2017-06-01

Using Multiview Annotation to Annotate Multiple Images Simultaneously

Timothy C. Price
Brigham Young University

Using Multiview Annotation to Annotate Multiple Images Simultaneously

Timothy C. Price

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Ryan Farrell, Chair
Bryan Morse
David Wingate

Department of Computer Science

Brigham Young University

ABSTRACT

Using Multiview Annotation to Annotate Multiple Images Simultaneously

Timothy C. Price
Department of Computer Science, BYU
Master of Science

In order for a system to learn a model for object recognition, it must have a lot of positive images to learn from. Because of this, datasets of similar objects are built to train the model. These object datasets used for learning models are best when large, diverse and have annotations. But the process of obtaining the images and creating the annotations often times take a long time, and are costly. We use a method that obtains many images of the same objects in different angles very quickly and then reconstructs those images into a 3D model. We then use the 3D reconstruction of these images of an object to connect information about the different images of the same object together. We use that information to annotate all of the images taken very quickly and cheaply. These annotated images are then used to train the model.

Keywords: Multiview segmentation, 3D annotation, 3D modeling

ACKNOWLEDGMENTS

First, I would like to acknowledge my wife. She has been a strength for me in her support for me with this paper and her helping me stay motivated to finish it. And for the hours that she spent helping me gather and sort the data. Next I would like to acknowldge my professor, Ryan Farrell. He provided me with a lot of guidance in creating and writing this thesis. I would like to acknowledge Connor Anderson who helped me with the capturing of the data and Pei Guo and Michael Brodie who helped me with developing the machine learning models. Finally, I would like to acknowledge Brian Davis who often provided a second pair of eyes when looking for those nasty coding bugs that just eluded me.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

While the human visual system can recognize and identify objects in a scene quite easily, it has proven to be very difficult to program computers to have this capability. Programs can, however, learn to recognize objects through many different methods. These methods are in the programming field of Computer Vision. Object Recognition is the process of detecting an object within an image and then being able classify what that object is. It is currently an area of great interest in Computer Vision research. One of the earliest technologies in Computer Vision that emerged was Optical Character Recognition (OCR) [15]. This was probably due to the fact that identifying characters in a typed paper was a relatively easy task compared to the rest of the computer vision problems that exist. And the demand for OCR is high. OCR is used to convert text in print into electronic / digital form to make documents searchable or for future analysis. However, there are other aspects of Computer Vision than OCR. License plate readers are used for security systems to keep track of cars passing through a specific area or for law enforcement to identify the owners of cars of interest [1]. These and many other applications utilize object recognition technologies and algorithms to detect items within an image and identify these items.

1.1 Standard Object Recognition Technique

Machine learning is a technique in computer science that allows a computer to predict a desired result based off of input information. It works by feeding the computer a lot of prior information of a similar type of data allowing it to "train" on that data. Then using that previous training information, it can predict what the result would be for a new item that has not been seen before but is similar to the trained data. It is often used in Computer Vision to identify the type of object in an image. This is known as classification. It can also be used to segment out an object within an image. It is utilized to train a recognition system such that it can recognize specific objects. During the process, the current object it is attempting to recognize and classify is known as the target object.

Object recognition can help in video surveillance [17], increases biometric security [9], increases the safety of otherwise dangerous tasks [19] and is often used in scientific studies to further knowledge. More will be discussed on Machine Learning in Chapter 2.

1.2 Problems that occur

However, there are a few factors that can make it more difficult for a system to learn. Object position, illumination, viewing angle and occlusion are some of the common factors. As the set of objects that a system needs to distinguish grows, the amount of data needed to teach the system grows with it. While datasets often provide sufficient information to learn the data, they usually do not provide specific information about the image that could help the system learn more quickly. If the system has this extra information, it can learn more quickly and accurately than having just images and class labels alone. However, this specific information is costly and time-consuming to annotate at a large scale. This thesis focuses on a scalable approach that dramatically decreases the cost of gathering and organizing large datasets.

