INSTAWEIGHT

Atif Mehmood
Sayyda Sahar Fatima
Mohammad Ateeb Ahmed

HABIB UNIVERSITY
KARACHI, PAKISTAN
2020

INSTAWEIGHT

The Kaavish Report
presented to the academic faculty

by

Atif Mehmood

Sayyda Sahar Fatima

Mohammad Ateeb Ahmed

in partial fulfillment of the requirements for
Bachelor of Science
Computer Science
Dhanani School of Science and Engineering

Habib University
Spring 2020

InstaWeight

This Kaavish project was supervised by:

Dr. Akhlaque Ahmed
Faculty of Computer Science
Habib University

Approved by the Faculty of Computer Science on _____.

"Your sickness is from you, but you do not perceive it and your remedy is within you, but you do not sense it. You presume you are a small entity, but within you is enfolded the entire Universe. You are indeed the evident book, by whose alphabets the hidden becomes manifest. Therefore, you have no need to look beyond yourself. What you seek is within you, if only you reflect." (Imam Ali A.S)

This project is dedicated to our beloved parents, respectable professors and supportive friends We pray to Allah Almighty for their health and pray to keep them saved from this Covid-19 and all other diseases.

Acknowledgements

By the grace of Allah Almighty this journey comes to its end. It was not possible without the sincere, cooperative, and brilliant people around us. We would like to give credit of our success to several individuals and of course to "Habib University". Firstly, we would like to express our sincere thanks to our supervisor, professor "Dr Akhlaque Ahmed" who supported us through out the project. His valuable advises, insightful comments, helpful information, and unceasing ideas, even in this pandemic and uncertain situation of COVID-19, helped us to successfully complete our project. It wasn't possible without his relentless help.

We also wish to express our sincere thanks to all our professors and"Kaavish Working Group", who guided us through out our journey, keep us updated in this worst time and answered every query on time. Furthermore, we would like appreciate the guidance given by other supervisors as well as the panel (Dr. Taj, Dr. Abdul Samad, Dr Umair Azfar and Miss Nadia Nasir) especially in our project presentation, who gave valuable comments and advises that improved our presentation skills.

We would also like to say thanks to Folio3 Software Inc, who gave their precious time to serve us and supervised us through out the project. We are really thankful to them to provide us Real Sense Depth Camera to capture depth images and dataset.

We are also thankful to our parents who support us and stand by us through the peaks and valleys of life.

Last but not least, we are also thankful to all our sincere friends who helped us to cheer up the mood and gave us moral support in bad times and special thanks to Syed Muhammad Hasan who helped us in dataset collection and annotation, Aniqa Ahsan and Ateeb Ahmed who also helped us in dataset annotation.

Abstract

The project, “instaWeight” aims to automate the process of weight estimation using machine learning and deep learning techniques. It makes the whole process of weight estimation and then tracking the history of cattle’s weight in a large farm or in live stock business automatic. It calculates the cattle’s weight and then maintain the history of it through admin panel, which keeps all the records saved for years.
This report gives the complete insight of the project and enlightens each idea in detail. It first describes the problem in the manual way of weight estimation then provides the solution, proposed by “Team - instaWeight”. It also gives a detailed literature review to get the idea of other similar work done in the same domain. The report also have separate sections on software requirement specification and software design specification, which provides the insight of project from requirement gathering to software’s structure and design. In the last section of the report, the experiments and their results are mentioned that have been done through out the project.

Contents

List of Figures

CHAPTER 1
INTRODUCTION

1.1 Overview

The livestock industry has played a vital role in world's economy. In colonial times animals were allowed to roam and hunt in the forest. This industry was dependent on free grazing. The livestock industry began to transform after the Civil War (1917 – 1922). This was the time when the advancement in technology began. The technology started to change agriculture , free range evolved into a more stable farming environment and different agriculture state, societies were formed [1].

The growth rate of livestock production is significantly more than the agriculture production in most developing countries. According to experts, this trend is likely to continue over the next 20 years and estimated at 4.5 percent per annum. In livestock business, to produce more intensive products, the feed is one of the major cost components. Therefore, It is needed to be carefully assessed with estimated feed requirements in mind [2]. The feed can be optimized based on the weight of the animal.

1.2 Problem Statement

The profit in livestock business depends on the weight and body size of the cattle. Weight is the most important factor in Livestock business. It is very effective in assessing the reproductive efficiency and growth performance of an animal. Weight is also used in measuring the correct dose of therapeutic pharmaceutical to treat diseases that affect cattle. The correct amount of feed can also be determined based on the weight to avoid underfeeding or overfeeding [3]. But Keeping track of the weight on a daily basis is the most tedious task and requires a lot of manhandling and is time consuming.

1.3 Proposed Solution

According to the work done by Chintan Bhatt et. al [4], image processing and the algorithms of machine learning can be used to estimate the weight of the cattle remotely without even touching them. It has been observed in research study that

by implementing this method 73 - 89% accuracy can be achieved [4]. But most of the work is dependent on additional cameras or hardware to achieve the goal.

In our project we want to achieve this goal only using a mobile phone's camera. And aim to develop a system which will be able to keep track of daily weight of Cattles. In our project, image processing, image segmentation and machine learning algorithms would be used to achieve the high accuracy for the task of estimating the weights of the cattle [5]. There are many ways to implement these techniques to get the task done but all those methods have their own advantages as well as limitations. Also, there is a difference of accuracy in between them. We will implement the algorithms in a way that high accuracy can be achieved.

Over the period of one year, we will be researching and developing a model, which will be able to predict animals weight.

Our project has three main components, the core and web client.

1.3.1 Core Component:

The core component will be responsible to detect the features of the animal from the image to estimate/predict the weight of the animal. This core component will be developed using machine learning and image processing algorithms.

1.3.2 Web Client:

Web client will consist of admin panel. The admin panel would display the overall growth of cattle, in a last past years. It will show the overall growth as well as growth of animals at individual level. All basic information of each animal can be seen by admin panel.

1.4 Intended User

Our InstaWeight is a special purpose system and therefore will be used only by ranchers and cattle farmers. The mobile application would be used by cattle farmer, while the admin panel would be used by the cattle farm's owner who will keep track of his animal growth to run his business.

1.5 Key Challenges

This section mentions the key challenges that we foresee in this project and possible ways to address them.

1.5.1 Availibilty of Cattle's Dataset and Weight Labels

The biggest challenge in our project is the availability of data related to our problem. Although we were able to collect RGB images of cattle by visiting some farms but we were unable to get weight labels. and after covid 19 our data collection process was impacted badly.

CHAPTER 2
LITERATURE REVIEW

2.1 Introduction:

The purpose of the literature survey is to get sufficient knowledge and to get familiar with the approaches, methods and strategies that are used before to solve the same problem on which we are going to work. In this paper, all approaches that we reviewed during our survey would be discussed. Moreover it will explain the limitations with the approaches and then it will answer the question 'what is our strategy to solve this problem and why we choose it.
In order to estimate the weight of the cattle by using the techniques of deep learning, computer vision and image processing, all research papers focuses on some of the common methods of pre-processing that can be used to segment out the region from the image. Once the image is segmented and separated from its background then some other advance techniques could be used for further calculation. All research papers that we read during our literature survey used different approaches to calculate the weight of the cattle. These approaches are discussed below in detail.

2.2 Literature Survey Review:

During the survey, we reviewed the research papers which are summarized below separately.

A measuring weight model of Timor's Beef Cattle based on image

In this paper [3], the overall approach that is used to estimate the weight of cattle is by calculating the heart girth and body length of cattle. The overall structure of the approach that is used in this research paper is explained below by the block diagram.

The paper didn't say much about the procedure that is used to obtain the results but it explains the steps that they followed to get the results. According to their methodology they followed the following steps that are mentioned below.

1. Obtained or captured a digital image.

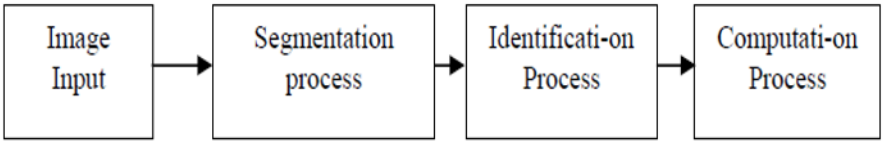


Figure 2.1: Diagram of image processing procedures

2. Performed image processing operations on image data.

3. Analyze and interpreted the image and use the results to get the body length, chest circumference and then use the standard formula to calculate the weight of cattle.

The overall structure of the image processing method to get the weight of beef cattle can be demonstrated by the following block diagram [3].

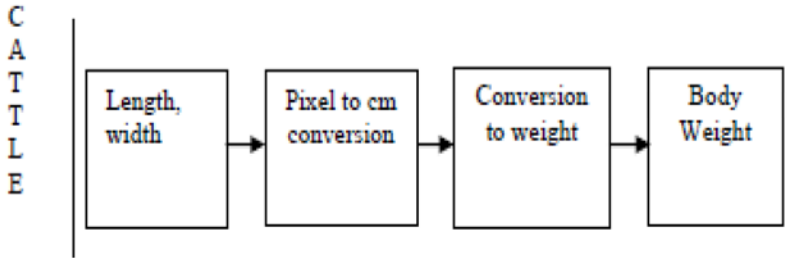


Figure 2.2: Diagram of general process to Weighting the Cattle

This paper cites some other research papers and explains the relationship between the body length and chest circumference. It gives reference to some formulas that can be used to determine the body weight of cattle. The author cites this research paper and according to Soeprapto [2], the body weight of cattle can be calculated by the formula:

$$BW = \frac{BL + (CC)^2}{10840} \tag{2.1}$$

Where, BW is the *body weight* in kg
BL is the *body length* in cm

CC is *chest circumference* in cm

It cites another paper and says that according Schoorl written by Siregar [1], weight can be obtained by getting the physical dimensions of cattle.

$$BW = \frac{(WoC + 22)^2}{100} \tag{2.2}$$

Where, BW is the *body weight* in kg WoC is the *width of chest* in cm

The author used the standardized formula based on the variables namely body length and chest width of cattle and obtained the results. He, in his paper claims that the results that were obtained by using this approach were not significantly different from the results of weighing using mechanical instrument [3].

Weight Estimation Using Image Analysis and Statistical Modelling: A Preliminary Study

The research was conducted previously on this topic and showed that there exists a correlation between the weight of the pig and physical features such as pig's length or two dimensional area when viewed from above. This paper tries to find the correlation between the body weight of cattle and its physical features.

In the paper [4], the author automated the weight estimation system by using image processing techniques. The steps that are used in this approach as a part of image processing are following.

