JIMMY ANDRÉS VARGAS DELGADO

AN IMAGE STITCHING ALGORITHM FOR COMPOUND VISION RESEARCH

The research reported in this dissertation has been presented at the Costa Rica Institute of Technology as a Final Graduation Project for the Licentiate degree in Mechatronic Engineering.

The research described in this dissertation has been carried out at the Faculty of Mathematics and Natural Sciences of the University of Groningen in the Netherlands.

AN IMAGE STITCHING ALGORITHM FOR COMPOUND VISION RESEARCH

Final Graduation Project

TEC | Tecnológico de Costa Rica

Academic Area of Mechatronic Engineering

university of groningen

Faculty of Mathematics and Natural Sciences
Department of Computational Physics

by

JIMMY ANDRÉS VARGAS DELGADO

August, 2016

SUPERVISORS:
Prof. dr. D. G. Stavenga
University of Groningen
Dr. ir. M. Muñoz-Arias
Costa Rica Institute of Technology

ASSESSMENT COMMITTEE:
Dr. J. L. Crespo-Mariño
Costa Rica Institute of Technology
Msc. M. E. Vílchez-Monge
Costa Rica Institute of Technology

To Ivania, my mother.

ABSTRACT

A robot, capable of capturing images of butterfly eyes while attaching precise goniometric data lacked the ability to link the overlapping information of the images. This hindered the automation of data collection. An image stitching algorithm, as well as other necessary tools, have been designed, implemented and/or improved upon to surpass this problem. Automated data collection is now possible.

OVERZICHT

Een robot, die in staat is tot het vastleggen van beelden van vlinderogen, tegelijkertijd met het verkrijgen van precieze goniometrische gegevens, ontbrak de mogelijkheid om de overlappende informatie met elkaar te verbinden. Dit bemoeilijkt de automatisering van het verzamelen van gegevens. Een beeld samenvoegen, alsook een algoritme en andere noodzakelijke instrumenten, zijn ontworpen, uitgevoerd en/of verbeterd om dit probleem te vermijden. Geautomatiseerde gegevensverzameling is nu mogelijk.

RESUMEN

Un robot capaz de capturar imágenes de ojos de mariposa mientras retiene datos goniométricos precisos carecía de la capacidad de vincular la información superpuesta en éstas. Esto dificultó la automatización de la recopilación de datos. Un algoritmo de fusión de imágenes, así como otras herramientas necesarias se han diseñado, implementado y/o mejorado para superar este obstáculo. La recopilación automatizada de datos es ahora posible.

ZUSAMMENFASSUNG

Ein Roboter, der Bilder von Schmetterlingsaugen erfasst, während präzise goniometrisches Daten Befestigung fehlte die Fähigkeit, die sich überlappenden Informationen in ihnen miteinander zu verknüpfen. Dies behindert die Automatisierung der Datenerfassung. Ein Bild Stitching-Algorithmus sowie andere notwendige Werkzeuge wurden entwickelt, implementiert und/oder verbessert um dieses Hindernis zu überwinden. Automatisierte Datenerfassung ist nun möglich.

ACKNOWLEDGMENTS

With sincerest appreciation I thank Dr. ir. M. Muñoz-Arias for being simply the most helpful person, absolutely beyond any expectation and also my imagination.

I am grateful for the oportunity to work with Prof. dr. D. G. Stavenga, his insight, the Dutch way of saying things and his pragmatic aproach to doing them.
I thank H. L. Leertouwer for his willingness to provide a helping hand and his ever pleasant attitude.

The members of my assessment committee Dr. J. L. Crespo-Mariño and Msc. M. E. Vílchez-Monge, I thank for the knowledge they have shared with me, which complemented this project.

This project was supported financially by the International Student Movility Program of the Costa Rica Institute of Technology, due to the support of the School of Electronic Engineering. I will warmly remember this.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

3D	Three Dimensional
CPP	Cornea Pseudopupil
DOF	Degree of Freedom
DPP	Deep Pseudopupil
EA	Elevation-Azimuth
FPS	Frames per second
GRACE	Goniometric Research Apparatus for Compound Eyes
HSV	Hue, Saturation, and Value color model
LED	Light-emitting diode
OLED	Organic light-emitting diode
RGB	Red, Green and Blue, additive color model
ROI	Region of Interest
XYZ	Last three letters in the Latin script, used in reference to the Cartesian coordinate system

Part I

INTRODUCTION

1

INTRODUCTION

Throughout the history of evolution, nature has developed highly efficient detection systems. These are of interest to state-of-the-art research and the development of applied science and technology.

In a biomimetic research context, with applications in artificial vision and navigation systems in mind. Arthropod compound vision is one of such inventions of nature which, despite being by far the most common animal vision mechanism [6], is understood in a rather limited way. Henceforth, while compound eye vision has been the subject of research now for centuries, there is vast knowledge to be gained from the characterization of the visual properties of arthropods.

In the past, characterizations of arthropod compound eyes have been done manually, requiring however an extended time frame to be completed [10]. A need exists to optimize the data collection process in order to scale it to more specimens and species quickly, especially if it is a goal to replicate these systems in human technology.

Successful replication has already been precedented, an example being the widely implemented reduction of lens reflection in optical systems by the "moth eye" principle [7]. Another example is how the structural components of the firefly bioluminescent lantern, used naturally for optical signaling, motivated highly efficient OLED applications [5].

Compound vision refers to animal vision systems that use compound eyes. These are made of hundreds or thousands of ommatidia, where each ommatidium can be considered as an independent optical system [6].

In Fig. 1.1 the images of the eyes of two butterflies can be appreciated. The eyes consist of ommatidia, which have a cylindrical shape and contain on the outside a facet lens, which together with a transparent crystalline cone channels light into the rhabdom. The rhabdom is the light receiving optical waveguide of the ommatidium, where light is absorbed to form an electrical signal[9].

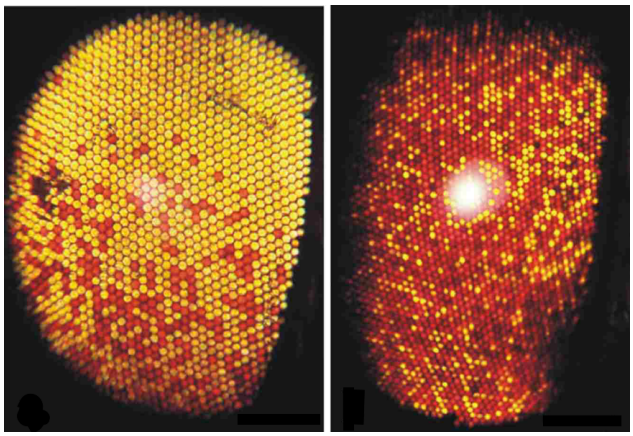


Figure 1.1: Images of the compound eyes of two butterflies.

The ommatidia together provide the butterfly a mosaic view of their surroundings formed by their adjacent fields of view [6].

Since the eyes of the butterflies are almost spherical, the optical axes of the ommatidia intersect approximately at the eye's centre of curvature [9]. It is of special interest to determine the position of the visual axes of the ommatidia.

Among arthropods, butterflies have a distinct apposition compound vision type [11]. An eye shine exists in them known as the *pseudopupil* phenomenon. Light which traveled down the ommatidium and escaped absorption is reflected back out of the eye, due to a reflecting tapetum [13]. The pseudopupil is a bright spot, which moves across the eye as the observer rotates around the animal; it provides a powerful non-invasive technique for characterization [6]. Since this phenomenon indicates the direction of the visual axes of its facets, one can track their orientation with a goniometer by aligning them with the optical axis of a microscope.

The pseudopupil is tracked at two different levels of the eye. Firstly at the level in which the eye shine first appeas, known as Deep Pseudopupil (DPP) and then at the level of the cornea, refered to as Cornea Pseudopupil (CPP) where for each position of interest the camera is focused at and then an image is taken.

Recent technological advancements in digital photography, created a paradigm shift for quantitative image analysis on a routine basis [10].

Some years ago [10] and more recently [2], state-of-the-art mechatronic systems have been used for rapid evaluation of compound eye specializations.
The Department of Computational Physics of the University of Groningen has developed a 6 DOF 3D robotic scanner (called GRACE) for the automation of the charaterization of the visual space of butterfly eyes [14].

Although the system is not yet fully automated, important progress has been made towards its development with an automatic image focusing function [1], a redundant angular position measuring system [4], a simultaneous motion control [8] and other important functions.

This document reports on an image stitching algorithm created, to meet the need of correlating data in overlapping images of a butterfly eye taken by GRACE, and mapping its optical devices while giving spatial orientation to their respective visual axes, thus extracting from these images the information desired by the research funding the system.

The following chapter elaborates on the mechatronic system in question, the materials and general disposition of testing conditions, the core functions of the scanning process and details of the image processing involved in data acquisition. The final sections present the obtained results, and later on provide a discussion about their implications and the therefore drawn conclusions, as well as thoughts towards further improvement.

Part II

MATERIALS AND METHODS

2

MATERIALS AND METHODS

2.1 BUTTERFLIES

Figure 2.1: *Pieris rapae* butterfly.

This project seeks to optimize the characterization of the visual space of butterfly eyes. The images used for testing the project's algorithm were obtained from butterflies of the *Pieris rapae* (see Fig. 2.1) species, however, the methods hereby employed should be easily adaptable to the use of different insects.

A key part of the process of scanning a butterfly eye is aligning the visual axis of the DPP to that of the camera before capturing an image. In this way the facets of the eye that appear in the image can be labeled according to their spatial position and the orientation of their axis. In Fig. 2.2 an ommatidium, the 'unit' of a compound eye [6], is outlined in red (om). It contains structures like the facet lens (f), corneal cone lens (cc), accessory screening pigments (sp), photosensitive rhabdom (rh, green), photoreceptor cells (rc), and axons (ax). Interommatidial angles ($\Delta\phi$) are defined by the separations between adjacent ommatidial axes, and light entering each ommatidium is mainly restricted to the acceptance angle (aa).

