, we aim to implement an algorithm that can be run on a simple machine such as a raspberry pi. That is why we can not afford such a high computational demand.

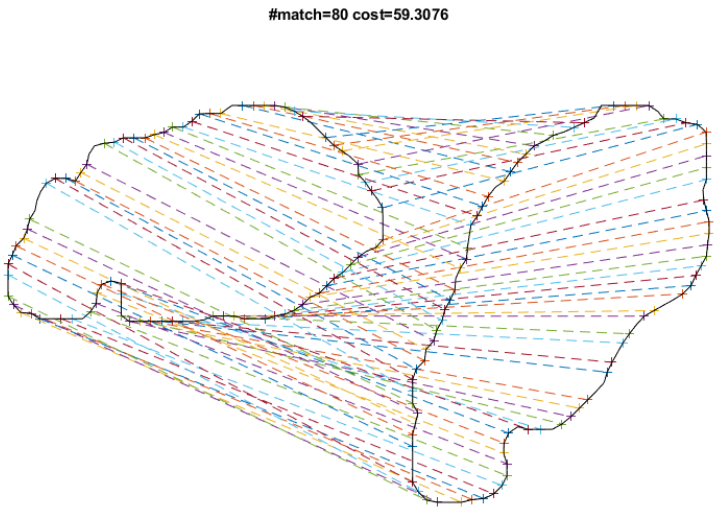


Figure 4.20: Point-to-point matching shown between generic mouse contour (on the right) and the reference contour (on the left). The figure was obtained using the MATLAB code provided by H. Lind and David W. Jacobs [11]. Matching shows IDSC invariance to rotation and translation. Matching cost and number of matched points are shown.

2. Using one reference shape only is definitely not enough to well track all points in all the possible shapes mice can assume. In fact, mice move very fast and their segmentation can assume shapes that do not show a geometry similar to the reference one, badly influencing the performance of the algorithm. That is why one reference shape is not enough to well identify landmark points on every possible shape assumable by a mouse.

Solving these problematics is not trivial. In addition, even though they do not seem to be related, the first one is strictly dependent on the second one. In fact, with one reference shape only, we obtain the best results in terms of speed of the algorithm. If we increase the number of reference shapes, we automatically increase the number of shapes that must be matched with the one belonging to the investigated mouse. Hence, the more is the number of reference shapes, the better is the result, but the higher is the computational demand. While the issue relative to the high computational demand can not be definitely solved, we can still find a good solution for the number of reference shapes employed for matching.

To face this problem we took 300 hundred different shapes and we decided to run a principal component analysis (PCA) to detect what are the parameters that allow us to better describe different shapes relative to different postures assumed by mice in their movement. Then, all 300 hundred shapes were plotted in a 2D plan with 2 principal components and a k-means clustering with no centroid initialization was run to detect 5 shapes that we take as reference. Reference shapes extraction through PCA went through the following steps:

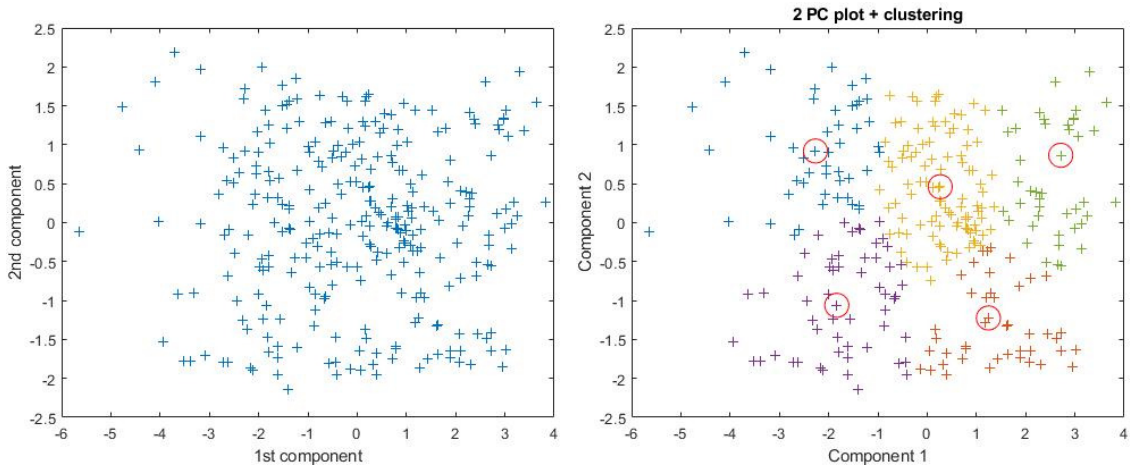
1. 300 hundred shapes were taken. Each shape is described by the following set of parameters: average speed, speed orientation, area, major axis, minor axis, eccentricity, ellipse orientation. Average speed and speed orientation are calculated on the distance run between two subsequent frames: the current and the previous one.

2. All data were weighted according to their variance.

3. PCA was executed on the data set, and 2 principal components resulted enough to explain more the 60% of the total variance of the data set. With 3 principal components, about the 70% of the variance was explained. We chose to describe the dataset with 2 principal components since the difference was not consistent.

4. K-means clustering with no centroid initialization was executed on the matrix of the scores. Five clusters have been identified and the five shapes identified by the five centroids were taken as reference shapes.

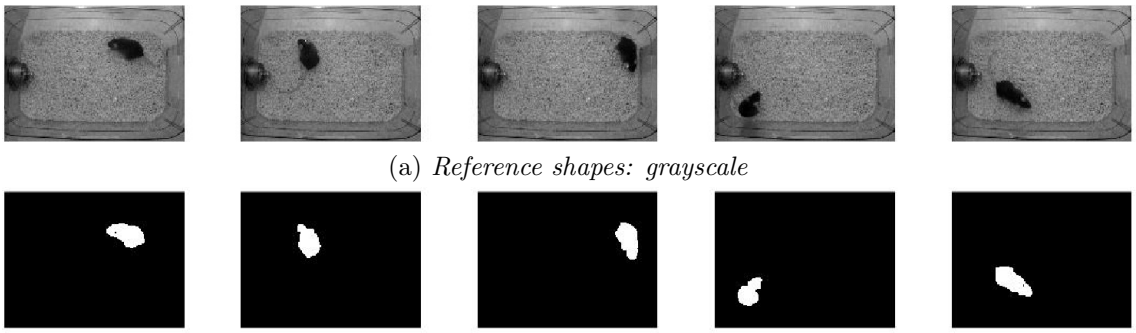
Results in terms of score matrix and clustering are shown in figure 4.21. The five reference shapes extracted by clustering of PCA results are illustrated in figure 4.22.



(a) *Score matrix on two principal components* (b) *Score matrix after clustering*

Figure 4.21: Score matrix before and after clustering. (**a**) shows the score matrix on the first two principal components that explain more than 60% of dataset total variance. (**b**) shows the five clusters obtained by k-means clustering of the score matrix. The five centroids are the five shapes taken as reference.

Results obtained by PCA made us understand that using five reference shapes instead of one only could be a good way to execute the point-to-point matching,



(a) *Reference shapes: grayscale*

(b) *Reference shapes: binary mask*

Figure 4.22: Illustration of the five reference shapes extracted by the dataset through PCA.

with any kind of shape, in the correct way. In fact, adopting this technique, results in terms of landmark points tracking and identity detection increased significantly. All these positive results made us choose to adopt five reference shapes instead of one, with consequent increase in terms of computational demand. Nevertheless, since the matching algorithm requires a long computational time even in the easiest case where only one reference shape is used, we believed that increasing the time of processing would not cause any additional problem. In fact, the algorithm could never work real time in both cases. That is why we chose to be more consistent and accurate instead of being faster. At the end of the dissertation, it will be explained why we chose to continue using an algorithm that can not work real time, even when a real-time application is our aim. In fact, we will see that this algorithm could be employed in an off-line environment to build a big "dictionary of shapes" where all shapes are described by simple geometrical and physical parameters. This big dictionary could then be used for shape matching in a fast-real-time application, since matching would consist only in comparisons based on easy algebraic operations.

The algorithm, as it is now, executes the matching between the mouse shape we are interested in finding landmark points, and the five reference shapes. Then, the best matching (the one showing minimum cost) is taken, and landmark points are detected. The positive features of IDSC-Tracker are listed below:

1. Accurate head, tail, left and right ears location detected every time it is called.

2. Robustness given by the five reference shapes employed for matching. The algorithm detects landmark points even in weak noisy conditions or when mice postures are not well defined.

3. Invariance to translation and rotation. This is extremely important to allow matching between any kind of shape with any other, no matter of their reciprocal geometry. This feature will be exploited in the identity detection algorithm.

4.8 Identity Detection and Preservation

The tracking algorithm described in the previous section partially fulfill our purpose. In fact, our tracker allows to keep track of landmark points when merging doesn't occur, and to restore them after merging occurred, when point location information is lost. Nevertheless, there are still two important features that must characterize our system in order to be suitable for automatic social behavior analysis:

1. Ability to keep track of mice identity frame by frame;

2. Eligibility for real-time applications.

The first condition is achieved by extending the IDSC-Tracker to track more landmark points, topic of discussion of this section. The second one, instead, represents not a trivial problem that can not be solved keeping the algorithm as it is. Nevertheless, a manner to exploit the algorithm for an offline shape characterization will be discussed as future work in the following chapters.

Observing the flowchart in figure 4.1, it is immediately clear that identity detection and preservation algorithm works differently according to the output of merging detector. As already discussed for the tracker, the output of merging detector is not considered in the current frame only, but in the previous too. In fact, if merging did not occur in both frames, the easy *Identity Preservation Algorithm* can be used to keep mice identities; otherwise, if merging is not occurring in the current frame, but it did occur in the previous one, it means that identities are lost and a more complicated *Identity Detection Algorithm* must be run to restore them. A partial flowchart showing a logical overview of the algorithm can be observed in figure 4.23.

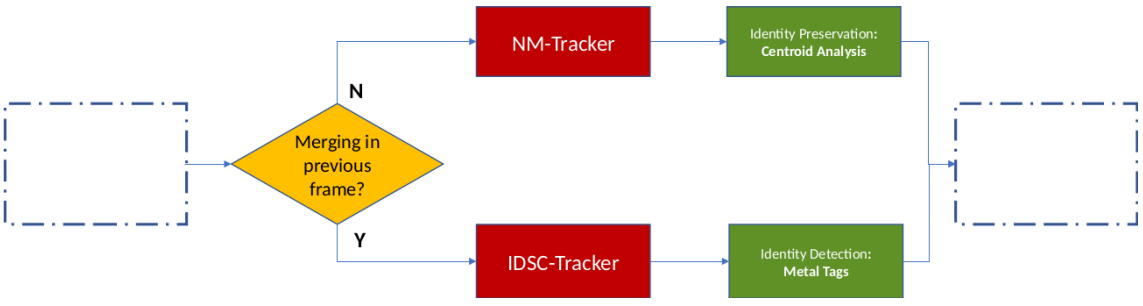


Figure 4.23: General overview of the identity detection and preservation algorithm. The flowchart implies that merging is not occurring in the current frame.