1. Object Detection
2. Segmentation
3. Filtering
4. Feature Extraction

Object Detection

For the object detection, the author mentioned the calibration process that they used in their project. For calibration they set a fixed threshold. This threshold was determined by an offline calibration step. To complete this step, they took two images from a fixed camera rig; one image containing pig and other without pig (just of the background). These two images were used to locate the largest region of

significant difference. Then they compared both images and based on the comparison results, a binary image was formed. Then a threshold was set based on brightness values in the binary image. They author suggests a method to detect the pig in his paper. According to this paper [4], Pig detection could then be performed by sampling an image and comparing the measured brightness from the pixels in the defined detection region with the detection threshold. If a large percentage of pixels passed the test, then a pig was declared to be present and the image saved for further analysis.

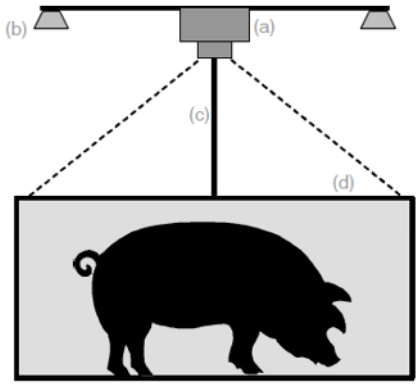


Figure 2.3: Camera and lighting setup

Segmentation

In the paper, fixed threshold method was used for segmentation. The author says that since the method was not entirely effective so in order to get the better results post filtering method was used to separate the object from the rest of the image.

Filtering

Once the image is segmented, the post processing is required to clean up the segmented image. The post-processing involves two forms of filtering; median filtering and morphological filtering. A median filter is able to correct isolated segmentation errors, thereby "filling in" missed detections without causing great distortion to the image. It was also used to prevent the next process, image opening, from having a detrimental effect. After applying median filter image opening process was applied.

Feature Extraction

After doing segmentation, the features were extracted to estimate the weight of cattle. The features that were employed in this study were area, length and spine length. The area of the object in pixels is trivially defined to be the sum of the binary pixel values over the whole image. This assumes that by this stage there will be only

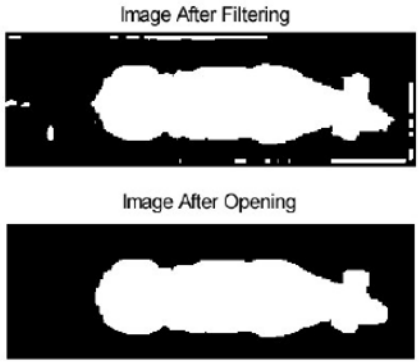


Figure 2.4: Image filtering and opening performed on the image of pig

one contiguous region in the image. The spine length was computed by isolating the largest branch in the skeleton and counting the number of pixels. The main branch was located by determining the location of junctions via a zero-crossing method [5].

Figure 2.5: Skeletonization performed on a segmented image

The length feature refers to the distance from the center of the pig's neck to its tail. The position of the pig's neck proved to be difficult to detect due to the unknown orientation of the pig's head. It was decided to estimate the pig's neck position as the point at which the main segment (referred to as the spine above) of the pig's skeleton terminated. After skeletonization as above, the two endpoints of the spine were found [4].
The "head" endpoint was deemed to be the one which gave rise to the most sub-branches, as the head was more topologically complex than the rump and usually produced a more structured sub-skeleton. The tail was assumed to be the most extreme pixel of the segmented image at the opposite end from the head. With these two data points the Euclidean distance could then be found, giving a close

approximation of the length of the pig. Euclidean distance was appropriate since the pixel dimension ratio was one-to one [4].
According to the author, the best results were obtained when using the neck-to-tail distance versus log weight (correlation coefficient 0.5640) and area versus weight (correlation coefficient 0.5295) to determine the weight. These correlation coefficients were both significant, having a probability of occurrence less than 1%. The resulting linear weight prediction formulas achieve an average absolute error just under 5% [4]. The author says, it confirms the claim that there exists a correlation between physical features and body weight.

Beef Cattle Weight Determine By Using Digital Image Processing

The body length, chest circumstance, height and width of the animal could be estimated.

Image Preprocessing: The paper first did some image preprocessing depending on the image of cattle to make it segmented to do further calculation. Once the image is segmented it is divided into segmentation parts as shown in the figure below. [6]

Suggested Segmentation Division:

1. Round and rear shank
2. Sirloin, short loin, tenderloin, top sirloin, bottom sirloin, flank
3. Rib and short plate
4. Chunk, brisket and front shank

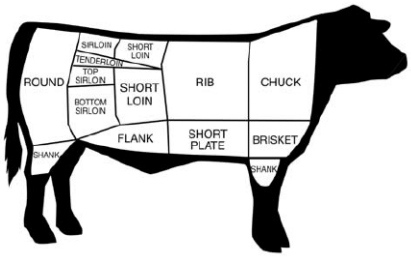


Figure 2.6: Types of Beef

Methodology:

1. Localized Region Based Active Contour

   Segmentation method that minimizes curves in the segmentation process.

$$E_{snake} = E_{internal} + E_{external} + E_{constraint} \tag{2.3}$$

   Make an initial contour surrounding the object, then with the energy of an object image (External) will cause the curve shrink and follow the pattern of the object. The curve can be moved closer towards the object and adjust the shape of the object because of the energy on the curve (Internal). Use active contour without edges which does not depend on the value of the gradient image as a condition to stop the contour changes. This model uses the average energy in the region. [6]

Figure 2.7: Objects covered by the curve cantour

2. Linear Regression

   Fitting a linear equation to observed data on two variables: explanatory variable, and dependent variable. Use scatter plot to determine whether or not there is a relationship between the variables of interest.[6]

3. Calculation Process

   To calculate the weight of each part, the part of side picture of cattle feature will be regressed with chest feature data.[6]

   The results that were obtained from this approach were as follow.

   (a) 73.213% accurate system;

(b) Accuracy between segmented object and without segmentation is the same because of anatomical shape.

Estimation of Calf Weight from Fixed Point Stereo Camera Images using 3D Successive Cylindrical Model

In order to improve productivity, the objective is to simplify the estimation of calf weight. In this paper, method to estimate weight by three-dimensional model of cattle shape using stereo images is proposed. This enables to solve the problem that requires no special environment to shoot images only from above in the previous work. Moreover, it can mitigate the issue of the posture of cattle by grasping it three-dimensionally. In the proposed method, firstly, the stereo camera is created by fixing two network cameras. The image of the calf is taken by these cameras, and the stereo matching method is applied to the captured image to calculate three dimensional coordinates. Next, only the body is modelled because chest girth and waist girth have the highest correlation with weight. Since the body has a rounded shape, a three-dimensional successive cylindrical model is used. Weight estimation by using the linear regression equation between the volume of the obtained three-dimensional successive cylindrical model is obtained and the weight measured manually. [7]

Basic Information of Cattle Body and Extraction of Body Parts Data

Two fixed network cameras shoot motion images of calves. These motion images are divided into frames and analysed as static images. The image can be used as a stereo image by calibration of cameras. In stereo matching, the distance between the camera and the target object is kept constant. This distance depends on the size of the target object. In this paper, the distance ranged from 1 to 3 meters. [7]

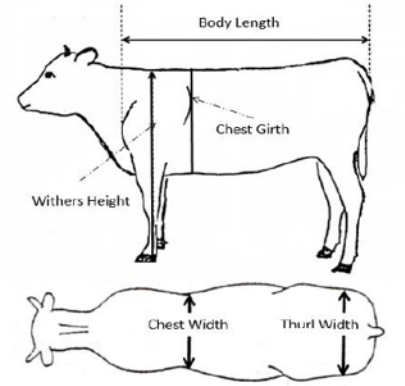


Figure 2.8: Each parts of cattle body

Two methods are used to extract body parts of the cattle from the stereo images. First method is background subtraction method where the background image is taken in advance. Second method is using parallax values obtained from the stereo matching method. Both of these methods output mask images. The common part of these two mask images is taken to extract the cattle body. This process eliminates the noise. The output image is converted to binary to give cattle body in white and remaining in black.

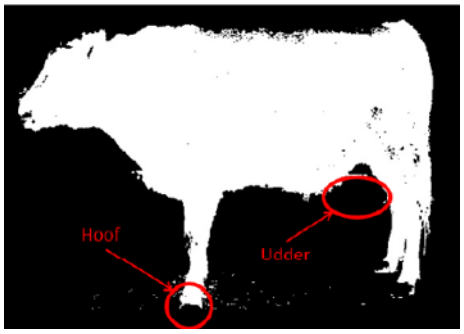


Figure 2.9: Binary image of cattle

Modeling Body of Cattle

Measuring chest girth precisely is difficult so it model the entire body using a successive cylindrical model to find correlation with weight. Three-dimensional point cloud data is calculated from stereo images using the stereo matching method. Surface of cattle is smooth with little change in colour, therefore the errors are high and volume cannot be determined precisely. Therefore, we attempt to model it by using a simple and fixed three-dimensional model. [7] A cylindrical model is chosen since it best models the rounded shape of the cattle body. It is necessary to divide the body into several parts in order to change the radius of the cylindrical model according to the position in the x direction of the body. [7]

The radius of the cylindrical model is determined by using the circle closest to the point cloud data. This enables to detect the body shape while reducing errors. To develop the circle nearest to the point cloud data, a circle fitting using a generalized Hough transform is employed.

First, when comparing the cases where the body is divided and not divided, the correlation coefficient and the average error are better when divided. Then, comparing the case where each result was individually examined and the volume was averaged, all of the result was better when the volume was averaged. Thus, it verified the effectiveness of slicing the image of the body with the proposed method and to

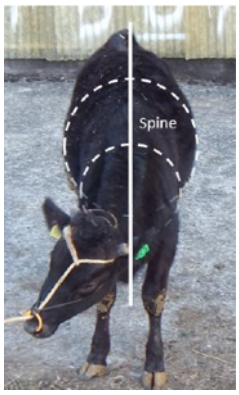


Figure 2.10: Cattle from above

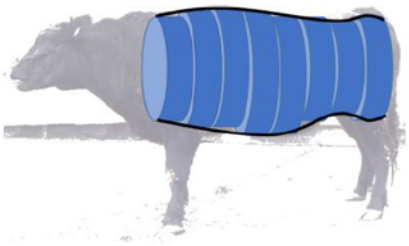


Figure 2.11: Successive cylindrical model

shoot and use multiple images of the same cattle. [7] The experimental results have revealed the following:

1. There is a correlation between weight and volume obtained by modeling the body.

2. The proposed method that models by slicing the image of the body is highly effective.

3. There is a possibility to improve accuracy of weight estimation by using multiple stereo images

Weight Estimation through Image Analysis

In this paper [9], image segmentation based technique has been used to segment out the region from the image and then using the dimension of the image object weight has been calculated using density based formula. According to the author, it

was observed through this experimental process, results were 92%-94% accurate. [9] The strategy that is used in the paper can be understood by the following flow chart diagram.

