DESIGN OF SOFTWARE CODE TO IMPROVE THE ACCURACY OF SHRIMP GRADING MACHINE

CHAN CHAO SHIUNG

B. ENG (HONS.) MECHATRONICS ENGINEERING

UNIVERSITY MALAYSIA PAHANG

DESIGN OF SOFTWARE CODE TO IMPROVE THE ACCURACY OF SHRIMP GRADING MACHINE

CHAN CHAO SHIUNG

Thesis submitted in partial fulfilment of the requirements
for the award of the degree of
Bachelor of Engineering in Mechatronics Engineering (Hons)

Faculty of Manufacturing Engineering
UNIVERSITI MALAYSIA PAHANG

JUNE 2016

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name	:	CHAN CHAO SHIUNG
Identification Card No	:	890824-06-5147
Title	:	DESIGN OF SOFTWARE CODE TO IMPROVE THE ACCURACY OF SHRIMP GRADING MACHINE
Academic Session	:	2015/2016

I declare that this thesis is classified as:

CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1972)
RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
OPEN ACCESS	I agree that my thesis to be published as online open access (Full text)

I acknowledge that Universiti Malaysia Pahang reserve the right as follows:

1. The Thesis is the Property of University Malaysia Pahang.
2. The Library of University Malaysia Pahang has the right to make copies for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

_____ (Author's Signature)

Date: _____

_____ (Supervisor's Signature)

Name of Supervisor

Date: _____

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Bachelor of Engineering in Mechatronics Engineering.

Signature	:	
Name of supervisor	:	DR. FAIZ BIN MOHD TURAN
Position	:	SENIOR LECTURER
Date	:	

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is my own except for quotations and summaries which have been duly acknowledged. The thesis has not been accepted for any degree and is not concurrently submitted for award of other degree.

Signature	:	
Name	:	CHAN CHAO SHIUNG
ID Number	:	FB12061
Date	:	

ACKNOWLEDGEMENT

First of all, with my greatest pleasure and honor, I would like to take this opportunity to thank my supervisor, Dr.Faiz Bin Mohd Thuran. While having him as my supervisor, he guided me for my Final Year Project by giving me the opportunity and chance to learn and in order to make the project a success.

Next, I would like to say thanks to my family, friends, staffs and lecturers of Manufacturing Engineering Faculty that involved no matter directly or indirectly for the continuous support, guidance, and encouragement throughout my completion in university education. Especially my family who has gave me love and sacrifies, supported me emotionally and financially since the beginning with no doubt.

Last but not least, I would like to sincerely thank Hannan Food Sdn Bhd for giving me the opportunity to work on the shrimp grading project with them along with their co-operation and willingness for all the requirements needed at various unexpected situations.

ABSTRACT

Shrimp is a famous dish all around the world, it is famous with its taste and variety of dish be match with shrimps. In the pass, it is possible to grade shrimp manually using manpower as peoples are not aware of the juicy taste of shrimp. By using manual shrimp grading, people often creates human error and did some mistakes in grading shrimp when it comes to mismatching the size with the standard grading parameter. To grade food, it requires precise accuracy on differentiating the weight of the food as every grams of the food comes with price depending on how much weight it produces and types of food being grade. All the while, beside manual shrimp grading by manpower, there are other method such as grading shrimp mechanically and image processing been used. This project uses weight sensor to grade shrimp because it suits the needs of buyers and the market. By using weight sensor, it is faster then the image processing method because the sensor is placed under the conveyor to measure the product such as shrimp once the shrimp goes on top of it. Weight sensor also produces less error when it does not depends on external source such as light to operates, the only limitation for weight sensor is the range of weight prefered by the factory to grade shrimp. The main objective of this project is to create a software code using MATLAB and C programming so that it could help improve the shrimp grading accuracy interms of quality, cost effectiveness, and man power.

ABSTRAK

Udang adalah hidangan yang terkenal di seluruh dunia, ia terkenal dengan rasa dan pelbagai hidangan enak yang melibatkan udang. Ketika masa dahulu, ia adalah mungkin untuk mengasingkan udang secara manual menggunakan tenaga manusia kerana ramai daripada kita tidak mengenali keenakan udang lagi. Dengan mengasingkan udang secara manual, kita sebagai maunusia sering menimbulkan kesilapan dan melakukan kesilapan dalam mengasaingkan udang apabila ia berlaku semasa mengasingkan saiz dengan piawai yang ditentukan oleh kilang udang. Untuk mengasingkan makanan mengikut piawai, ia memerlukan ketepatan yang sangat tepat pada berat makanan yang berbeza kerana setiap gram makanan diambil kira dengan harga bergantung kepada berapa banyak berat yang produk menghasilkan dan jenis produk yang diambil kira. Disamping menggunakan kaedah pengasingan udang secara manual oleh tenaga manusia, masih ada kaedah lain yang digunakan seperti pengasingan udang secara mekanikal dan pemprosesan imej telah digunakan. Dalam projek ini, sensor pemberat telah digunakan kerana ia adalah cara terbaik untuk proses pengasingan udang dan ia sesuai dengan keperluan pembeli dan pasaran. Dengan menggunakan sensor pemberat, ia adalah lebih cepat berbanding kaedah pemprosesan imej kerana ia diletakkan di bawah penghantar untuk mengukur produk seperti udang semama udang lalu di atasnya. Sensor pemberat juga menghasilkan kurang ralat kerana ia tidak bergantung kepada sumber luar seperti cahaya untuk beroperasi, hanya had bagi sensor pemberat adalah julat berat udang diinginkan oleh kilang untuk gred udang. Objektif utama projek ini adalah untuk mewujudkan satu kod perisian menggunakan pengaturcaraan MATLAB dan C supaya ia boleh membantu meningkatkan ketepatan proses pengasingan udang dari segi kualiti, keberkesanan kos, dan tenaga kerja.

TABLE OF CONTENTS

CHAPTER 3 INITIATING

CHAPTER 4 RESULTS AND DISCUSSION

CHAPTER 5 CONCLUSION

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

σ	Weighted sumation of variances
ω	Weight
t	Time
f	Turn angle function 'f'
g	Turn angle function 'g'
n	Amount of sliding
$\stackrel{\text{def}}{=}$	Equal to by definition
m	Number of data point
$\subset$	Subset
$\in$	In
R	Segmentation reference
$\forall$	For all
min	minimum

LIST OF ABBREVIATIONS

HSV	Hue, Saturation, and Value
GUI	Graphical user interface
DV	Daily value
US	United State
GPO	Glutathione Peroxidase
SOD	Copper-zink superoxide dismutase
DHA	Docosahexaenoic acid
EPA	Eicasapentaenoic acid
LDL	Low-density lipoproteins
HACCP	Hazard Analysis and Critical Control Point
TADA	Turn Angle Distribution Analysis
TAC	Turn Angle Cross-correlation
RGB	Red, green, and blue
SAD	Sum of absolute differences
SM	Similarity measurement

CHAPTER 1

INTRODUCTION

1.1 PROJECT BACKGROUND

Shrimps are like lobster and crab but with smaller and weaker legs, it is categorized as arthropods. Arthropods are living things that have their skeleton on the outside of the body and is called exoskeleton which gives shrimps their unique look. There are thousands of total shrimp species worldwide, they can be found in saltwater, freshwater, brackish water, or combination of habitats. Researchers often use different ways to categorize shrimps, for example warm-water shrimp of tropical waters from south, and cold-water shrimp from north where the water are colder.

Approximately thirty percent (30%) of consumed seafood in U.S. are shrimps which can be explain by about 4-5 pounds out of 14-15 pounds of total seafood consumed per year come from shrimp. While average amounts of tuna and salmon are consumed in about 2-3 pounds a year which makes them fall to second place. Shrimp are captured, processed, market, and cooked in many different ways which includes frozen, breaded, fresh, dried, cooked, and in the form of paste. Therefore, in the market, we will find shrimps that are processed into different categories which is with vein, without vein, peeled, unpeeled, and with head or without head.

The terms "shrimp" and "prawns" are confusing. Larger shrimps are called "prawns" where its habitats are mostly from freshwater, while smaller shrimp are called "shrimp" where its habitats are mostly from saltwater. As for size, shrimps are often measured with per kilogram where large shrimps will get about 40 or less per kilogram compared to 50 or less for medium size shrimp and 60 or less for small. But from the view of science, shrimps and prawns can both come from either freshwater or saltwater,

and there are no specified method for measuring the size of a shrimp into small, medium, or large categories.

There is always one question most people would ask about, which is how shrimp sizes are categorised as jumbo, large, medium, and small. But there is no standard method used for measuring shrimp size, number of shrimps per kilogram method is mostly being used to measure the size of shrimps. For small shrimps it is usually measured around 60 shrimps per kilogram. While for medium shrimp, it reduced to about 50 shrimps per kilogram because the shrimp is bigger, and with more weights. Large shrimp will be measuring around 40 shrimps per kilogram. As for super-size jumbo shrimp the measurement is about 30 shrimps per kilogram.

Shrimp has become more and more favourable among all of us for its crunchy bite and its rich nutrients. Shrimp contains carotenoid nutrient astaxanthin which provides antioxidant support to nervous systems and musculoskeletal system. Studies shows that astaxanthin can helps reduce the risk of colon cancer and also some diabetes related problems.

Besides, shrimps also contain mineral such as selenium which helps glutathione peroxidase (GPO) to function properly, GPO is a type of enzyme that cannot function without selenium. GPO is an enzyme that protects most of our human body systems such as lungs from damage caused by small particles in air. Research shows that the selenium found in shrimp can be absorb efficiently by human body which is 80-85% of total intake of selenium. The lack of selenium mineral in human body may also cause cardiovascular disease such as heart failure along with other problems which includes type 2 diabetes, depression, and cognitive function.

On the other hand, copper-zinc is also found in shrimp and is also classified as antioxidant mineral. In our body, copper-zinc is required to help the functions of copper-zinc superoxide dismutase (SOD), which is a type of enzyme. SOD is located in most fluid compartment of our cells which is cytosol, it helps regulating oxygen metabolism and also preventing oxidative stress.

Recent researchers have proven some useful side of the fatty acids found in shrimps. Omega-3 is one of the fatty acid found in shrimps. In omega-3, there are DHA (docosahexaenoic acid) and EPA (eicosapentaenoic acid) which is very important to our nervous and cardiovascular system. At the same time, the ratio of omega-6:omega-3 in shrimps are very unique with around 1:1 ratio. This high ratio of omega-6 and omega-3

will help to reduce the risk of many chronic diseases such as obesity, diabetes, and high blood pressure. In addition, according to recent studies, there are other sterol found in shrimp beside cholestrol, it is a type of fat which can be found in the form of clionasterol and campesterol in small amounts. It is similar to cholesterol chemically and function as anti-inflammatory molecules and are useful is decreasing the level of LDL-cholesterol. Therefore, it can be considered as a health benefit diet.

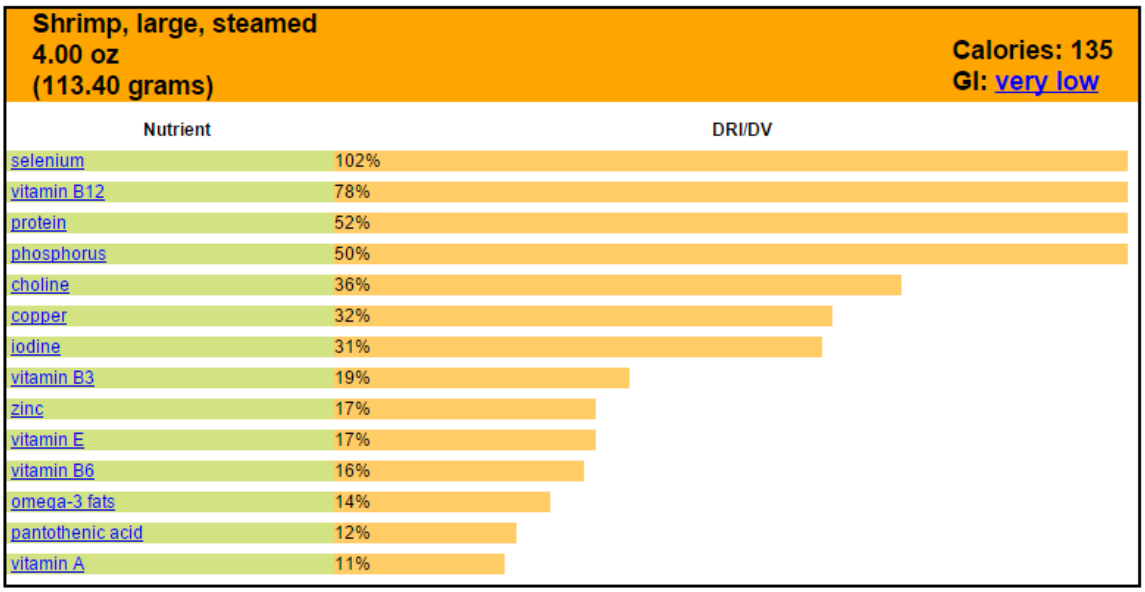


Figure 1.0: This chart illustrate the percentage of Daily Value (DV) nutrients in each shrimp serving.

Source: WHFoods (2013)

In Selinsing District Perak Malaysia, a company named Hannan Food Sdn Bhd was established in 2009 where the main business of this company is shrimp aquaculture. The company has set up an aquaculture complex with 150 vannamei shrimp culture ponds. Until today, the company has started its operation with thirty ponds and will increase more in the future. The raw shrimps in Hannan Food Sdn Bhd is processed and categorized manually by man power using HACCP (Hazard Analysis and Critical Control Point) standard where food safety is the main focus through the analysis and control of food from beginning till the final product.

1.2 PROBLEM STATEMENT

The current method of Hannan Food Sdn Bhd processing fresh captured raw shrimp is facing some problem which is the accuracy in term of quality, cost effectiveness, and productivity. The existing system using image processing, mechanical machines, and manpower needs improvement.

Referring to the existing journal, the method proposed was to use vision machine which will visualise the shrimp, change the image to binary codes, analyse using algorithm and turn angles, then determine the shape of the shrimp automatically by computer, and finally categorise the shrimp.

However, the vision machine uses vision sensor which is costly and requires precise coding which will consumed plenty of times to revise the codes. When problem occur, operators will need a lot of times to troubleshoot the problem and this may lead to delay of production line which may cause the company to loose valuable sales.

1.3 OBJECTIVE

The project objectives are:

- To establish the relationship between characteristic of shrimps and machine's parameters.
- To develop the software code using MATLAB and C programming to improve the accuracy of shrimp grading process.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

After reading and going through journal, It is obvious that there are not much open source journal and thesis about shrimp grading machine, but through studies, most of the proposed method of grading shrimp accurately was to use vision machine, it has been experimented and proven to be applicable to grading shrimp. Below are some of the method used and some pictures to help better understanding on how the method works.

Image processing machines are widely used in agriculture and food industries for grade evaluation for it is highly efficient and non-destructive, it is used to determine the beans, sizes of seeds, and to grade fruits such as apples and oranges. Fish, on the other hand, were analysed using image processing which includes fish length and shape. As for shrimp, weights and visual quality were also analysed by image processing in vision machine and is utilized to evaluate length and shape of shrimp.