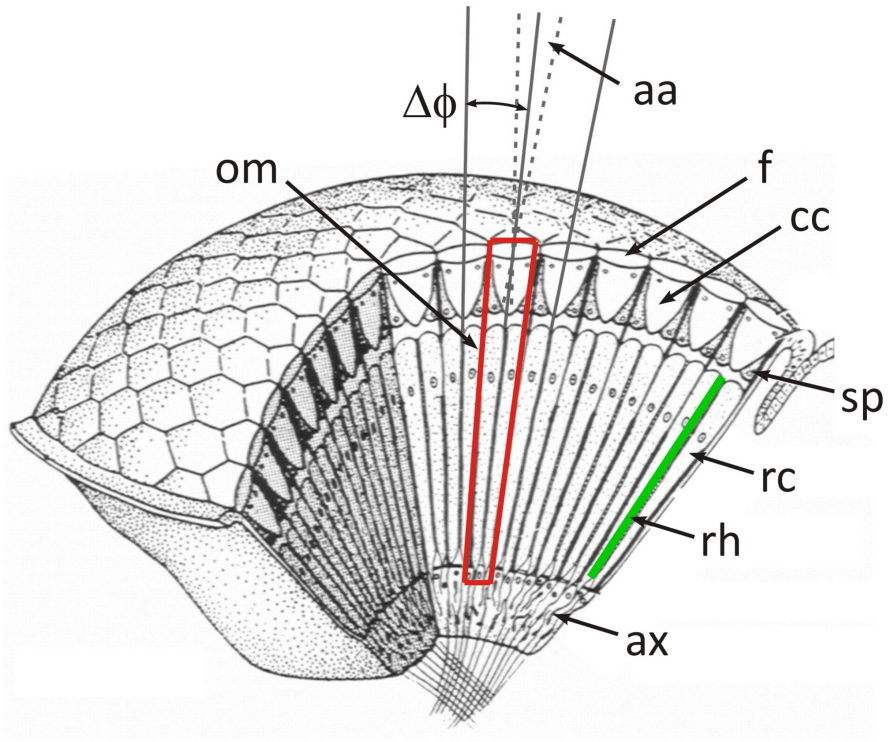


Figure 2.2: Structure of an apposition compound eye [3] [2].

The butterflies were either captured in the wild or obtained through pupae cultured by the Wageningen agricultural University.

2.2 GRACE

Goniometric Research Apparatus for Compound Eyes (GRACE), this is the device used to photograph the surface of insect eyes. It is a six DOF robotic scanner based on a former design of Stavenga's [9], see Fig. 2.8. It contains six stepper motors (one per DOF), a digital camera with a 5x magnification, a LED light source, a goniometer (two rotational joints), and four linear joints, of which three move the objective in a Cartesian coordinate system and the last moves the camera along its visual axis. The system is displayed in Fig. 2.3.

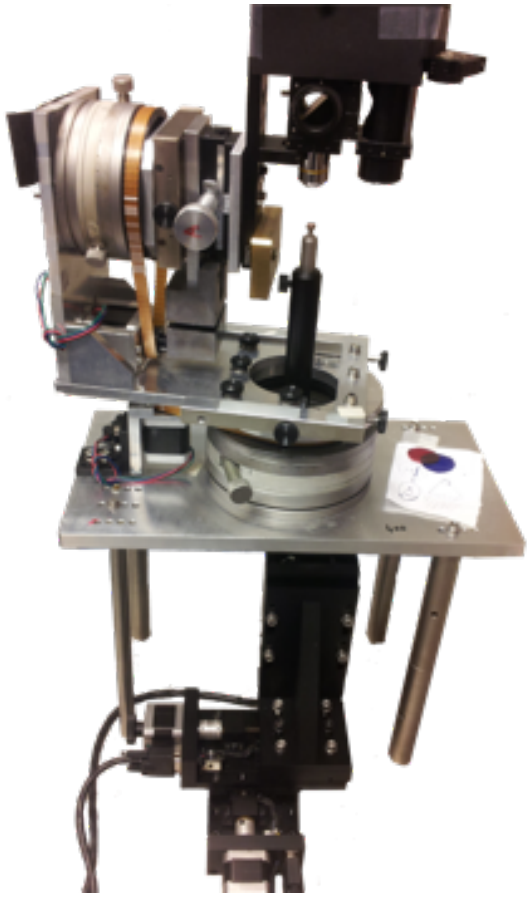


Figure 2.3: GRACE.

2.3 CONTROL

A diagram of the system's control is displayed in Fig. 2.4. The system is controlled through a Windows computer using Matlab. It communicates via a serial port with an Arduino Leonardo microcontroller. The Arduino controls the LED light source and the 'EasyDriver' stepper motor drivers which ultimately handle the operation of all six stepper motors. The camera is controlled directly from Matlab via a USB 3.0 connection.

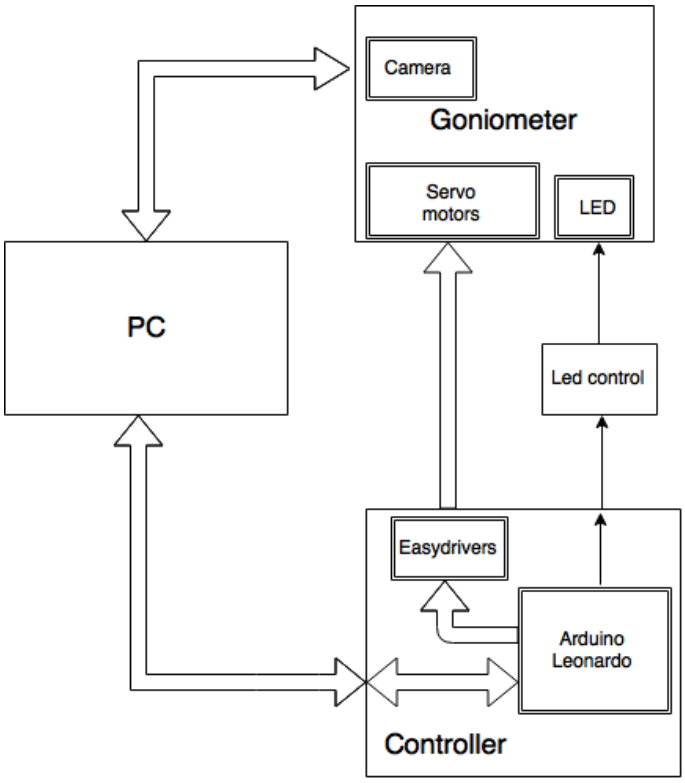


Figure 2.4: System control diagram.

The system's control has benefited from the help of collaborators, see Table 2.1. Their collaboration and resulting algorithms are elaborated on explained further in the document.

CONTROL ALGORITM	COLLABORATOR(S)
Autowaisting	M. Muñoz-Arias
Autocentering	M. Muñoz-Arias, A. Vargas-Delgado
Autofocusing	G. J. Doornbos, M. Muñoz-Arias
Simultaneous XYZ-motion	T. R. Spanier
Angular position mesurement	M. van der Horst

Table 2.1: Collaboration.

2.3.1 *Mechanics*

The motion of the system is quite complex given its 6 DOF. It is made of three different stages listed in Table 2.2.

Every axis is controlled by its own Wantai stepper motor, see Fig. 2.5. These motors need a 2.5 A current supply and have a 1.8° angular step, however they are configured (except for the camera motor) to have that step size reduced eight times requiring 1600 steps for a revolution. This allows extremely precise movements. Each motor is referred to according to the axis it is charged with moving: 'X', 'Y', 'Z', 'E', 'A', or 'camera'.

Figure 2.5: Wantai stepper motor.

STAGE	INFORMATION
XYZ Set	Three linear joints which move the objective.
EA Set	Two rotational joints which move the system's optics.
Camera axis	A linear joint which moves the camera along its visual axis.

Table 2.2: System Sets.

Table 2.2 elucidates what the diferences are between the three sets defined for the system, and the way motion is achived in each of them. The EA set is the one providing goniometric position tracking and their axes are setup as shown in Fig. 2.6. With the Elevation axis position being determined by θ and the Azimuth axis defined by ϕ.

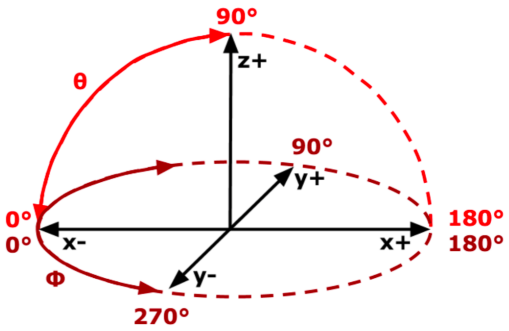


Figure 2.6: XYZ and EA set disposition.

Fig. 2.6 shows how the XYZ and EA sets are setup. The objective is to be placed at the origin of the cartesian coordinate system (XYZ set), and centered in the rotation axis of ϕ (Azimuth).

Due to physical constraints, the Elevation axis is only able to move along an approximate 180° range, Azimuth however, has a 360° range of motion.

2.3.2 *Optics*

The camera (see Fig. 2.7) mounted on GRACE is a Point Grey Flea3 USB 3.0 camera. It takes 3.2 megapixel color images with a 60 FPS frame rate and a resolution of up to 2080 by 1552 pixels.

Figure 2.7: Flea3 digital camera.

Similar to the design implemented in 2002 by Stavenga, the camera is placed in a diferent axis than the light source. Mirrors are then used to direct them both to the objective. In Fig. 2.8 the system's optical setup is shown, accurate to GRACE's case however with the camera replacing the photomicroscope. The objective lens L1 has a large aperture. A light source is focused at lens L2 in its back focal plane, where the field diaphragm D1 is positioned. D1 is in the focal plane of lens L3, which is confocal with L1 because of a half-mirror placed at 45° with respect to the optical axis of L1 and L3. A more-or-less parallel beam, depending on the size of D1, enters L1 and is focused on the DPP in the center of the butterfly's eye. The telescope lens pair L1 and L4 images the DPP in the back focal plane of L4, where diaphragm D2 is positioned. The image of the CPP, projected by lens L5, confocal with L4, is photographed by the flea3 camera. The dotted lines are the back-focal planes of L1 and L5 [9].