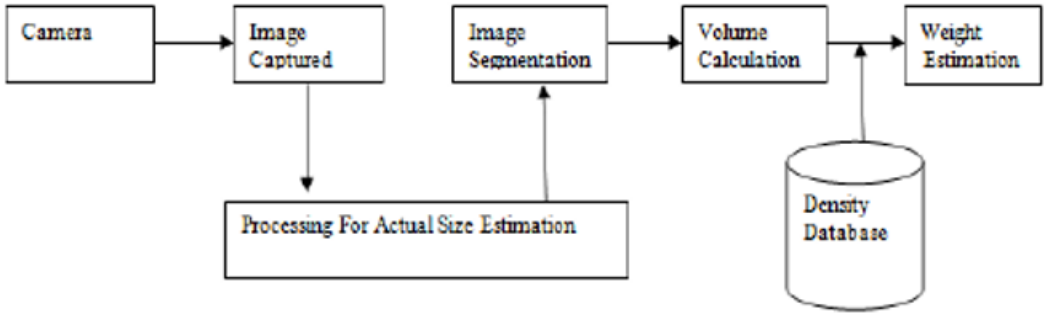


Figure 2.12: Process Flow Chart

The propose strategy in paper [9] consists of the following steps.

1. Pre-Processing
2. Image Segmentation
3. Volume Calculation
4. Mass Calculation

Pre-Processing

The image is a 2D array which consists of finite and discrete pixel values. In order to segment out the foreground object (cattle in this case) from background some pre-processing is required. In pre-processing phase the image is downscaled for faster computation. The pre-processed image is segment using K-means clustering algorithm to extract the object from the image. The image is divided in which subtraction of the image comprising the object cluster is used for further processing. [9]

Image Segmentation

In this phase, the image that was obtained from the pre-processing step is then segmented using watershed segmentation algorithm to extract the object from the image.

The paper [9] explains the method that is used to calculate the volume in detail. Let f(x, y) be the input from the front view and s(x, y) be the input from the side

view which is then preprocessed to get f‘(x, y) and s‘(x, y). These preprocessed images are passed through the segmentation function, as mentioned above, to get clustered images of front and side view, that is g(x, y) and h(x, y) respectively. The subsection of g(x, y) and h(x, y) containing the object's cluster. The grid area technique is applied on the segmented images. The M x N subsection is selected for grid area calculation. The subsection consists of object's cluster O and background cluster G is further divided into smaller grid of size m x n. The M x N is then iterated for every grid and for each grid if more than 50% pixels in the grid belong to object's cluster O then the grid is considered to be a part of the object's cluster. The iteration gives the total number of grids belonging to O, that is, O[n]. The count O[n] is stored in a vector for every column in the grid array. The grid area returns two vectors, C_{ocside} for side image and $C_{ocfront}$ for front image. After applying this technique, the following cases might arises.

1. The vector have equal dimensions
2. The vector have unequal dimensions

Volume Calculation

If the dimension are not equal then extra elements from the larger vector are removed and average of these elements is inserted back into vector. Volume can be calculated using the following formula.

$$Volume = (\sum_{i=0}^{n} C_{ocfront}(i) + C_{ocside}(i)) * V_{grid} * sf^3 \tag{2.4}$$

$Where, V_{grid} =$ *Volume of the grid cube*
$sf =$ *Scaling Factor*

Mass Calculation

Using the volume of object obtained from the above mentioned process, the mass can be calculated as follow.

$$Mass = Density * Volume \tag{2.5}$$

Learning Deep Features for Discriminative Localization

In this research paper, we visit the global average pooling layer and shed light on how it explicitly enables the convolutional neural network (CNN) to have remarkable localization ability despite being trained on imagelevel labels. While this technique was previously proposed as a means for regularizing training, we see that it actually builds a generic localizable deep representation that exposes the implicit attention of CNNs on an image. Despite the apparent simplicity of global average pooling, we are able to achieve 37.1% top-5 error for object localization on ILSVRC 2014 without training on any bounding box annotation. We can see in a variety of experiments (given below) that the network is able to localize the discriminative image regions despite just being trained for solving classification. [8]

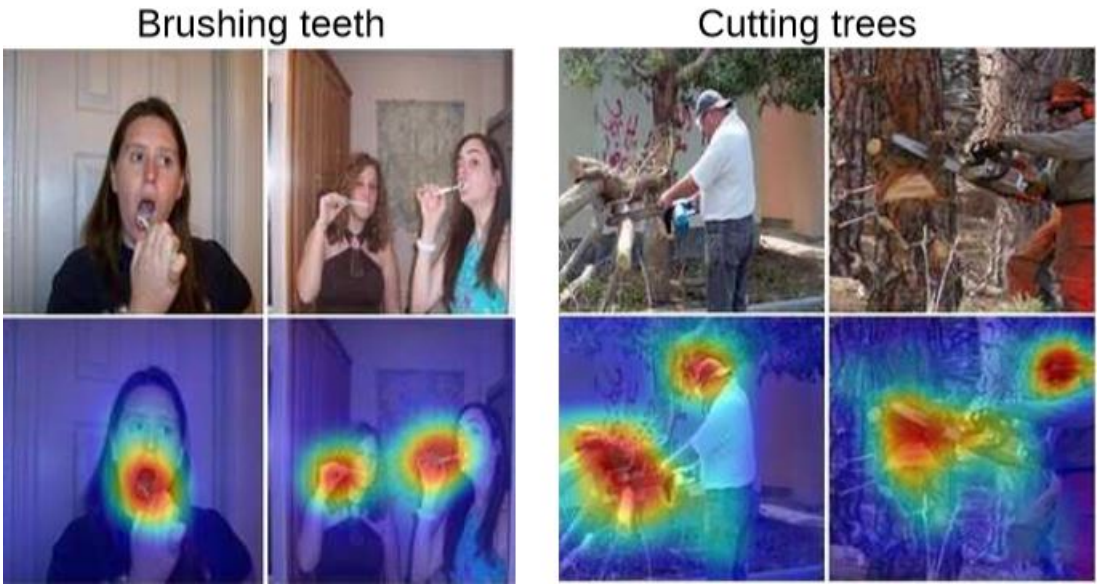


Figure 2.13: Simple modification of the global average pooling layer combined with our class activation mapping

The final output above is called Class-Activation-Map(CAM). It is actually the regions of the image that are the deciding factors during classification i.e. Discriminative-Regions (DR). These maps are calculated by first finding the node with the highest classification score and then finding the weights that connect the neurons from Global Average Pooling (GAP) layer to the final layer (classification layer). We can then

find the linear combination of these weights and feature maps obtained from the last convolutional layer to find the CAM. The figure below illustrates that.

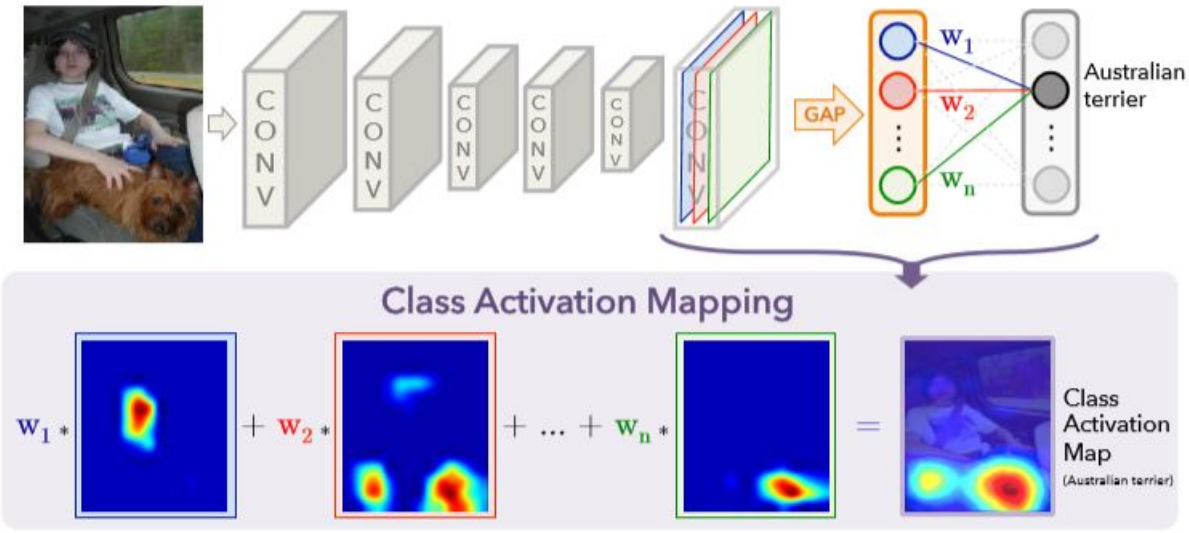


Figure 2.14: Class Activation Mapping: the predicted class score is mapped back to the previous convolutional layer to generate the classactivation maps (CAMs). The CAM highlights the class-specific discriminative regions

The results shown are evident that localization problems(with reliable results) can be solved by using a vanilla CNN without going into the complexity of object detection.[8]

2.3 Conclusion

The research papers we reviewed did not incorporate the depth information which in our case would greatly increase accuracy. To counter this, we explored 3 different approaches in order to predict the weight of the cattle as accurately as possible.

2.3.1 Approaches

Automated Heart Girth Measurement Where depth is Fixed

The heart girth measurement is the most reliable technique of estimating the weight. In this approach, we take two variables; Heart Girth and the Body Diagonal. We assume that the depth of the animal i.e the distance of the animal from the camera

is either known or fixed. We take two images, the front view and the side view. The width of the animal is determined from the front view. To get both the variables, we convert the number of pixels in the image to real-world measurements using the formula described in curve length estimation section.

The following vectors are obtained from the images:

$$\vec{H} = [p1, p2, p3...pN]$$
$$\vec{D} = [dp1, dp2, dp3...dpN]$$

Where H = Heart Girth Pixel Vector and D = Diagonal Pixel Vector.

Heart Girth Measurement with Per-Pixel Depth

This approach builds upon the previous approach. We take the same variables i.e the Heart Girth and the Body Diagonal. This method however requires only 1 image; the side view of the animal. We find the inner and outer bounding boxes around the required parts of the animal. The curvature of the animal is estimated using the per pixel depth information.

The following vectors are obtained from the images:

$$\vec{H}p = [p1, p2, p3...pN]$$
$$\vec{D}p = [dp1, dp2, dp3...dpN]$$

Where H = Heart Girth Depth per Pixel Vector and D = Diagonal Depth per Pixel Vector.