There are plenty of image recognition methods being exploit to carry out object recognition task in industries. The most common ones are landmark point analysis, whole shape matching and shape recognition. The shape analysis method used to find critical landmark points produces inaccurate results in most applications, this may cause problem because landmark points cannot be accurately located most of the time. Few shape descriptors were analysed to match the desired shape and few of the matching methods such as Fourier descriptors, polygon approximation, and line segmentations were proven to perform better recognition than landmark point matching, but it does not improve much because their recognition accuracy are still below 60%.

On the other hand, a vision machine system that is able to sort oysters by its size and irregular shapes detection was developed. The design of this system involves software for computer vision algorithms, a special camera and optics arrangement, a

custom-designed light chamber, specific control and delivery mechanism, and friendly graphical user interface (GUI). Such design can be easily applied in shrimp grading.

Later, a better proposed method with better improved performance for shape matching was developed in Contour Matching for Fish Species Recognition and Migration Monitoring. It is a fast and accurate shape matching method called Turn Angle Distribution Analysis (TADA), it is developed to allow the input contour to be matched with different contours from the database for shape-based recognition in this approach. The TADA test results was proven to be more accurate, and it improves the effectiveness of the fish recognition system. Furthermore, a method to measure turn angle similarity in oyster grading using Turn Angle Cross-correlation similarity measure (TAC) was proposed.

Grading shrimp using TAC method is more accurate and simple. Section 2.2 and 2.3 will explain about what the colour segmentation and contour extraction is. While the whole grading algorithm will be shown in section 2.4.

2.2 COLOUR SEGMENTATION

When undergoing colour segmentation in the latest shrimp quality evaluation system, it is best to have a background colour that is different from any species of shrimp from around the world. Therefore, blue colour background is selected because most of the agricultural products have colours that are not blue to ease the colour segmentation process so that it can locate the product and extract the contour without facing any problem.

Now a days, most colour cameras in machine projects images in three colours which is red, green, and blue (RGB) channels. Because of fine changes in lighting or inconstant lighting, a commonly used thresh holding method in the blue channel may not produce desired results of segmentation. And because of that, it is a very common practice to convert the red, green, blue image data to another colour space for processing in colour image processing. Even though there are many colour spaces that can be utilize for colour image processing but some of them are complicated and are not suitable for applying on actual vision machine. But HSV (hue, saturation, and value) can help solve the problem and it is a commonly preferred colour space for colour segmentation and is a straightforward process in converting RGB to HSV.

There are several categorization of colour segmentation algorithms which can be feature space clustering, histogram threshold, edge-based detection, region-based, neural network, and fuzzy set. From all of the choices, simple thresh holding has been chosen for the application because even with inconstant lighting, the blue background hue value stays relatively consistent. By thresh holding hue values in the blue channel will be improved in the new proposed method.

From the proposed method, in blue channel the HSV statistical values were obtained by selecting 50 blue background samples. In Table 2.0, it shows the statistics curves that can be used to obtain characteristic parameter of the blue background with HSV colour space. The risk and the percentage error must be at its minimum point to avoid to increase the accuracy. In this case, hue is the best channel among the three HSV channels for segmentation because it has the lowest changing range in risk and percentage error. To avoid missing the product, the thresh holding range must be expanded from 110-120 to 100-130. Figure 2.0 shows the sample of the segmentation result.

Table 2.0: Thresh holding parameters of HSV colour space

Colour channel	Hue	Saturation	Value
blue	110-120	140-190	90-240

(a) (b)

Figure 2.0: (a) Original image with blue background and (b) Shrimp body that is segmented from the background.

Source: Dah-Jye Lee (2012)

2.3 CONTOUR EXTRACTION

A. Initial Contour Extraction

Referring Figure 2.1(a), when converting the result to binary image, the binary image may contains small blobs also known as pixel clusters from image noise or some minor defects inside the binary image. So, to analyse the shape of shrimp contour, an operation called shape operations are performed to do some repairs on the noise and defects of shrimp contour. In Figure 2.1(c), the x and y coordinates of the shrimp contour is extracted by using a fast and simple eight-neighbourhood contour trace algorithm.

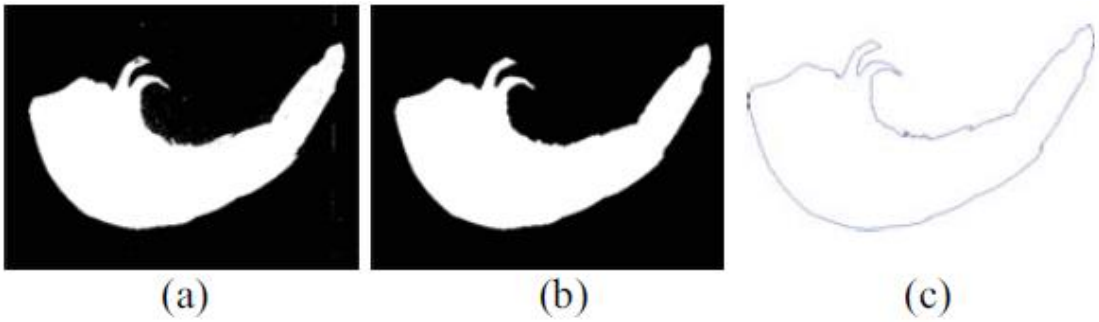


Figure 2.1: Shape operations and contour extraction.

Source: Guangming Xiong (2012)

The proposed contour extraction method can be applied to most cases except when the legs are included in the contour as illustrated in Figure 2.2(a). It is critical when shrimp legs are included in shape analysis because the legs could cause the system to categorize a good shrimp as defect. A further process is required to trim the legs of shrimp from contour.

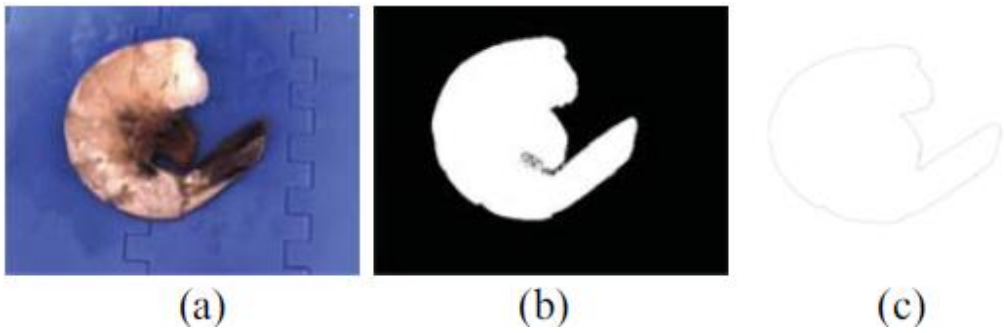


Figure 2.2: Example of shrimp contour that includes the legs.

Source: Guangming Xiong (2012)

B. Contour Trimming

The colour of shrimp legs illustrated in Figure 2.2(a) is black which is darker than the other part of shrimp body. In Figure 2.2(b), a method called Otsu's method is applied to exclude the primary contour in the intensity channel (V) for the area highlighted by the binary image shown where pixels in the blue background will not be evaluated. To minimize the intra-class variance in Otsu's method, they have to continuously search for a threshold that classified as weighted sum of variances of the two classes:

$$\sigma_{\omega}^{2}(t) = \omega_{1}(t)\sigma_{1}^{2}(t) + \omega_{2}(t)\sigma_{2}^{2}(t) \tag{2.1}$$

Weights ω are two classes separated by a threshold t and changes of classes.

By applying threshold, it can differentiate the shrimp legs from the shrimp body. It can be done for the pixel to mark as a foreground pixel when in a given pixel, the grey value is approximately more than or equal to threshold. But the pixel will be highlighted as part of the legs or tail when less than threshold. A binary image is created using these information, where ‘1’ representing shrimp body pixel while ‘0’ representing it is a leg or tail. In this way, an improved version of contour for shape analysis can be done. Figure 2.3 shows the results of binary image and also the extracted contour.

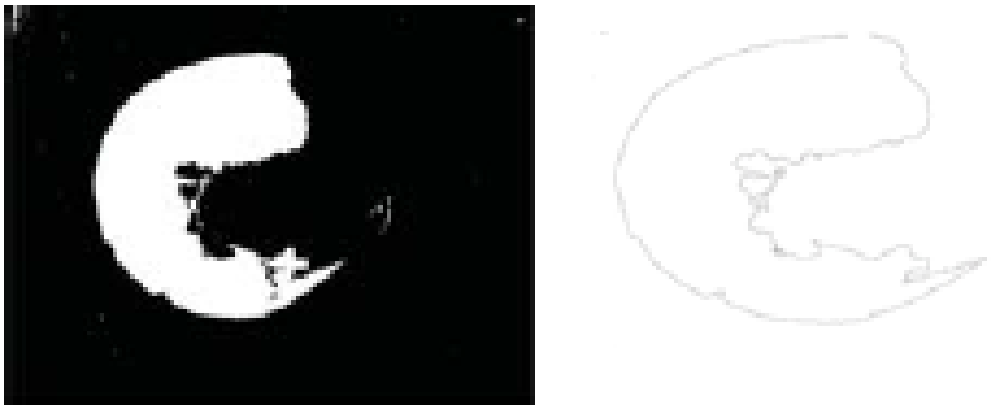


Figure 2.3: Contour extraction result.

Source: Dong Zhang (2012)

2.4 TURN ANGLES CROSS-CORRELATION

A. Data Point Reduction

It is better to exclude those imperfect data points by applying filter before performing shape analysis because most of the x and y coordinates on the extracted contour are imperfect and describe of overall shape is unecessary. Besides, by reducing the number of data points on the contour to a reasonable number is also necessary so that the contour can be evaluated using shape similarity measurement and the overall shape can be better describe by the contour.

There are two data reduction method being tested. The first method is by fixing interval sub-sampling of the contour data points. And the second method is a technique developing curve that repeatedly compares all of the related measures of the vertices on the contour.

Refering Figure 2.4(a), it illustrate the result of contour extraction method, contour of a shrimp which originally having 1827 points using this method. While in Figure 2.4(b) and 2.4(c) when using fixed interval sampling, it shows 50 and 30 data points. When using curve development technique, Figure 2.4(d) and 6(e) shows 50 and 30 data points were obtained.

B. Turn Angle vs. Normalized Length

When the data points are reduced, it is implimented to depict the overall shape using turn angle function against normalized length. Then turn angle will be calculated with respect to the orientation of the previous segment as shown in Figure 2.4(a), so that when it is turning clockwise it will gives a negative angle and when it turns counterclockwise it gives a positive angle. The closed contour and line segment orientations are illustrated in Figure 2.5(a) where solid line segments are parts of a closed contour and dotted lines are the line segment orientations.

Because the objects may change in size and the image will be analyse using different direction and at different location of the image the shape descriptors and classification algorithms should be unchangeable in rotation, translation, and scaling. It is very important to take note aboout the suggested technique, polygon representation, uses turn angle against normalized length so that the representation scaling is unchanged. On the other hand, the turn angles does not have any data apprehensing the shrimp position, and therefore translational is unchanged because the contour contain only turning angles. The same shrimp represent the same turn angle function depending on the location in the image and its orientation. As illustrated in Figure 2.4(b), a sample of turn angle function, the turn angle ranging from -180 to 180 is illustrated on y-axis and the range of normalized length from 0 to 1 is illustrated on x-axis.

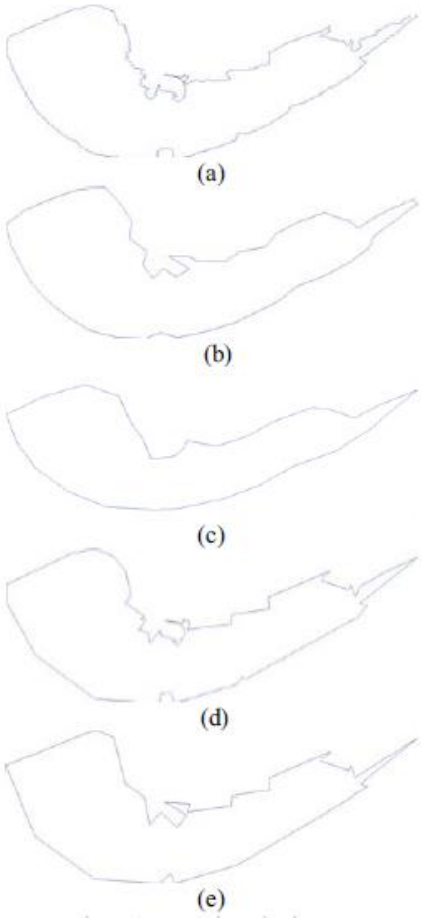


Figure 2.4: Data point reduction.

Source: Robert M. Lane (2012)

C. Turn Angles Cross-Correlation

The fish species recognition is analysed and matched with a newly developed technique called Turn Angle Distribution Analysis (TADA). To obtain best match, the contour of a trial image has to be adapted with the reference contours, a fast and effective shape matching method is used to characterize the contour data set. By using the TADA method, the shape-matching accuracy will improves and the shrimp shape grading will be adapted and modified. After experimenting, the method of measuring turn angle appear to have some similarities with Turn Angle Cross-correlation measurement (TAC) and is already implemented in oyster shape grading. Therefore, the TAC method is implemented into grading shrimp.

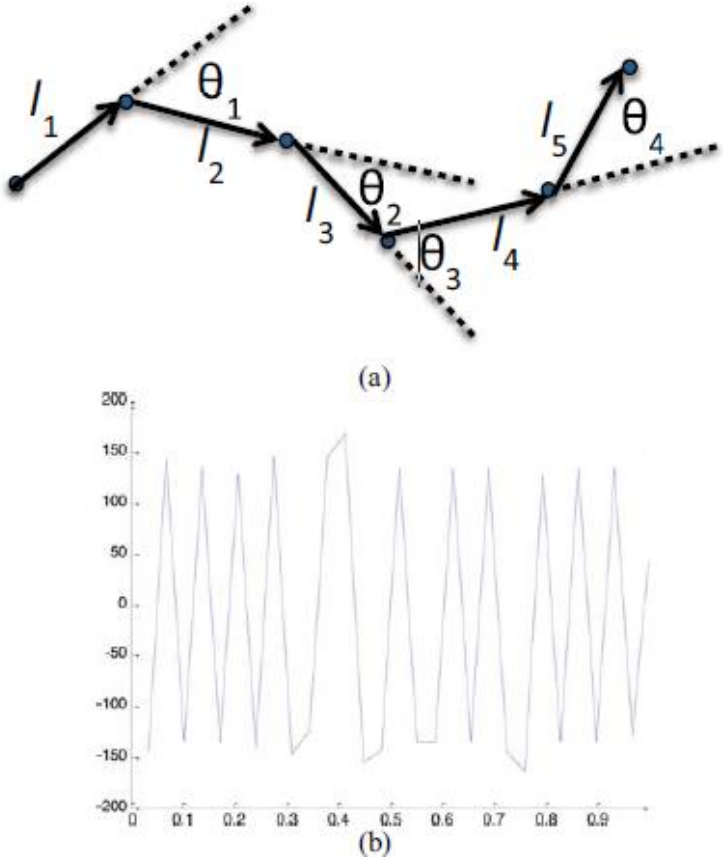


Figure 2.5: Shape description using turn angle vs. normalized length. (a) Turn angle calculation and (b) an example of turn angle function.