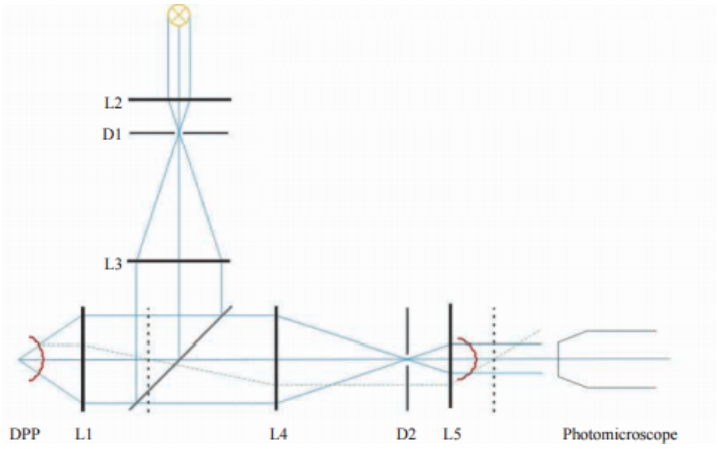
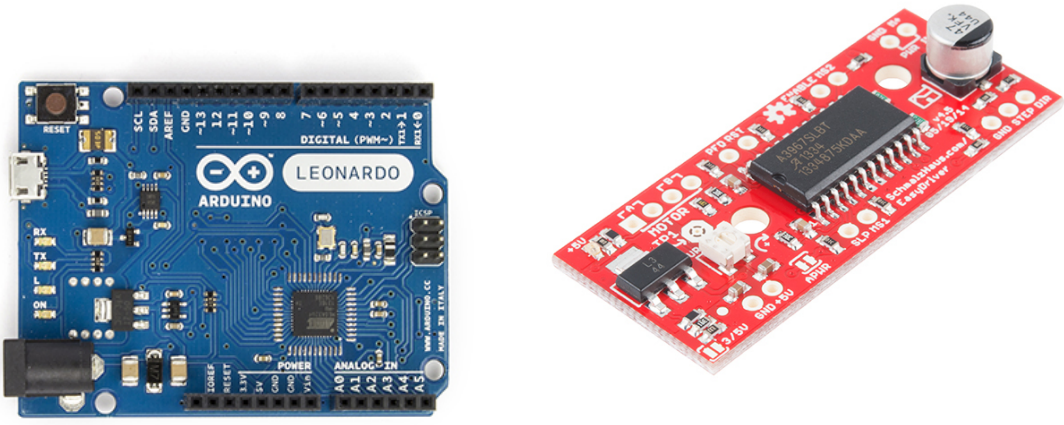


Figure 2.8: Optical device used to photograph butterfly eye shine.

2.3.3 *Electronics*

An Arduino Leonardo, Fig. 2.9, is the controller used to receive commands from the computer through serial port and then trigger the stepper motors appropriately, the Arduino code used is listed in A.1. The motors are driven by the EasyDriver stepper motor driver, conviniently assembled onto a breakout board by Sparkfun Electronics (see Fig. 2.9).



(a) Arduino Leonardo microcontroller.

(b) Stepper motor driver.

Figure 2.9: System microcontroller and motor driver.

2.4 SCANNING

While scanning an eye (Fig. 2.10), the visual axis of the camera must be to aligned to that of the DPP. This is done by finding a correct intensity of the DPP and centering it on the image provided by the camera and then mantaning that alingnment while moving to the CPP to capture the facets (Fig. 2.11). Once all intended captures have been made the stitching algorithm can be applied to the image group to extract the data from them.

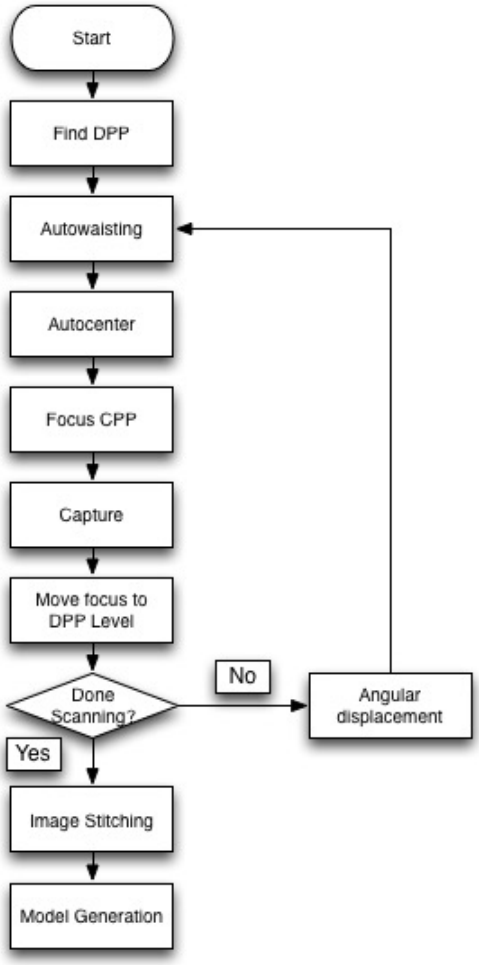
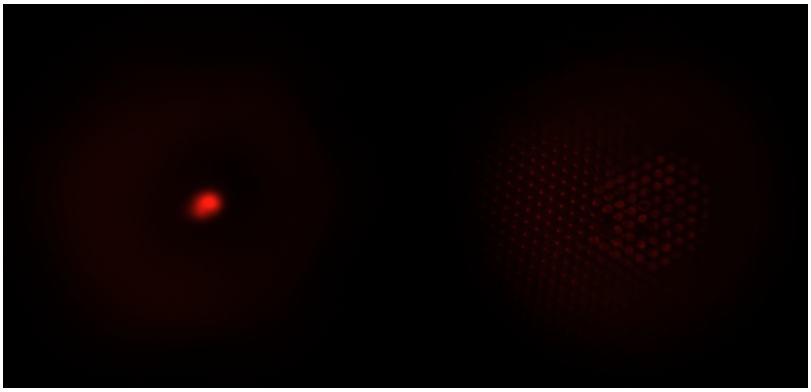


Figure 2.10: Auto-scanning process diagram.

Fig. 2.11 (a) shows an image of the DPP. This is the level at which the autowaisting and autocentering are applied. In 2.11 (b) we have the same eye but now focused at the cornea level (CPP), here the facets are appreciated much more clearly.

The butterfly *Pieris rapae* contains mostly red pigment along its rhabdom. Because of this, its eye shine is red to human eyes. The butterfly eye shine can be strained by the constant incidence of light, and much

like a human pupil it will shrink to avoid it. To avoid this, a red filter has been placed on top of a lens, increasing the butterfly's comfort and therefore stabilizing the quality of the images taken of the phenomena.



(a) DPP. (b) CPP.

Figure 2.11: Pseudo pupil phenomena.

2.4.1 *Setup*

Due to the requirement of finding and aligning the DPP to the camera, it is necesary to use live butterflies to capture images. Therefor care must be taken to mount them securely and rigidly onto the system, which is done by placing the butterfly into an approximately 3 cm long piece of plastic straw and then applying molten wax to its head so that it may not move, potentially altering scanned information.

Before starting the scanning the center of the eye is aligned to the visual axis of the camera with GRACE at the 90° elevation and 0° azimuth position.

Figure 2.12: A butterfly mounted on GRACE.

2.4.2 *Autowaisting*

This is an iterative function which simply computes the highest intensity level of the DPP in a series of images, returning its respective position as optimal. Basically, it makes the DPP bigger and easier to detect.

2.4.3 *DPP centering*

This also referred to as 'Autocentering' and the Matlab function is called simply 'Autocenter'.

This function is illustrated in Fig. 2.13. Basically the DPP is a point which must be moved from its current position to the center of the image plane. This is done with the XYZ set, so the algorithm involves a parametric conversion from the image plane coordinates (vertical ν and horizontal μ) to the XYZ position of the objective.

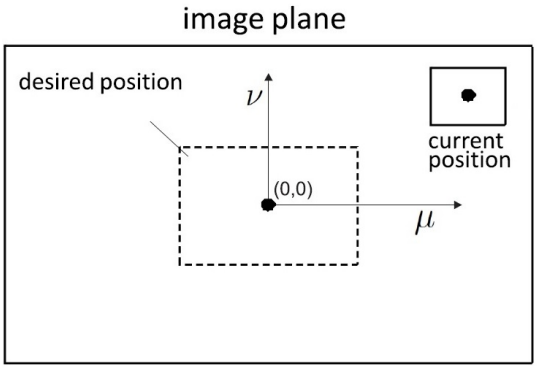
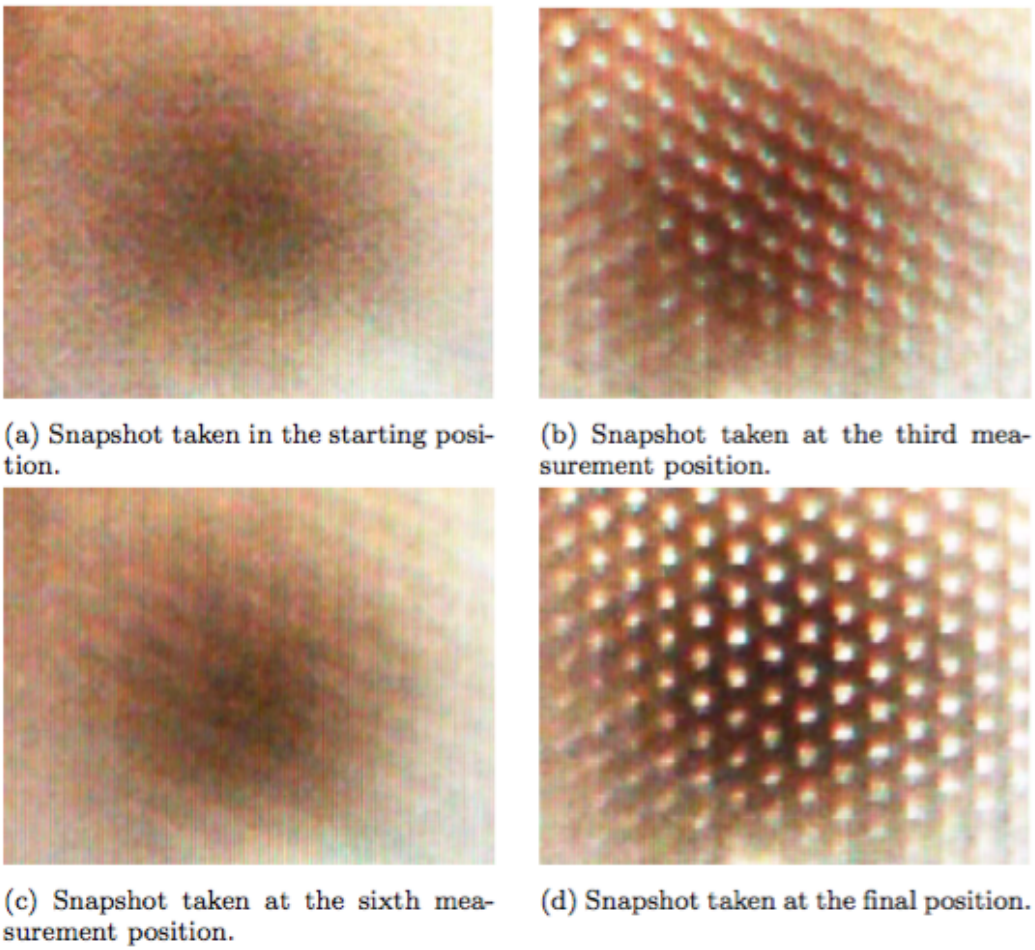


Figure 2.13: DPP position correction.