Curve Length Estimation:

In the above mentioned approaches the final result we get is a bounding box from our core component using our deep learning model which encapsulates the information of body length in the diagonal and heart girth measurements in height of the box. Since these values are just only represented by two pixel coordinates top-left and bottom right. As mentioned earlier to get the measurements accurate we have to incorporate the depth information. Since we have depth map and RGB image of same dimensions and from same camera we can safely map this bounding box to depth image as well.

Once we have a this image box on depth image we will extract out the vector of pixels from the heart girth line of the bounding box and diagonal vector from pixels starting from bottom left and the top right point in this image. Once we have these two vectors we can convert it into real world measurements using the procedure defined below in Appendix.B, Math section.

CHAPTER 3
SOFTWARE REQUIREMENT SPECIFICATION (SRS)

3.1 Introduction

3.1.1 Purpose

The purpose of this document is to present a detailed description of our Final Year Project, InstaWeight. It will explain the purpose and features of the system, the core components and the interface of the system. It will give in-depth information of our system that we will developed.

3.1.2 Scope

The aim of the system is to estimate the weight of the cattle accurately. The system is designed to automate the process of measuring cattle's weight on a large scale in a farm. Moreover, it will keep track of the daily growth of the cattle's weight to take decisions related to live stock business. The core component of the system will be responsible to detect the features of the animal from the image to estimate/predict the weight of the animal. This core component will be developed using machine learning and image processing algorithms.

On client side, there will be a admin panel to the store the weight of cattle and see the growth trend of each cattle. The system will also contain a relational database containing the basic information such as id, age, gender, weight etc. of each cattle.

3.1.3 Overview of the document

The next section of this document, 'the overall description' gives an overview of the functionality of the system. It describes the functional and non-functional requirements, use case diagrams, system diagram and external interfaces.

3.2 Overall Description

3.2.1 System Environment

The autonomous weight estimating system has two active actors a system with three components. A normal user/ farmer can capture the image of a cattle to measure

the weight of an animal. The camera would be connected to the core component of the system. The weight would be saved in admin panel. The owner of the farm or a rancher can access the admin panel to keep track of the growth of the cattle.

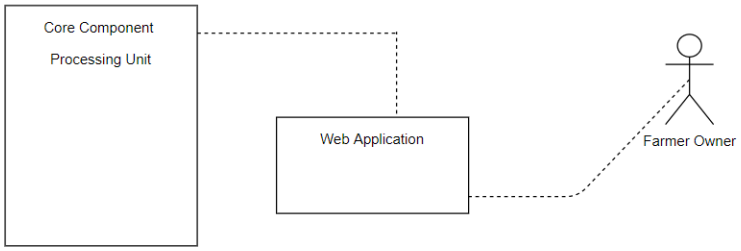


Figure 3.1: System Environment Diagram

3.3 Functional Requirements

3.3.1 Business Requirements

1. The system should be 90% to 95% accurate.

3.3.2 Web Application (Admin Panel):

The core functionalities of admin panel would be following:

1. **Authentication:**
   (a) The sign up of all users would be done by the admin panel.
      i. To sign up into the system, the basic information of the user would be required and will be added to the system.
2. **Dashboard:**
   (a) The admin panel would display the overall growth of all cattle in last six months through pi-chart.
   (b) It would show the growth progress of each cattle on individual level in last six months.
   (c) Admin panel would have different tabs and button to add, update and delete.
   (d) Record of the animal can be update by clicking on the "Update Record" tab.
   (e) The cattle's record can be deleted by "Delete" button.
3. **Cattle Management:**
   (a) **Add Cattle**
      i. To add an animal into the system, it would be required to add some details such as age, gender, type etc. into the system.
   (b) **List View**
      i. List of all animals would be displayed.
   (c) **View Details**
      i. On clicking view details, the complete details of every individual animal would be displayed on screen.

4. **Under Observation:**
   (a) All cattle who are diagnosed ill or under stress by the system and needs checkup.
5. **Settings:**
   (a) Personal configuration section from where admin can change password and other login details.

3.4 Non-functional Requirements

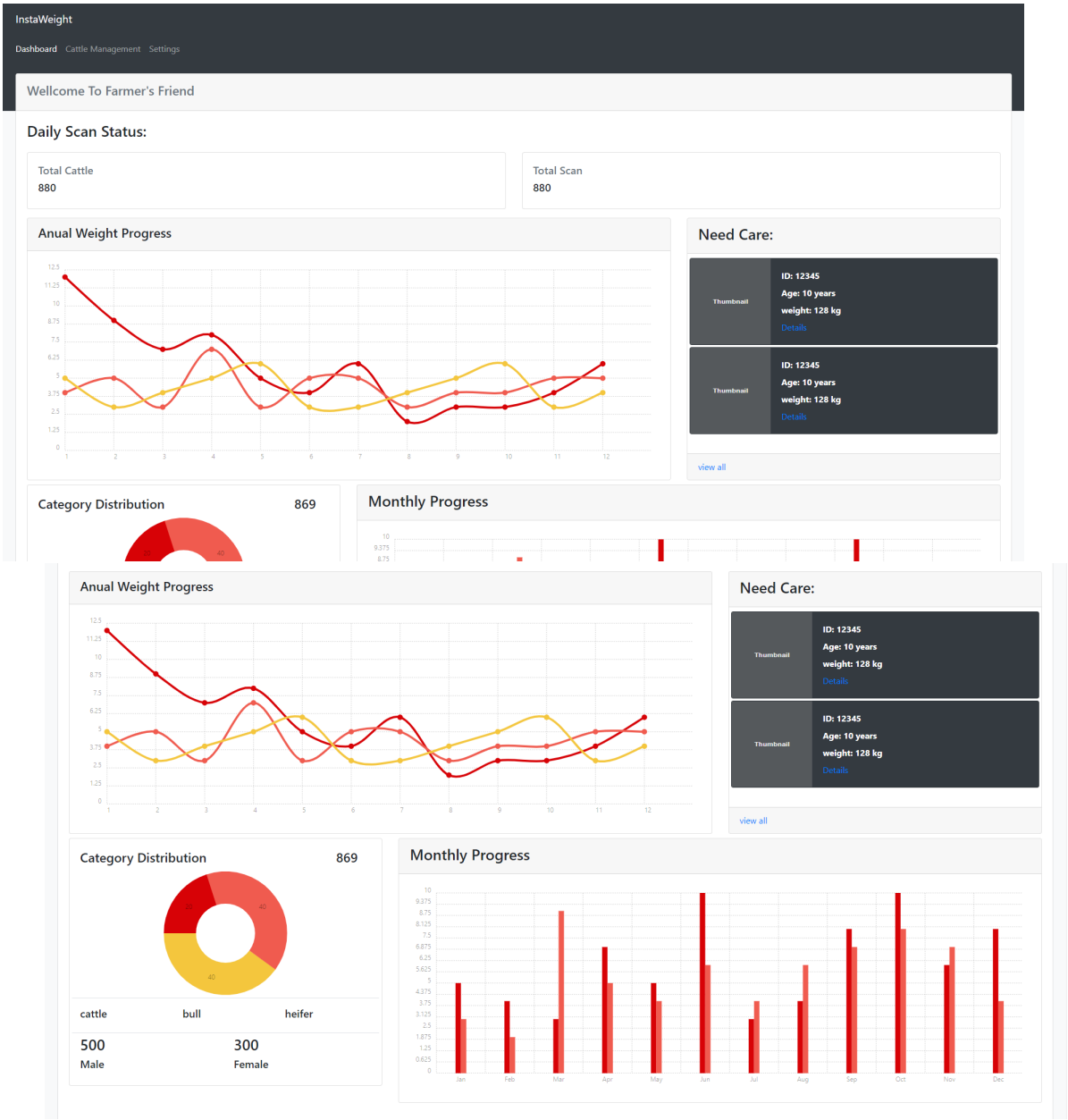
Following are the non-functional requirements of our system.

1. **Security:**
   The user's data that would be required during sign up should be confidential, safe and secure.
2. **Performance:**
   The system should less than a minutes to process the image and estimate weight.
3. **Compatible:**
   The system should be compatible with all android devices.
4. **Memory Efficient:**
   The mobile application should be space efficient and should not occupy large space of memory.
5. **Usability:** The interface of the system should be simple and flow of app should be understandable by farmer who is not well educated.
6. **Scalability:**
   The system should be scalable so that it can later be used in real world or at industry level.
7. **Maintainability:** The system should be easily maintainable so that it can become more advance and efficient when use in industry.
8. **Legal:**
   According to the intellectual property rights, the system should only be used by Folio3.