Exactly how it was already done for previous steps, *Identity Preservation* and *Identity Detection* will be treated separately.

Identity Preservation

In case merging is not occurring in the current frame and it did not happen in the previous one, mice identities can be easily tracked with a small extension of *NM-Tracker*. In fact, assuming that a mouse can move not too far between two subsequent frames, identities can be kept only measuring reciprocal centroid distances. Assuming that three mice are moving withing the cage, given the centroid $C_i(x, y)$ of the investigated mouse in the current frame, and given the centroid locations $C_1(x, y)$, $C_2(x, y)$, $C_3(x, y)$ of the three mice in the previous frame, whose identities are known, we compute the following Euclidean distances:

$$D_{ij}(C_i, C_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \qquad \forall j \neq i \tag{4.9}$$

We obtain $D_{i1}(C_i, C_1)$, $D_{i2}(C_i, C_2)$ and $D_{i3}(C_i, C_3)$ which represent the Euclidean distances from centroid C_i in the current frame, to centroids C_1, C_2 and C_3 in the previous one. The identity of the investigated mouse is assigned taking the minimum distance. An illustration of the process is shown in figure 4.24

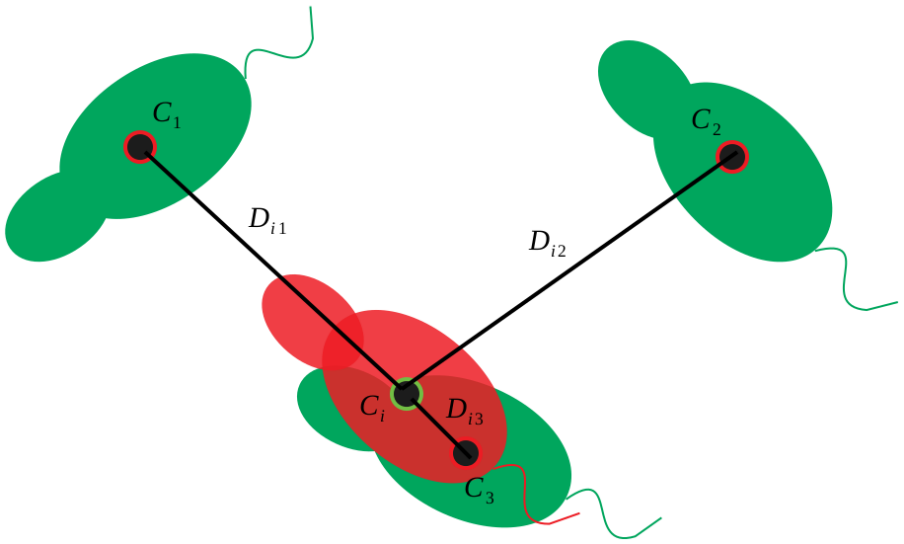


Figure 4.24: Illustration of Identity Preservation Algorithm. Green mice refer to the previous frame while one mouse only (the red one) is shown in the current frame. Distances between investigated mouse (red) and mice in previous frame (green), whose identities are known, are calculated. The minimum distance is taken and identity is assigned. In the example, investigated mouse i corresponds to mouse 3 in the previous frame. Identity is assigned. The same is repeated for the other mice in the current frame.

Since mice move separately for the most of the time, this algorithm is the most called by the system. According to this, it must be as light as possible not to influence too much the time of processing. Fortunately, this method exploits Euclidean

distances to assign new mice identities in current frame. Hence, since the mathematics is quite easy, the computational demand is very low and the algorithm can work in real-time.

Identity Detection

In case merging is not occurring in the current frame, but it did occur in the previous one, identities are lost and must be restored. Unfortunately, *Identity Preservation Algorithm* is not useful in this situation, since identities are not available from the previous frame. Therefore, a stronger algorithm is required to face this issue. In chapter 3 the experimental setup was described, and metal tags were introduced as objects employed in identity detection. In fact, spherical metal tags with a diameter of $4mm$ were bound to mice ears. To detect three mice, we realized a system following the metal tags configuration shown in table 3.1. Since we used three mice only, we didn't need the "both tags" configuration of the fourth mouse.

Identity Detection Algorithm was feasible thanks to an extension of *IDSC-Tracker*. In particular, the tracker was asked to output more landmark points. According to this, not only head and tail are tracked, but also left and right ear. Why we need left and right ear for tag segmentation will soon be clear. In fact, the first idea that comes to the mind when dealing with tag segmentation is executing a global thresholding filtering to segment tag on the whole area of the mouse. Unfortunately, even executing the best pre-processing before segmentation, it would not be possible to segment only metal tags, since other noisy pixels belonging to mouse contour would not be filtered. According to this, tag research should be executed within a small area surrounding the tag itself. Here comes the utility of knowing left and right ear position. As a matter of fact, left and right ear would allow us to create a small neighborhood around each ear, and to execute segmentation only in that specific area. In this way, all possible artifacts on mouse shape would be excluded from the segmentation, and only metal tags would be segmented. An important feature of *IDSC-Tracker* that has been exploited for metal tags detection was already noted down in the previous section, but it is here underlined again, since it represents the most important feature that allowed us to obtain a fast tag segmentation. This feature consists in tracker's invariance to rotation. In fact, this invariance makes the tracker able to identify left and right ear without executing any additional geometric operation. Since this feature makes us know which is the left and which the right ear, we don't have to do anything else but segmenting the tag, and identity will be immediately assigned. This characteristic makes the algorithm very fast, since it exploits the tracker for its aim. Fortunately, *IDSC-Tracker* is required exactly in the same situations as the *Identity Detection Algorithm*. In fact, when merging occurred in the previous frame and landmark points must be detected again, identities must be restored too. This detail is very important since *IDSC-Tracker* is not required in

any additional situation if not when already needed for landmark points. According to the flowchart shown in figure 4.23, *Identity Detection Algorithm* is run only in frames where mice are moving separately and merging occurred in the previous frame, proceeding as follows:

1. Left and right ear locations are tracked by *IDSC-Tracker*;
2. A Small neighborhood (21x21) is generated around each ear;
3. Variance filtering is carried out in both neighborhoods to highlight pixels belonging to tags;
4. Global thresholding is applied in both neighborhoods to segment eventual tags;
5. The neighborhood that shows the largest number of segmented pixels is taken as metal tag;
6. Identity is assigned according to where the tag was found (or not found in case of mouse with no tags).

Tag area and threshold for tag segmentation were estimated experimentally. Obviously, the number of segmented pixels must be similar to the number of pixels composing tag area. In case no tag is found at point 4, a bigger neighborhood (41x41) is generated around each mouse and research is executed again. In fact, if no tag is found, it is possible that the identity of the investigated mouse corresponds to the one with no tags, but it could be also an error of segmentation. As a matter of fact, artifacts occurring in mouse segmentation could be imperfect and the ear position not precise. If ear position is not precise, the neighborhood could be too small to include the tag, that is consequently not found by segmentation. A bigger neighborhood is tried to understand whether the tag is effectively not present or it is present, but the neighborhood is too small to include it. A flowchart of the algorithm is shown in figure 4.25 while figure 4.26 shows the performance of variance filtering, highlighting the neighborhoods around ears to illustrate how the algorithm works.

Once identities are assigned in the current frame, our algorithm comes to a conclusion, since no additional steps are performed. According to what we already said in chapter 1, our study was focused on tracking and identity detection because no automatic behavior analysis is achievable without a robust and consistent algorithm that is able to keep landmark points and mice identities during time. In this chapter we described only the methods used for the implementation of the algorithm. Next chapter will discuss the results obtained, and a final one will conclude our dissertation discussing the future work to achieve automatic social behavior analysis.

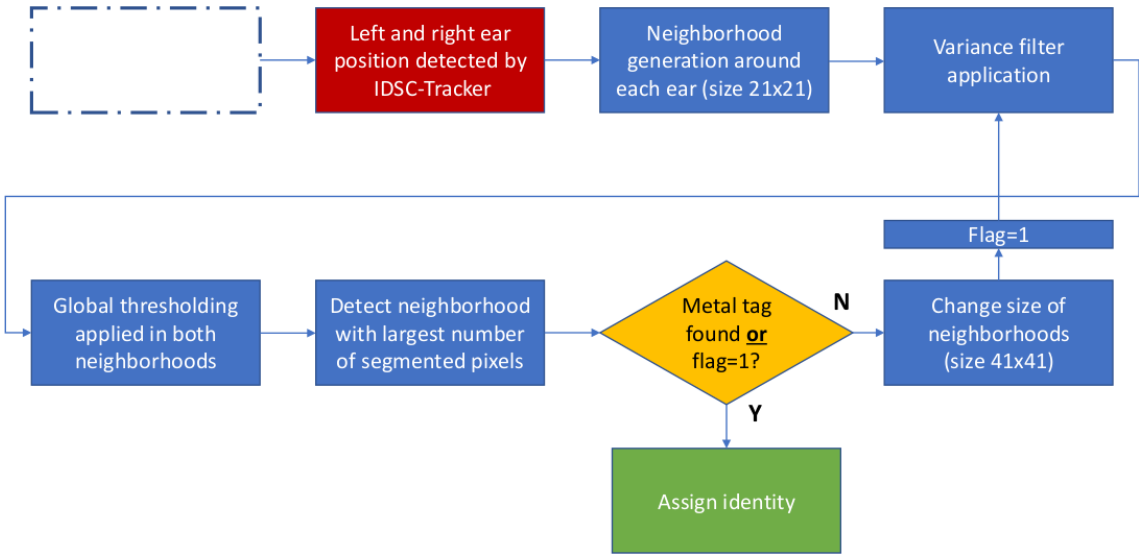
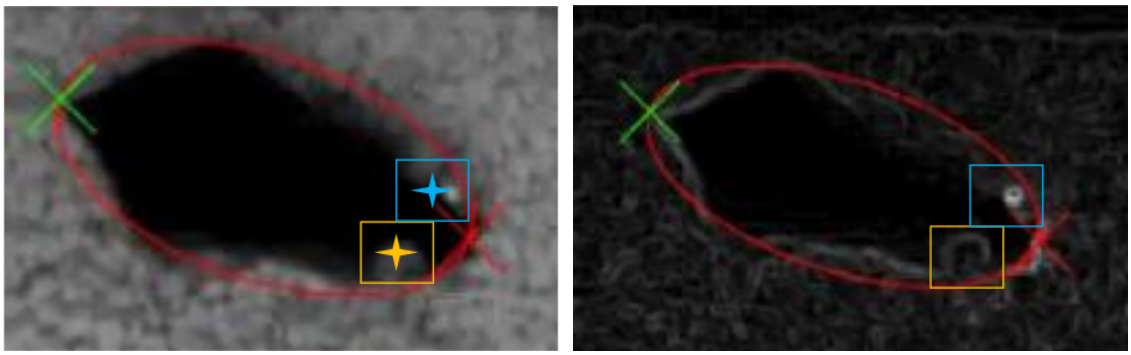


Figure 4.25: Identity Detection Algorithm overview. The square with dashed line refers to the previous steps of the algorithm. In case no tag is found, sizes of neighborhoods are increased to 41x41 for a second trial. If no tag is found again, "no tag" identity is assigned.