Source: Dong Zhang (2012)

There is no specified point that can be used as a reference point in shrimp shape analysis. But it is better to resolve the unchanged-movement problem due to the lack of a reference point on the contour in order to use the fast and effective TADA channel for shrimp shape grading. The turn angle function could be changed horizontally in Figure 2.5(b) to create 30 possible shifts in turn angle function for a 30 data points contour. To determine if the two turn angle functions have same shape characteristics and belong to the same shape category, calculations have to be made to see the difference between the two turn angle as a similarity measure by aligning two turn angle functions as close as possible.

The normalized length in turn angle function that moves along the x-axis is treated as a separate function and the cross relationship between the two separate functions can be compared to as the sum of absolute differences (SAD) using the equation bellow:

$$(f * g)[n] \stackrel{\text{def}}{=} \sum |f[m] - g[n+m]| \tag{2.2}$$

In this equation, n is the amount of sliding. For the best alignment to happen, cross relationship value is minimized. Thus, the similarity measurement (SM) with m data points between two turn angle functions is calculated as bellow:

$$SM = \{min \sum |f[m] - g[n+m]| \, \forall_n \subset [1, m]\} \in R \tag{2.3}$$

D. Building a Model for Each Shape Category

To build a shrimp model, 30 samples of shrimp images are chosen for shape categorization to perform turn angle functions calculation. A shrimp is chosen as reference using the proposed TAC similarity measurement method is performed on each of the 30 turn angle functions which has aligned with the turn angle function.

The similar shape category will never be hundred percent identical because there will be fine changes in shrimp shape and some minor segmentation error where the 30 resulting aligned turn angle functions belong to. The average of the 30 aligned turn angle functions is used as a type model or a model for every shape category after outliners being eliminated. The good shape quality shrimp is generated using type model generation the steps are illustrated in Figure 2.6.

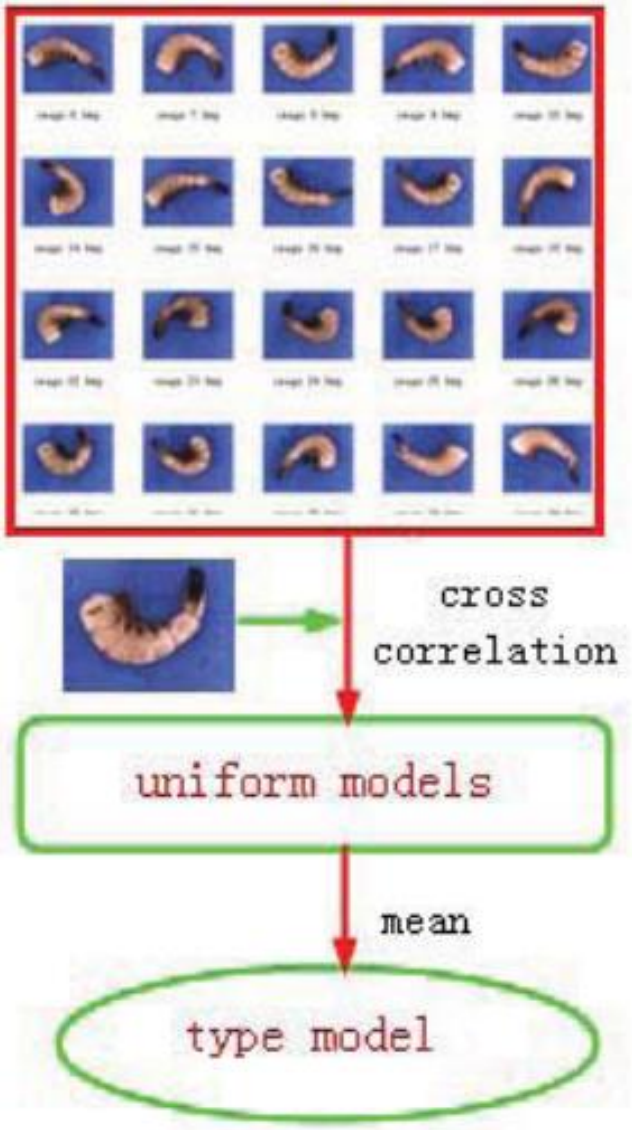


Figure 2.6: Building a type model for each shape category.

Source: Guangming Xiong (2012)

E. Shape Analysis and Grading

The whole process of analysing shrimp shape and algorithm of grading shrimp using TAC similarity measuring method is illustrated in Figure 2.7. As discussed in the previous section a turn-angle distribution type model or model for each shape category is created to perform accurate shape analysis and kept for similarity measurement in the database.

A shrimp with a major axis near to the horizontal axis is chosen as reference and the turn angle function of that particular model is used as a reference. Next, by using the proposed TAC similarity measurement, the turn angle function of each tested shrimp contour in this reference model is aligned and shifted to filter the turn angle function shift caused by rotational movement where the resulting aligned turn angle function is called the uniform model. The comparison between good and broken type models are perform on each uniform model. A model of shrimp is selected as reference to perform the implemented TAC similarity measurement method to determine the category of the shape which is the smaller TAC similarity measurement of the good and broken models to determine the category of the tested shrimp's shape.

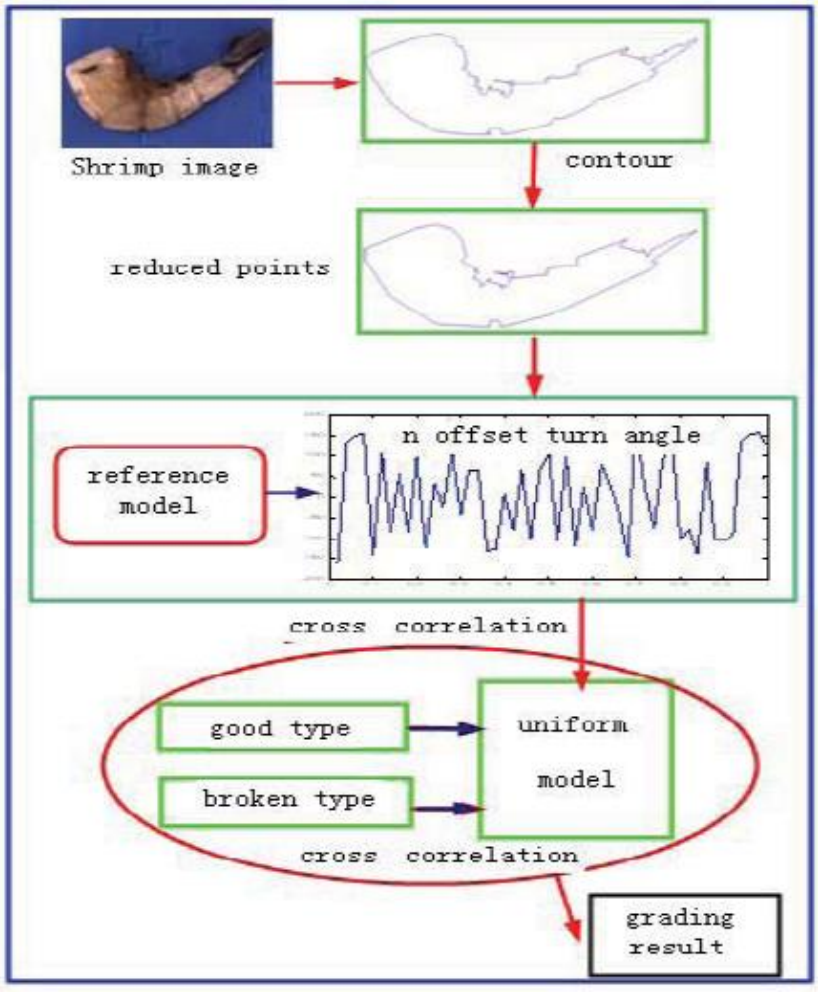


Figure 2.7: Shrimp shape analysis and grading diagram.

Source: Dah-Jye Lee (2012)

CHAPTER 3

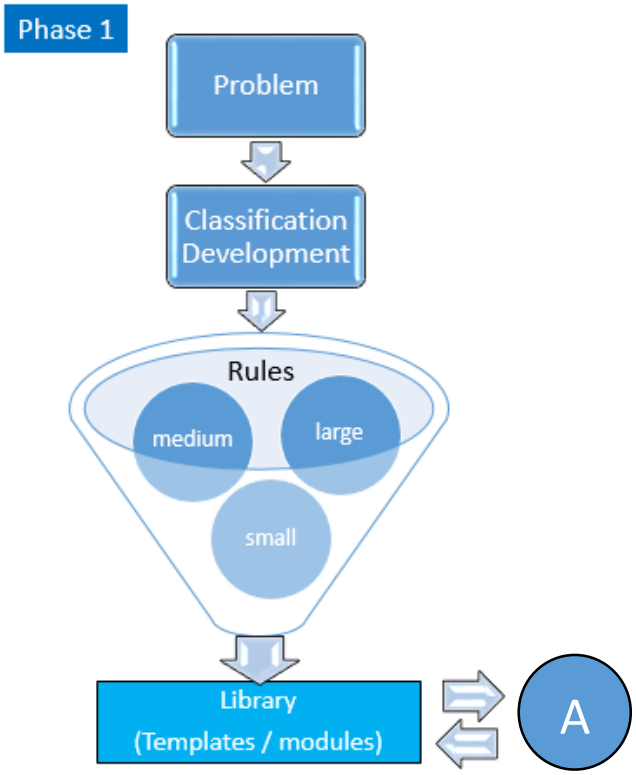
INITIATING

3.1 INTRODUCTION

In chapter 3 will discuss about how to create GUI using MATLAB and to link Simulink with GUI using C code programming. Data of shrimp sizes are obtained from Hannan Food Sdn Bhd, Perak, Malaysia.

3.2 METHODOLOGY

Below is the flow chart that shows about what it is done during the project from the beginning until the end of project.



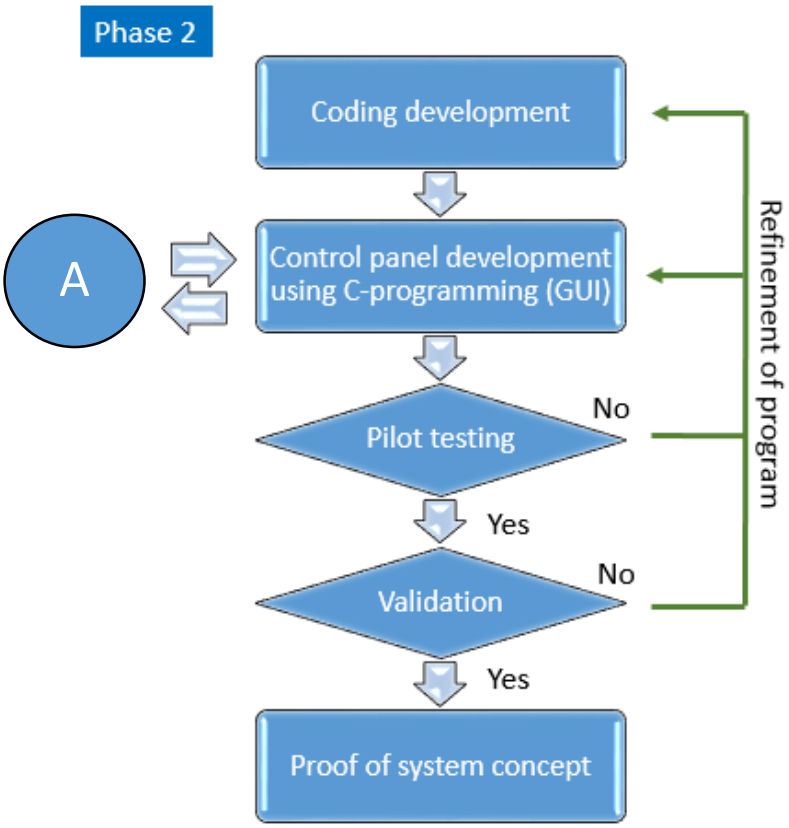


Figure 3.0: Flow chart of shrimp grading software code design.

Phase 1

1. To initiate this project, the first thing to do is to troubleshoot problem face by industry, according to problem statement.
2. Develop the classification of shrimp from the literature review.
3. Then create rules in MATLAB and by using C programming to classify shrimp into small, medium, large, and jumbo.

Phase 2

1. Develop coding for classification of shrimp in C programming.
2. Develop control panel with Generated User Interface (GUI) using C programming referring to library.
3. Pilot test and validate the prototype with refinement of program.
4. Finally proof the concept of the system.

3.3 EXPECTED OUTCOME

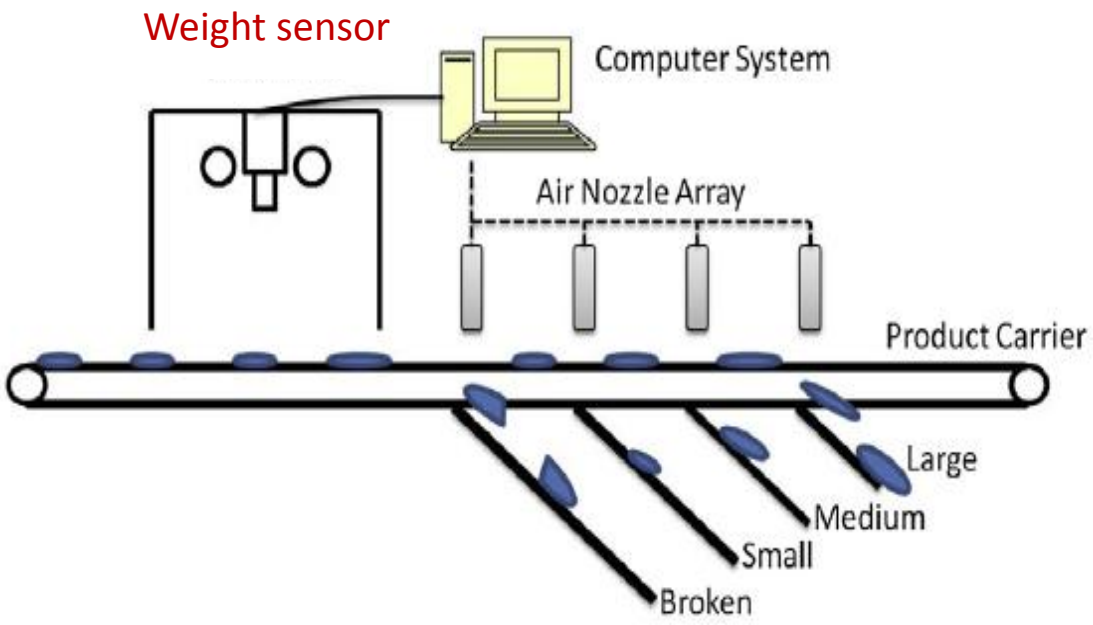


Figure 3.1: Expected shrimp grading system.