3.5 External Interfaces

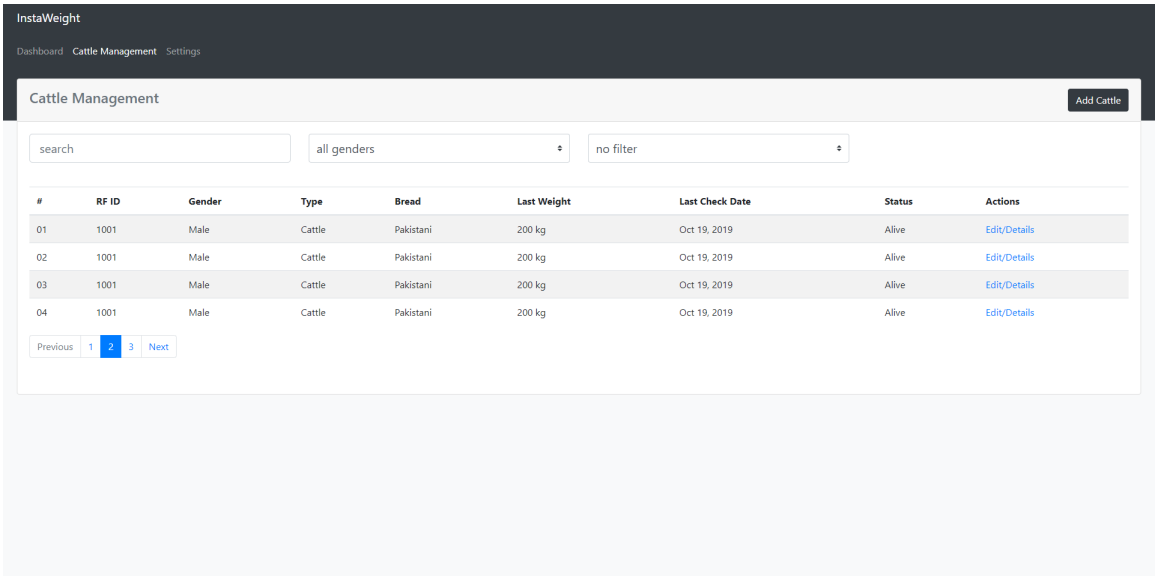
3.5.1 User Interfaces

Admin Dashboard

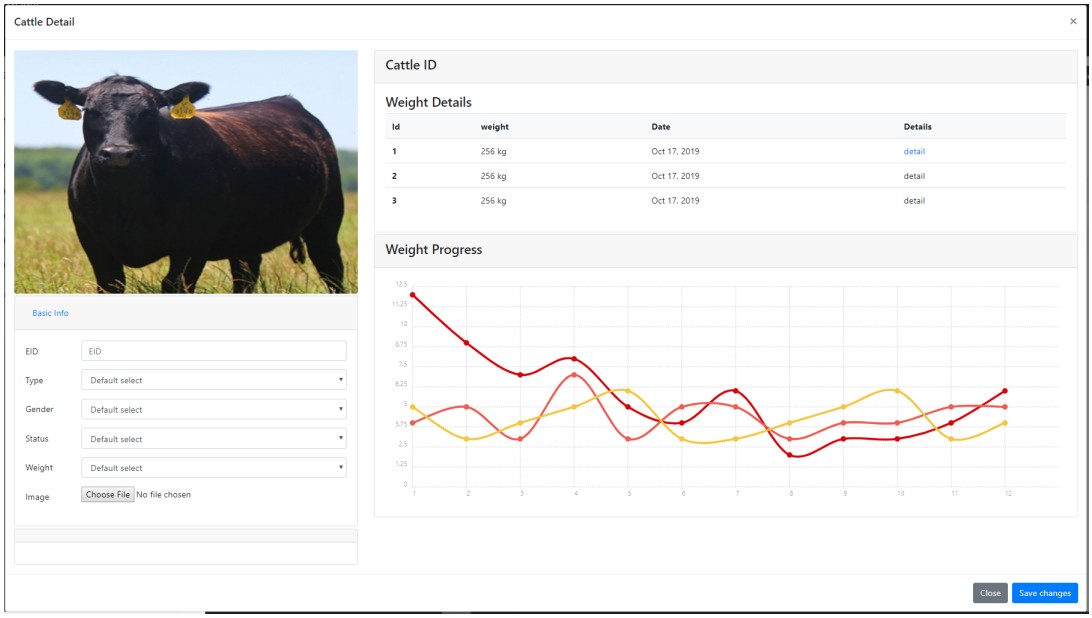


This page will display some graphes which will show annual and monthly animals' progress. it will also display animal which need special care.

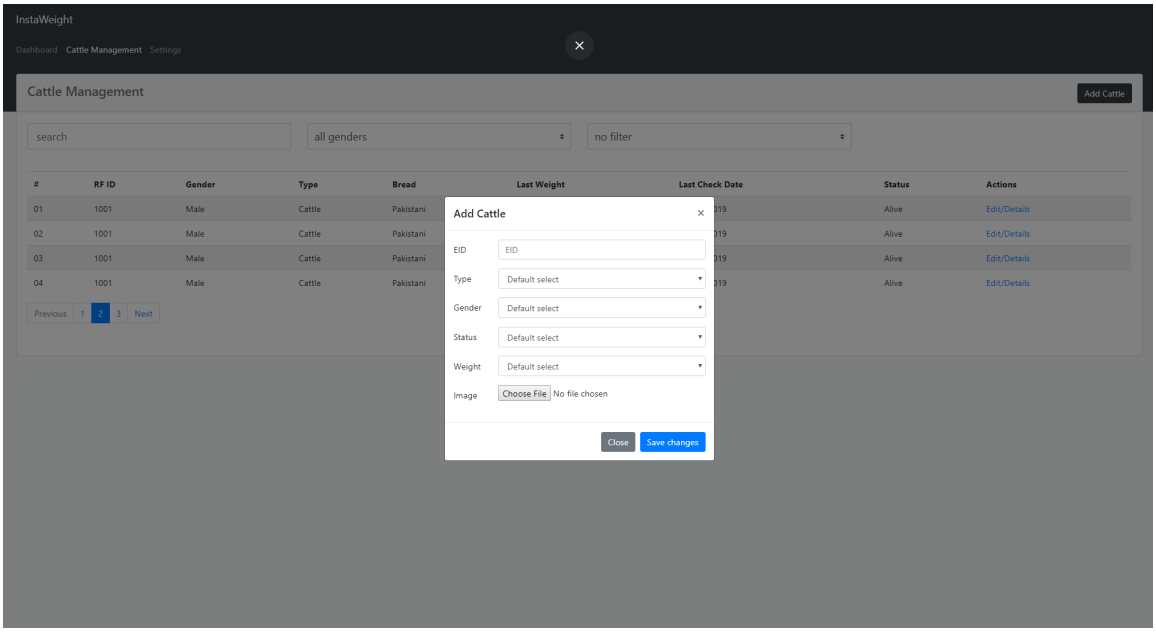
3.5.2 Cattle Management:



This will display all animals in a tabular form and will let user filter the animals and see their details.



This will display details of a single cattle and lets user edit it.



This will will be the interface though which a new cattle can be added details of a single cattle and lets user edit it.

3.5.3 Dataset and Augmentation

Dataset Collection:

The dataset was collected 3 times. Here are the details of the dataset:

1. **Dataset V1** - This dataset was collected on a farm having about 150 animals and approximately 1200 RGB images were collected. All of the dataset was annotated for two bounding boxes, one to enclose the whole cattle (outer bounding box) and the other to enclose abdomen of the cattle (inner bounding box). Out of those 1200 images 229 were usable. We also scrapped about 700 images from google and around 277 were usable. These images only consisted of a side view of the Cattle and no depth and weight labels were available. These images were used for inner and outer bounding box detection. We augmented this dataset by making each image brightened with gamma = 2.0. We then darkened the image and introduced noise to simulate low light conditions. The images were flipped on a vertical axes. hence generated 3 augmented images out of one. The total number of images after augmentation were about 2,024.

Figure 3.2: Annotation on V1 for inner and outer bounding box

2. **Dataset V2** - This dataset was collected on a farm having about 65 animals. Approximately 1235 RGB images with depth were taken. We annotated these images for the Inner and outer boxes as well as for Semantic Segmentation. These were also side views of the cattle and no informatio of cattle's weight was available. Total images in our dataset were 1235 with depth, 229 (original from V1 without depth) and 2,024 (including original, augmented and scraped images) without depth. These images were used for Semantic Segmentation and bounding boxes.

Figure 3.3: Annotation on V2 for mask RCNN

3. **Dataset V3** - This dataset was taken urgently and only 8 animals were available. This dataset was created because the previous image with depth has some noise in depth values and a lot of zero values appear in the depth. The problem aroused because the camera was not calibrated correctly. After correct calibration, Approximately **1000** RGB images with Depth were taken. This dataset was also annotated for inner and outer rectangles but was only used for depth reated experiments.

All of these images were annotated by our own team. We were unable to get cattle's weight labeled dataset because most of the farms do not keep it. And due to Covid-19 we were unable to make any further progress regarding data collection.

3.6 System Diagram

This diagram gives a high-level view of the different components of our system and the interactions between them. Each component and the particular tools/technologies/libraries used to build it are described.

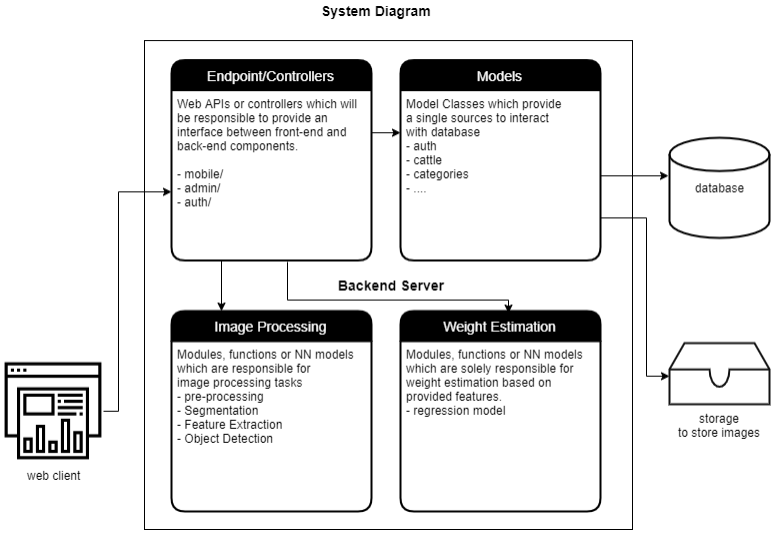


Figure 3.4: System Diagram

3.6.1 Database

This component will be responsible of storing data into relational format. This will be only accessible by Models (in MVC) component in the back-end.

Tools and Technologies: PostgreSQL, SQL.

3.6.2 Back-end Server

This component will have further subcomponents:

1. **Models:** This component is responsible to defining the schema of the database and interact with the database. This is the only module which interacts with the database, so that there is one source of truth for insertion and retrival of data.

2. **Image Processing module:** This component will have methods responsible for segmentation, object detection and features extraction.

3. **Weight Estimation:** Modules, functions or NN models which are solely responsible for weight estimation based on provided features.
   i.e. regression model

4. **Endpoints / controllers:** These Endpoints will define communication routes between server and clients and will bridge between the frontend and backend components.

Collectively we refer to **Image Processing** and **Weight Estimation** as **Core**.

Tools/Technologies: OpenCV, Django 2, MXnet,

3.6.3 Webclient

This component will be helping in managing the information of animal and keeping tack of their weights on the go.
Tools/Technologies: Html, CSS, JavaScript

3.7 Architecture Diagram

The following is the architecture diagram of the system which would be consists of four different layers. Since the system follow the MVC structure, these layers would be of view, controller, and model layers. There would be another layer of storage, which consists of database of the system. Layers are in hierarchical order, it means that every layer can communicate with its neighbour only.

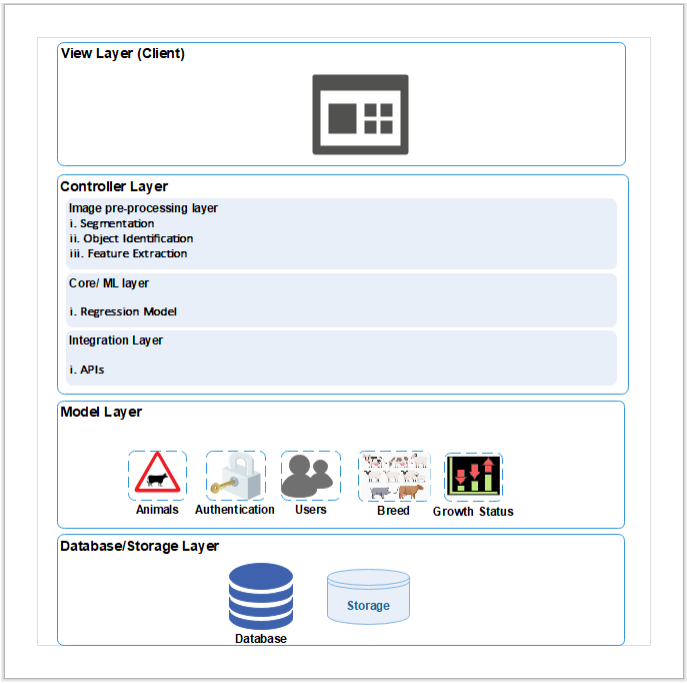


Figure 3.5: Architecture Diagram

3.8 Use Cases

This section presents detailed use cases of our system.This diagram illustrates the basic breakdown of our system in terms of use cases and actors. The expanded use cases are provided ahead.