2.4.4 *CPP focusing*

This is refered to as 'Autofocusing' and the Matlab function is called simply 'Autofocus'.

Fig. 2.14 illustrates the iterative approach in which this is carried out. A focus value is obtained at different levels of the eye and the level with the highest value is returned as the correct focus level for image capture.



(a) Snapshot taken in the starting position.

(b) Snapshot taken at the third measurement position.

(c) Snapshot taken at the sixth measurement position.

(d) Snapshot taken at the final position.

Figure 2.14: CPP focusing method.

2.5 IMAGE PROCESSING

This section describes how images are manipulated in order to gain the information desired of them. Several changes are made to an image in order to make objects more accurately identifiable and characterizable. After this, a process known in relevant literature as segmentation is applied to label these objects and attribute properties to them. Later on, the data generated may be used to generate models of interest for the study of compound eyes.

2.5.1 *Filtering*

The filtering process, all within the function listed in A.5, follows a specific sequence, see 2.15, extracting from a base image an optimal version for property labeling. The resulting stages of this information filtering can be observed in Fig. 2.16, following the sequence, in which the image data is manipulated to obtain the desired end image, where elements may be easily detected and its properties labeled.

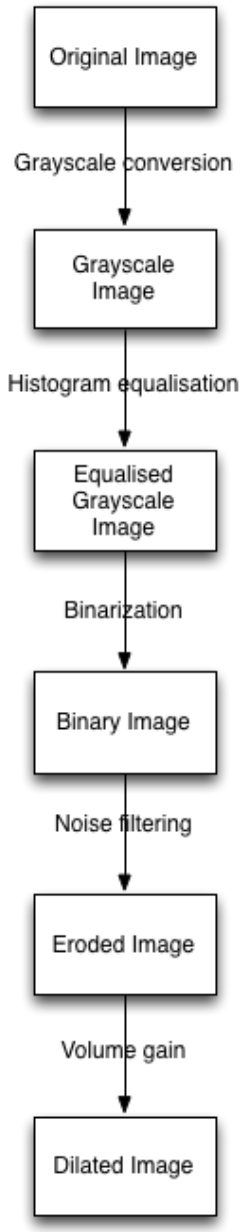
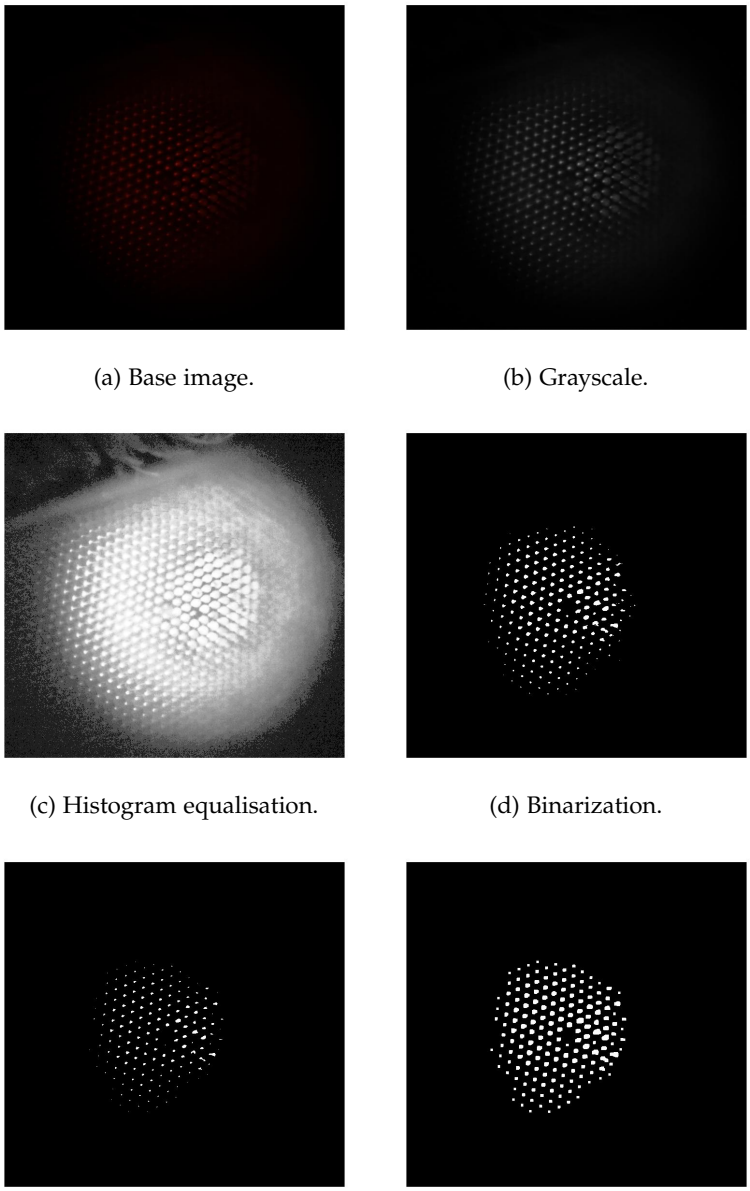


Figure 2.15: Filtering flow diagram.



(a) Base image. (b) Grayscale.

(c) Histogram equalisation. (d) Binarization.

(e) Erosion. (f) Dilation.

Figure 2.16: Image filtering.

The original image 2.16 (a), is converted from a RGB to an HSV format anticipating the fact that some butterfly species do not always have red pigments in their rhabdom making their DPP not red, but a different color. By doing this transformation, a grayscale version of the image can be obtained, as seen in 2.16 (b), from the intensity channel of the HSV image.

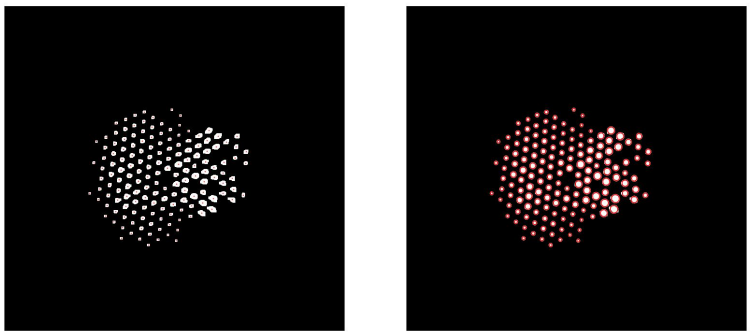
That last image has an histogram equalization applied to it later on, yielding image 2.16 (c). This process re-scaled the intensity channel image to fit the whole dinamic range of the image format [12], which given the use of Matlab's 'uint16' format is $(1, 2^{16}) = (1, 65536)$. This is important because later on a binarization is applied to the image

where pixels are turned either to white or black, depending on their intensity level. This level threshold is made general by the applied equalization, because without it a threshold value would need to be chosen for each specific image.

After a binary image is obtained 2.16 (d) certain imperfections or erroneous objects may remain. In order to exclude them an erosion filter is applied yielding 2.16 (e). Finally, for the surviving elements to gain volume, another manipulation is applied, this time a dilation filter resulting in 2.16 (f). This final image is quite different from the original, however. It is now ideal for property segmentation.

2.5.2 *Labeling*

Labeling consists of finding objects in an image and determining its properties. The two properties that are detected are the centroid (which is naturally the most important) and the approximate radius of the object. The radius is used mostly to enhance the display of information so that it is simpler to observe. However, its value has no importance due to the filtering done; only its correlation to other facets in a given image is a quantitative property, which is not calculated or displayed here.



(a) Centroids. (b) Radii.

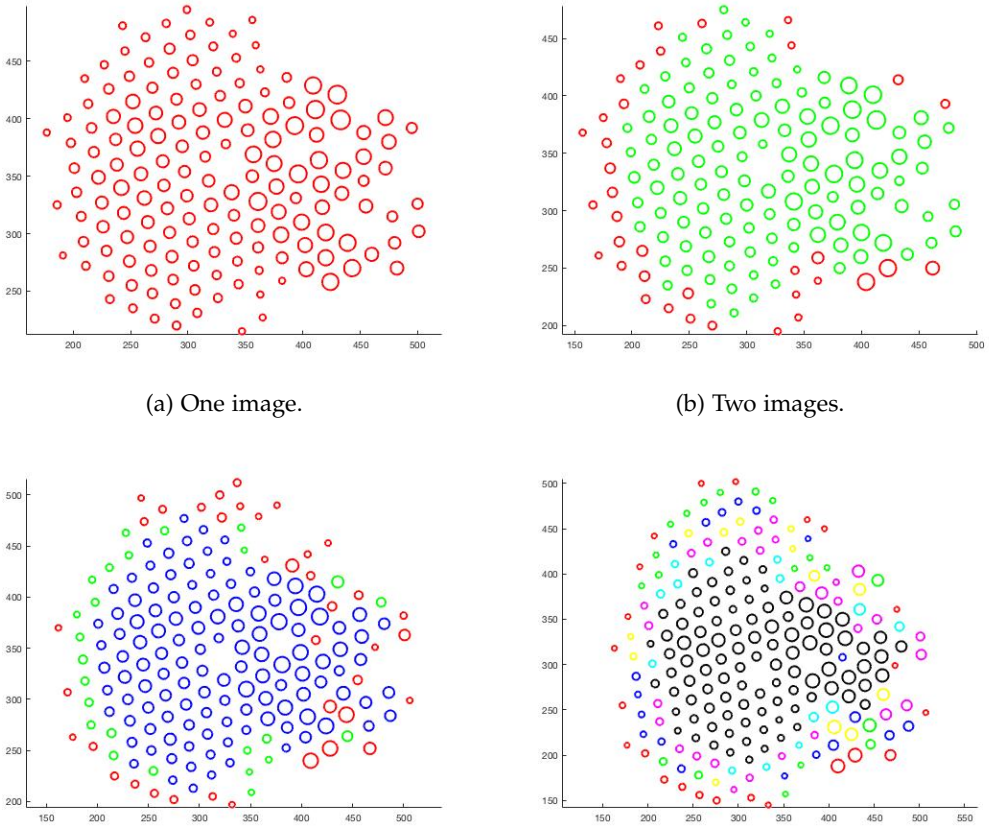
Figure 2.17: Labeling.