(a) *Before variance filtering* (b) *After variance filtering*

Figure 4.26: Illustration of the working principle of Identity Detection Algorithm. Cyan crosses indicate the left ear while yellow crosses the right one. Colored squared boxes indicate the neighborhood wherein research is carried out. Variance filter performs very well in highlighting pixels inherent to the tag (**b**). Metal tag is segmented around left ear and identity is assigned.

Chapter 5

Results

The evaluation of performance and confidence of our system was carried out on two different kinds of videos, depending on which part of the algorithm was tested. In particular, one video where three mice move within their home cage was used to evaluate the tracking algorithm, and videos where only one mouse moves were employed for the evaluation of the identity detection algorithm, that is applied frame by frame. The whole system (tracker + identity detector) on three mice was not directly tested in videos recording three mice moving, but its testing will constitute the first step of the future work. According to this, this chapter deals with results collected by the two main parts our algorithm is composed of: the tracker and the identity detector. The final section shows a result of the whole system (tracker + identity detector) applied on a video where three mice move. The result is just a preview of the future work that must be performed to get a real validation.

5.1 Tracker

Our tracking algorithm, composed of *NM-Tracker* and *IDSC-Tracker*, was tested on a 20-minutes-long video where three mice are moving within their home cage. A simple test on one video could seem a bit weak for the validation of a tracker. Nevertheless, since mice are forced to move within the cage, and since the variability of their movements is not so large, one video should easily cover the whole amount of postures mice can assume during their movement. According to this, and given a short time available we had to validate the system, we did not believe that other tests should have been carried out for a first performance validation of the algorithm.

The tracking algorithm showed very good results in terms of head and tail detected frame by frame, when merging does not occur. In fact, the validation executed showed only the 6.3% of frames with wrong points detection on a 20-minutes-long video. This result could still be improved through a post processing correction. As a matter of fact, the wrong head and tail labeling consists in the inversion of

the points. In particular, if a wrong tracking happened, it means that head was labeled as tail, and tail as head. This means that the error could be corrected by swapping again the point labels. According to this, a post processing correction could be performed to correct some among the wrong labels assigned during tracking. For example, if a mouse has head and tail labeled in the same way for a certain number of frames, and if then points are swapped for a single frame, and immediately inverted again, it means that an error occurred. This error is easy to detect, since a mouse can not move so much to invert head and tail in two subsequent frames. A high-performance error detector could be implemented to realize that an error is occurring and to finally solve it. This is just an anticipation that will be described more comprehensively in the future work section of next chapter. *NM-Tracker* showed no particular problems in geometrically tracking points frame by frame, and *IDSC-Tracker* well identified landmark points after merging, at the expenses of a higher computational time. Figure 5.1 shows a frame where landmark points were correctly tracked and labeled by our tracker. In fact, each mouse shows its ellipse with a red cross indicating head, and a green one representing tail. This result was obtained in the 93.7% of frames where merging does not occur, and can still be improved in a future work.

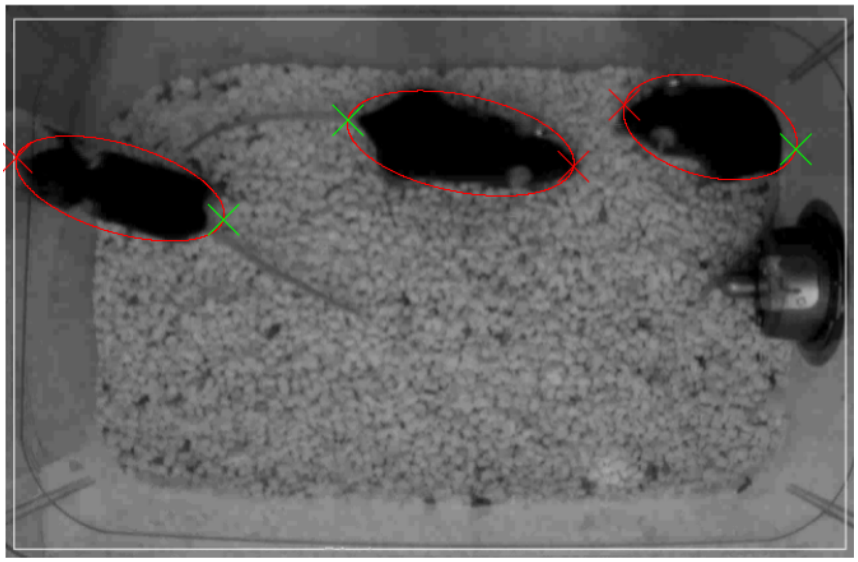
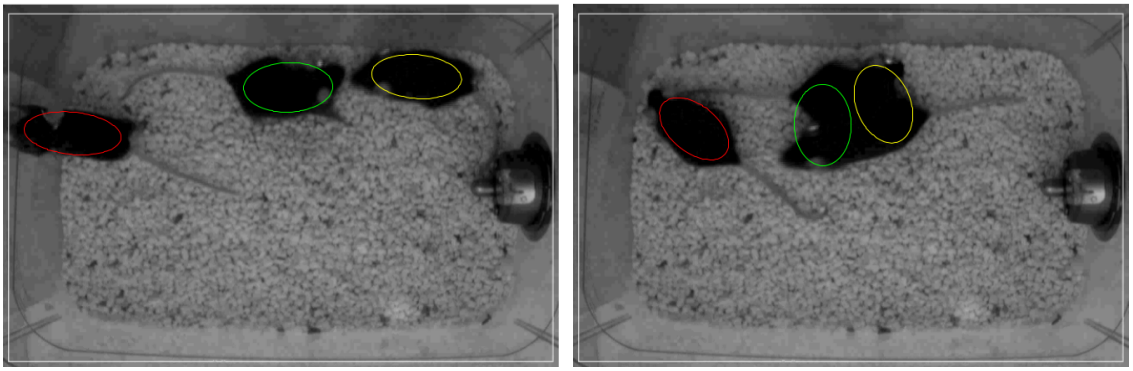


Figure 5.1: Illustration of tracking algorithm results. Head and Tail are represented by a red and a green cross, respectively.

When merging occurs, the algorithm attempts to separate points belonging to different mice, and to keep their basic geometrical parameters such as ellipse and centroid. In fact, keeping track of the centroid could be of big interest in trying to keep mice identities without using the *Identity Detection Algorithm*, as it will be explained in the future work section in chapter 6. Results of k-means clustering and

ellipse-drawing algorithm showed a good performance in separating points belonging to different mice, when these lasts get close between each other not along their sides. In fact, when they move close along their flanks, the clustering error described in chapter 4.6.2 and visible in figure 4.10, happens, and ellipses are swapped between mice. In almost every other situation, unsupervised clustering was able to correctly separate the merging blob, avoiding the employment of *Identity Detection Algorithm* (and so of *IDSC-Tracker*) to restore identities, saving a lot of computational time. Results in terms of good and bad ellipse separation are shown in figure 5.2.



(a) *Good clusters separation* (b) *Bad clusters separation*

Figure 5.2: Illustration of ellipse separation algorithm during merging. (**a**) shows a frame where mice position allows to get a good separation of points belonging to different mice. Ellipses are correctly drawn, and centroids are detected. (**b**) shows a frame where mice get close along their flanks, not allowing a good clustering. The result is the loss of centroids position, and a bad ellipse tracing.

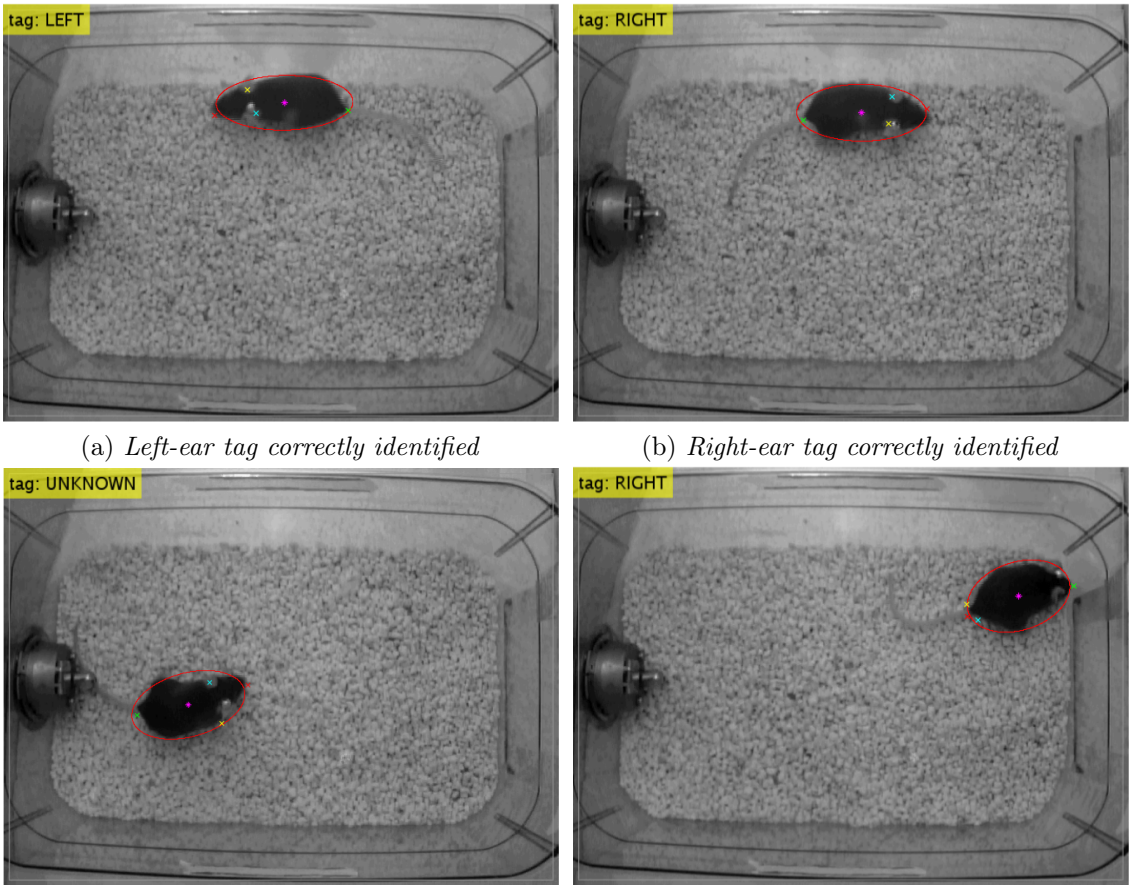
5.2 Identity Detector

Our identity detector, composed of the *Identity Preservation Algorithm* and the *Identity Detection Algorithm* was tested on twenty different videos where only one mouse was put into the cage. Specifically, in ten videos a left-ear-tag mouse was recorded, while a right-ear-tag one was recorded for the other ten. All videos are 1-minute-long, for a total of 20-minute-recording. Actually, *Identity Preservation Algorithm* was tested on the same video the tracker was validated on, since three mice were required for it. The twenty videos were used to test the *Identity Detection Algorithm*, since it represents the most delicate algorithm, hence the most important to be validated.