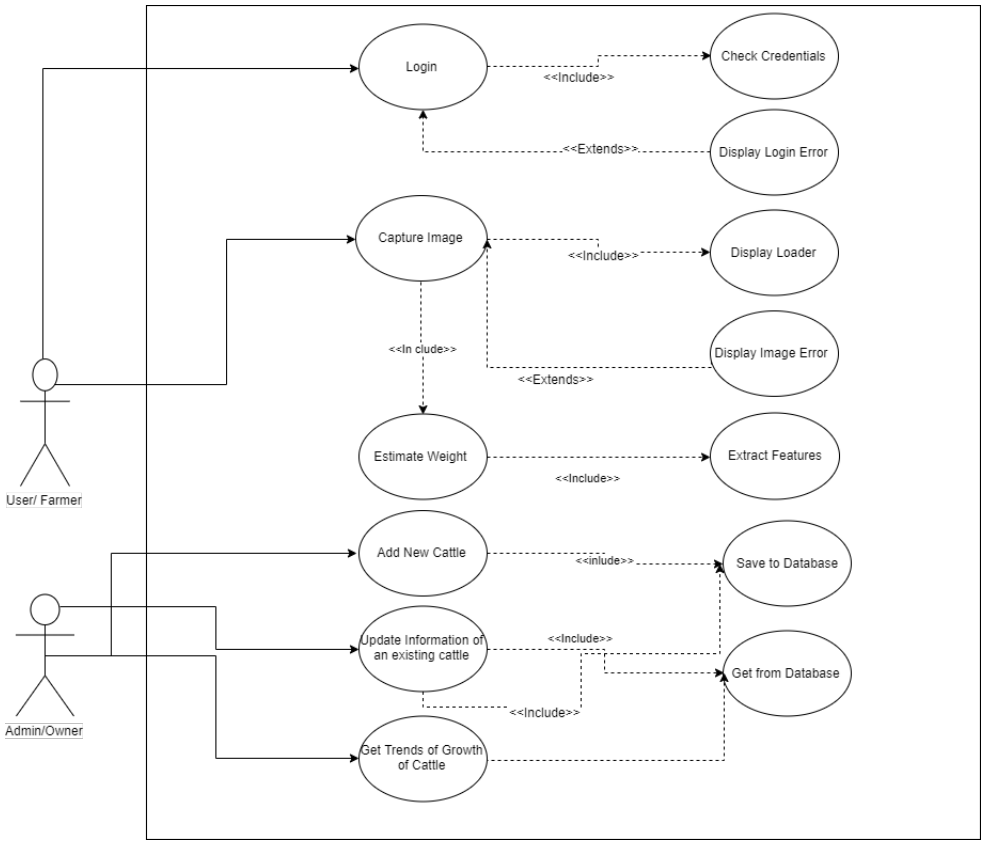


Figure 3.6: Use Case Diagram

Use Case 1: Login

Use-Case Name:	Login
Use-Case ID:	UC-01
Priority:	High
Source:	None
Primary Business Actor:	User/Client
Other Participating Actors:	No one
Description:	The user would login the application as well as admin panel to use the system. Only authenticated users will be allowed to login.
Precondition:	User must be registered.
Trigger:	When user enters his credentials and clicks on submit.
Typical course of events:	**Actor Action:** 1. User enters the credentials and clicks on submit button. **System Action:** 1. System checks the credentials of the user and validate them. 2. If the credentials are valid. User is allowed to use the system, else error generated.
Conclusion:	If credentials are valid, them user will logged in.
Post Condition:	None

Use Case 2: Capture Images

Use-Case Name:	Capture Images
Use-Case ID:	UC-02
Priority:	High
Source:	None
Primary Business Actor:	User/Client
Other Participating Actors:	No one
Description:	The user would capture the image of a cattle from front and side view. The images would be captured from a particular angle and only two images would be captured.
Precondition:	User must be authenticated.
Trigger:	When user is logged in into the system and click on the camera button to capture the image.
Typical course of events:	**Actor Action:** 1. The actor clicks on the button to capture image. **System Action:** 1. After capturing the image, the image is sent to server for further processing.
Conclusion:	The image is captured by the mobile application at a certain angle and sent to server for further processing.
Post Condition:	The image is sent to the server for further processing.

Use Case 3: Add New Cattle

Use-Case Name:	Add New Cattle
Use-Case ID:	UC-05
Priority:	High
Source:	None
Primary Business Actor:	Farm's owner/ Admin.
Other Participating Actors:	System/Admin Panel
Description:	When new cattle is added in the farm, the basic details of the cattle such as age, gender, unique id is stored in the system's database through admin panel
Precondition:	User must be authenticated.
Trigger:	On clicking the "Add new cattle" button on admin panel.
Typical course of events:	**Actor Action:** 1. Admin click on "Add new cattle" tab and enters the information related to animal and click on the save button. **System Action:** 1. System take those information and send it to database to save.
Conclusion:	The record of the new cattle is saved into the database.
Post Condition:	The information can be shown through admin panel.

Use Case 4: Update Cattle Information into the Database

Use-Case Name:	Update cattle information into the database.
Use-Case ID:	UC-06
Priority:	High
Source:	None
Primary Business Actor:	Farm's owner/ Admin.
Other Participating Actors:	System/Admin Panel
Description:	The cattle information can be updated by admin at any time when needed. The updated information will overwrite the previous information.
Precondition:	The cattle information is already added and saved in the system.
Trigger:	When information changes, the admin will update the information into the system.
Typical course of events:	**Actor Action:** 1. Admin click on "Update cattle info" tab and updates the information click on the save button. **System Action:** 1. The system save the updated information into the database.
Conclusion:	The information is updated into the database.
Post Condition:	The updated information is displayed through admin panel.

Use Case 5: Get Trends of Growth of Cattle

Use-Case Name:	Get trend of growth of cattle.
Use-Case ID:	UC-07
Priority:	High
Source:	None
Primary Business Actor:	Farm's owner/ Admin.
Other Participating Actors:	System/Admin Panel
Description:	The cattle growth trend will be shown on admin panel on the basis of their weight estimated by the system. The trends will be shown on individual level. The pi-chart would also be displayed on panel to show the overall progress. The system will display the last six months record.
Precondition:	The cattle information is already added and saved in the system.
Trigger:	It would be the core functionality of the admin panel and will be displayed by default on admin panel.
Typical course of events:	**Actor Action:** 1. No action would be done by user. **System Action:** 1. The system will display the trends to show the growth progress.
Conclusion:	The growth trends will be shown on the panel.
Post Condition:	The user can keep track of the animal's weight of last six month.

CHAPTER 4
SOFTWARE DESIGN SPECIFICATION (SDS)

This chapter provides important artifacts related to design of our project.

4.1 Software Design

This section presents the UML class diagram and gives a brief description of each class in our system. Attributes and methods of each class and relationship among classes are clearly presented. Our project contains the following classes. All the following classes except for Static classes are derived from Django Model Class because they will be directly communicating to database and define schema of it.

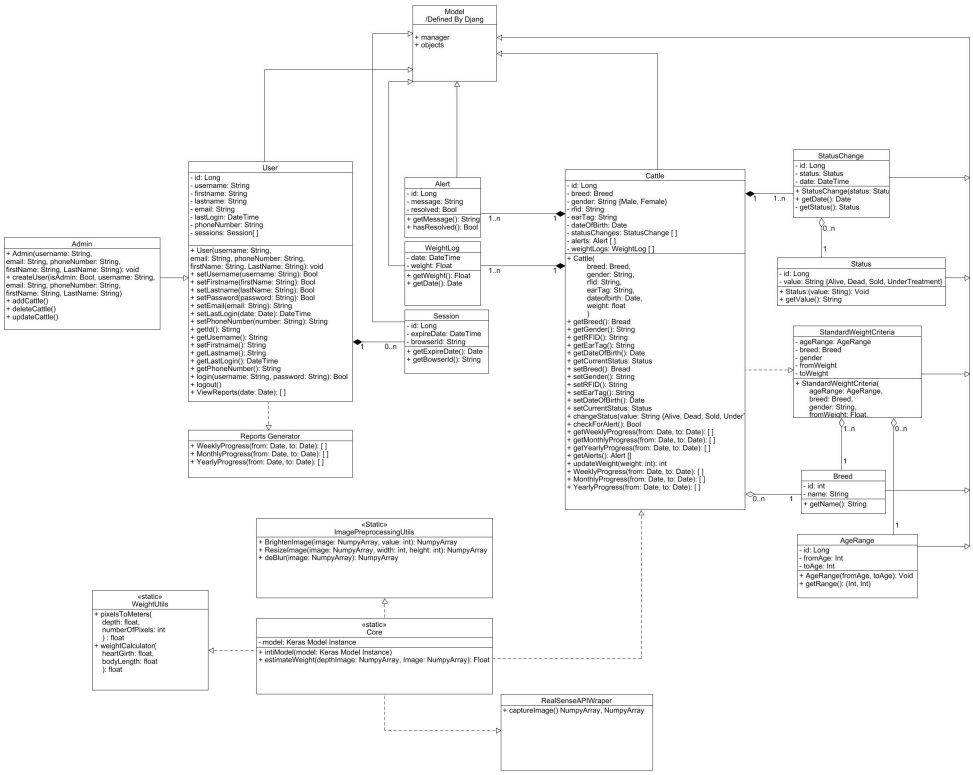


Figure 4.1: Software Design Diagram (Class Diagram)

4.1.1 User

This class represents a generic user in out system. In the system there are two types of user **Admin** and **Simple User** therefore another class '**Admin**' is derived from this class.

The class encapsulates information like **firstname, lastname, username, password, currently active session** for a perticular instance of a user.

4.1.2 Admin

This is a subclass of user which has additional access of **creating**, **updating** and deleting a users also, only the admin has access to **CRUD** operations on **Cattle**.

4.1.3 Sessions

Session encapsulates the information about a particular session of a user. User has an array of his/her currently active sessions.