Figure 3.1 is the expected outcome of the project previously. It is to put the weight sensor underneath the conveyor and uses air nozzle to grade the shrimp into different sizes.

3.4 MODELING AND DESIGNING

3.4.1 Block Diagram

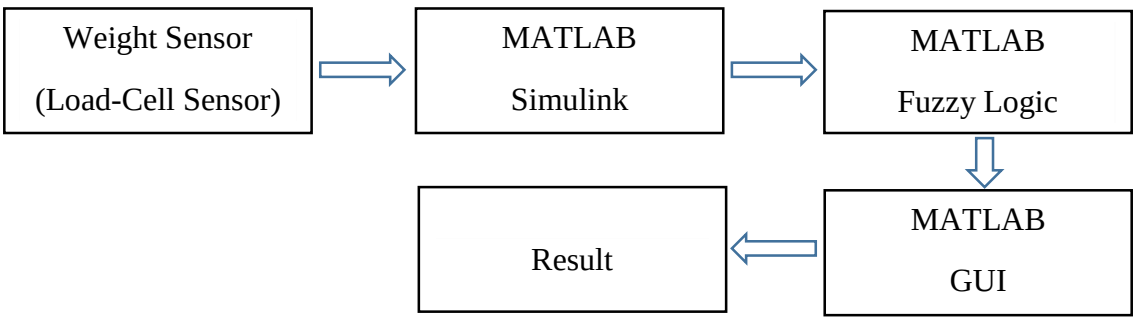


Figure 3.2: Shrimp grading system block diagram.

To create block diagram, we will have to first understand why use Simulink in this project rather than other open source software. Simulink is a tool integrated in MATLAB which provides graphical editor, self-designed block libraries, and provides solutions for modelling and simulating dynamic systems. Simulink is capable of incorporating with MATLAB algorithms into models and export simulation result to MATLAB for further analysis.

Referring to the shrimp quality evaluation block diagram in Figure 3.2, the input signal (shrimp weight) is sent into the Simulink from conveyor to be further process by MATLAB using fuzzy logic. After processed by fuzzy logic, the result will be displayed in a GUI created using MATLAB GUIDE tool which will show the category of the shrimp in the range of small, medium, large, or reject. The category range will be set by the shrimp industry according to their preference.

3.4.2 Procedure

1. Create a MATLAB GUI using “GUIDE”.

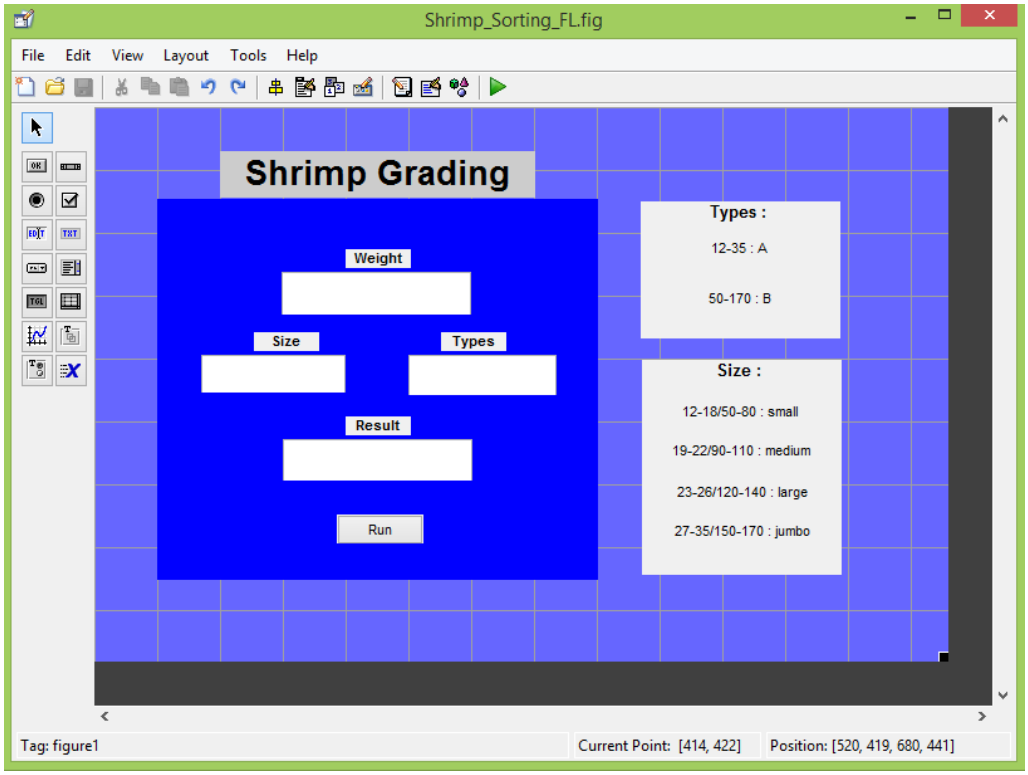


Figure 3.3: MATLAB GUIDE.

In MATLAB GUIDE, we are free to create whatever GUI designs. From Figure 3.3, when creating GUI, we can start by dragging those tool box from the left selection panel. In this shrimp grading project, my main motive of creating this GUI is to display the size and types of shrimp according to the weight of the shrimp.

On the GUIDE design panel, there are four displaying text box on the blue main panel which is labelled as Weight, Size, Types, and Result. From there, the Weight text box is to display the weight of the shrimp from the weight sensor, Size and Types are to display the results of MATLAB fuzzy logic in numerical form, while the Result is to display the summary of the Size and Types in alphabetical form.

At the same time, there are two big text box on the right side of the GUIDE design panel which is created as reference for the user based on the range provided by factory. On the other hand, there is a small push button to initiate the GUI display. The display can be initiate by clicking the push button labelled as “Run”.

After completing the design, right-click on the tool box to configure the properties such as font size, colours, or font pattern by selecting “Property Inspector”. Buttons can be assigned to initiate operations such as displaying results, run programs, or even doing calculations.

2. GUI code can edit in “Editor” to link between objects.

- For “display” example:

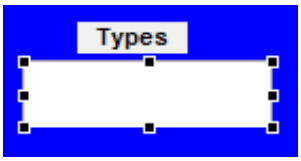


Figure 3.4: GUI display.

```
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Figure 3.5: GUI code to link display with Simulink.

In Figure 3.5 are the codes to program display text box in Figure 3.4. First, to edit the code of the display text box, right-click on the display text box then click on "View Callbacks" and click on "Callback". By doing so, it will direct us to MATLAB editor so that we could do some modifications on the code.

In this project, the function of the text box is added with the function to display the result of MATLAB fuzzy logic from Simulink. The text box is commanded to get data from the workspace where the fuzzy logic data run by Simulink is placed. The result of the Simulink fuzzy logic will be displayed in the text box in numerical form. The results of the fuzzy logic will be link to GUI for user to view and be recorded.

- For "push button" example:

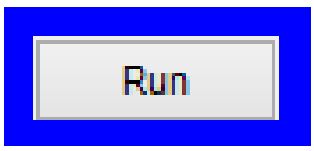


Figure 3.6: GUI push button.

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
a=evalin('base','input.signals.values');
set(handles.edit3,'string',num2str(a));
b=evalin('base','size.signals.values');
set(handles.edit1,'string',num2str(b));
c=evalin('base','types.signals.values');
set(handles.edit2,'string',num2str(c));
```

```
sv = str2num(get(handles.edit3,'string'));
if (sv >= 12) && (sv <= 18);
    S='small,A';
    disp(S)

elseif (sv >= 19) && (sv <= 22);
    S = 'medium,A';
    disp(S)
elseif (sv >= 23) && (sv <= 26);
    S = 'large,A';
    disp(S)
elseif (sv >= 27) && (sv <= 35);
    S = 'jumbo,A';
    disp(S)
elseif (sv >= 50) && (sv <= 80);
    S = 'small,B';
    disp(S)
elseif (sv >= 90) && (sv <= 110);
    S = 'medium,B';
    disp(S)
```

Figure 3.7 (a): GUI push button code for run result on display.

```
elseif (sv >= 120) && (sv <= 140);
    S = 'large,B';
    disp(S)
elseif (sv >= 150) && (sv <= 170);
    S = 'jumbo,B';
    disp(S)
else
    S = 'Sorry, no that kind of shrimp';
    disp(S)
end

sp_v = str2num(get(handles.edit3,'string'));
if (sp_v >= 12) && (sp_v <= 35);
    s='A';
    disp(s)
elseif (sv >= 50) && (sv <= 170);
    s = 'B';
    disp(s)
else
    s = 'Sorry, no that kind of shrimp';
    disp(s)
end

set(handles.edit4,'string',s);
set(handles.edit4,'string',S);
```

Figure 3.7 (b): GUI push button code for run result on display.

Figure 3.7 (a) and 3.7 (b) are the codes to program the push button in Figure 3.6 so that when the push button is clicked, it will run certain function based on what it is programmed to do. First, to edit the code of the display text box, right-click on the display push button then click on “View Callbacks” and click on “Callback”. By doing so, it will direct us to MATLAB editor for adding some commands to the push button.

In this project, the function of the push button is to run the display of fuzzy logic results and Simulink flow. When clicked, the push button will get data from the workspace where the fuzzy logic data run by Simulink is placed. The result of the Simulink fuzzy logic will be displayed in the text box in numerical form. The results of the fuzzy logic will be link to GUI for user to view and be recorded.

3. Simulink input value from weight sensor.

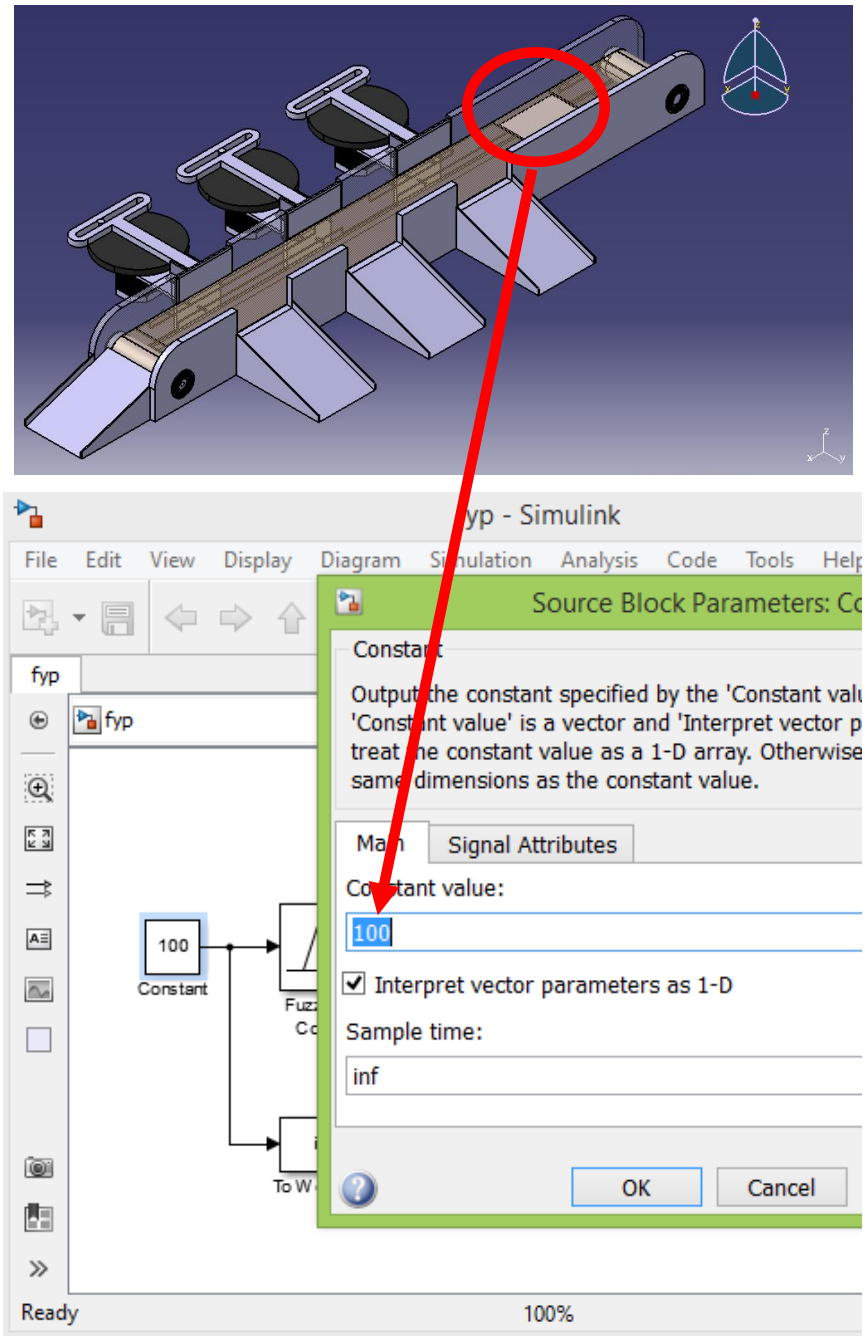


Figure 3.8: Input from weight sensor.

According to the main objective of this project, it is to create a more accurate shrimp grading method using software coding. In this project, the software code is calibrated with weight sensor to grade shrimp by measuring the weight.

The Figure 3.8 above shows that the signal from weight sensor is sent to Simulink so that the trained fuzzy logic controller can read the weight of shrimp and grade shrimp according to its size and types.