Identity Preservation Algorithm didn't show any particular problem in keeping track of mice identities when merging does not occur in the current frame, and it did not occur in the previous one. The main errors are due to artifacts affecting

segmentation and to the relative position that mice assume in certain frames. There are some frames where mice get close between each other in a way that makes the current centroid of a mouse be closer to the old centroid of the other mouse, causing a swapping of identities. Nevertheless, these errors are not very common since they take place in a very tiny percentage of frames, not constituting a big obstacle for tracking.

Identity Detection Algorithm was run on the twenty videos and statistics was performed. The algorithm was applied frame by frame, hence all landmark points and identities were tracked all the time by *IDSC-Tracker*. The algorithm showed excellent results in terms of identity detection. In fact, averaged results showed that left-ear-tag mouse identity was correctly detected in more than 80% of frames, while right-ear-tag mouse identity was properly labeled in more than 90% of frames. In addition, the wrong tag identification rate was very low in both cases. This is an excellent result, since avoiding the assignment of the wrong label is extremely important; a mouse with unknown identity is preferable to a mouse with a wrong identity to prevent the wrong identity from being kept for further frames. Unfortunately, this excellent result is opposed to the high computational demand required by the detector. In fact, *IDSC-Tracker* takes $0.28s$ to execute a matching between the five reference shapes and the investigated one, on a i7 processor running at $3GHz$, meaning that about $0.9s$ is the time required by the algorithm to find out the identities of three mice (times calculation were carried out considering the whole algorithm, pre-processing included). This time demand is huge and can not be afforded by our algorithm that aims to a real-time application. Nevertheless, an off-line application of our detector, that lays the groundwork for a real-time application, is described in the future work section of next chapter. Figure 5.3 shows two frames where identities were correctly detected by our system, and two frames where they were not detected and wrongly detected, respectively. The unknown detection is due to *IDSC-Tracker*, which assigned the right ear position to a point laying too far from the real ear. This wrong assignment influenced the performance of the detector, that was not able to find the tag within the right-ear neighborhood. The wrong identity detected in 5.3 (**d**) is due again to a malfunction of *IDSC-Tracker*. In fact, despite head and tail are well identified, left and right ear are detected in a totally wrong position, close to the tail. This is certainly due to a problem relative to reference shapes. Despite the five shapes we took as reference come from a good pca selection, they are not enough to cover the whole amount of possible postures a mouse can assume. Using two components we were able to explain something less than the 70% of variance of the dataset used. The selection of 5 shapes from that dataset reduces a lot the variance explained, making the algorithm subjected to possible errors in phase of landmark points detection. The *Identity Detection Algorithm* was run on the twenty videos and statistics was obtained. In both cases, the algorithm was able to well identify the mouse in the major parts of frames. Results are averaged on ten videos



(a) *Left-ear tag correctly identified*

(b) *Right-ear tag correctly identified*

(c) *Right-ear tag not identified*

(d) *Left-ear tag wrongly identified*

Figure 5.3: Performance of the *Identity Detection Algorithm*. (**a**) shows a frame where a left-ear-tag mouse was correctly identified, while (**b**) shows the good result obtained with the right-ear-tag mouse. (**c**) shows a missed identity detection due to the wrong tracking executed by *IDSC-Tracker*. Right-ear lies too far from the real ear position; the consequence is a research neighborhood too far to include the metal tag, which therefore is not found. (**d**) illustrates a frame where identity is wrongly detected, due to an error in landmark points tracking (all points are in the wrong position, as if the mouse was positioned in the opposite way). In fact, the mouse is labeled as right-ear-tag, but the tag is on its left ear.

for each mouse, and are illustrated in table 5.1 in terms of average and standard deviation. Figure 5.4 shows the bar chart and the pie chart of averaged results obtained from the twenty recordings. Specifically, ten with left-ear-tag mouse, and ten with right-ear-tag one. The average shows that the 80% of correctly labeled frames was reached in both cases and the wrong tag identification rate was very low, as already said. In addition, standard deviation doesn't show a big variance in the results, demonstrating a certain robustness of the algorithm.

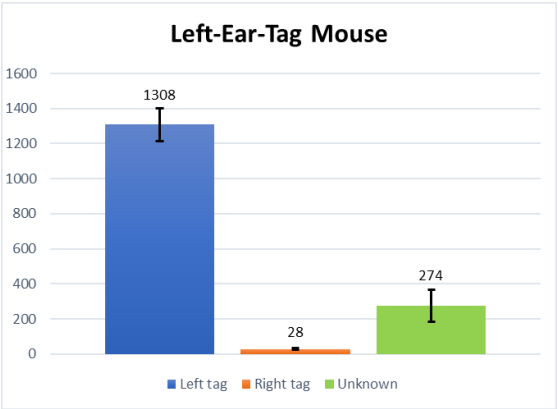
(a) *Left-ear-tag mouse*

Tag	avg (frames)	std (frames)
Left	1308 (81%)	93 (7%)
Right	28 (2%)	6 (21%)
Unknown	274 (17%)	91 (33%)

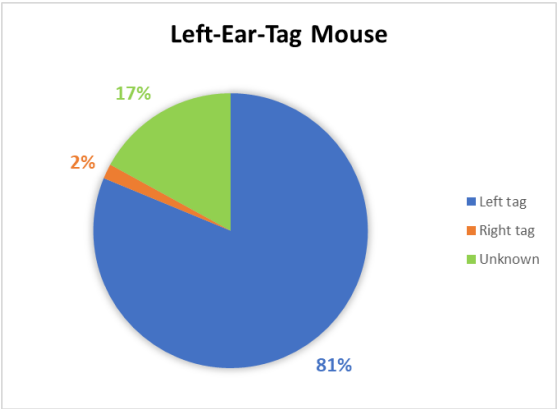
(b) *Right-ear-tag mouse*

Tag	avg (frames)	std (frames)
Left	13 (1%)	5 (38%)
Right	1512 (94%)	53 (3%)
Unknown	85 (5%)	53 (62%)

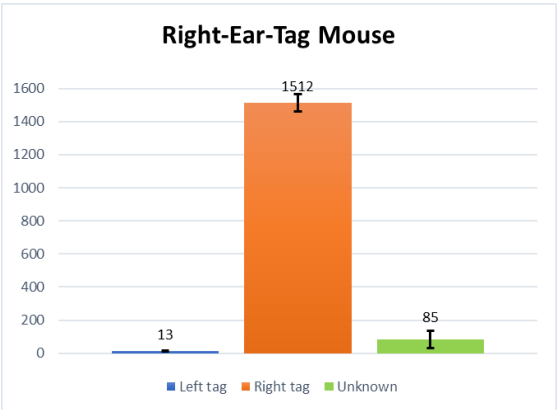
Table 5.1: Averaged results collected by ten videos for left- and-right-ear-tag mouse. The total number of frame for each video was 1610. *avg* percentages are relative to the total number of frames of the video. *std* percentages represent the intervals of confidence of the relative *avg* values.



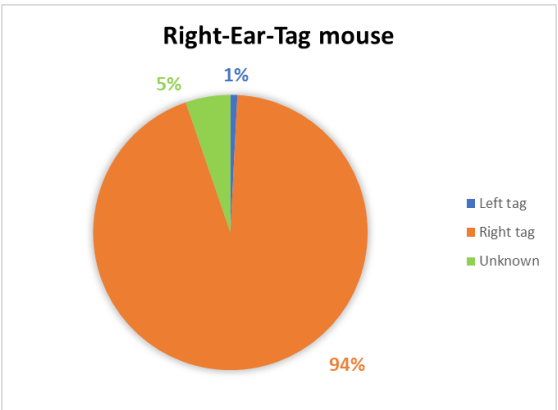
(a) *Left-ear-tag mouse: bar chart*



(b) *Left-ear-tag mouse: pie chart*



(c) *Right-ear-tag mouse: bar chart*



(d) *Right-ear-tag mouse: pie chart*

Figure 5.4: Statistics relative to *Identity Detection Algorithm* results. (**a**) and (**b**) show the bar chart and the pie chart of the averaged results obtained processing the ten videos in which the left-ear-tag mouse was recorded. (**c**) and (**d**) are relative to the averaged results obtained with the ten videos showing the right-ear-tag mouse.

5.3 Validation of the Whole System

The whole system was validated on a video where three mice were recorded. In the complete system, the algorithm of identity detection is run only after the occurrence of merging, when identities are lost. Results showed that the algorithm was able to identify and correctly label mice identities in the majority of cases. In particular, identities were correctly labeled in more than the 50% of frames, allowing a correct detection with a rate of about one frame every two. This means that we were able to detect the correct identities in two frames (at most) after merging occurred. Since the camera recorded videos at $30fps$, if our system could work in real time, it would be able to identify mice in $67ms$ at most. Unfortunately, *IDSC-Tracker* takes $0.9s$ to detect landmark points on three mice, lengthening the processing time to $1.8s$, since two frames are required for the identification. Here again is underlined the impossibility of our system, as it is now, to be employed in a real-time application. Anyway, an example of the good results collected are visible in figure 5.5, where a frame immediately after merging is illustrated with labels indicating the right identities of mice.