4.1.4 Cattle

This class encapsulates all details of a cattle object. some of the attributes are **rfid, earTag, dateOfBirth**. Since cattle must belong to a specific **Breed**, it store reference to the breed it belongs to. It also has some object level and some class level methods to access reports. All class methods are capitalized. While generating alerts for a cattle it is dependent on standard weight criteria.

4.1.5 Alert

This class encapsulates details of an alert object. An alert will be generated if cattle is exceeding its **Standard Weight Criteria**.

4.1.6 Standard Weight Criteria

This class encapsulates information about a weight criteria for a specific breed for a specific age range.

4.1.7 Age Range:

This class encapsulates information about an age range. An Age range is a range of age which is required to define a standard weight criteria.

4.1.8 Breed:

The class encapsulates information about a breed.

4.1.9 Status

The class encapsulates information about a Status of a cattle. Status specifies whether the cattle is under overweight, underweight, good or bad condition.

4.1.10 Status Change

This class encapsulates change in the state of a cattle at a particular instance. A Cattle object has an array of status changes in it.

4.1.11 Weight Log

This class encapsulates information about weight change of a particular cattle object at a particular instance. A Cattle object has an array of Weight Logs in it.

4.1.12 Core (Static Class)

This class has some static methods for weight estimation and model initiation. the classes is dependent on the static methods of other three static classes of **WeightUtils, IntelRealsenseAPIWraper, and ImagePreprocessingUtils** .

4.2 Data Design

This section presents the structure of our database that caters to persistent data storage in our project. The structure is shown as a normalized data model for relational databases. It clearly shows entities, attributes, relationships with their cardinalities, and primary and foreign keys. We have used DB designer (or any other similar data modeling tool) to build our data model.

The system has following Entities:

4.2.1 User

This represents a generic user and its attributes.

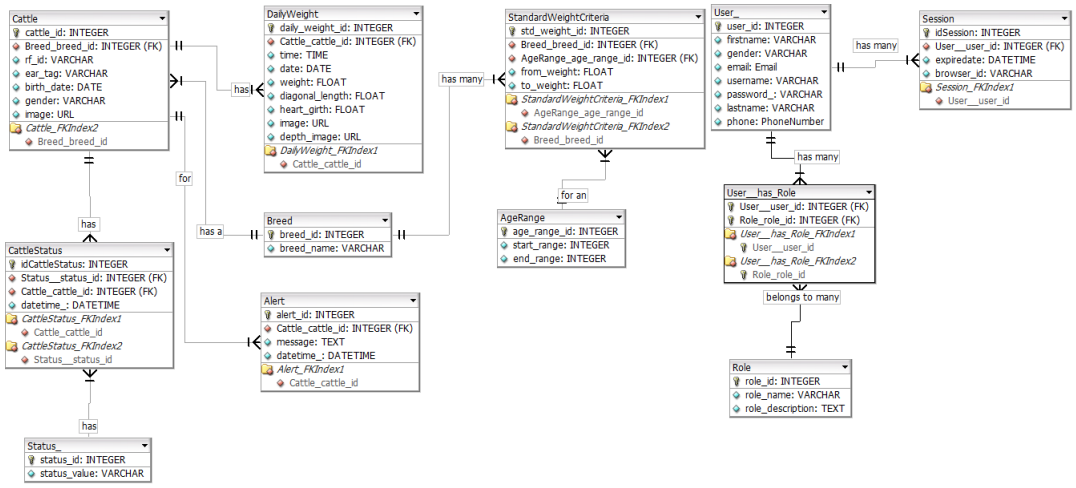


Figure 4.2: Entity Relationship Diagram

4.2.2 Session

This represent a serverside session of a user. User and Session have a one to many relationship therefore userId is stored as foreign key in Session.

4.2.3 Role

Role is a generic role representation in database. User and Roles have many to many relationship therefore a middle entity **UserHasRole** is use to resolve this.

4.2.4 Cattle

This entity represents a Cattle's attributes which are stored in database. A cattle must have a **Breed** therefore it stores BreedId as a foreign key.

4.2.5 Breed

The entity represents a breed in database. Since a breed can belong to many cattle its id is stored as foreign key in cattle's table. A breed can also have a different

weight criteria based on gender and age range therefore its id is stored as foreign key in **StandardWeightCriteria** .

4.2.6 Alert

Alert are generated when an animals state is changed. Alert entity shows the attributes stored in database. Since an alert is related to a particular cattle, cattle id is stored as a foreign key in Alert.

4.2.7 Standard Weight Criteria

Standard weight Criteria is different for each breed and gender. This entity represents its representation in database. It has a one to many relation with breed and AgeRange their ids are stored in this entity.

4.2.8 Age Range:

An Age range is a range of age which is required to define a standard weight criteria.

4.2.9 Status

Status specifies whether the cattle is under overweight, underweight, good or bad condition. Its id is stored for each status change for a cattle in Status Change Entity along with Cattle Id.

4.2.10 Status Change

This entity represents change in the state of a cattle at a particular instance. It stores Cattle id and State id in it.

4.2.11 Weight Log

It represents a weight change of a particular cattle object at a particular instance.

CHAPTER 5
EXPERIMENTS AND RESULTS

The main component of our system is the "Core Component" which is responsible for estimating the weight of a cattle given an RGB image along with a per-pixel depth map. So, most of the time was spent in in the development of this component. The experiments which we carried out throughout the span of this project can be divided into the following two main categories.

1. Deep learning Models
2. Camera and Depth Related Experiments

5.0.1 Deep Learning Models

Almost 75% of the time was dedicated to collect data whose description is in 3.5.5 Dataset and Augmentation section, to do experiments, and to come up with a better model. The biggest challenge for this part was the availability of dataset and also weight of the cattle. All dataset, which is used throughout this project, is collected by our team.

Experiments on Dataset V1 for Localization

On dataset V1, we tried to solve our problem as a localization problem through transfer learning. In localization problem we assume that there is only one object in an image which we are trying to detect. This fact give us an advantage that we know exactly how many bounding boxes we have to predict. In this experiment Pre-trained networks VGG16 and Resnet were used as base networks. At the end of these networks we attached our own fully connected network for regression on the features extracted by the base networks.

Initially, we trained only fully connected layers for two bounding boxes. Then we trained only fully connected layers for one bounding box. We then unfreeze some convolution layers and train them for both two bounding boxes and one bounding box. Global Average Pooling was used for Localization. But we were unable to get the satisfactory results. Our min loss was 4.68 on mean absolute error.

Figure 5.1: Output of Models for Object Localization trained on V1

Experiments on Dataset V2 for Localization

In this experiment, cattle images were cropped and we tried to predict inner bounding boxes. We assumed that it will learn better features due to the lack of background. However, this did not produce the results we were looking for

The problem we observed in both of the experiments was the over fitting of train data. They seemed to perform good on train data while it performed worse on test data. But in order to make our solution work we needed very tight bounding box. After a lot of effort we then trained Yolo and Mask RCNN.

YOLO

After experimenting with object localization and not getting satisfactory results we turned towards YOLO object detector. YOLO stands for You Only Look Once. The advantages of using YOLO are numerous and one of them is speed, YOLO is the fastest deep learning based object detector which offers reasonable accuracy. We trained YOLO on Dataset V2 containing 1200 images. After 100 epochs we got loss of 11.54 units on YOLO custom loss function. On the test set we acheived an average IoU of 80%. The bounding box detected by YOLO trained on dataset V2 is shown below.

Experiment#	Model	Model Architechture	# of Train Images	# of test in	Final loss	Epochs	Loss function
1	VGG16	Output was taken from the last convolutional layer of network and a single FC layer was added with 8 neurons for regression.	229	30	9	500	Mean Squarred Error
2	VGG16	Output was taken from the last convolutional layer of network and a single FC layer was added with 8 neurons for regression.	1900	200	51	500	Mean Sqarred Error
3	VGG16	Output was taken from the 3rd convolutional block and a single FC layer was added with 8 neurons for regression.	1900	200	44	500	Mean Squarred Error
4	VGG16	Output was taken from the 3rd convolutional block and a single FC layer was added with 1024 neurons and another FC layer was added with 8 neurons for regression.	2000	100	17	500	Mean Squarred Error
5	ResNet50	Output was taken from the last convolutional layer of network and a single FC layer was added with 8 neurons for regression.	2000	100	261	500	Mean Squarred Error
6	ResNet50	Output was taken from the last convolutional layer of network and a single FC layer was added with 8 neurons for regression with another FC layer with 1024 neurons added in between.	2000	100	147	500	Mean Squarred Error
7	ResNet50	Output was taken from the last convolutional layer of network and a single FC layer was added with 8 neurons for regression with another FC layer with 1024 neurons added in between. Also newly collected dataset was used.	1900	256	8	500	Mean absolute Error
8	ResNet50	Output was taken from the last convolutional layer of network and a single FC layer was added with 8 neurons for regression with another FC layer with 1024 neurons added in between. Also newly collected dataset was used. With Some convolutional layers retrained.	1900	256	6	500	Mean absolute Error
9	VGG16	Output was taken from the last convolutional layer of network and a single FC layer was added with 8 neurons for regression with another FC layer with 1024 neurons added in between. Also newly collected dataset was used. With Some convolutional layers retrained. Probabliy overfitted.	1900	256	4.68	500	Mean absolute Error

Figure 5.2: Results of Models for Object Localization trained on V1

The results from YOLO may look reasonable but they could be made better with little adjustment. Or we can collect a bigger dataset and train YOLO on it better performance in terms of detection accuracy. In an ideal case there should not be a single pixel on the box that is outside cattle body.

Mask R-CNN

In our quest for a perfect weight estimater we also explored Mask R-CNN to get a better more robust system. The idea was to segment out the cattle from the RGB image. Then superimpose the cattle mask on the depth channel. Essentially, the extracting all the depths the lies on the cattle. We then input the depth values to an Artificial Neural Network (ANN) and frame the problem as regression problem. The problem with this approach is that it needs weight labels for each cattle in the dataset, which is not available. Nevertheless we went on to explore this approach as weight labels can be collected later.

We trained a Mask-RCNN on Dataset V2 and got the following results.

But we can not proceed any further with this approach to make an actual weight estimator as we dont have weight labels of each cattle. But what we can do is prove that the ANN which we are going to train once we get the weight labels will work, by calculating the proxy Factor in the next sub heading.

Factor

5.0.2 Camera and Depth Related Experiments

The purpose of our deep learning models to get the rectangle so that we can take out the pixels' vector which lies on the heart girth of the cattle once we have this vector we have to get the real world measurement of the heat girth of the cattle.

Curve Length Estimation:

In the above mentioned approaches the final result we get is a bounding box encapsulating the information of body length in the diagonal and heart girth measurements in height of the box. Since these values are just only represented by two pixel coordinates top-left and bottom right. As mentioned earlier to get the measurements accurate we have to incorporate the depth information. Since we have depth map and RGB image of same dimensions and from same camera we can safely map this bounding box to depth image as well.