Fig. 2.17 shows an image of a cornea which resulted from the filtering process. A centroid property has been assigned to each object, illustrated with red dots in (a). The approximate radius property is used to display a circle around each object in (b), since the end goal of this image processing is to merge objects from different images. The uniformity of these circles is indicative to the researcher in order to visually recognize problems in facet detection or subsequent merging algorithms.

2.5.3 *Modeling*

An explication of how the obtained data is going to be displayed may be useful before discussing the applied methods any further.

The end result of the image stitching is two arrays. The first contains the total number of elements found within all images processed with four properties. These are: horizontal coordinate, vertical coordinate, radius, and number of times merged. The second is an array containing the relative horizontal and vertical position of every visual axis. Naturally, the amount of elements in this array is equal to the amount of images analyzed.



(a) One image.

(b) Two images.

(c) Three images.

(d) Seven images.

Figure 2.18: Data display.

In Fig. 2.18 (a), the same circles have been plotted in their respective centroids as in Fig. 2.17 (b). In Matlab however, since this is not done on top of an image figure, an inversion appears to happen. That is due to the scale of the pixel position in a displayed image, starting in the top-left corner with pixel $(1, 1)$ and ending in the bottom-right corner with pixel (pixel width, pixel height). Images taken by GRACE

COLOR	MERGE COUNT
Red	0
Green	1
Blue	2
Yellow	3
Magenta	4
Cyan	5
Black	6 or more

Table 2.3: Color coding in element display.

have a $(672, 704)$ dimension because it was determined as the ROI for capturing buterfly eyes, despite the $(2080, 1552)$ resolution available from the systems camera. This apparent inversion does not mean any data has been lost, but it is nonetheless important to clarify.

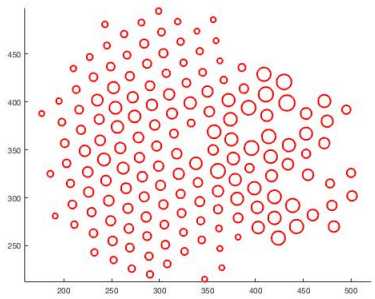
A color code was implemented to display the amount of times an element has been merged, see Table 2.3. It is demonstrated in Fig. 2.18 (b), (c), and (d) with merged arrays of 2, 3 and 7 images respectively.

2.5.4 *Merging*

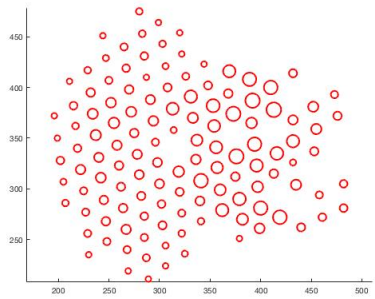
What the Merge function, listed in A.6, acomplishes is to combine the facet position and radius data in two images after finding a correction offset (see offset function A.7) between them.

The Offset function finds the offset between elements which appear in two images, by aligning them and then moving one within a fixed area, then the 'Matches' function A.8 returns a count of the matching elements for each position.

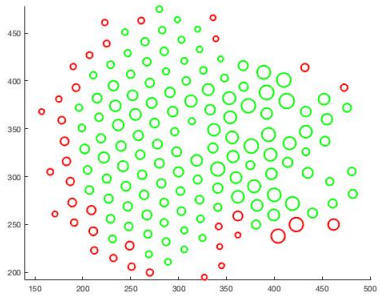
Following the sequence in Fig. 2.19, when having two facet centroid arrays, Fig. 2.19 (a) and (b), which overlap with each other, the function will create a new 'Merged' array Fig. 2.19 (c), which contains all the matched facets as well as the unmatched ones left in both starting arrays. All matched facets will have their properties averaged. The data is complemented by adding a new property, which counts the times each facet has been matched (color code, mostly created to add feedback on how the Merging/Stitching is working) and tracking the center pixel of each image. These are the visual axes and are illustrated by red dots in Fig. 2.19 (d).



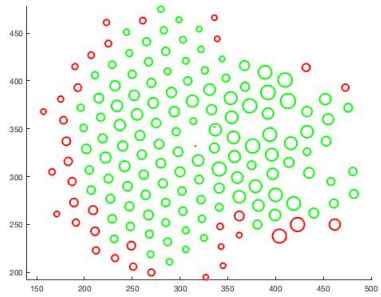
(a) Centroid array A.



(b) Centroid array B.



(c) Merged array.



(d) Merged array with visual axes (red dots).

Figure 2.19: Merging.

2.5.5 *Stitching*

The function of the image stitching algorithm developed (A.10) is to create a mapping of an eye's ommatidia from multiple images, which have overlap between each consecutive image. Inherent to these images are the so-called visual axes, which due to the scanning process followed should be located at the center of each image.

These visual axes allow for the angular position of each ommatidium to be interpolatable, since their angular separation is determined beforehand by the person doing the scanning, who can ultimately set it to any desired value allowed by the system.

Having elaborated on ealier image processing stages, the end goal Stitching function becomes quite simple to define as a generalization of the 'Merge' function (A.6), enabling the system to handle a group of any number of consecutive images.

The 'Merge' function, in a way, combines data from just two images. 'Stitch' is a more robust tool which can combine as many images as the user feeds into it. As long as there exists sufficient overlap between consecutive images. Beyond this, the function keeps track of the visual axes specific to each image, by accumulating the offsets applied by each time a combination is made.These visual axes are

represented in data by red dots as seen in Fig. 2.19 (d).

This allows for an automated data collection process to be achievable, following the sequence in Fig. 2.20.

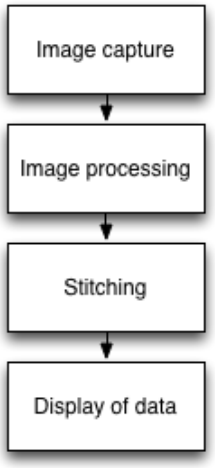


Figure 2.20: Automated data collection process sequence diagram.

Image capture involves the previously explained algorithms that lead to a based image, striving for the best posible quality. The image procesing stage leads to the acquisition of object properties for every individual image. The stitching unites all those data arrays into one, and then it is displayed for a visual analysis of the data.

Part III

RESULTS

3

RESULTS

Photographs were taken of butterflies of the *Pieris rapae* species. Several sample scans of different individuals were made to make sure the results the algoritm provided were being effectively reproduced. Since overlap is extremely desirable, the angular step used when scanning sample eyes was the minimum allowed physically by the system, of 1°.

The stitching of facet position data has been completed for several samples, each consisting of images taken from the 90° in elevation and 0° in azimuth position with angular steps between each image of 1° in the Elevation axis while maintaining Azimuth in 0°.

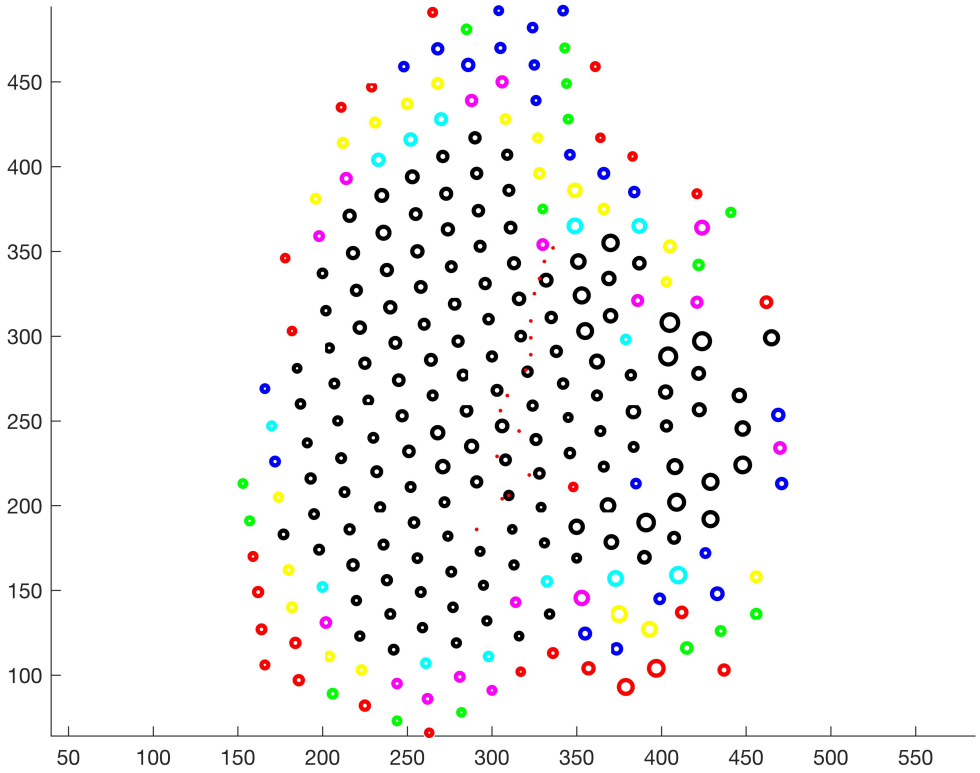


Figure 3.1: Stitching of facets in 16 images covering a 15° range.

Fig. 3.1 shows the result of running the Stitch Matlab function (A.10) working on 16 images, covering a range of 15° from $Elevation = 90°$ to $Elevation = 105°$.