4. Simulink and fuzzy logic

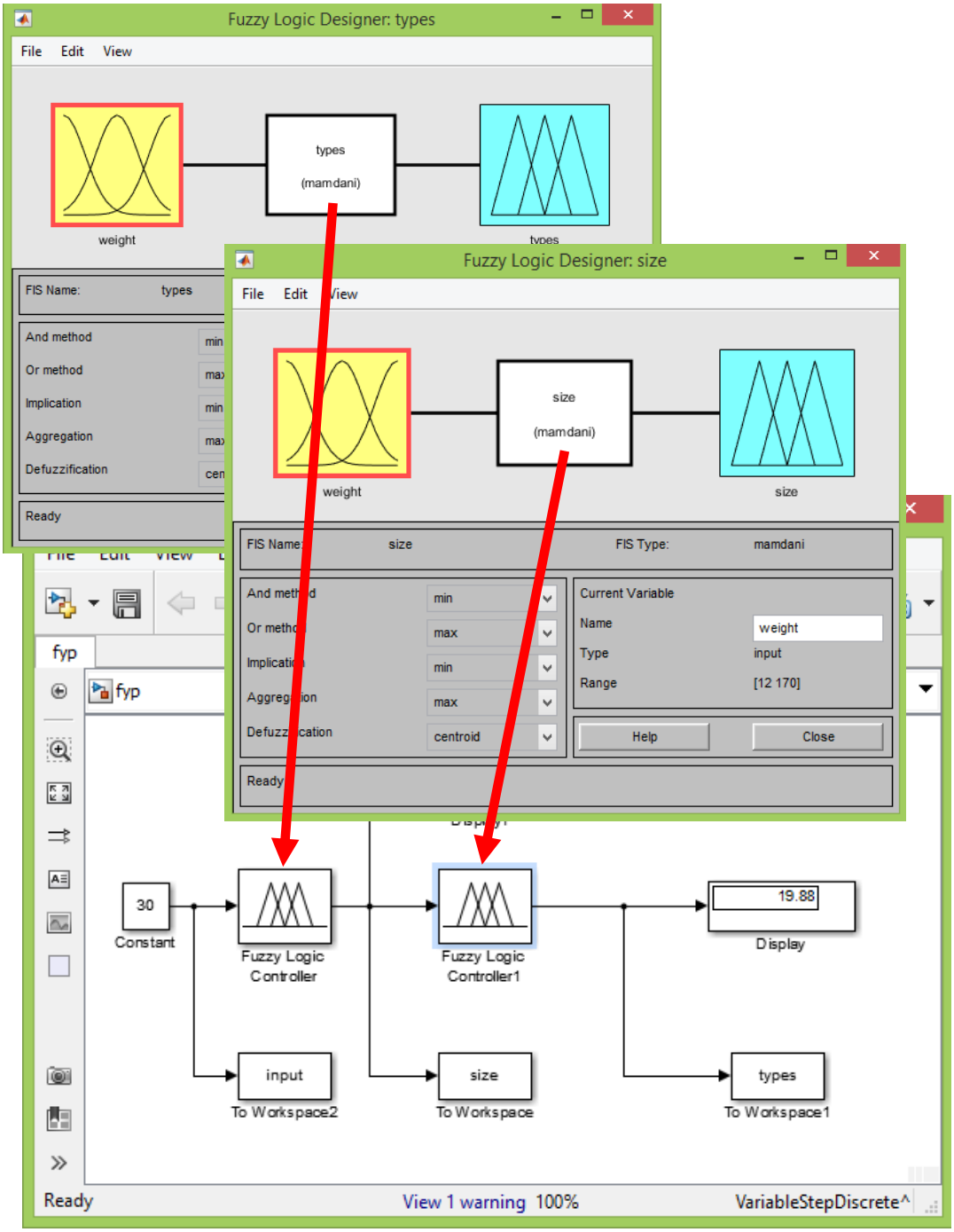


Figure 3.9: Implementing trained fuzzy logic into Simulink.

5. Simulink and GUI

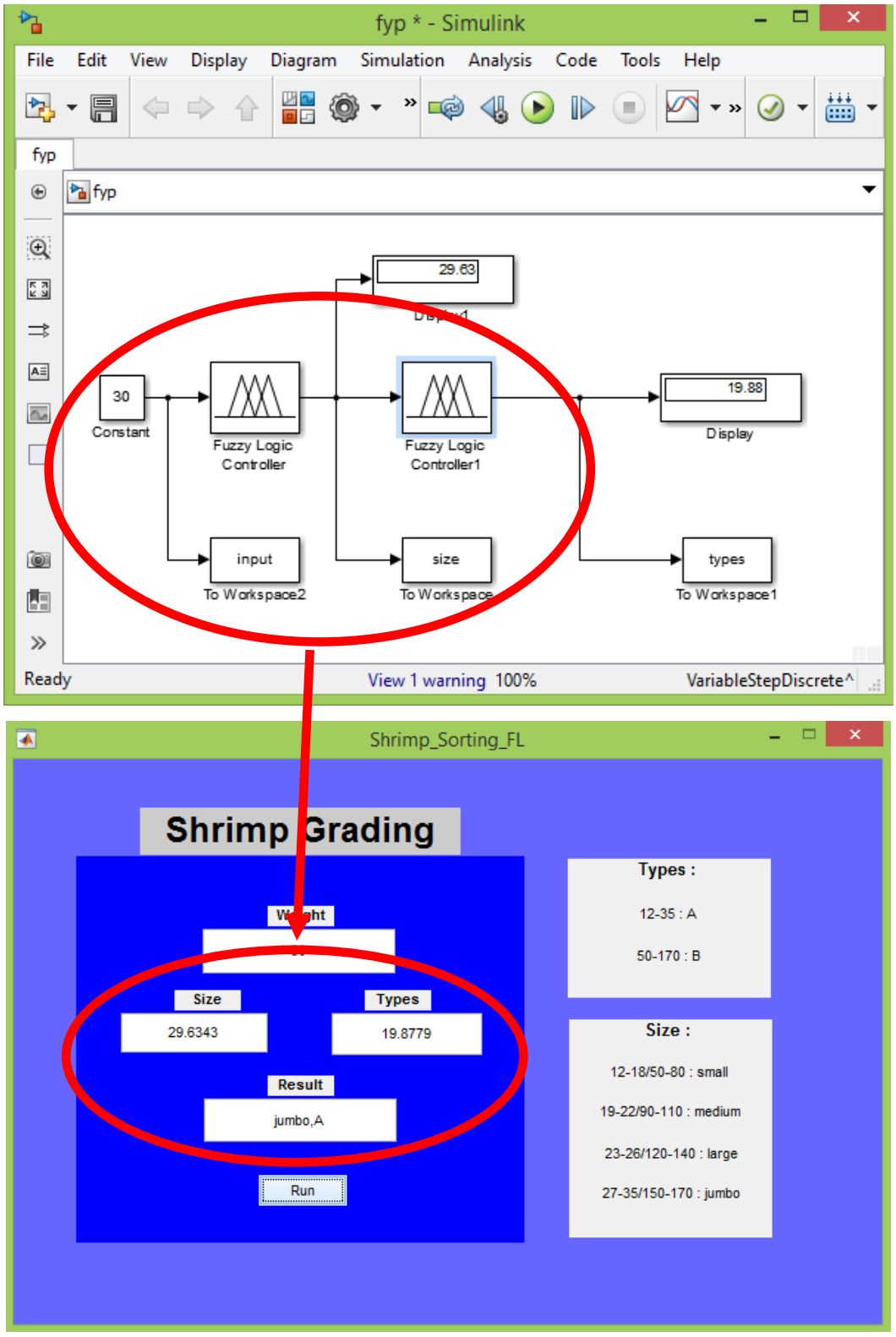


Figure 3.10: Results are displayed into GUI when run.

Fuzzy logic plays an important role in this project, referring Figure 3.9, the two trained fuzzy logic designer will have to implement into the fuzzy logic controller in Simulink in order to grade the shrimp.

Figure 3.10 shows that the fuzzy logic controller results are displaying in the GUI for operator to monitor the system on computer.

3.4.3 Modelling and designing fuzzy logic

1. Create “weight to size” converter in fuzzy logic toolbox.

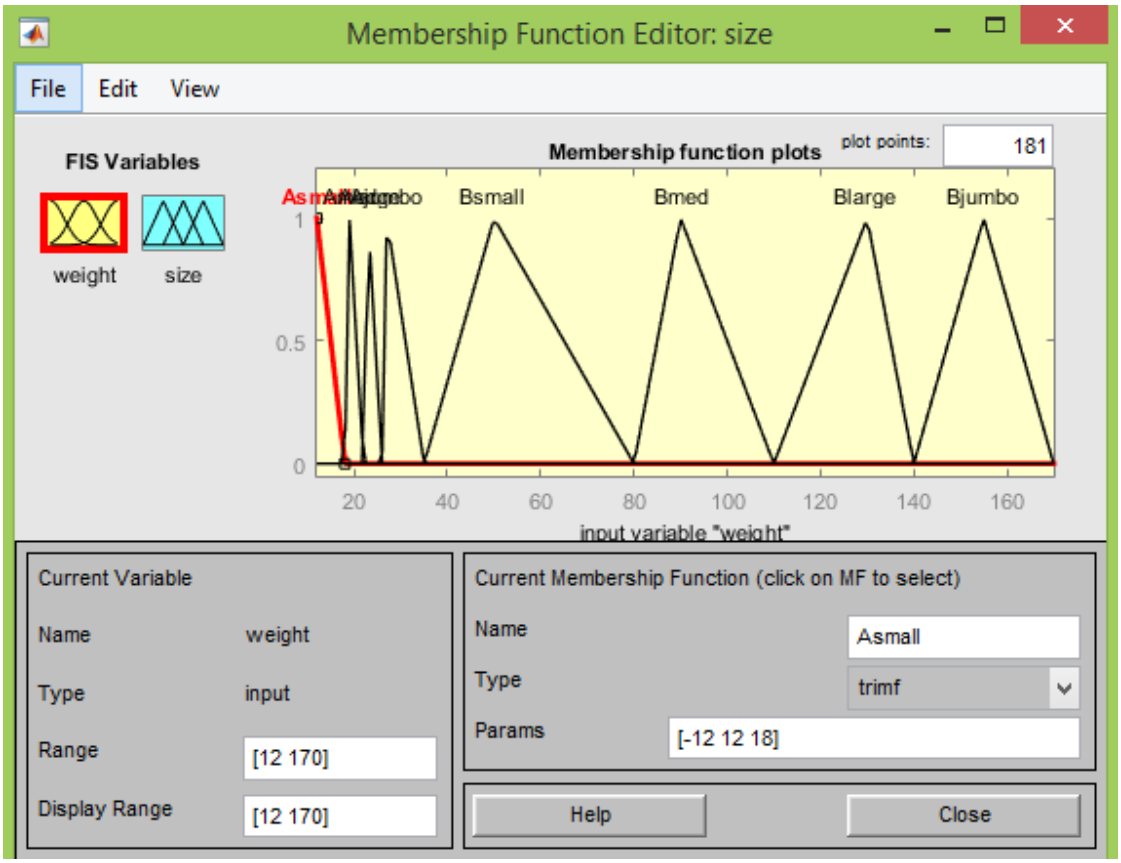


Figure 3.11: Membership function of weight input ranging from 12 to 170 grams.

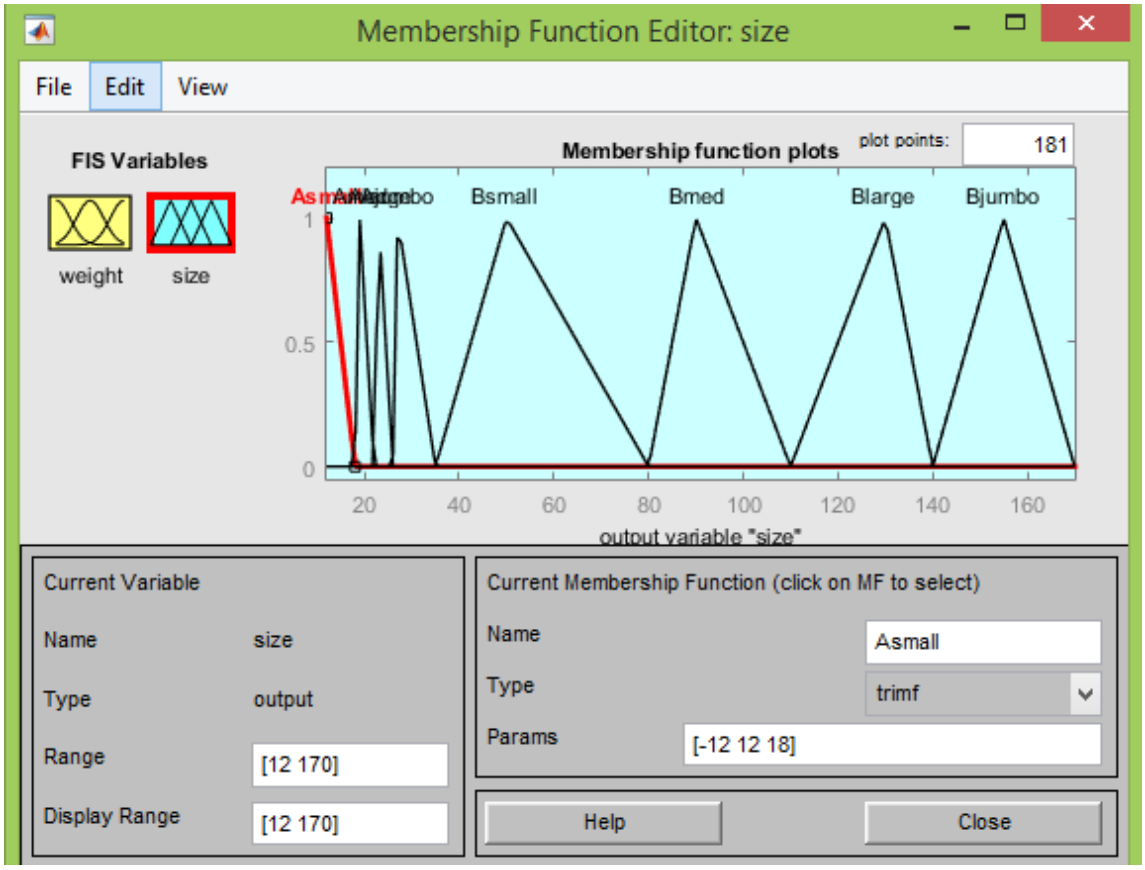


Figure 3.12: Membership function of output (size) ranging from 12 to 170 grams.

Figure 3.11 and 3.12 illustrate the membership function which allow the fuzzy logic to be trained according to the given parameters of size and types. Figure 3.12 shows the membership function of input weight variable from the weight sensor while Figure 3.12 shows the membership function of output size variable from fuzzy logic controller.

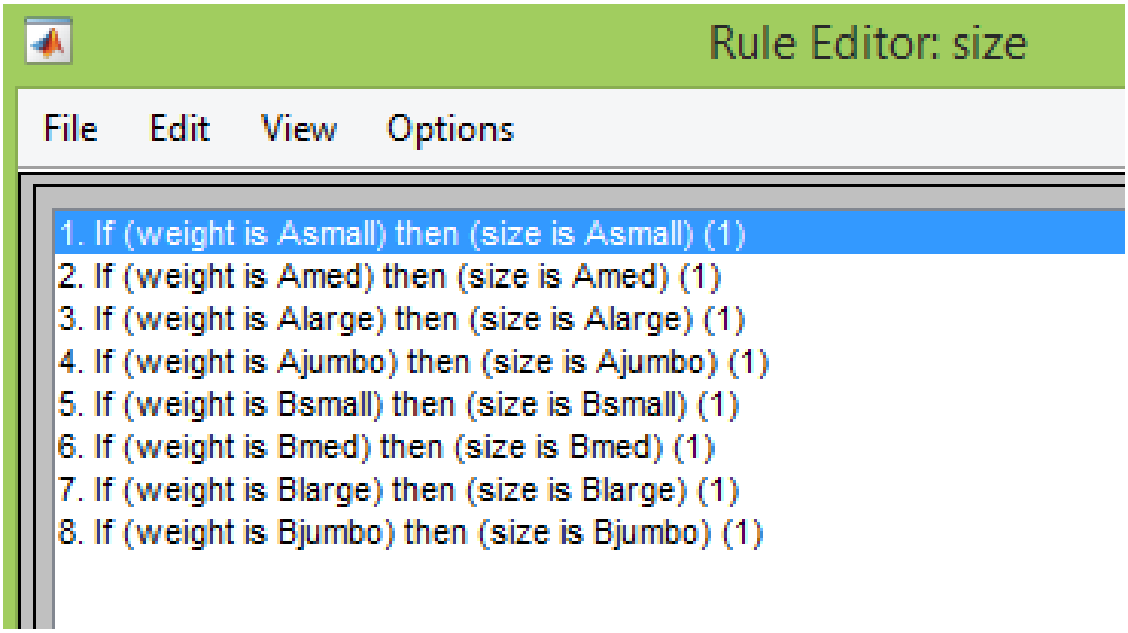


Figure 3.13: Rules of size controller (If "weight", then "size").