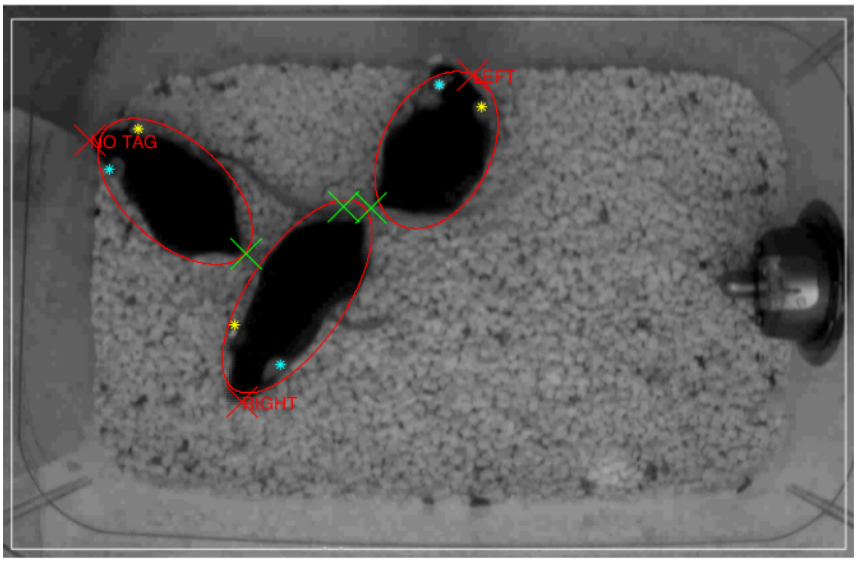


Figure 5.5: Illustration of a frame showing a situation immediately after merging. The algorithm was able to correctly identify mice. Mice identities are visible in the red labels.

Chapter 6

Future Work

The project we aim to is a big challenge which a huge amount of biological laboratories are taking part to. Even though our study is very promising in terms of collected results, the final achievement of automatic social behavior analysis is still very far away from where we are now. Nevertheless, every big construction is built brick by brick, and we are glad for having taken part to one of this series of steps that lays the basis for the future work that will have to be carried out. According to this, we believed important to give some information about the future steps required for a good employment of the algorithm that was comprehensively described in this dissertation. In this chapter, we provide a short description about the next steps we propose us to go through in our future work. In particular, we start describing small extensions that should be performed on our algorithm to obtain better results and to get a robust validation that would allow our system to be compared with other similar state-of-the-art systems to evaluate outcome differences.

6.1 Database of Videos

The first important step that must be performed is the recording of a specific set of videos that will be used as reference dataset for all application of any kind of algorithm. Only in this way a good validation of the system can be performed, and a cross evaluation of the performance of different algorithms can be executed. Until we don't create a database of videos that could become a standard in tracking algorithm validation, we won't be able to validate our system and to characterize it in terms of sensitivity and specificity. That is why this part of the future work is the most urgent, since a system with no strong validation can not be of any interest for any kind of application.

We propose us to build a database containing the following videos:

- Ten one-minute-long videos in which one single left-ear-tag mouse is recorded;

- Ten one-minute-long videos in which one single right-ear-tag mouse is recorded;
- Ten one-minute-long videos in which one single no-tag mouse is recorded;
- One 20-minute-long videos in which three tagged mice are recorded;

Videos must be recorded with the camera placed in the center of the lid of the cage as illustrated in figure 3.1, at 30fps. In general, the whole experimental setup described in chapter 3 must be respected as it is. Videos can be recorded using different cameras to evaluate differences given by the acquisition system. Anyway, for each camera, the entire set of videos must be recorded. In addition, it is important that cameras do not have an IR filter, since IR radiation is the one we employ to lighting the environment. In case a camera shows an IR filter, it must be removed. In fact, IR filters are designed to reflect or to block IR radiation while passing visible light. Since our environment is mainly lighted up by IR radiation, we can not afford a recording with IR filter.

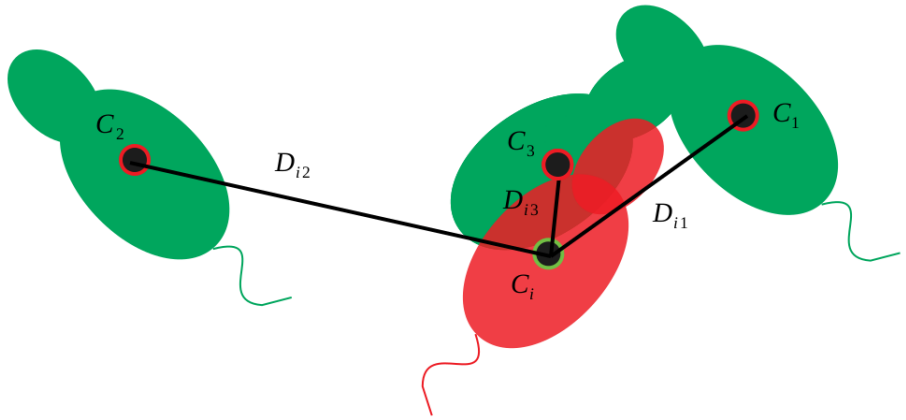
Another important requirement that videos must respect consists in the movement of mice. In particular, recorded mice must move during the acquisition for the most of the time, and do not have to show a static behavior. Only in this way we will be able to characterize our algorithm, since both static and dynamic conditions will take part to validation.

Once the whole database of videos will be created, it can be shared with the scientific community in order to make it available to anyone interested in comparing the performances of different algorithms. In other words, the database could become a standard in multiple mice tracking validation.

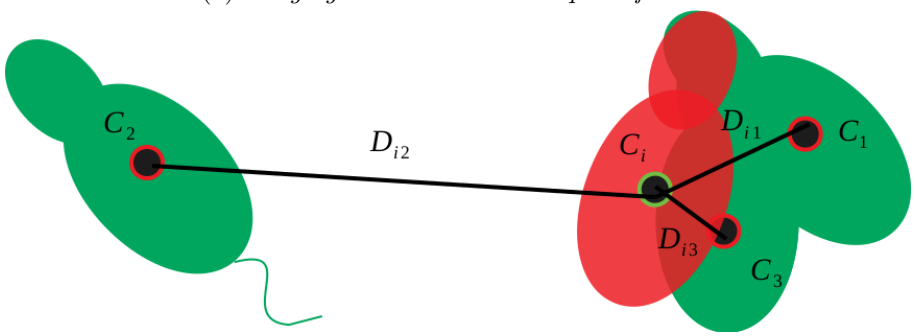
6.2 Centroid Tracking During Merging

Our algorithm does not attempt to evaluate social interactions during merging, as already repeated several times in the dissertation. Hence, landmark points tracking is not required in frames where merging occur. Nevertheless, keeping track of positions of centroids, belonging to different mice during merging, would be an excellent way to avoid the employment of the heavy *IDSC-Tracker* and *Identity Detection Algorithm* to restore landmark points and identities. In fact, if we were able to keep centroids frame by frame during merging, the easier *Identity Preservation Algorithm* would be largely enough to keep identities. As a matter of fact, when more mice move so close to get a single segmented blob, if centroids are still available, the identity of a single mouse can be kept measuring distances between its centroid in the current frame and the three centroids in the previous one. Then, the minimum distance is taken and identity is assigned. Figure 6.1 shows an illustration of this method, very similar to the one showed in figure 4.24 for the *Identity Preservation Algorithm*. Identity preservation in case of good centroids detection is illustrated

in two different situations: when merging is finishing and when merging continues between subsequent "frames". Since this method would be feasible only if positions



(a) *Merging ends between subsequent frames*



(b) *Merging continues between subsequent frames*

Figure 6.1: Illustration of Identity Preservation Algorithm during merging. Investigated mouse (red) belongs to the current frame, while other mice (green) belong to the previous one. If centroids position are detected and available during merging, keeping mice identities would be fast and easy, since the simple *Identity Preservation Algorithm* would be enough. In fact, in both cases, the distance D_{i3} represents the shortest one. According to this, identity 3 is assigned to the investigated mouse i.

of centroids were well detected during merging, a better algorithm than the simple k-means clustering must be implemented. Kalman filter can be exploited for this aim. A Kalman filter is an optimal estimator that infers parameters of interest from indirect, inaccurate and uncertain observations. It's a recursive method, so new measurements can be processed at the same time they arrive. It's an "optimal" estimator because it attempts to minimize the mean square error of the estimated parameters, reaching an "optimum". Kalman filter "learns" and minimize the error during time in real time. This characteristic could be employed in our algorithm to estimate mice centroids positions, especially during merging. In this way, we could

get information about centroids in any kind of situation, and the *Identity Preservation Algorithm* would be employed in the largest amount of frames instead of the demanding *Identity Detection Algorithm*, allowing a real-time processing.

6.3 Post-Processing Correction

According to what anticipated in section 5.1, a post-processing error correction could be executed to increase the performance of the tracker in terms of head and tail detection. To achieve this, a high-performance error detector must be implemented. Since all our study was carried out under the hypothesis that a mouse can not move much between two subsequent frames, the implementation of an error detector does not represent a big deal. Let's suppose that head and tail labels were correctly assigned for a certain non-small number n of subsequent frames. Then, let's say that an error occurred in one frame and, after that, labels were correctly assigned again for another non-small number q of frames. If an error occurred between the n and the q frames, it means that head and tail labels were swapped, ending up with a head labeled as tail, and a tail labeled as head. Since a mouse is not able to move much between two subsequent frames, and since labels were correctly assigned for a large number $n + q$ of frames with just one occurrence of mislabeling, it means that an error occurred. This error could be solved in a post processing phase, swapping again the labels of head and tail. Therefore, despite the unknown causes of the error, we could be able to solve it restoring the correct labels. Figure 6.2 shows a graph where the trend of x and y coordinates of head and tail points is illustrated. It is easy to detect an error in points labeling since there is an unexpected crossing between the coordinates of the two points, result of an error in landmark points detection. According to this, a post-processing error correction technique could be developed to avoid a good part of head and tail wrong detections.