1.3 Multiview Geometry

In order to obtain and annotate a set of images quickly and easily, we propose the use of Multiview Geometry. By taking images of an object from multiple angles in a controlled environment, the angle and location, of the camera called the extrinsic parameters, can be calculated for each image. The geometric relationships between cameras defined by these extrinsic parameters can then be leveraged for tasks such as segmentation. By manually creating the segmentation masks for a few images, the segmentation mask of the same object in other views can be estimated using a technique called voxel-carving [13].

In this work we will compare the accuracy of multiview segmentation predictions with ground-truth segmentations done by hand through manual methods. This comparison will use varying quantities of manual segmentation images to generate the predicted segementation. We will also compare the accuracy of computational recognition models created both using segmentated data and not using segmented data. For a given amount of human annotation effort, multiview annotation provides more richly annotated training data. Through the use of multiview annotation, a large set of images taken in a controlled environment can be annotated with a fraction of the human effort that would normally be required.

1.4 Organization

The rest of this work is organized in the following manner. Chapter 2 describes related work. Chapter 3 describes the methods used to collect the image data. Chapter 4 describes the methods used to segment the data. Chapter 5 describes the methods used to validate the segmentation data and the machine learning techniques. Chapter 6 presents conclusions and future work.

Chapter 2

Related Work

A lot of research has been put into creating good recognition models and creating good datasets for training them. Many methods have been developed that will collect images from the internet and then attempt to organize this data. All of these approaches use data that is inherintly unorganized and attempt to organize and annotate. The proposed approach starts with organized data in the first place and therefore provides better control over it. A discussion of related techniques from the literature is below and provides both context and motivation for the proposed approach.

2.1 Structure From Motion

Structure from Motion [10, 25] is a method used to extract a three dimensional point cloud. It works by tracking similar points between images and using those similar points to calculate the rotation and translation for each image as well as the scene geometry in a three-dimensional model. Essentially it allows you to take many images of the same object and reconstruct the three-dimensional model of that object as well as where the cameras were when each picture was taken. This information can be leveraged when doing Multiview Geometry.

2.2 What makes a good dataset

Berg, et al. [3] suggest that "The perfect dataset will be big ... It should contain images of objects in the contexts in which they occur. The images should cover most significant categories, and there should be many images of each category. Within a category, the images should show all variations important for training and for testing." In addition, the dataset images should contain "different viewpoints, under different illumination conditions." The authors also state that "It is a fallacy to believe that, because good datasets are big, then big datasets are good." The proposed approach raises the quality of big datasets by increasing the range of viewing angles and by combining with the already created large datasets found elsewhere adding more data to those sets.

ImageNet [6] is a database consisting of over 14 million images of objects across more than 10,000 categories. The categories of which are organized in a semantic hierarchy. ImageNet's goal is to "provide the most comprehensive and diverse coverage of the image world." ImageNet was constructed with the intent that "objects in images should have variable appearances, positions, view points, poses as well as background clutter and occlusions." While they constructed the database with this goal in mind, it is difficult to achieve this objective for all subjects; common views and poses are sampled densely but the less common views and poses, the "long-tail" of the view/pose distribution, are rarely sampled. Our approach provides additional data to the already provided datasets, sampling diverse views.

The more examples a dataset has of each given object to be recognized, the better. It is especially helpful to have images from multiple angles under different conditions. Because of this, a lot of work has gone into developing large sets of data. The Stanford Cars dataset provides imagery of 196 different makes and models of vehicles. Caltech and UCSD jointly built a dataset with nearly 12,000 bird images in 200 different categories [22]. ImageNet [6] includes images for many different types of objects. While these datasets are large and provide images of their respective objects under various conditions to help improve recognition accuracy, they do not provide much detailed information on each object, parts of the object or what angle it was taken from.

2.3 Pose Estimation

The Poselet framework of Bourdev, et al. [4] attempts to use known three-dimensional pose information from two-dimensional images of humans to estimate the poses of humans in new two-dimensional images where the three-dimensional information is not available. It takes each joint in a human separately and attempts to determine the pose based off of the statistics of where each Poselet could be relative to others [1]. Our work follows this paradigm, leveraging known three-dimensional information in training images to train better identification models.