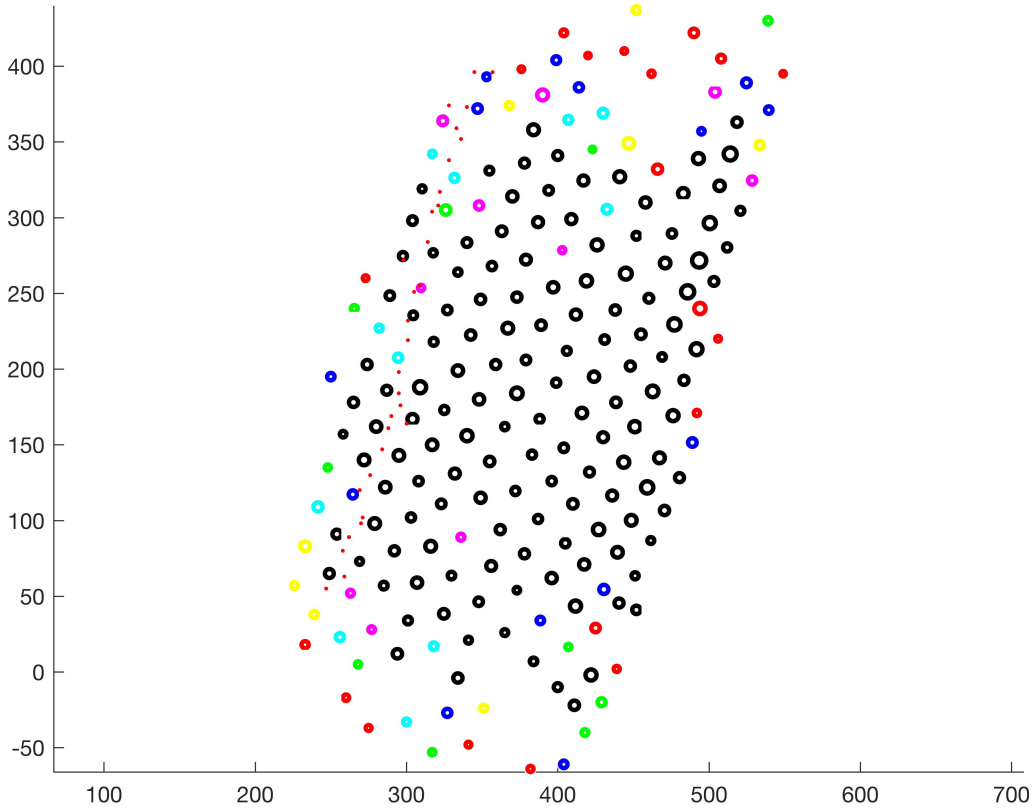


Figure 3.2: Stitching of facets in 31 images covering a 30° range.

Fig. 3.2 shows the result of running the same function working on 31 images, covering a range of 30° from $Elevation = 90°$ to $Elevation = 120°$.

The image groups that resulted in Figures 3.1 and 3.2 were taken from two different specimens and captured with close to a month of time in between (May 18th, and June 16th). Both figures follow the modeling structure explained previously and have the visual axes of ther respective images represented by red dots. The distance between these axes was averaged to $14,3213pixel$ for 3.1 and $15,4936pixels$ for 3.2. Using a factor of $1pixel = 1.0526\mu m$, obtained by calibrating the camera with a micrometric scale at the cornea level, the average interommatidial distances are $15,1\mu m$ and $16,3\mu m$.

As mentioned in the previous chapter, the objective is centered in the Azimuth axis. Therefore, if ϕ is given an angular displacement, there will be no new facets appearing in the image, they will simply be the same facets now rotated. Because of this, the scanning and following stitching are done along the Elevation axis motion range, and then repeated with a displacement in Azimuth to complete the scanning of the eye.

These greater arrays, view Figures 3.3 and 3.4, can then be also stitched together to obtain a final mapping of the ommatidia. To elaborate another two samples were captured (June 28th), with 10 images each. Now covering a range of 9° from $\mathtt{Elevation} = 90°$ to $\mathtt{Elevation} = 99°$, but with a 1° in the Azimuth axis between samples.

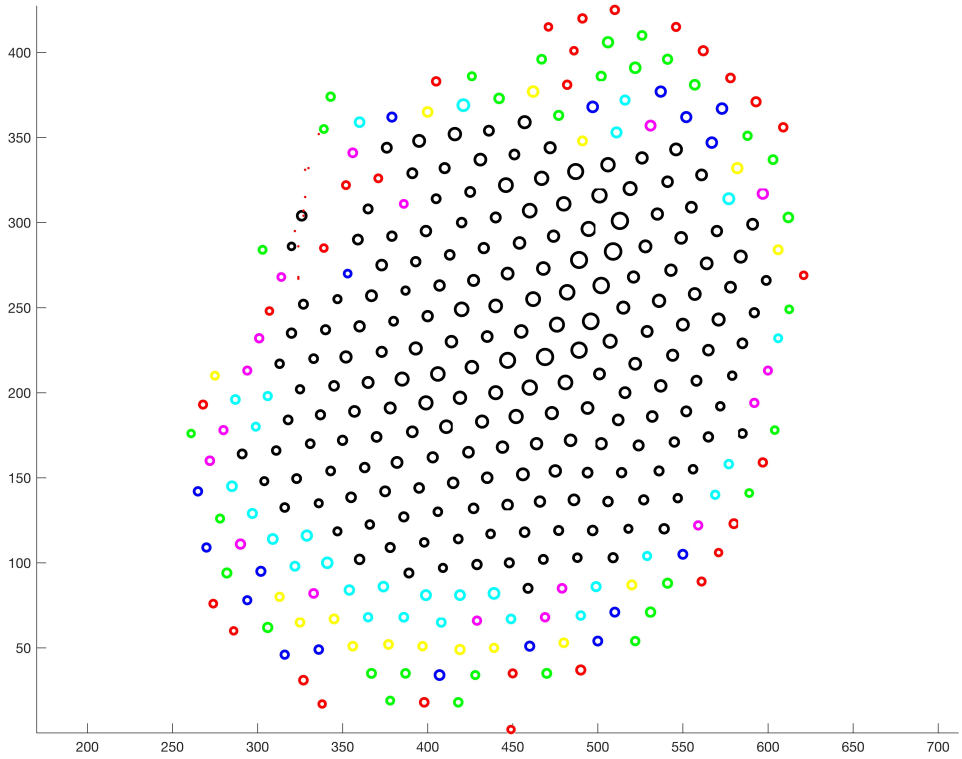


Figure 3.3: Stitching of 9 images covering a 9° range on $\mathtt{Azimuth} = 0$.

Finally these two resulting arrays of data were merged, yielding Fig. 3.5.

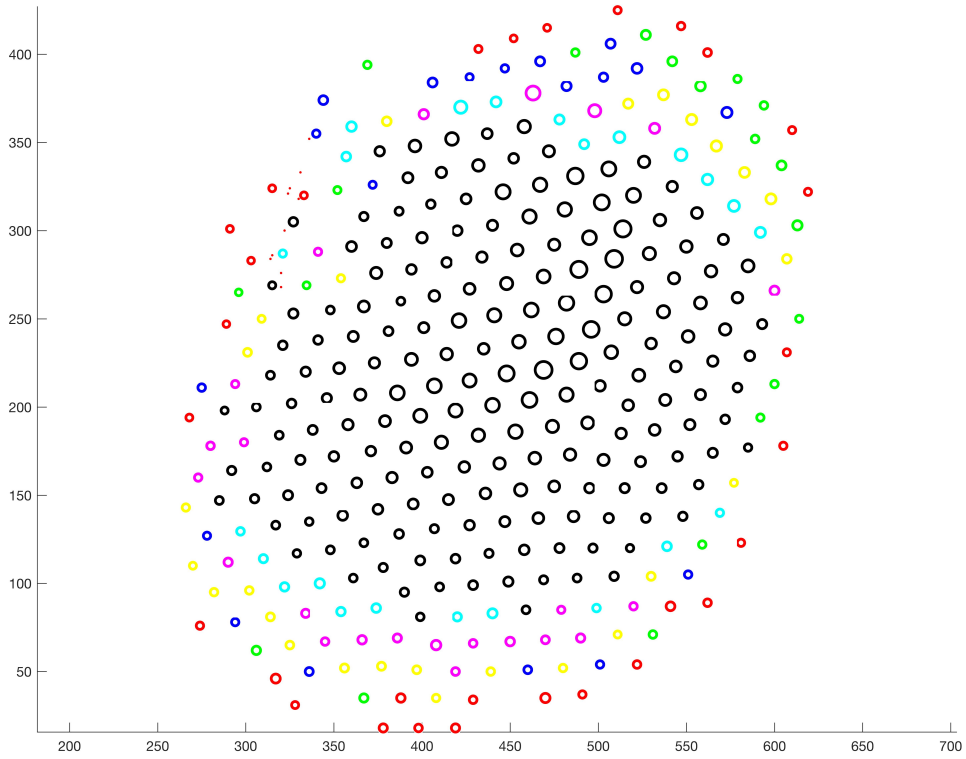


Figure 3.4: Stitching of 9 images covering a 9° range on $Azimuth = 1$.