The fuzzy logic is not consider as trained if the rules of the fuzzy logic is not set. In order to train fuzzy logic, the rules must be set according to the input and output variables such example is shown in Figure 3.13 where from the first line means if the weight ranged between Asmall, and the size is consider as Asmall shrimp.

2. Create “weight to types” converter in fuzzy logic toolbox.

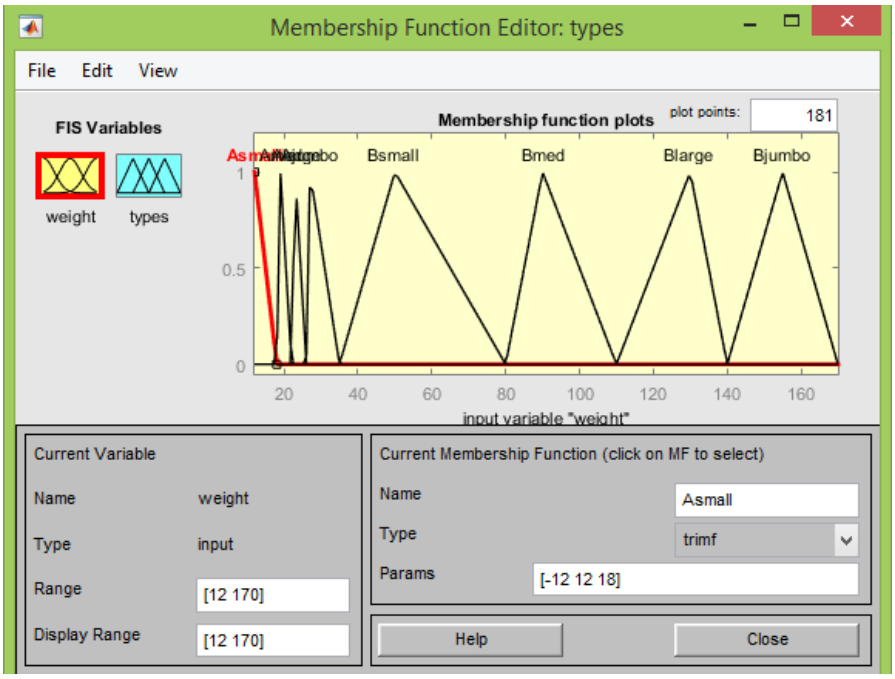


Figure 3.14: Membership function of weight input ranging from 12 to 170 grams.

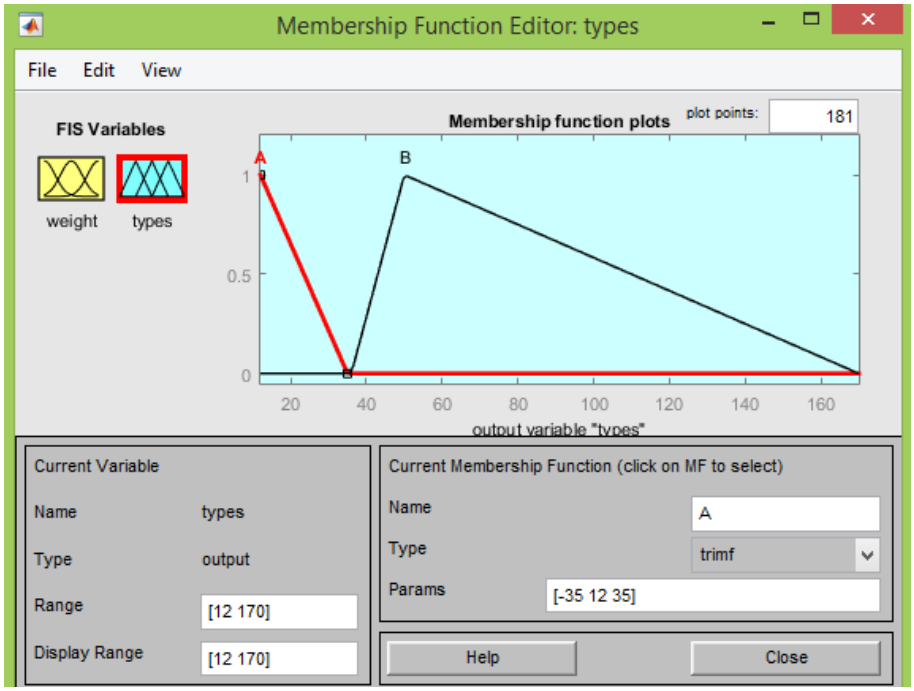


Figure 3.15: Membership function of output (types) ranging from 12 to 170 grams.

While in Figure 3.14 and 3.15 illustrate the membership function which allow the fuzzy logic to be trained according to the given parameters of weight and types. Figure 3.14 shows the membership function of input weight variable from the weight sensor while Figure 3.15 shows the membership function of output size variable from fuzzy logic controller.

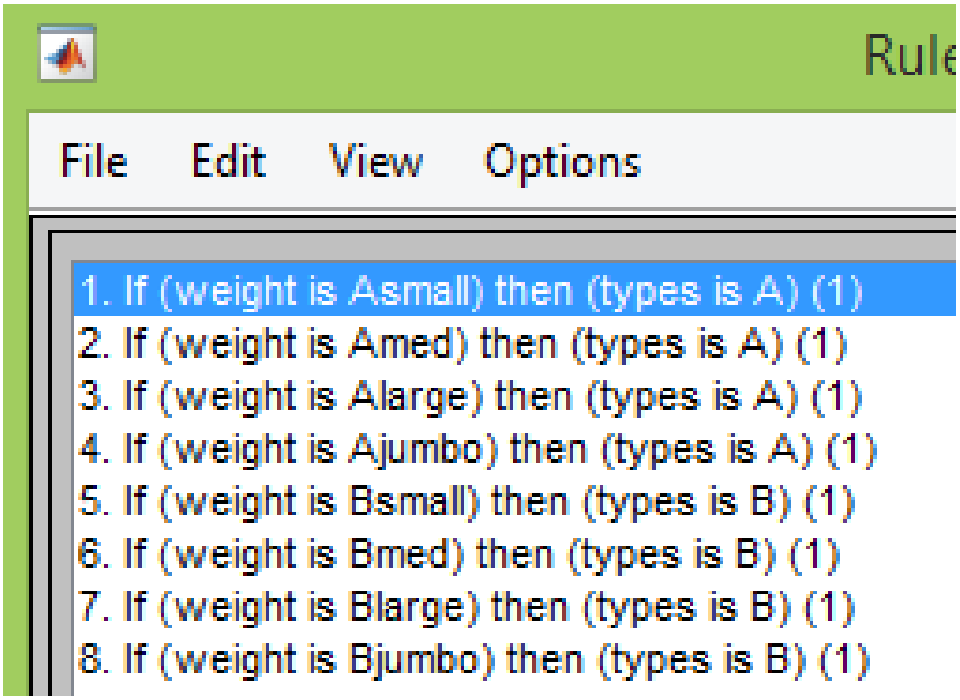


Figure 3.16: Rules of "types" controller (If "weight", then "types").

To train the fuzzy logic of types, rules must be set according to the input and output variables as shown in Figure 3.16 where from the first line means if the weight ranged between Asmall, then the shrimp is consider as A type shrimp.

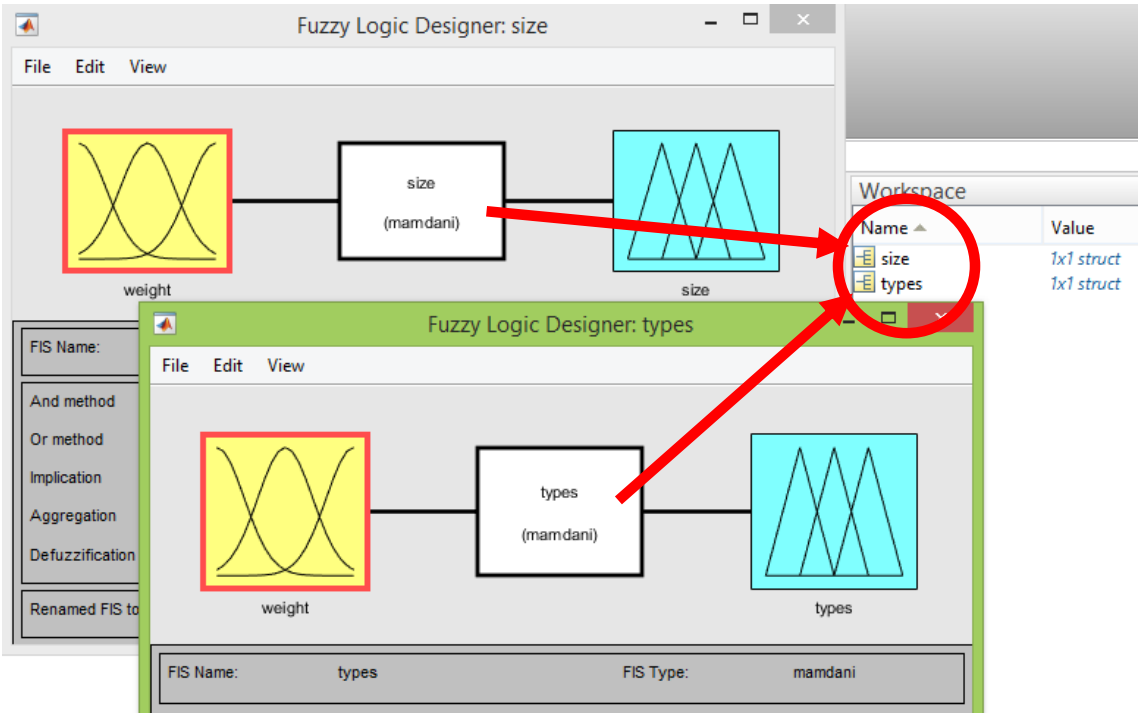


Figure 3.17: Trained fuzzy rules are exported to workspace.

In order to run the Simulink program, fuzzy logics have to be exported to MATLAB workspace because the Simulink can only get data from there. From figure 3.17, The Simulink will implement the fuzzy logic data into fuzzy logic controller in Simulink so that it could identify the size and types of shrimp with the trained fuzzy logic.

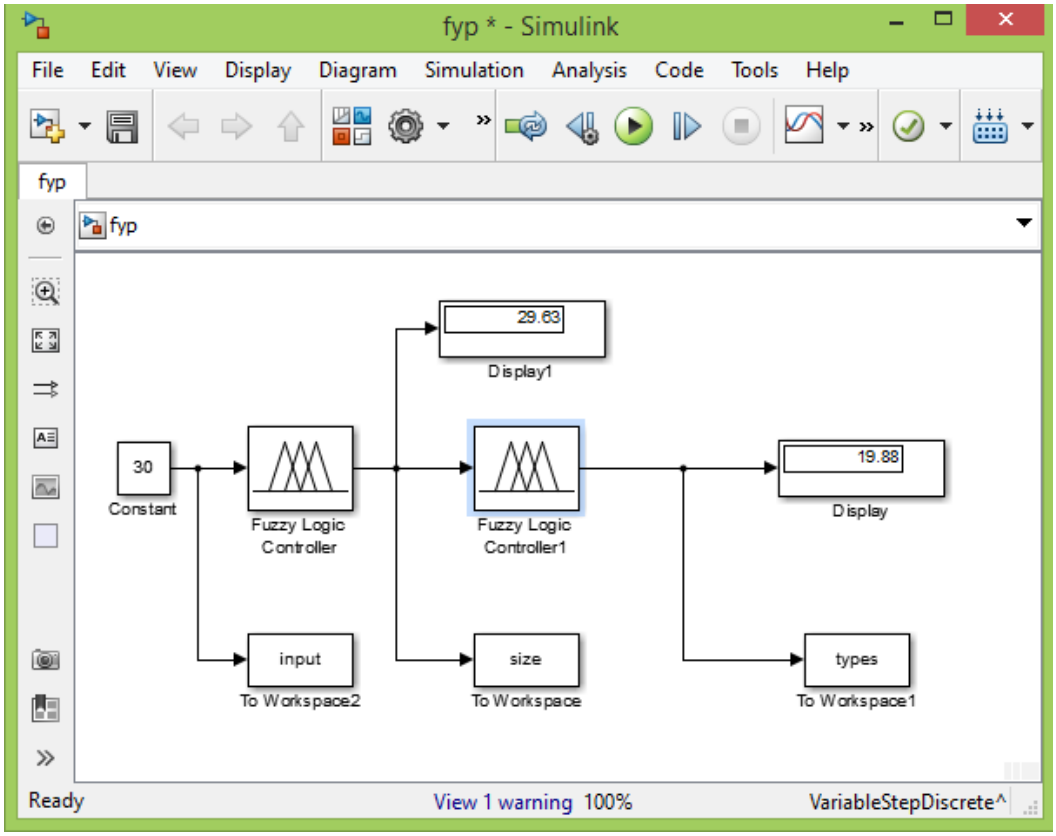


Figure 3.18: Simulink model to implement in shrimp grading system.

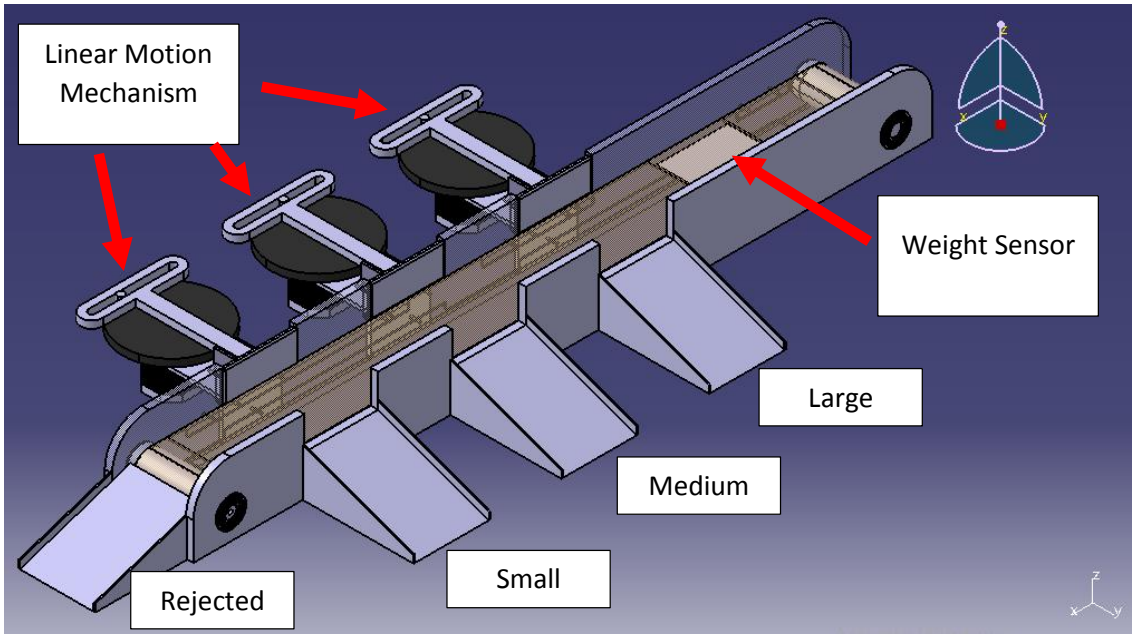


Figure 3.19: Draft of shrimp grading machine.