The poselet approach is then extended by Birdlets [8]. The authors attempt to normalize the poses of birds within an image so that the effective pose is the same between images. This way, the learning algorithm can concentrate on the features of the bird without having to deal with the differences in the birds pose or the camera's viewing angle. The body of the bird is modeled as an ellipsoid and the ellipsoid is rotated to match that of the other birds in the data set.

2.4 Annotation and Active Learning

Active learning is an approach used when annotating images. Instead of a user attempting to provide the labels for all of the images in a system, the system will actively ask the user to provide labels for the images that it feels it is most uncertain about. This type of system can potentially learn faster in this case, reducing the human effort needed to train the system [5, 11]. Our paper takes a different approach to reducing the amount of human interaction by using epipolar geometry between images to annotate multiple images at the same time. We do a similar thing in that we attempt to label images automatically, but instead rely on epipolar geometry (which describes relationships between cameras).

[1] As defined by the paper, a poselet is a part of ones pose

2.5 Learned Segmentation

Long, et al. [14] demonstrated how convolutional networks can be used to segment objects within an image. The idea is to train a model to do segmentation of an image for you. A deep neural network with multiple layers is trained from exact segmentations. This network does not have any fully connected layers. It is made up entirely of convolutional layers, and as such, was called a fully convolutional network. This type of network has the property that it acts as a nonlinear filter to an input image of potentially any size, while other deep networks tend to act more as a nonlinear function producing a small number of output values for a fixed-size input.

Badrinarayanan, et al. [2] also use fully convolutional networks for the task of segmentation. They provide a slightly different approach to the structure of the network separating it into two groups, an encoder and a decoder. The image first runs through an encoder encoder. The encoder is derived from a VGG16 network, which has a lower resolution output. Instead of immediately going to a final layer that does pixelwise prediction, the decoder then takes that output and does a nonlinear upsampling back to the resolution of the input image. It uses values calculated in the pooling layer of the encoder to calculate the values in the decoder. This way, the decoder does not have to go through the same learning process the encoder does in order to upsample. Once trained, the model can segment a variety of images of the same type of object. However, training the model can require a large number of images that are segmented manually.

Newell, et al. [16] took this idea of a encoder decoder model which could be thought of as an hourglass model and improved upon it. They stacked hourglass-shaped sets of network layers together, each set bringing its input down to a small layer and then back up to the same size as the original layer. This allows it to aggrogate information across scales and more effectively handle objects of different size. They developed and trained a convolutional network using this model that was purposed for human pose estimation.

The method used in this thesis for segmentation is based on the stacked hourglass model developed by Newell, et al. In this work, the stacked hourglass method has been selected for segmentation due to the fact that the accuracy of the model from Newell, et al. was higher than many other methods used for segmentation and the authors had already pretrained a model using this method on segmenting images of humans. We were then able to use this pretrained model and fine tune it to segment butterflies. This research can also be used to segment multiple images automatically. While a machine learning model built to do segmentation can be used to continuously segment many images of a target object, the process this thesis proposes will only be able to segment a set of images taken in a controlled environment. However, unlike segmentation through machine learning, it does not take nearly as many images to start segmenting the process using our process. Only a handful of images are needed in order to segment a large number of other images. In this way, our process can be used to feed the machine learning segmentation process.

2.6 Object Recognition through Machine Learning

At its core, object recognition consists of these two phases; first detection, and second classification.

In the first phase, the program cannot make prior assumptions about the target object location or size. The program therefore must identify if and where the target object exists in the image and how much space the target object takes up.

The general approach is to create subwindows of various sizes and locations and search through those sub windows for the target object. Features are extracted from within each subwindow and then run through a machine learning model to determine if an object of interest is present within that window.

Some machine learning models do not require breaking the image down into sub windows to find the target object. Instead it is based off of machine learning models that are trained to segment the target object within the image. The segmentation is then used to generate a bounding box and crop the image around the bounding box.

Once an object has been located in the image, the program moves over to the second phase and must identify which category or class, if any, out of a list of known classes, that target object belongs to. This identification of an object's class, called recognition, is often used to simplify and improve the effectiveness of common perception tasks.