6.4 The Challenge of the Dictionary of Shapes

Despite the good outcomes obtained with our system, we must recognize that some changes should be carried out to reduce its computational demand. In fact, as already highlighted in the previous chapter, the system, as it is now, is not suitable for a real-time application. Since biological laboratories are very big, hosting thousands and thousands of cages, we can not afford a simultaneous recording of all videos with consequent storage before processing. This would require huge systems of memory and it would considerably slow down the whole process. As a matter of fact, every single recorded video should be sent to a central memory. Since organizing a lab in order to send all videos by cable connections is not an easy deal, wi-fi transfer would be used, drastically lengthening the process. This is something that nowadays

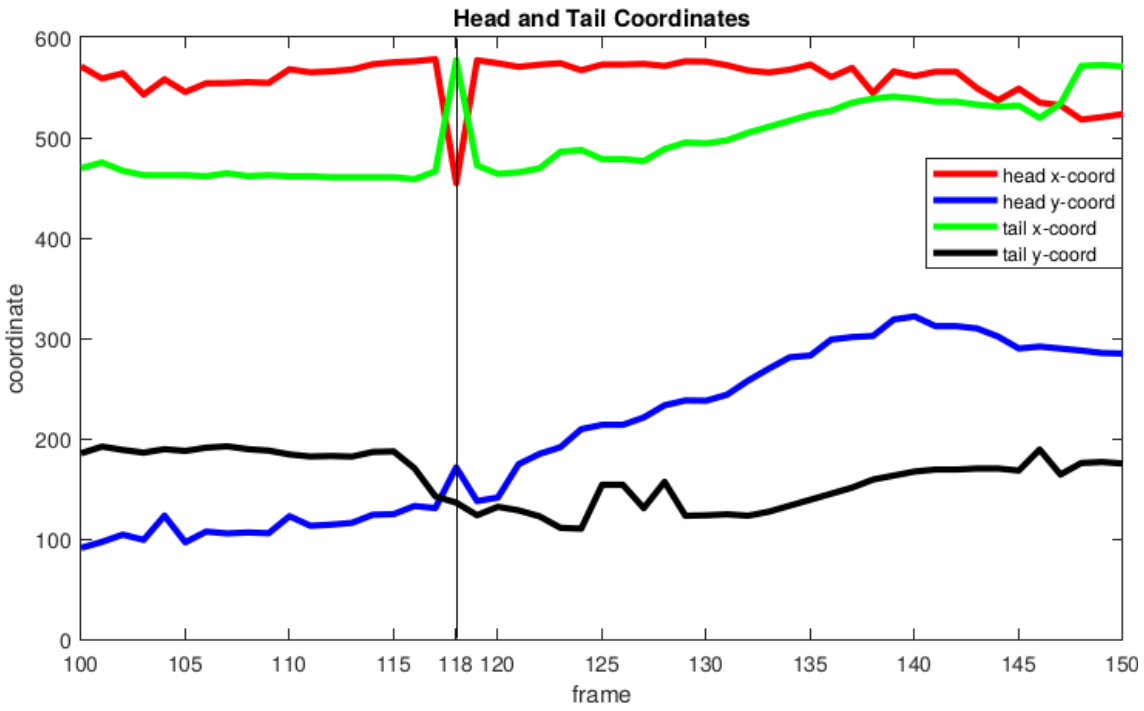


Figure 6.2: Graph illustrating the trend of x and y coordinates of landmark points. Head and tail are correctly labeled from frame 100 to 117 and from frame 119 onwards. It is immediately clear that an error occurred in frame 118. In fact, x and y coordinate of head and tail results swapped. The phenomenon is visible by a crossing of lines inherent to points coordinates.

laboratories do not want. Hence, improvements to our system should be performed. Even though a series of measures could be taken to speed up our algorithm, they would probably not be enough to reach a real-time processing. In fact, the only adjustable parameters are related to the algorithm of shape matching through Inner Distance Shape Context, but even reducing them to the minimal processing time, it would still be not enough to reach a real-time implementation. According to this, we could think to reduce the number of sampled points from shapes contours, in order to decrease the number of points for matching, or we could use just one reference shape, decreasing of $\frac{1}{5}$ the processing time, but we would obtain a considerably worse result with no significant improvements in terms of computational time. Hence, we did not consider to make big changes to the algorithm, but to find another application that could be of help for our final aim. We found this application in an off-line processing.

The strength of our algorithm consists in its ability to precisely detect landmark points, frame by frame. This ability could be extended also to any matching point we are interested in, even the whole set of sampled points for mice contour. Basically,

the whole point-to-point matching could be used for an off-line application, since time is not an important constraint. This means that all this extremely significant information about shapes could be used with a set of other geometrical parameters extracted by the algorithm to characterize a big set of shapes containing all possible postures mice can assume. With "shape characterization" we mean finding out a parameter, that we call *energy*, that could operates as a specific fingerprint of a specific shape. This parameter would be extracted as linear combination of a big set of other parameters extracted from the shape by our algorithm. For instance, this set of parameters could include: head, tail and ears position, shape size, ellipse parameters, shape asymmetry (to evaluate relative position between head and tail), shape similarity with reference shapes, etc. All this process of shape characterization would be executed off-line on standard videos recorded to obtain a number of segmented shapes that would be able to cover the majority of possible postures assumable by mice. Since the process would run not real-time and time would not be a big issue, we don't really care about the number of shapes but only about the results. Once all shapes would be characterized by their energy, a big *dictionary of shapes* could be created. In this hypothetic dictionary, each shape would be described by its one and one only energy value. If this *dictionary of shapes* was created, the shape matching would consist in a simple algebraic comparison between shapes energies. In fact, hypothesizing that we want to find landmark points in a current shape segmented by our algorithm, we should just extract the energy of the investigated shape, and calculating the difference between the energy of the shape we are interested in and all the energies on the dictionary of shapes. The best matching would consist in the shape with the energy that brought to the minimum difference. Once the best matching would be found, landmark points assignment would happen very fast. Resuming, an hypothetic algorithm to build this challenging *dictionary of shapes* could consist in the following steps:

1. Record a set of videos where mice move in a way to collect a number n of shapes $s_1, s_2, ..., s_n$ that could cover approximately all possible postures;

2. For each shape s_i, calculate its energy parameter e_i as combination of a set of geometrical parameters extracted by the algorithm;

3. Eliminate all shapes with similar energy. Each shape must be described by one and one only energy;

4. Build the dictionary of shapes.

The previous steps are meant to be performed in an off-line environment on videos recorded in a previous phase. Once the dictionary of shapes would be created, the shape matching, meant to happen in a real-time processing, would consist in the following steps:

1. Given the current frame, take the segmented shape on which landmark points must be detected;
2. Find the energy of the investigated shape;
3. Calculate the difference between the energy of the investigated shape and all energies of the dictionary of shapes. The best matching is the one that minimizes the difference;
4. Export landmark points from reference shape (in the dictionary) to the similar investigated shape (in the current frame);
5. Continue with tag identification algorithm.

The hypothetic shape matching is illustrated in figure 6.3. Basically, we propose a system where the heavy algorithm composing *IDSC-Tracker* would be used only in an off-line environment to build the dictionary of shapes. This dictionary of shapes can be as big as we want, since computational time is not a big issue in a non-real-time system. Then, the algorithm could be lightened up, since shape matching would happen executing an easy set of algebraic operations for different shapes energy comparison. Obviously, this is just a rough anticipation of an hypothetic future work that we propose us to carry out. Even though this approach can seem very promising, there is a couple of non-trivial obstacles that must be overcome. First, a good algebraic definition of *energy* must be found. Second, the exportation of landmark points from the reference shape on the hypothetic dictionary, to the investigated shape, must be defined and implemented. These two tasks are not easy to perform, since energy must defined uniquely for each shape and landmark points must be transferred with high precision to the investigated shape for a correct tag identification.

Nevertheless, we believe that our algorithm deserves to be employed in a future work to reach a real-time application, since its excellent results make it the best state-of-the-art system for multiple mice tracking. Once all the future work concerning this part of the project will be performed, new studies on social behavior analysis will start, and new forward steps will be done for the final goal of automatic social behavior classification.

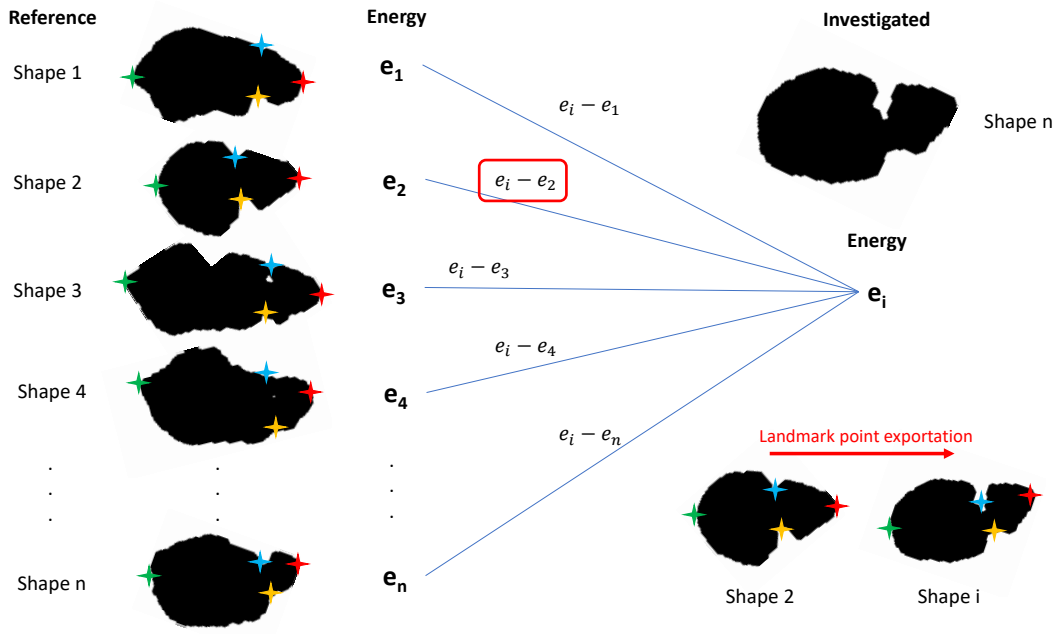


Figure 6.3: Illustration of the hypothetic dictionary of shapes. On the left, shapes with landmark points and relative energy are shown. Red, green, cyan and yellow landmark points refer to head, tail, left and right ear, respectively. On the right, the current investigated shape is shown with its relative energy. Differences between investigated shape energy and all other energies are executed. At the bottom, the shape showing the minimum difference is taken, and landmark points are transferred from the shape on the dictionary to the investigated one.

Chapter 7

Conclusions

Established research institutions, such as Johns Hopkins University School of Medicine, conduct large scale behavioral studies on specimen such as mice. As the sample size grow, the amount of human labor to conduct becomes a strong limit in the experiment, yielding to a cost-inefficient and time-consuming process, with all the problems relative to lack of standardization and low reproducibility, intrinsic in systems handled by human. Experiments revealed head and tail to be crucial points for automatic behavior analysis, and on these points our attention has been focused. State-of-the-art systems generally exploit methods which range from RFID-based technologies to machine-learning algorithms to recognize different mice moving within the same home cage. Unfortunately, despite the good results achieved in terms of landmark points tracking, these systems do not allow to track more than two mice at the same time, avoiding the possibility to classify social behaviors in a multiple mice environment for drug and illnesses characterization. The incapability of these systems to track more than two mice is mainly due to their inability to face the main issue affecting multiple object tracking: merging of segmented blobs. Merging is defined as the phenomenon that happens when more mice move so close between each other to be segmented as one big blob instead of more blobs well separated. When merging occurs, landmark point positions and mice identities are lost and must be restored. Many attempts have been performed to overcome this issue, with very poor results.