Once we have a this image box on depth image we will extract out the vector of pixels from the heart girth line of the bounding box and diagonal vector from pixels starting from bottom left and the top right point in this image. Once we have these two vectors we can convert it into real world measurements using the procedure defined below. For the simplicity forget the consider that we have a formula which converts given number of pixels into meters and also take a distance from camera to take care of the scaling, because otherwise far objects appear small in image. now given this formula consider the abdomen of the cattle. It is mostly symmetric from both sides. If we take the measurements of a heart girth and body length curves on the surface of the abdomen, which is like a semi-ellipse from one side, will have huge factor in measurements. So to preserve this information we will use the mathematical approach, discussed in Appendix A, to get the length of heart girth circumference and body length more accurately.

CHAPTER 6
CONCLUSION AND FUTURE WORK

If we look at the progress of this project so we were able to develop the Admin Panel success fully and sub programs to estimate the weight given depth vectors were also developed. The only part where we still struggled was the "Core Component". Our dataset collection was impacted due to Covid-19. The best we have now is YOLO object detector which is giving us on average 80% workable predictions, it gives us the bounding box for abdomen which we use to extract heart girth's information . Since we also have proposed that we can segment out the mask of cattle from Mask RCNN, which we trained on our own, because existing one was not as accurate as we wanted on cattle's side view, and extract depth information corresponding to this mask from image then given this information if we have cattle's weight labeled dataset we can train a regression model on this depth mask to get the accuracy up to $90 - 95\%$ because the depth values encapsulate the whole 3D information about the cattle. This is the future work we propose for which Mask RCNN is trained for better masks for this problem. Also we have compiled a very diverse dataset for this problem which we think is going to help a lot if someone tries to solve this problem in future.

Appendix A. More Math

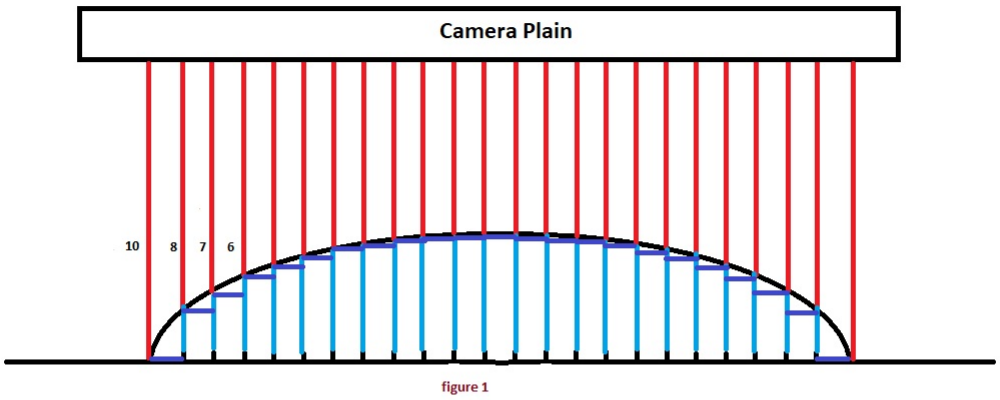


Figure A.1: Heart girth measurement approach

Consider figure A.1. Here picture of a surface is taken from a dept camera. Black curve represents the surface and the red lines represent per-pixel distance from camera. Since we assumed that we have a function which can convert number of pixels covered at certain depth to meters, we can get the measures for each pixel in me-

ters. But our problem is to get the length of the length of the curve in meters while preserving the curve, we break the curve into small segments each segment is break after one pixel. If we have to estimate the curve of this segment we can do so by fitting a triangle on it. Where base of the triangle is

maxpoint = argmax(depth of P1, depth of P2) b = pixelToReal (depth = maxpoint, numberofpixels = 1)

Now we have the base in meters we have to calculate the perpendicular or the height this is simply the absolute difference between the depth of P1 and P2

$$p = |depthofP1 - depthofP2|$$

once we have the the base and perpendicular we can use the Pythagoras theorem to calculate the hypotenuse or the length of the segment

$$h = \sqrt{p^2 + b^2}$$

now we have the formula we can apply this over all segments sum the all segments' length to estimate the length of the curvature. hence by using this technique we can get the the length of heart girth circumference and body length more accurately.

Appendix B. Data

The complete dataset our project can be found at this Google Drive link.

Appendix C. Code

Here is the part of our code. This code is to capture the images of cattle and to collect dataset. The camera would be connected to laptop and this script would run to capture the cattle images. This would take two paths as parameters to save RGB and depth images and also image number of the captured image.

```
# First import the library
import pyrealsense2 as rs
from PIL import Image
import numpy as np
import cv2
import glob


def collect_data(image_save_path, depth_save_path, from_image_no):

    image_no = int(from_image_no)
    # Create a context object.
    # This object owns the handles to all connected realsense devices
    pipeline = rs.pipeline()
    config = rs.config()

    config.enable_stream(rs.stream.depth, 1280, 720, rs.format.z16, 30)
    config.enable_stream(rs.stream.color, 1280, 720, rs.format.rgb8, 30)
    profile = pipeline.start(config)
    depth_sensor = profile.get_device().first_depth_sensor()
    depth_scale = depth_sensor.get_depth_scale()

    # post processing
    # temporial filter
    temporal_filter = rs.temporal_filter(
        smooth_alpha=0.4, smooth_delta=20, persistence_control=2)
    # threshold filter
```

```
threshold_filter = rs.threshold_filter(
    min_dist=0.15000000596046448, max_dist=16.0)
# Colorize
colorizer = rs.colorizer(0)
# decimation filter
decimation_filter = rs.decimation_filter(3)
# Spatial Filter
spatial_filter = rs.spatial_filter()

# Hole filling filter
hole_filling_filter = rs.hole_filling_filter(1)

while True:
    # for preview of images
    while(True):
        frames = pipeline.wait_for_frames()
        color = frames.get_color_frame()
        color = np.asanyarray(color.get_data())
        cv2.imshow("Image", color)

        align = rs.align(rs.stream.color)
        frames = align.process(frames)

        aligned_depth_frame = frames.get_depth_frame()
        # filter
        filtered = threshold_filter.process(
            aligned_depth_frame) # aligned_depth_frame
        filtered = hole_filling_filter.process(filtered)
        filtered = colorizer.process(filtered)

        depth = filtered.get_data()
        depth = np.asanyarray(depth, dtype=np.float) * depth_scale
        cv2.imshow("Depth", depth)

        k = cv2.waitKey(30)
        if k == 27:  # if ESC is pressed, close the program
            exit()
        elif k == 32:
```

```
            break
        else:
            continue

    # This call waits until a new coherent set of frames is available
    # Calls to get_frame_data(...) and get_frame_timestamp(...)
    # on a device will return stable values until wait_for_frames(...)
    frames = pipeline.wait_for_frames()
    depth = frames.get_depth_frame()
    color = frames.get_color_frame()

    color = np.asanyarray(color.get_data())
    img_color = Image.fromarray(color)
    img_color.save(f'{image_save_path}/{image_no}.jpg')

    # Create alignment primitive with color as its target stream:

    align = rs.align(rs.stream.color)
    frames = align.process(frames)

    # Update color and depth frames:
    aligned_depth_frame = frames.get_depth_frame()

    # filter
    #filtered = temporal_filter.process(aligned_depth_frame)
    filtered = threshold_filter.process(
        aligned_depth_frame)  # aligned_depth_frame
    #filtered = decimation_filter.process(aligned_depth_frame)
    #filtered = spatial_filter.process(filtered)
    filtered = hole_filling_filter.process(filtered)

    depth = filtered.get_data()
    depth = np.asanyarray(depth, dtype='float') * depth_scale

    # saving the depth data
    np.save(f'{depth_save_path}/{image_no}.npy', depth)
    image_no += 1
    print(f"Image #{image_no} saved")
```

```
if __name__ == "__main__":
    # input start image
    image_no = input("Please input starting image number: ")
    image_save_path = './Dataset/Experiments'
    depth_save_path = './Dataset/Experiments'

    collect_data(image_save_path, depth_save_path, image_no)
```

The complete code of our project can be found at this GitHub link.

CHAPTER 7
REFERENCES

1. Siregar,S. B.,2007. Fattening Cattle. Jakarta : Penebar Swadaya.

2. Soeprapto,H., Z.Abidin. 2006. How Fattening Beef Cattle. Jakarta : Agromedia Press

3. M. D. Deddy B. Lasfeto, "A measuring weight model of timor's beef cattle based on image," International Journal of Engineering and Technology (IJET), p. 12.

4. T. M. Banhazi, "Weight Estimation Using Image Analysis and Statistical Modelling: A Preliminary Study," Applied engineering in agriculture, 2007.

5. Mehrotra, R., and S. Zhan. 1996. A computational approach to zero-crossing-based two-dimensional edge detection. Graphical Models and Image Processing 58(1): 1-17.

6. Z. H. Pradana, B. H. and S. D., "Beef Cattle Weight Determine By Using Digital Image Processing," The 2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), 2016.

7. A. Yamashita, "Estimation of Calf Weight from Fixed-Point Stereo Camera Images Using Three Dimensional Successive Cylindrical Model," Proceedings of the 5th IIAE International Conference on Intelligent Systems and Image Processing 2017, 2017.

8. Zhou, B. (n.d.). Learning Deep Features for Discriminative Localization. Retrieved from `http://cnnlocalization.csail.mit.edu/Zhou_Learning_Deep_Features_CVPR_2016_paper.pdf`

9. R. Dalai and K. K. S. , "Weight Estimation through Image Analysis," International Journal of Computer Trends and Technology (IJCTT), vol. 49, pp. 1-5, July 2017.

CHAPTER 8
VITA

Sayyda Sahar Fatima

Sayyda Sahar Fatima is graduated from Habib University in 2020, with a degree in Bachelor of Computer Science. She is Top Performer student and has received Dean's List award multiple times. She is very focused and punctual and does her work with full concentration. She enjoys her work and tries to explore innovative things and technologies. Furthermore, she has experienced in the field of web development and has worked as a software developer at WeUno.

Atif Mehmood

Atif Mehmood is one of the hard working graduate of Habib University Class of 2020. He is a quick learner and is always excited about solving new problems and learning something new. He has a particular interest in Software Development and Computer Vision. He has worked at Afiniti and currently working as a Software Engineer for Stellic on their reimagined platform for degree planning, advising and auditing.

Mohammad Ateeb Ahmed

Mohammad Ateeb Ahmed is a computer science graduate of Habib University, really excited about theoretical computer science and physics. He is more interested in doing research work and tries to solve the a problem by applying different algorithms and problem solving techniques.