This learning process consists of presenting a set of example images and training a model to identify which features are unique to each class. The training of models consists of running multiple positive images of the objects through a Machine Learning model. This model usually uses either a series of features or filters that get applied to the images one by one. Features are properties of the image that can be measured. Filters are layers of values the pixels in the image get multiplied or added to to change the image in a way that will bring out relevant parts of the image that help determine the class. During the training process, the algorithm will change the values of the filters as it runs through the various images to more closely associate with the given class. The values in each filter will eventually converge so that subsequent training will change the values very little. During testing or detection this model gets applied to an image window, which generates a probability for each particular class. This becomes the estimated object. While a well-trained model will achieve an accuracy in the 90% range, no model will provide 100% accuracy.

2.7 Segmentation through Machine Learning

Another application of machine learning is segmenting the target object out from the rest of the image [14]. The model uses a fully convolutional network and sets the output layer to be the same size as the input layer. When the model gets trained, both the original image and the objects segmentation mask are provided as inputs to the training model. The correct segmentation helps reinforce the training. During testing, a test image is sent through the model. The model creates an output image of the same size, which holds a heatmap as to the segmented object.

2.8 Object Labeling

Xiao, et al. [26] focused on reconstructing an entire room as a three-dimensional model merging human annotations of RGB-D images with Structure from Motion (SfM).

The attempt was to annotate objects in every frame and use those annotations to produce a “better three-dimensional reconstruction” of the scene. They attempted to create a tool that would allow them to annotate each frame of a video more easily. This was accomplished by essentially merging the two needed tasks together; annotating the images would better allow them to create the three-dimensional reconstruction and the three-dimensional reconstruction would allow them to annotate the images more easily. A user uses this to go through frames in a video and label objects within each frame. For a new frame, the system initially attempts to automatically label it by transferring information from other previously-labelled frames. The user is then able to correct any errors in this automatic annotations.

Xiao, et al. are attempting to merge human annotation with three-dimensional reconstruction to allow for easier annotation. However, they are using RGB-D cameras instead of color-only cameras. Most RGB-D cameras can only be used indoors where infrared radiation is limited, theoretically restraining the possible recording locations. They also require the user to label every frame, or at least look at every frame to verify that the label is correct. Our method, however, can leverage the labels provided on just two to three frames and automatically label all of the other frames without the need for human oversight or the requirement to be indoors.

Chapter 3

Building a Dataset

Most image datasets used for object recognition provide only a set of images and the labels associated with the primary object present in each image. ImageNet [6] is one of the largest of such datasets. Such data, however, is unstructured as it does not provide any other associated information about the image. Some datasets attempt to provide additional structure such as bounding boxes surrounding the target object. Segmentation, a detailed surrounding shape of silouettes of the target object, may provide better information, but is even more rare.

Another system, LabelMe [21], attempts to add more information to each image by using the idea that there is more than one object within an image and each one needs to be identified. An outline is provided for each object within each image. While that information is beneficial, it is costly and takes time to create. This thesis is focused on creating needed annotation information for all images in a group by only needing to manually annotate a few of them. This will, in turn, both save time and money. To do this, all of the images in a group must be taken of the same exact object (and scene), but should be taken from multiple angles.

By doing so, we create image recognition datasets of high quality that provide more detailed information about the objects which can then be used while training vision systems. Adding this type of data to an existing dataset can increase the accuracy with which a system can learn to recognize a target object under varied imaging conditions. Instead of trusting that a dataset has a sufficiently diverse set of viewpoints in its set, this work seeks to explicitly provide examples of the object from multiple viewpoints, thus creating the diversity needed for the dataset.

This thesis attempts to solve a few common problems observed in the process of building image datasets. The first is providing images of an object from a wide range of viewing angles. Taking a set of images from below the object is most likely not necessary, as we anticipate that most images will be taken from the side or on occasion above the object. This will be done by taking multiple images of the same object from multiple angles using one of two methods described later. It is important that the object remains static, i.e. in the same location and position, until all images within a group has been captured. This is needed to reconstruct the 3D locations where each image was captured, locations which allow the keypoints and segmentations of the object to propagate to all of the images in the group.