Shape matching was introduced for the first time by Belongie et al. with the *shape context* descriptor, and then extended and applied by Jacob et al. for different pattern recognition and human silhouette tracing [2] [11]. Jacob introduced a new parameter called *inner distance* as the shortest path connecting two points within the shape they belong to. This parameter was exploited for shape context descriptor to extend it to articulated shape matching. Braun applied the method for mice matching, obtaining excellent results in terms of landmark point tracking and separation of merged blobs [4]. Unfortunately, even in this case, tracking is executed on two mice only and the extension to three mice is not so trivial. In fact, the algorithm

implies that a big database of shapes representing two mice interacting is created. Then, social interaction is classified executing the Inner-Distance-Shape-Context between the reference shapes (whose social interaction they represent is known) and the investigated merged one. Finally, the new interaction is identified and labeled. This method executes several times the point-to-point matching through IDSC, a method that was comprehensively described in this dissertation. This makes the algorithm extremely demanding, excluding any kind of real-time application. Hence, we could not afford its employment but it was extremely useful as starting point for the development of our system.

Our study aims to develop an automated tracking system for identification of tagged mice for automatic social behavior analysis using advanced computer vision techniques. Even though social behavior classification is our aim, we had to face some important issues affecting multiple mice tracking. According to what was said about merging, our study was mainly focused in solving the loss of landmark points and mice identities when merging occurs. According to the Neuropathology department of Johns Hopkins University School of Medicine, social behavior identification is not required during merging, representing a secondary problem. More important is landmark points tracking in frames immediately before and after merging and mice identity preservation. Having accurate results in these terms would allow us to well identify social interactions (exploiting landmark points), always knowing which mouse is who (by identity detection).

Following this important information, we started to develop our system, whose steps are resumed in the flowchart shown in figure 4.1. In essence, a pre-processing phase with background estimation and subtraction, with consequent foreground extraction, is executed. Then, an algorithm that identifies and removes static objects which could interfere with mice segmentation is performed. Finally, the important merging detector, implemented to understand whether merging is happening or not, is run. According to the output of merging detector, geometric information is differently extracted. After that, the core of the algorithm, consisting in landmark point tracking and identity detection, is run only in frames where merging is not occurring, according to the output given by the merging detector in the previous frame. In fact, if merging did not occur, *NM-Tracker* is run to detect landmark points, and the *Identity Preservation Algorithm* is employed to keep mice identity, following centroids position; if merging occurred, the heavier *IDSC-Tracker* is used to detect landmark points again, and the *Identity Detection Algorithm* is run to identify and segment metal tags, bound around mice ears, to restore their identities. In this last case, the algorithm is much more computational demanding since point-to-point shape matching techniques are employed for landmark points detection. Metal tags segmentation is carried out detecting not only head and tail, but left and right ear too. Then, a neighborhood is generated around each ear, and tag research is run. Finally, tag position is assigned exploiting the properties of invariance to rotation

and translation of the Inner-Distance-Shape-Context algorithm.

The tracking algorithm showed very good results in terms of landmark point positions, detected frame by frame. A validation showed that head and tail position was wrongly detected only in the 6.3% of frames in a 20-minutes-long video where three mice move. This result is even improvable implementing a post-processing correction introduced as future work in section 6.3. The identity detection algorithm was validated on twenty different one-minute-long videos where one mouse moves. In ten of them, a left-ear-tag mouse was recorded, while a right-ear-tag mouse was employed in the recording of the other ten. Even in this case, results are very promising, since the right identity was detected in more than the 81% of frames composing the twenty videos. The price we pay for this good result is a high computational time demanding algorithm that makes it not suitable for real-time applications on common instrumentation. Nevertheless, the high accurate results obtained made us believe that our system could be implemented in an off-line phase of the processing that will represent the base of the real-time application. Future work will exploit this system for an off-line analysis and characterization of all possible shapes that mice can assume while moving within their home cage. The idea is to characterize each shape with a series of parameters that can be extracted frame by frame, thanks to the always available location of head, tail and ears, given by our algorithm. Once each shape will be characterized with these parameters, a sort of *energy* could be obtained by a combination of all the other geometrical parameters that can be extracted. Then, each shape will be entirely described by its energy that will be its univocal fingerprint, and a big *dictionary of shapes*, functioning as reference shape dataset, could be built. If we will manage to get this result, the process of finding ears (so identity) will turn into a simple arithmetic comparison between parameters, then to a fast-real-time identity detection that will lay the groundwork for social behaviors classification.

Appendix A

Ellipse-Drawing Algorithm

The ellipse-drawing algorithm introduced in 4.6.2 was described for the first time by Fitzgibbon et al. and later improved by Halir and Flusser [7] [9]. The algorithm is based on the idea that an ellipse represents a special case of a general conic that can be described by an implicit polynomial:

$$F(x, y) = ax^2 + bxy + cy^2 + dx + ey + f = 0$$

with the ellipse specific constraint

$$b^2 - 4ac < 0 \tag{A.1}$$

where a,b,c,d,e,f are the coefficients of the ellipse and (x,y) the coordinates of points. Since the long implicit form of the polynomial is uncomfortable to handle, introducing the vectors

$$\mathbf{a} = [a, b, c, d, e, f]^T$$
$$\mathbf{x} = [x^2, xy, y^2, x, y, 1]$$

we can rewrite it in a vector form

$$F_{\mathbf{a}}(\mathbf{x}) = \mathbf{a} \cdot \mathbf{x} = 0 \tag{A.2}$$

One of the most common methods for conic fitting consists in minimizing the sum of squared algebraic distances of the points (x_i, y_i) belonging to the conic represented by coefficients $\mathbf{a}$):

$$\min_{\mathbf{a}} \sum_{i=1}^{N} F(x_i, y_i)^2 = \min_{\mathbf{a}} \sum_{i=1}^{N} (F_{\mathbf{a}}(\mathbf{x_i}))^2 = \min_{\mathbf{a}} \sum_{i=1}^{N} (\mathbf{x_i} \cdot \mathbf{a})^2 \tag{A.3}$$

The parameter vector $\mathbf{a}$ must be constrained in order to avoid the trivial solution where $\mathbf{a}$ is a null vector, and to make the conic representable by any multiple of a

solution **a**. Since we want to find an ellipse, and not a generic conic, we must consider the ellipse constraint A.1. So, we want to limit the vector **a** so that the conic it represents is forced to be an ellipse. According to Fitzgibbon, the constrained problem given by A.1 is hard to solve. Since we have the freedom to scale conic parameters (**a**) because any coefficient we put as multiplier of **a** doesn't change the conic, we can turn the inequality constraint A.1 into an equality constraint:

$$4ac - b^2 = 1 \tag{A.4}$$

At this point, the problem can be formulated as:

$$\min_{\mathbf{a}} ||\mathbf{Da}||^2 \tag{A.5}$$

with the constraint condition

$$\mathbf{a}^T\mathbf{Ca} = 1 \tag{A.6}$$

where:

$$\mathbf{D} = \begin{bmatrix} x_1^2 & x_1y_1^2 & y_1^2 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_i^2 & x_iy_i^2 & y_i^2 & x_i & y_i & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N^2 & x_Ny_N^2 & y_N^2 & x_N & y_N & 1 \end{bmatrix} \tag{A.7}$$

is the design matrix and represents minimization A.3, and

$$\mathbf{C} = \begin{Bmatrix} 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{Bmatrix} \tag{A.8}$$

is the constraint matrix representing equation A.1.
Introducing the Lagrange multiplier λ and differentiating, we obtain:

$$2\mathbf{D}^T\mathbf{Da} - 2\lambda\mathbf{Ca} = 0 \tag{A.9}$$

always with the constraint

$$\mathbf{a}^T\mathbf{Ca} = 1$$

The same equation can be written introducing the scatter matrix S:

$$\mathbf{Sa} = \lambda\mathbf{Ca} \tag{A.10}$$

with the constraint

$$\mathbf{a}^T\mathbf{C}\mathbf{a} = 1$$

where the scatter matrix

$$S = \mathbf{D}^T\mathbf{D} = \begin{Bmatrix} S_{x^4} & S_{x^3y} & S_{x^2y^2} & S_{x^3} & S_{x^2y} & S_{x^2} \\ S_{x^3y} & S_{x^2y^2} & S_{xy^3} & S_{x^2y} & S_{xy^2} & S_{xy} \\ S_{x^2y^2} & S_{xy^3} & S_{y^4} & S_{xy^2} & S_{y^3} & S_{y^2} \\ S_{x^3} & S_{x^2y} & S_{xy^2} & S_{x^2} & S_{xy} & S_x \\ S_{x^2y} & S_{xy^2} & S_{y^3} & S_{xy} & S_{y^2} & S_y \\ S_{x^2} & S_{xy} & S_{y^2} & S_x & S_y & S_1 \end{Bmatrix} \tag{A.11}$$

where the operator S indicates the sum

$$S_{x^a y^b} = \sum_{i=1}^{N} x_i^a y_i^b \tag{A.12}$$

At this point, equation A.10 is solved using generalized eigenvectors. Six real solutions $(\lambda_j, \mathbf{a}_j)$ exist, but since

$$||Da||^2 = \mathbf{a}^T \mathrm{D}^T\mathbf{D}\mathbf{a} = \mathbf{a}^T\mathbf{S}\mathbf{a} = \lambda\mathbf{a}^T\mathbf{C}\mathbf{a} = \lambda \tag{A.13}$$

we just want the eigenvector $\mathbf{a}_k$ corresponding to the minimal eigenvalue λ_k. The solution of the problem represents the ellipse that best fit the studied set of points [7].