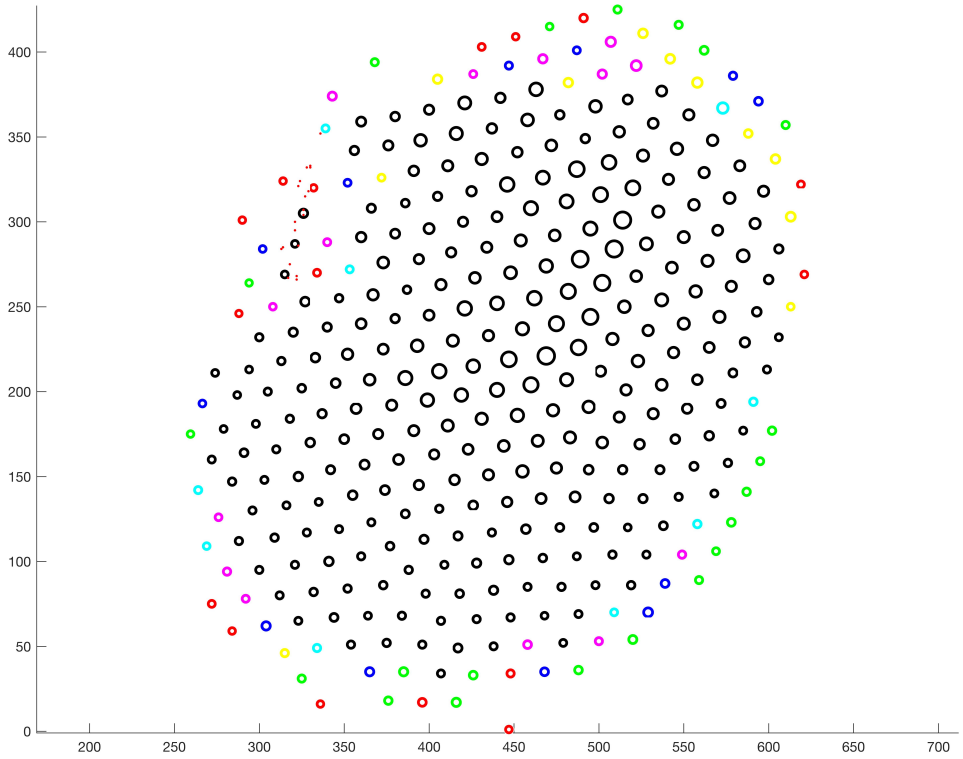


Figure 3.5: Merging of two seperate Elevation scans with a 1° displacement.

Part IV

DISCUSSION

4

DISCUSSION

The algorithm proposed has been proven to correlate adequately overlapping objects found in series of consecutive images, while being a highly scalable and mantaining compilation time below five minutes. This can be appreciated by how imperfections of the eye like the one in Fig. 3.1 at approximately $\mu = 300$ and $\nu = 200$, which is visibly apparent in the original image (Fig. 2.16 (a)), are maintained despited being in a region where most surrounding facets have been matched 6 or more times.

The color code helps to visualize how the merging of centroid arrays is being carried through, as well as the general direction of the scanning, matching that of the trajectory traced by the points representing the visual axes of each image.

As is to be expected the colors representing lower amounts of combinations are found in the outer layers of the described ommatidium, with bordering facets showing a trend towards red and green, meaning 0 or 1 respective combinations.

It is highly useful to increase the overlap between consecutive images as much as possible, because this will decrease the possibility that a false pattern generates more matching elements than the real one. For the samples displayed in this thesis this point occured at a 7° angular step. However, in other image groups it happened at an angular step of 5° due to decreased image quality.

Increasing overlap between images also makes the computing of the Offset function **??**lst:Offset more efficient.

The most crucial factor to achieve an accurate scanning is to collect images with the absolute best quality. This is a tricky suggestion since this quality should be understood as relative to the image processing end goal, or in this case the optimal identification of the ommatidia within an image.

As a follow up proposition for further improvement, it could be desirable for increasing the effectiveness of the stitching algorithm to change the way in which images are binarized from a fixed binarization of an equalized image to an unique-to-each-image threshold

finding option, in which an optimal level threshold is determined for an image before binarization and the subsequent filtering. Again a cumbersome notion, since (learned from experience while testing) it is not enough to simply tweak this threshold to maximize the amount elements detected in an image. It is also necessary to add the criteria: "while maintaining no errors in facet detection". This error concept is mostly to be understood as ‚but not limited to, that there may not be two neighboring facets wrongly labeled as one. This, however found to be non-trivial, may be possibly achieved by iteratively using a preliminary centroid array to draw a projected hexagonal grid on top of the eye's lattice, determining in this way the expected position of dead facets and detecting the suggested concept of error in facet detection.

Part V

APPENDIX

A

APPENDIX

A.1 ARDUINO CODE

The code used by the Arduino Leonardo is listed. It takes string commands coming from Matlab, interprets them and proceeds to send the respective actuation commands to the motor drivers.

A.2 MATLAB FUNCTIONS

The Matlab functions that carry out the algorithms explained in this document.

Listing A.1: Arduino code for system control from Matlab via serial port.

```
//Program that runs the Goniometer based on the comands of Matlab
//The Goniometer consists on a 3D Robotic Scanner with 6 DOF such
     as:
//XYZ: Three servomotors for the lower stage control by
//     a conversion between Spherical and Cartesian coordinates
//EA:  Two for the upper stage, i.e., Elevation and Azimuth.
//C:   The servomotor that controls the movement of the camera.
#include <AccelStepper.h>
//Counter integers for steps
int cx = 0;
int cy = 0;
int cz = 0;
int ce = 0;
int ca = 0;
int cc = 0;
//Integers containing the number of steps
int ix = 0;
int iy = 0;
int iz = 0;
int ie = 0;
int ia = 0;
int ic = 0;
//Integers containing the step pins
int sx = 3;
int sy = 5;
int sz = 7;
int se = 9;
int sa = 11;
int sc = A2;
//Integers containing the direction pins
int dx = 2;
int dy = 4;
int dz = 6;
int de = 8;
int da = 10;
int dc = A5;
//LED for the control of the shutter of the microscope
int led = 12;
//Important variable to see if the serial channel was reinitiated
     again after a comand from Matlab.
int serialk = 0;
int serialkEA = 0;
int serialkC = 0;
String ccheck;
int icheck;
String x;
String y;
String z;
String e;
String a;
String c;
//Only for the Elevation and the Azimuth
int positions1 = 0;
int positions2 = 0;
int positions3 = 0;
int speedEint = 0;
int speedAint = 0;
int speedCint = 0;
String speedEstring;
```

Listing A.2: Autowaisting function code.

```
function autowaisting(vid,leo,theta,phi)
frame1 = getsnapshot(vid);
    %Butterfly XYZ
    MoveXYZ(leo,200,theta,phi);
    Cycles = 20; %MoveXYZ
    delta = 10;   %Step size of autofocuisng
    MaxPoint  = zeros(Cycles,1);
    MaxPoint  = zeros(Cycles,2);
    MaxPoint  = zeros(Cycles,3);
for i=1:Cycles
    MaxPoint(i,1) = i;
    MaxPoint(i,3) = -delta;
    MoveXYZ(leo,MaxPoint(i,3),theta,phi);
    pause(0.1)
    lon(leo);
    pause(0.2);
    frame = getsnapshot(vid);
    pause(0.1);
    lof(leo);
    pause(2);
    H = frame(:,:,1);
    [maxChx,imaxChx] = max(H);
    [maxmaxChx,imaxmaxChx] = max(maxChx);
    H = H';
    [maxChy,imaxChy] = max(H);
    [maxmaxChy,imaxmaxChy] = max(maxChy);
    MP = H(imaxmaxChx,imaxmaxChy);
    MaxPoint(i,2) = MP;
    [maxMP,imaxMP] = max(MaxPoint(1:i,2));
    if MaxPoint(i,2) == maxMP
        frame1 = H';
        i
    end
end
% Detecting the Maximun
[~,imaxChx] = max(MaxPoint(1:Cycles,2));
% Returning to the Maximun
MaxPoint(Cycles+1,3) = (Cycles-imaxChx)*delta;
pause(0.5);
MoveXYZ(leo,MaxPoint(Cycles+1,3),theta,phi);
end
```

Listing A.3: Autocenter function code.

```
function autocenter(vid,leo,theta,phi)
i=0;
Iterations = 4;   %Amount of times that system will compesante
    for the
                  %image error
StepsX = zeros(Iterations,1);
Centered = false; %Boolean that tells if the Deep PseudoPupil was
     successfully
                  %centered.
DRoCZ = 5;       %Desired Radius of Centered Zone for the Deep
    PseudoPupil of
                  %the Butterflies' eyes
RoCZ = zeros(Iterations,1); %Radius of Centered Zone
while (Centered == false) && (Iterations >= 1)
i = i + 1;
pause(1)
lon(leo);
pause(0.1);
frame = getsnapshot(vid);
pause(0.1);
lof(leo);
frame = imrotate(frame,-15); %Compensation of rotation of Camera
    by 15 degrees
mR = mean(mean(frame(:,:,1)));
mG = mean(mean(frame(:,:,2)));
mB = mean(mean(frame(:,:,3)));
ratio = mR/mG;
H = frame(:,:,1)-frame(:,:,2)*ratio;
[R C] = size(H);
Center = [R/2,C/2];
%-------------METHOD OF MAXIMA-----------------
 [maxChx,imaxChx] = max(H);
 [maxmaxChx,imaxmaxChx] = max(maxChx);
 H = H';
 [maxChy,imaxChy] = max(H);
 [maxmaxChy,imaxmaxChy] = max(maxChy);
 MaxPoint = H(imaxmaxChx,imaxmaxChy);
 Pixel = [imaxmaxChy,imaxmaxChx]   %R - Pixel_{y} is done due to
     fact
Error(i,:)   = Pixel - Center;
RoCZ(i)=sqrt(Error(i,1)^2+Error(i,2)^2);
%Coordinate transformations
u = Error(i,1); %Mu of the Image plane
v = Error(i,2); %Nu of the Image
%Rotational matrix after change of optics. Switch of Camera 90
    degress
%Also change of reference of Error
R = [ sind(theta)*sind(phi)  cosd(phi);...
      sind(theta)*cosd(phi) -sind(phi);...
      cosd(theta)            0];
tau = [u;...
       v];
p = R*tau*3.7/2; % p = [x;... %3.7/2 is a factor between quantity
     of pixels vs
if (RoCZ(i) > DRoCZ)
    pause(0.1)
    MoveXYZSteps(leo,-p(1),-p(2),-p(3));
else
    Centered = true;
```

Listing A.4: Autofocus function code.

```
function [Focus frame] = autofocus(vid,leo)
    Cycles = 80;
    delta = 50;   %Step size of autofocuisng Camera
    Focus  = zeros(Cycles,3);
    figure
    ylim([5000 20000])
    xlim([0 Cycles])
    grid on
    hold on
for i=1:Cycles
    Focus(i,1) = i;
    if i == 1
        Focus(i,3) = 5*200;
    else
        Focus(i,3) = delta;
    end
        MoveCamera2(leo,Focus(i,3));
        pause(0.1)
    [Focus(i,2) frameE] =  EnGraAc(vid);
    [maxF,imaxF] = max(Focus(1:i,2));
    if Focus(i,2) == maxF
        frame = frameE;
        i
    end
    plot(Focus(1:i,1),Focus(1:i,2),'r');
end
GoingBack = sum(Focus(:,3));
        pause(1)
        MoveCamera2(leo,-GoingBack);
end
```