3.1 Internal Camera Calibration

The first thing that needs to be done is to calibrate the cameras used during the capture. Cameras inherently cause distortion in the images they capture. While this distortion is often so small that it goes unnoticed by human eyes, it can significantly hinder the calclulations for certain processes like structure from motion. Because of this, the images must be undistorted before applying Structure from Motion.

There are two main causes of distortion, radial distortion which produces the fisheye look and tangential distortion which is caused by an unaligned lens. Both are caused by imperfections in the creation of the camera and more specifically the creation and mounting of the lens within the camera. Because all cameras have some imperfections, all images have some distortion. In order to generate a more accurate 3D model of the object and therefore 3D location of the cameras taken of the object, the images must be modified to correct for these imperfections.

The process begins with estimating parameters of the distortion. One of the most common methods to do this is is to use the camera to take multiple pictures of a checkerboard where the location, scale and orientation of the checkerboard relative to the camera are varied. Because the checkerboard consists of multiple straight lines, any deviation of those lines shown on the images is due to distortion. These distortions are then estimated and then formed into a camera matrix and a vector of distortion coefficients, used together to reverse the distortions.

In our implementation, OpenCV is used to calculate these parameters and then undistort the images. These parameters are then applied to every image taken from the same camera, reversing the distortion on those images. Once the distortion has been calculated, the same distortion parameters can be used to undistort multiple images and captures taken by that camera so long as properties like zoom, lens used, etc. remain the same. If multiple cameras are being used, then each one must be calibrated independently.

3.2 Arch

The next step is the capturing process. We use one of two methods to capture the images. The first is a rotating multi-camera system that allows for many images to be captured automatically. The second is by freehand using a camera and moving it around the object taking multiple pictures from different angles

Using the arch method consists of taking a wooden semi-circular arch has sixteen GoPro cameras evenly spaced along the inside of the arch. The cameras are staggered so that each camera on one side of the arch is placed vertically half-way between two cameras on the other side. This allows us to sample more densely latitudes of the hemisphere of viewing angles than if the cameras were mounted symmetrically. There is no camera directly above the object as rotating on that camera would not provide any new information. Instead, the first camera is offset from the axis of rotation to allow for more information to be collected by that camera and to allow the cameras to be staggered evenly. The arch is unable to record a moving object as it takes time for the arch to turn around the subject. Due to the GoPro's wide-angle lens, the subject shown in the image takes up a smaller area on the image than if