Since the solution of the equation is considerably demanding in terms of computational time, the original approach was improved by Halir and Flusser [9]. In their study, Halir and Flusser noticed that matrices **S** and **C** have a structure that can be simplified as follow:

$$D = (D_1|D_2) \tag{A.14}$$

where

$$D_1 = \begin{pmatrix} x_1^2 & x_1y_1 & y_1^2 \\ \vdots & \vdots & \vdots \\ x_i^2 & x_iy_i^2 & y_i^2 \\ \vdots & \vdots & \vdots \\ x_N^2 & x_Ny_N^2 & y_N^2 \end{pmatrix} \tag{A.15}$$

and

$$D_1 = \begin{pmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_i & y_i & 1 \\ \vdots & \vdots & \vdots \\ x_N & y_N & 1 \end{pmatrix} \tag{A.16}$$

The scatter matrix $\mathbf{S}$ can also be simplified as follow:

$$\mathbf{S} = \begin{pmatrix} \mathbf{S}_1 & \mathbf{S}_2 \\ \mathbf{S}_2^T & \mathbf{S}_3 \end{pmatrix} \tag{A.17}$$

where

$$S_1 = \mathbf{D}_1^T \mathbf{D}_1 \quad S_2 = \mathbf{D}_1^T \mathbf{D}_2 \quad S_3 = \mathbf{D}_2^T \mathbf{D}_2$$

In the same way, constraint matrix $\mathbf{C}$ is simplified as follow:

$$\mathbf{C} = \begin{pmatrix} \mathbf{C}_1 & 0 \\ 0 & 0 \end{pmatrix} \tag{A.18}$$

where

$$\mathbf{C}_1 = \begin{pmatrix} 0 & 0 & 2 \\ 0 & -1 & 0 \\ 2 & 0 & 0 \end{pmatrix}$$

In the end, vector of coefficients $\mathbf{a}$ is split as follow:

$$\mathbf{a} = \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{pmatrix} \tag{A.19}$$

where $\mathbf{a}_1 = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$ and $\mathbf{a}_2 = \begin{pmatrix} d \\ e \\ f \end{pmatrix}$

According to these simplifications, the equation A.10 can be rewritten as follow:

$$\begin{pmatrix} \mathbf{S}_1 & \mathbf{S}_2 \\ \mathbf{S}_2^T & \mathbf{S}_3 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{pmatrix} = \lambda \cdot \begin{pmatrix} \mathbf{C}_1 & 0 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{pmatrix} \tag{A.20}$$

All the simplifications carried out, with additional algebraic solution steps, takes to the definition of the *reduced scatter matrix* $\mathbf{M}$ of size 3x3

$$\mathbf{M} = \mathbf{C}_1^- 1(\mathbf{S}_1 - \mathbf{S}_2 \mathbf{S}_3^- 1 \mathbf{S}_2^T) \tag{A.21}$$

that allows us to rewrite the problem expressed by A.10 as the set of following equations: merging flag in current frame

$$\mathbf{M}\mathbf{a}_1 = \lambda \mathbf{a}_1 \tag{A.22}$$

$$\mathbf{a}_1^T \mathbf{C}_1 \mathbf{a}_1 = 1 \tag{A.23}$$

$$\mathbf{a}_2 = -\mathbf{S}_3^- 1 \mathbf{S}_2^T \mathbf{a}_1 \tag{A.24}$$

$$\mathbf{a} = \begin{pmatrix} \mathbf{a}_1 \mathbf{a}_2 \end{pmatrix} \tag{A.25}$$

The problem expressed in terms of the last four equations described can be solved considering the minimization problem already discussed, where eigenvector $\mathbf{a}$ that yields a minimal value λ represents the vector of coefficients of the ellipse that best fits the data points.

Appendix B

Pseudocode

This appendix aims to make the reader acquainted with the code described step by step in this dissertation. Since the algorithm is quite long and nested, we believed that dividing it in different parts, according to the phase each part belongs to (pre-processing, processing, post-processing), would have made it easier to read and follow. That is why this appendix is divided in different subsections. Nevertheless, since the post-processing methods were just outlined as a possible future work, we believed not useful inserting a section inherent to them.

B.1 Pre-processing

Video Reader and Variable Definition:

1: videoReader Initialization
2: load reference shapes ▷ *reference shapes found through PCA*
3: video selection executed by the user
4: counters initialization $ii = 0$
5: initialization of Blob Analysis Block: output are set to be major axis, minor axis, centroid, area, orientation, eccentricity.
6: minimum blob area representing mice set to $2000 pixels$

Background Estimation:

1: **while** video is not finished **do**
2: $ii = ii + 1$
3: calculate normalized frame
4: store normalized frame in the layer ii of the matrix $frames$
5: **end while**
6: sum all frames of matrix $frames$
7: find background by averaging: $background \leftarrow \frac{SumOfFrames}{ii}$
return background

Static Object Identification:

```
binarize background (global threshold set to 0.3)
fill connected components
for every pixel of the binarized background do
    if pixel background(i, j) belongs to static object then
        T(i, j) ← 0.99
    else
        T(i, j) ← 0.80
    end if
end for
    return threshold matrix T
```

B.2 Frame-by-Frame Processing

```
while video ≠ isDone do
    take the current frame
    subtract background
    segment foreground
    perform morphological operations
    mask ← segmentedForeground
    if n_blobs ∈ mask > 3 then
        while n_blobs ∈ mask > 3 do
            delete blob with smallest area
            n_blobs = n_blobs − 1
        end while
    end if
    find centroids
    if merging in current frame then
        isMerged=1                          ▷ merging flag in current frame
        find number of blobs that must be divided
        initialize k-means with oldCentroids
        [clusters, centroids] = kmeans(micepoints)
        newCentroids ← centroids
        mouse1 ← points ∈ cluster1
        mouse2 ← points ∈ cluster2
        mouse3 ← points ∈ cluster3
        for each mouse m_i do
            find best fitting ellipse for points ∈ cluster_i
            plot ellipse
        end for
```

else if No merging in current frame **then**
  for each mouse m_i **do**
    run Blob Analysis block
    extract major axis, minor axis, orientation, centroid, eccentricity
    plot ellipse around mouse m_i
  end for
  if isMergedOld=0 (no merging in previous frame) **then**
    run **NM-Tracker** ▷ *No-Merging Tracke*r
    assign head and tail landmark points
    run the centroid-based **Identity Preservation Algorithm**
    assign mice identities
  else if isMergedOld=1 (merging in previous frame) **then**
    run **IDSC-Tracker** ▷ *Inner-Distance-Shape-Context Tracker*
    assign head, tail, left and right ear
    run the **Identity Detection Algorithm**
    assign mice identities (left-ear tag, right-ear tag, no tag)
  end if
end if
$oldCentroids \leftarrow newCentroids$
$isMergedOld \leftarrow isMerged$ ▷ *merging flag in previous frame*
return mice identities
return landmark points
return geometrical parameters
end while

Bibliography

[1] United States Department of Agriculture (USDA), ed. *Annual Report Animal Usage by Fiscal Year*. 2016. URL: https://speakingofresearch.files.wordpress.com/2008/03/usda-annual-report-animal-usage-in-research-2016.pdf.

[2] Serge Belongie, Jitendra Malik, and Jan Puzicha. "Shape Matching and Object Recognition Using Shape Context". In: *IEEE Transaction on Pattern Analysis and Machine Intelligence* (2002).

[3] Fred L. Bookstein. "Principal Warps: Thin-Plate Splines and the Decomposition of Deformations". In: *IEEE Transaction on Pattern Analysis and Machine Intelligence* (1989).

[4] Stav Braun. "Tracking Multiple Mice". Master's Degree Thesis. Boston, MA, USA: Massachussets Institute of Technology, chap. 3.7.

[5] Luca Catarinucci, Riccardo Colella, Luca Mainetti, et al. "An animal tracking system for behavior analysis using radio frequency identification". In: *Nature America* (2014).

[6] Fu Chang, Chun-Jen Chen, and Chi-Jen Lu. "A linear-time component-labeling algorithm using contour tracing technique". In: *Computer Vision and Image Understanding* (2003).

[7] Andrew W. Fitzgibbon, Maurizio Pilu, and Robert B. Fisher. "Direct Least Squares Fitting of Ellipses". In: *Proceedings of 13th International Conference on Pattern Recognition* (1996).

[8] Luca Giancardo, Diego Sona, Huiping Huang, et al. "Automatic Visual Tracking and Social Behaviour Analysis with Multiple Mice". In: *PLOS ONE* (2013).

[9] Radim Halíř and Jan Flusser. "Numerically Stable Direct Least Squares Fitting of Ellipses". In: *Proceedings of 13th International Conference on Pattern Recognition* (1999).

[10] Joanna L. Jankowsky, Tatiana Melnikova, Daniel J.Fadale, et al. "Environmental Enrichment Mitigates Cognitive Deficits in a Mouse Model of Alzheimer's Disease". In: *The Journal of Neuroscience* (2005).

[11] Haibin Ling and David W. Jacobs. "Shape Classification Using the Inner-Distance". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2007).

[12] MathWorks. *Matlab documentation*. Online, visited on 12th of August 2018. URL: `https://www.mathworks.com/help/images/ref/bwmorph.html`.

[13] MathWorks. *Matlab documentation*. Online, visited on 13th of August 2018. URL: `https://www.mathworks.com/help/vision/ref/blobanalysis.html`.

[14] Jacqueline N. Crawley. "Mouse Behavioral Assays Relevant to the Sympotms of Autism". In: *Brain Pathology* (2007).

[15] Shay Ohayon, Ofer Avni, Adam L. Taylor, et al. "Automated Multi-day Tracking of Marked Mice for the Analyis of Social Behavior". In: *Journal of Neuroscience Methods* (2013).

[16] Alfonso Pérez-Escudero, Julià n Vicente-Page, Robert C. Hinz, et al. "idTracker: tracking individuals in a group by automatic identification of unmarked animals". In: *Nature America* (2014).

[17] Alfonso Pérez-Escudero, Julià n Vicente-Page, Robert C. Hinz, et al. *idTracker: tracking individuals in a group by automatic identification of unmarked animals Supporting Text*. 2014.

[18] Andreas T. Schaefer and Adam Claridge-Chang. "The Surveillance State of Behavioral Automation". In: *Current Opinion in Neurobiology* (2012).

[19] Amanda N. Smolinksy, Carisa L. Bergner, et al. *Mood and Anxiety Related Phenotypes in Mice*. Chapter 2. Springer, 2009.

[20] A.J. Spink, R.A.J. Tegelenbosch, et al. "The EthoVision video tracking system-A tool for behavioral phenotyping of transgenic mice". In: *Physiology & Behavior* (2001).

[21] Noldus Information Technology, ed. *EthoVision XT*. URL: `https://www.noldus.com/animal-behavior-research/products/ethovision-xt`.

[22] Noldus Information Technology, ed. *EthoVision XT technical specification*. 2013.

[23] Jakob Unger, Mike Mansour, Marcin Kopaczka, et al. "An Unsupervised Learning Approach for Tracking Mice in an Enclosed Area". In: *BMC Bioinformatics* (2017).

[24] Remco C. Veltkamp and Michiel Hagedoorn. *State-of-the-Art in Shape Matching*. 1999.

[25] Wikipedia. *Luminance*. Online, visited on 11th of August 2018. URL: `https://en.wikipedia.org/wiki/Luminance`.

[26] Wikipedia. *Radio-frequency identification*. Online, visited on 31st of July 2018. URL: `https://en.wikipedia.org/wiki/Radio-frequency_identification`.