Listing A.5: GetCentroid function code.

```
function [Centroid,Radii] = GetCentroid(RGB)
%% Binarization.
HSV = rgb2hsv(RGB); % Transform RGB image to HSV format.
Gray = histeq(HSV(:,:,3)); % Use the equalized intensity channel.
BW = im2bw(Gray,0.99); % Convert into binary image.
%% Filtering.
seER=strel('square',3);%structuring element for erosion.
seDL=strel('square',5);%structuring element for dilation.
imero = imerode(BW,seER); % Erode image (filter noise).
imdil = imdilate(imero,seDL); % Dilate image (gain volume).
%% Segmentation/Labeling.
[L,N] = bwlabel(imdil);
properties = regionprops(L,'Centroid','EquivDiameter');
% Outputs.
Centroid = zeros(N,2);
Radii = zeros(N,1);
for i=1:N
    Centroid(i,1) = round(properties(i).Centroid(1)); %
        Horizontal axis.
    Centroid(i,2) = round(properties(i).Centroid(2)); % Vertical
        axis.
    Radii(i,1) = properties(i).EquivDiameter/2; % Radius.
end
end % Main.
```

Listing A.6: Merge function code.

```
function [MergedC,MergedR,OS] = Merge(C1,R1,C2,R2)
[Matches,OS] = Offset(C1,C2);
C1(:,1:2) = [C1(:,1) - OS(1),C1(:,2) - OS(2)];
[N1,Fields] = size(C1);
[N2,~] = size(C2);
Mark = zeros(N2,1);
Total = N1+N2-Matches;
MergedC = zeros(Total,3);
MergedR = zeros(Total,1);
if Fields == 2
    MergedC(1:N1,1:2) = C1;
elseif Fields == 3
    MergedC(1:N1,:) = C1;
end
MergedR(1:N1,1) = R1;
for i=1:N1
    for j=1:N2
        X = C1(i,1)-C2(j,1);
        Y = C1(i,2)-C2(j,2);
        if X >= -1 && X <= 1 && Y >= -1 && Y <= 1
            MergedC(i,1) = round((C1(i,1)+C2(j,1))/2);
            MergedC(i,2) = round((C1(i,2)+C2(j,2))/2);
            MergedC(i,3) = MergedC(i,3) + 1;
            MergedR(i) = (R1(i)+R2(j))/2;
            Mark(j) = 1;
            break;
        end
    end
end
Count = 1;
for i=1:N2
    if Mark(i) == 0
        MergedC(N1+Count,1:2) = C2(i,:);
        MergedR(N1+Count) = R2(i);
        Count = Count + 1;
    end
end
Distance = zeros(Total,1);
Minimum = zeros(Total,2);
for i=1:Total
    for j=1:Total
        if i==j
            Distance(j) = 100; % Assigned so this position is
                ignored.
        else
            Distance(j) = sqrt((MergedC(i,1)-MergedC(j,1))^2+(
                MergedC(i,2)-MergedC(j,2))^2);
        end
    end
    [Minimum(i,1),Minimum(i,2)] = min(Distance);
end
Count = 0;
DuplicateA = []; % Twice the amount of pairs.
for i=1:Total
    if Minimum(i,1) <= 0.5*mean(Minimum(:,1)) % IMPORTANT!
        Count = Count + 1;
        DuplicateA(end+1,:) = [i,Minimum(i,2)];
    end
end
```

Listing A.7: Offset function code.

```
function [Count,Position] = Offset(C1,C2)
Count = 0;
Position = [0,0];
for X=-30:30
    for Y=-30:30
        Aux = Matches(C1,C2,X,Y);
        if Count < Aux
            Count = Aux;
            Position = [X,Y];
        end
    end
end
end % Main.
```

Listing A.8: Matches function code.

```
function [Count] = Matches(C1,C2,X,Y)
Count = 0; % Number of matching elements returned.
[N1,~] = size(C1);
[N2,~] = size(C2);
CX = [C2(:,1) + X,C2(:,2) + Y];
for i=1:N1
    for j=1:N2
        if C1(i,1)-CX(j,1) >= -1 && C1(i,1)-CX(j,1) <= 1 && ...
           C1(i,2)-CX(j,2) >= -1 && C1(i,2)-CX(j,2) <= 1
            Count = Count + 1;
        end
    end
end
end % Main.
```

Listing A.9: DrawCircles function code.

```
% Function draws Circles on each Centroid Element of "Radius"
    radius.
% by JAVD, June 6th.
function [] = DrawCircles(Centroid,Radius)
[N,F] = size(Centroid);
if F == 2
    viscircles([Centroid(:,1),Centroid(:,2)],Radius(:),'EdgeColor
        ','r');
elseif F == 3
    for i=1:N
        if Centroid(i,3) == 0
            viscircles([Centroid(i,1),Centroid(i,2)],Radius(i),'
                EdgeColor','r');
        elseif Centroid(i,3) == 1
            viscircles([Centroid(i,1),Centroid(i,2)],Radius(i),'
                EdgeColor','g');
        elseif Centroid(i,3) == 2
            viscircles([Centroid(i,1),Centroid(i,2)],Radius(i),'
                EdgeColor','b');
        elseif Centroid(i,3) == 3
            viscircles([Centroid(i,1),Centroid(i,2)],Radius(i),'
                EdgeColor','y');
        elseif Centroid(i,3) == 4
            viscircles([Centroid(i,1),Centroid(i,2)],Radius(i),'
                EdgeColor','m');
        elseif Centroid(i,3) == 5
            viscircles([Centroid(i,1),Centroid(i,2)],Radius(i),'
                EdgeColor','c');
        else%if Centroid(i,3) >= 6
            viscircles([Centroid(i,1),Centroid(i,2)],Radius(i),'
                EdgeColor','k');
        end
    end
end
end
```

Listing A.10: Stitch function code.

```
function [C,R,VA] = Stitch(Images)
[X,Y,~,frameCount] = size(Images);
cX = round(X/2);
cY = round(Y/2);
VA = zeros(frameCount,2);
if frameCount == 0 || frameCount == 1
    errordlg('Insuficient input images. Input must have at least
        two.');
    C = [];
    R = [];
else
    [C,R] = GetCentroid(Images(:,:,:,1));
    VA(1,1) = cX;
    VA(1,2) = cY;
    for i = 2 : frameCount
        [Ci,Ri] = GetCentroid(Images(:,:,:,i));
        [C,R,OS] = Merge(C,R,Ci,Ri);
        % Apply current Offset to (Visual Axes) list.
        VA(:,1) = VA(:,1) - OS(1);
        VA(:,2) = VA(:,2) - OS(2);
        % Add current visual axis to (Visual Axes) list.
        VA(i,1) = cX;
        VA(i,2) = cY;
    end
end
end % Main.
```

BIBLIOGRAPHY

[1] G. J. Doornbos. "Design and implementation of an autofocusing algorithm for a 3D robotic scanner." University of Groningen, 2015.

[2] John K. Douglass. "Rapid mapping of compound eye visual sampling parameters with FACETS, a highly automated wide-field goniometer." In: *Journal of Comparative Physiology A* (2016).

[3] Stewart Duke-Elder. *System of Ophthalmology Vol. 1 The Eye in Evolution*. Henry Kimpton, 1958.

[4] M. van der Horst. "Design of an angular position measurement system and controller for a 3D robotic scanner." University of Groningen, 2015.

[5] Jae-Jun Kim, Jaeho Lee, Sung-Pyo Yang, Ha Gon Kim, Hee-Seok Kweon, Seunghyup Yoo, and Ki-Hun Jeong. "Biologically Inspired Organic Light-Emitting Diodes." In: *Nano letters* 16.5 (2016), pp. 2994–3000.

[6] Michael F. Land and Dan-Eric Nilsson. *Animal eyes*. Oxford University Press, 2002.

[7] Theobald Lohmueller, Robert Brunner, and Joachim P. Spatz. *Improved properties of optical surfaces by following the example of the Moth eye*. 2010.

[8] Tamara R. Spanier. "The Design and Implementation of the Full Actuation for the Five DOF 3D Robotic Scanner." University of Groningen, 2015.

[9] Doekele G. Stavenga. "Reflections on colourful ommatidia of butterfly eyes." In: *Journal of Experimental Biology* 205.8 (2002), pp. 1077–1085.

[10] Doekele G. Stavenga. "Modern Optical Tools for Studying Insect Eyes." In: *Methods in Insect Sensory Neuroscience*. CRC Press, 2004. ISBN: 978-0-8493-2024-8. DOI: doi:10.1201/9781420039429.sec3. URL: http://dx.doi.org/10.1201/9781420039429.sec3.

[11] Doekele G. Stavenga, S. Foletti, G. Palasantzas, and Kentaro Arikawa. "Light on the moth-eye corneal nipple array of butterflies." In: *Proceedings of the Royal Society of London B: Biological Sciences* 273.1587 (2006), pp. 661–667.

[12] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[13] Kürt Johan André Vanhoutte. *Butterfly visual pigments: molecular cloning and optical reflections*. [University Library Groningen][Host], 2003.

[14] Andrés Vargas-Delgado, Mauricio Muñoz-Arias, and Doekele G. Stavenga. "Characterization of the visual space of butterfly eyes via a robotic scanner." In: *Book of Abstracts 35th Benelux Meeting on Systems and Control*. Ed. by Raffaella Carloni, Dimitri Jeltsema, and Mircea Lazar. 2016, p. 102. ISBN: 978-90-365-4035-3.

DECLARATION

I certify that this thesis to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text.

Cartago, Costa Rica, August, 2016

Jimmy Andrés Vargas
Delgado

ACT OF APPROVAL

INSTITUTO TECNOLÓGICO DE COSTA RICA

CARRERA DE INGENIERÍA MECATRÓNICA

PROYECTO DE GRADUACIÓN

ACTA DE APROBACIÓN

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Mecatrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

Dr. Juan Luis Crespo Mariño
Profesor lector

MSc. Marta Eugenia Vílchez Monge
Profesor lector

Dr. Ing. Mauricio Muñoz Arias
Profesor asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Carrera de Ingeniería Mecatrónica

Cartago, 16 de Agosto, 2015