Figure 3.18 shows the overall block diagram designed in Simulink which will be implemented in the conveyor in Figure 3.19. From the drafted conveyor, the product, which is the shrimp will moves from top right end where the weight sensor is placed, then the signal will send to Simulink so that the size and types of the shrimp can be determined.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 INTRODUCTION

From this project, the results of fuzzy logic tools and rule viewer will be shown for a better understanding purposes. There will be the view of fuzzy logic rule and surface view in the shape of graph.

4.2 RESULTS

All figures below are the results obtained from MATLAB GUI software development. It is the overview of working principle of MATLAB fuzzy logic.

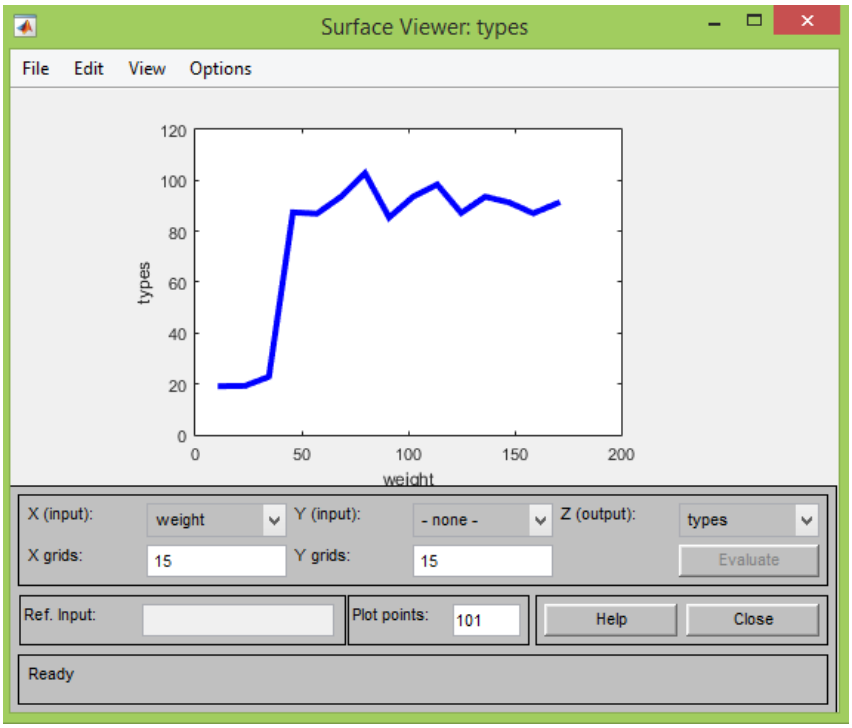


Figure 4.0: Weight vs type graph.

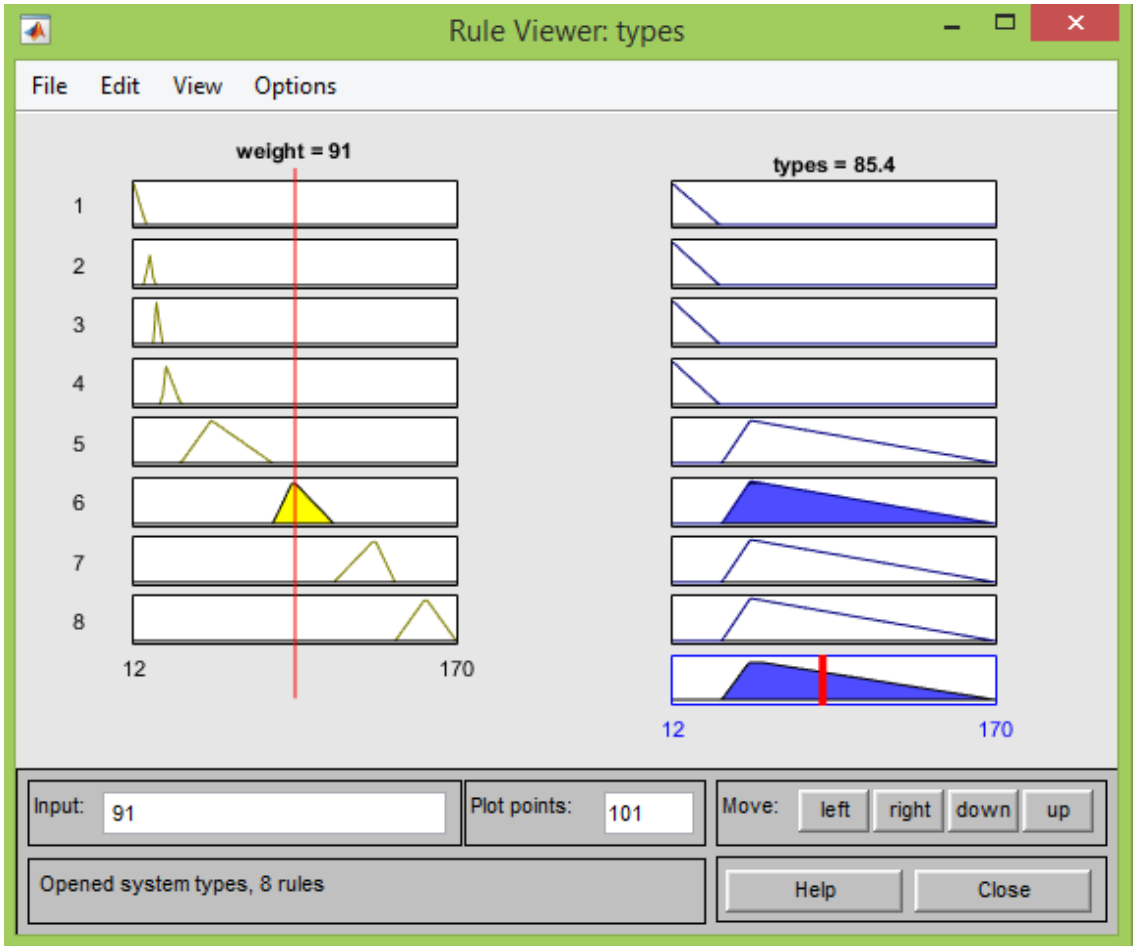


Figure 4.1: Weight vs type rule viewer.

According to Figure 4.0 and Figure 4.1, the results illustrated are the results obtained from the fuzzy rules where the relationship of weight and types are shown in graph and rule viewer. From Figure 4.1, when the weight of shrimp is 91 grams, the value for shrimp type is 85.4 where it falls into range of type B.

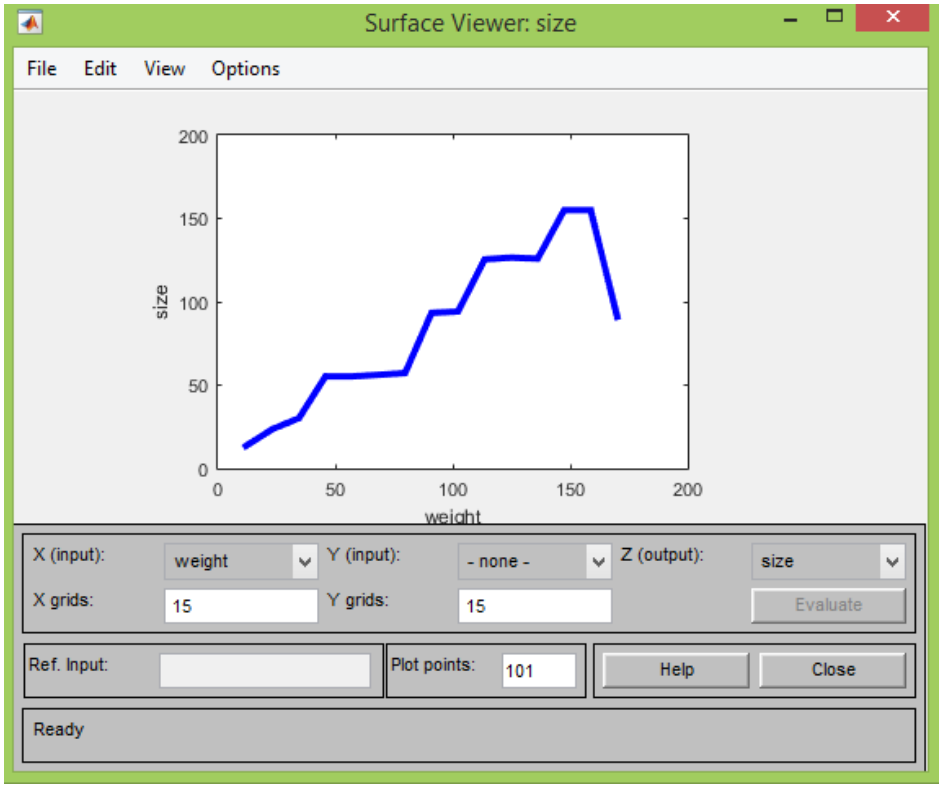


Figure 4.2: Weight vs size graph.

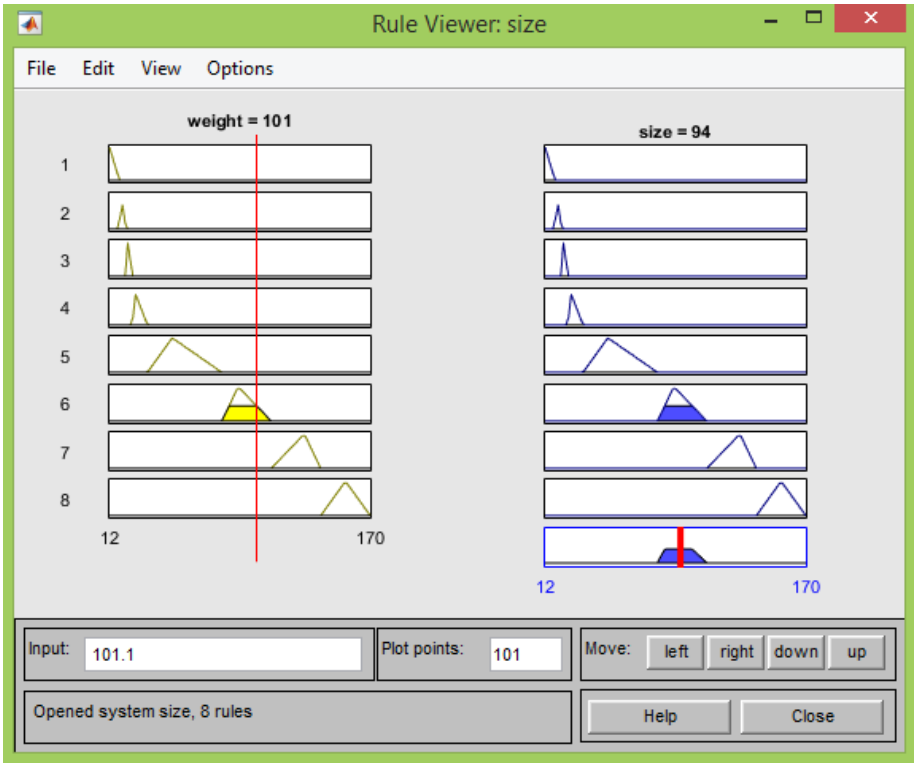


Figure 4.3: Weight vs size rule viewer.

As illustrated in Figure 4.2 and Figure 4.3, the results obtained from the fuzzy rules with relationship between weight and size are shown in graph and rule viewer. From Figure 4.3, when the weight of shrimp is 101 grams, the value for shrimp size is 94 where it falls into range of medium.

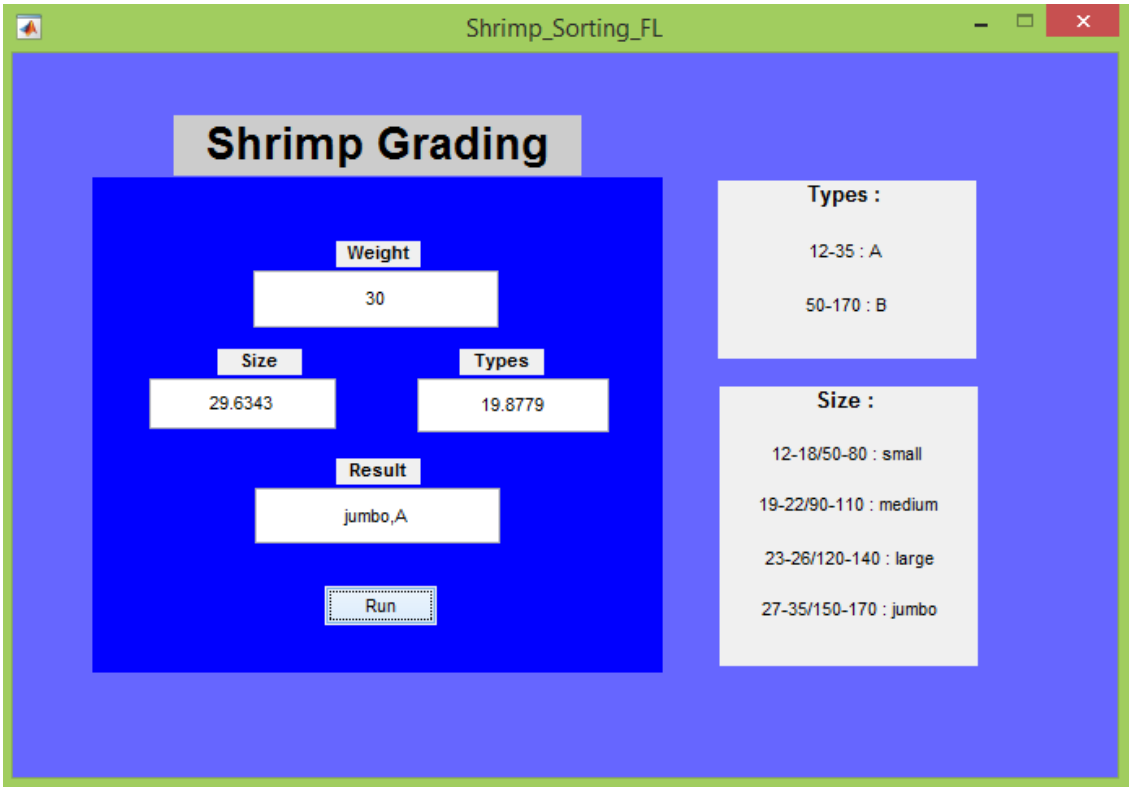


Figure 4.4: Shrimp grading GUI.

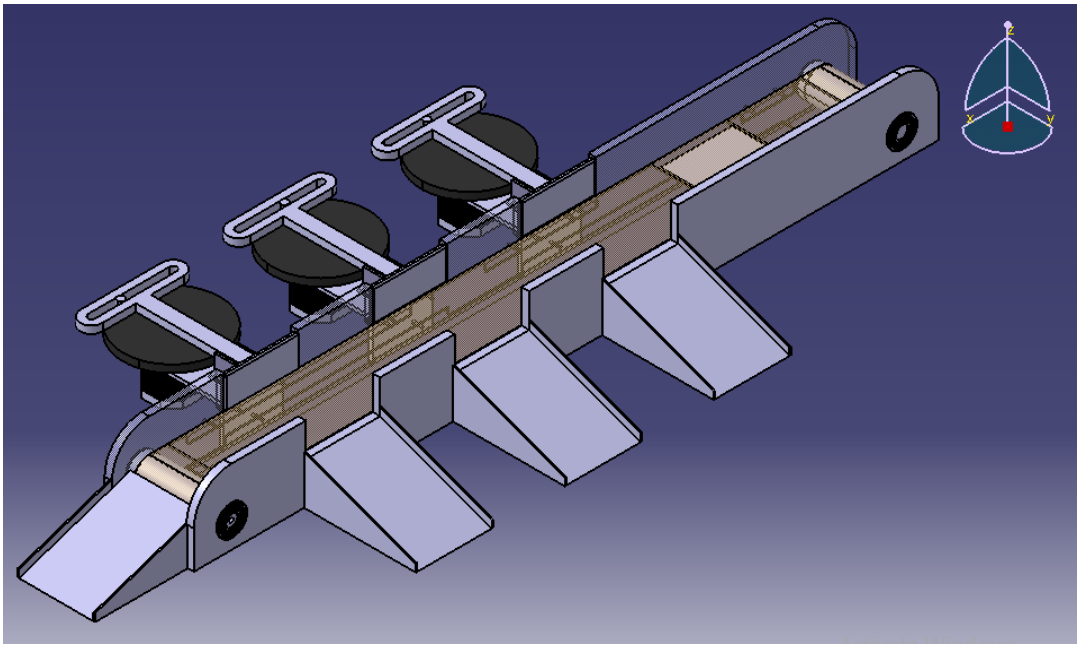


Figure 4.5: Draft of shrimp grading machine.

Above, Figure 4.5, is the draft of the shrimp grading machine where shrimp will be grade according to types and size using fuzzy logic. Operators can monitor the progress of the system through the created GUI in Figure 4.4 for a better view.

4.3 DISCUSSION

When doing the shrimp grading project, fuzzy logic deals with “degrees of truth” in shrimp grading which similar to human thinking that gives meaning to expressions like “big”, “medium” and “small” rather than absolute values of “0 and 1” or “big/small”. While on the other hand, Crisp logics are like computer software which understands only binary (0 and 1) functions like 3 category function in shrimp grading. The real world is complex and everything cannot have absolute values and follow a linear function.

However, in this project, there are still some limitations which stops the development of hardware of the system. The main limitation is the load-cell sensor, it is the main tool in this project and it is a problem acquiring it.

In the near future, this project can be improve for use in other industries such as fruits industries to determine the grade of the fruits or cement industries to determine weight of every bag of cement. Because everything on earth has weight, even from a huge heavy aircraft to a tiny little hair; they are still pulled by gravity which gives them weight. Therefore by using weight sensor, it will always be a better and cost effective method to grade an object into which ever preferred category according to various parameter.

CHAPTER 5

CONCLUSION

5.1 CONCLUSION

From this project, it is concluded that the project objective is achieved, software GUI code was successfully developed using MATLAB GUIDE along with Simulink to perform automated shrimp grading. The automated system has improved the accuracy of shrimp grading by 60% and reduces the cost for man power by 80%. The Table 5.0 below is generated from the GUI based on the weight of the shrimp.

Table 5.0: Results gained from GUI simulation.

			Result	
Weight [g]	Size	Type	Size [S,M,L,J]	Type [A,B]
15	13.95	19.86	Small	A
20	19.7	19.58	Medium	A
25	23.85	19.72	Large	A
30	29.63	19.88	Jumbo	A
35	91	85.42	Jumbo	A
40	56.31	86.62	Sorry, no that kind of shrimp	
50	54.98	86.17	Small	B
60	55.4	86.31	Small	B
80	90	85.42	Small	B
120	126.1	86.46	Large	B
150	155	85.33	Jumbo	B
170	91	85.42	Jumbo	B

Table 5.0 shows the overall results using the Simulink fuzzy logic to grade shrimp, where the weight of shrimp is picked randomly to test the effectiveness of the fuzzy logic system. If the input weight of shrimp is not in the range, it will be categorize as reject and the GUI will display “Sorry, no that kind of shrimp”.

5.2 RECOMMENDATION FOR FUTURE WORK

Some recomendation can be made for further improvement in the system and also for further development of software code in near future as there:

- Product for weight measuring can be set and change base on which ever products are applied
- Creates another GUI for showing the current progress of the shrimp grading in graph.
- Creates a standalone executing files for the GUI so that it is easier for the user to operate with just one click.
- Fuzzy logic rules can be modified for more advance measurment.

REFERENCES

This thesis is prepared based on the following references;

D.J. Lee, X. Xu, R.M. Lane, and P. Zhan (October 25-28, 2004), "Shape Analysis for an Automatic Oyster Grading System", SPIE Optics East,Two and Three-Dimensional Vision Systems for Inspection, Control, and Metrology II, vol. 5606-05, Philadelphia, PA,USA.

D.-J. Lee et al. (2008) Contour Matching for Fish Species Recognition and Migration Monitoring, Studies in Computational Intelligence (SCI) 122, 183–207.

Xiong Guangming, Lee Dah-Jye, Moon Kevin R, Lane Robert M. (September 2010) Shape similarity measure using turn angle cross-correlation for oyster quality evaluation. Journal of Food Engineering, v 100, n 1, p 178-186.

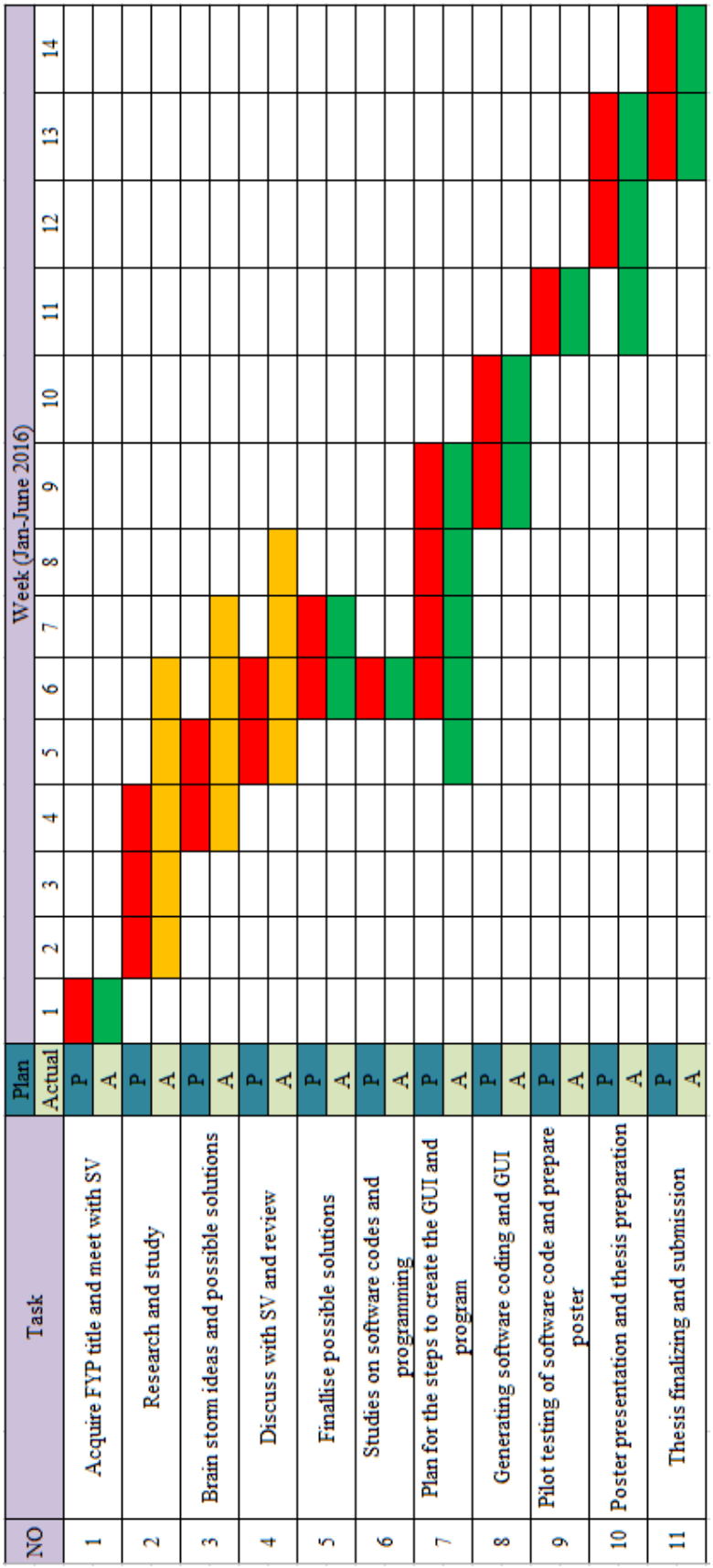
Nobuyuki Otsu. (1979) "A threshold selection method from graylevel histograms". IEEE Trans. Sys., Man., Cyber. 9: 62–66. doi:10.1109/TSMC.1979.4310076.

Latecki LJ, Lak amper R (2002), Application of planar shape comparison to object retrieval in image databases. Pattern Recognition 35:1529.

Lee DJ, Bates D, Dromey C, Xu X, Antani S (2003), An imaging system correlating lip shapes with tongue contact patterns for speech pathology research. In: Proc. 16th IEEE symposium on Computer-Based Medical Systems, pp 307313.

APPENDICES

Gantt Chart



```
function varargout = Shrimp_Sorting_FL(varargin)
% SHRIMP_SORTING_FL M-file for Shrimp_Sorting_FL.fig
%      SHRIMP_SORTING_FL, by itself, creates a new SHRIMP_SORTING_FL
or raises the existing
%      singleton*.
%
%      H = SHRIMP_SORTING_FL returns the handle to a new
SHRIMP_SORTING_FL or the handle to
%      the existing singleton*.
%
%      SHRIMP_SORTING_FL('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in SHRIMP_SORTING_FL.M with the given
input arguments.
%
%      SHRIMP_SORTING_FL('Property','Value',...) creates a new
SHRIMP_SORTING_FL or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before Shrimp_Sorting_FL_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Shrimp_Sorting_FL_OpeningFcn
via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Shrimp_Sorting_FL

% Last Modified by GUIDE v2.5 07-May-2016 12:46:31

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@Shrimp_Sorting_FL_OpeningFcn, ...
                   'gui_OutputFcn',  @Shrimp_Sorting_FL_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Shrimp_Sorting_FL is made visible.
```

```
function Shrimp_Sorting_FL_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to Figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Shrimp_Sorting_FL (see
VARARGIN)

% Choose default command line output for Shrimp_Sorting_FL
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Shrimp_Sorting_FL wait for user response (see UIRESUME)
% uiwait(handles.Figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Shrimp_Sorting_FL_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to Figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
of slider


% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

```
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double




% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double




% sv = str2num(get(handles.edit3,'string'));
% if (sv >= 12) && (sv <= 18);
%     S='A,small';
%     disp(S)
%
% elseif (sv >= 19) && (sv <= 22);
%     S = 'medium,A';
%     disp(S)
% elseif (sv >= 23) && (sv <= 26);
%     S = 'large,A';
%     disp(S)
% elseif (sv >= 27) && (sv <= 35);
%     S = 'jumbo,A';
%     disp(S)
% elseif (sv >= 50) && (sv <= 80);
%     S = 'small,B';
%     disp(S)
% elseif (sv >= 90) && (sv <= 110);
%     S = 'medium,B';
%     disp(S)
% elseif (sv >= 120) && (sv <= 140);
%     S = 'large,B';
%     disp(S)
% elseif (sv >= 150) && (sv <= 170);
%     S = 'jumbo,B';
%     disp(S)
% else
%     S = 'Sorry, no that kind of shrimp';
%     disp(S)
% end
%
% sp_v = str2num(get(handles.edit3,'string'));
% if (sp_v >= 12) && (sp_v <= 35);
%     s='A';
%     disp(s)
% elseif (sv >= 50) && (sv <= 170);
%     s = 'B';
%     disp(s)
% else
%     s = 'Sorry, no that kind of shrimp';
%     disp(s)
% end
%
% set(handles.edit4,'string',s);
% set(handles.edit4,'string',S);

% a=evalin('base','size.signals.values');
```

```
% set(handles.edit1,'string',num2str(a));
% b=evalin('base','types.signals.values');
% set(handles.edit2,'string',num2str(b));



% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4
as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
```

```
a=evalin('base','input.signals.values');
set(handles.edit3,'string',num2str(a));
b=evalin('base','size.signals.values');
set(handles.edit1,'string',num2str(b));
c=evalin('base','types.signals.values');
set(handles.edit2,'string',num2str(c));

sv = str2num(get(handles.edit3,'string'));
if (sv >= 12) && (sv <= 18);
    S='small,A';
    disp(S)

elseif (sv >= 19) && (sv <= 22);
    S = 'medium,A';
    disp(S)
elseif (sv >= 23) && (sv <= 26);
    S = 'large,A';
    disp(S)
elseif (sv >= 27) && (sv <= 35);
    S = 'jumbo,A';
    disp(S)
elseif (sv >= 50) && (sv <= 80);
    S = 'small,B';
    disp(S)
elseif (sv >= 90) && (sv <= 110);
    S = 'medium,B';
    disp(S)
elseif (sv >= 120) && (sv <= 140);
    S = 'large,B';
    disp(S)
elseif (sv >= 150) && (sv <= 170);
    S = 'jumbo,B';
    disp(S)
else
    S = 'Sorry, no that kind of shrimp';
    disp(S)
end

sp_v = str2num(get(handles.edit3,'string'));
if (sp_v >= 12) && (sp_v <= 35);
    s='A';
    disp(s)
elseif (sv >= 50) && (sv <= 170);
    s = 'B';
    disp(s)
else
    s = 'Sorry, no that kind of shrimp';
    disp(s)
end

set(handles.edit4,'string',s);
set(handles.edit4,'string',S);

% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --- Executes on key press with focus on pushbutton2 and none of its
controls.
function pushbutton2_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  structure with the following fields (see
MATLAB.UI.CONTROL.UICONTROL)
%   Key: name of the key that was pressed, in lower case
%   Character: character interpretation of the key(s) that was pressed
%   Modifier: name(s) of the modifier key(s) (i.e., control, shift)
pressed
% handles    structure with handles and user data (see GUIDATA)


% --- Executes when user attempts to close Figure1.
function Figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to Figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: delete(hObject) closes the Figure
delete(hObject);
```