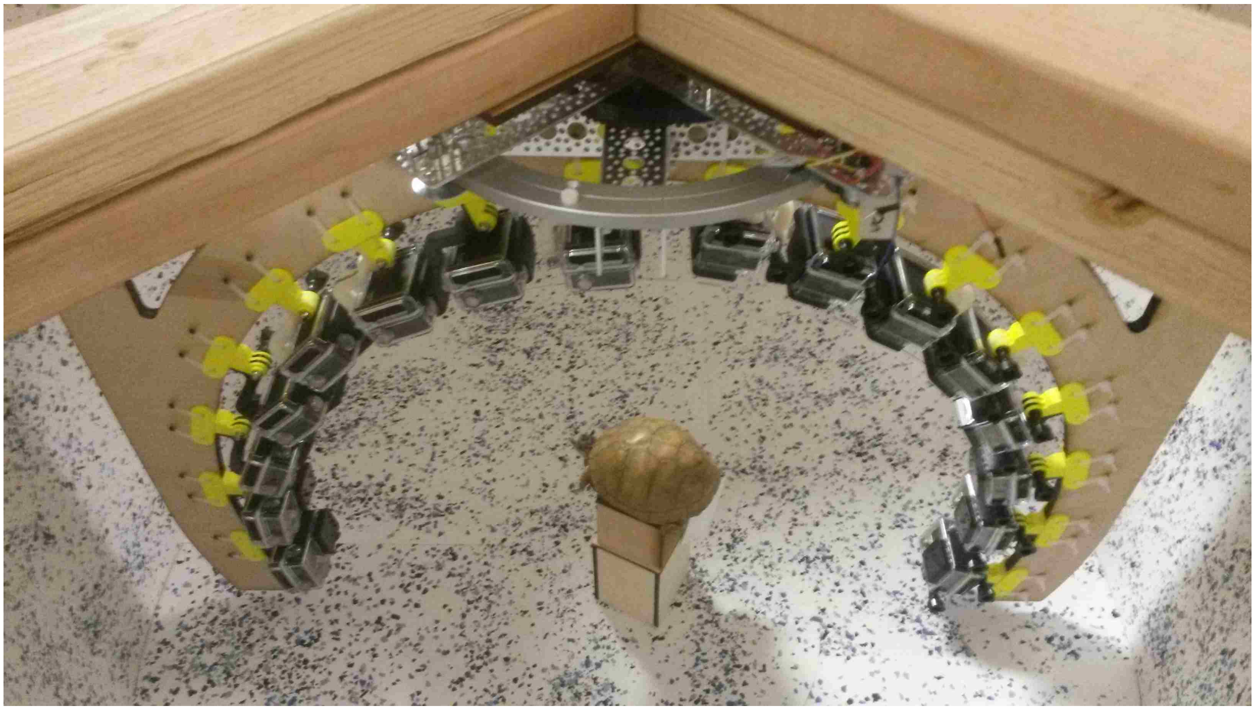
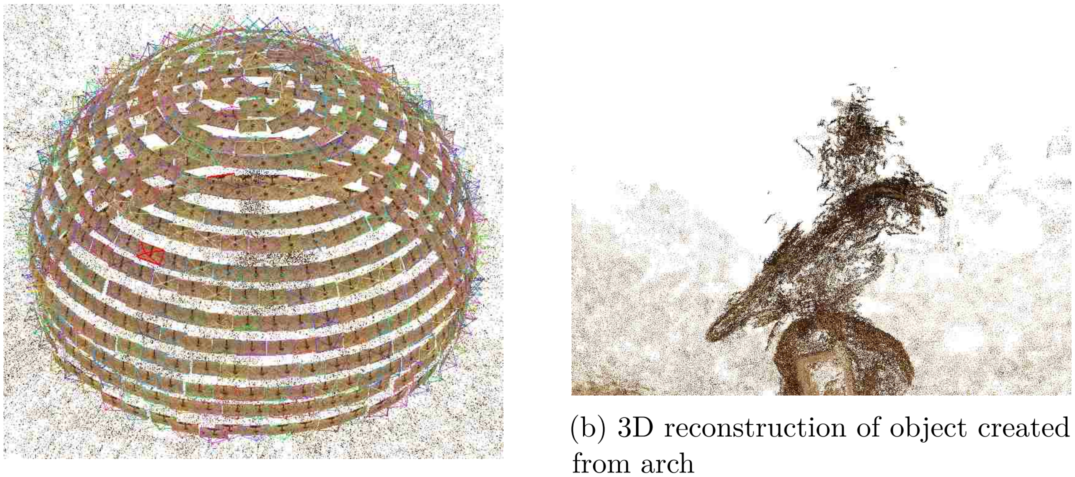


Figure 3.1: Arch containing 16 cameras

the camera lens was narrow. In order to allow the subject to take up a large enough area on the image, the subject must be relatively large in order to produce a detailed 3D model. In a relatively short amount of time, over 600 evenly-spaced images may be produced, resembling the creation of a higher resolution model. Figure 3.3 is a drawing of the design used to create the arch. The arch is roughly 30 inches in diameter in the interior. The cameras are mounted around the arch and cover over 180°of a full circle. The arch is suspended and rotates around the vertical axis of symmetry.

The cameras used in these data captures are GoPro Hero 3+ black which have a viewing angle of roughly 120°. Figure 3.1 shows the arch from an approximately top down view with all cameras around the perimeter. Each camera is pointing towards the target object in the center, the tortoise in this case. Figure 3.2a shows the resulting camera locations of the images used for the reconstruction and model training. Figure 3.2b is a resulting 3D reconstruction of a quail taken from the arch and generated through SfM.



(a) Resulting camera locations of the arch

(b) 3D reconstruction of object created from arch

Figure 3.2: Arch

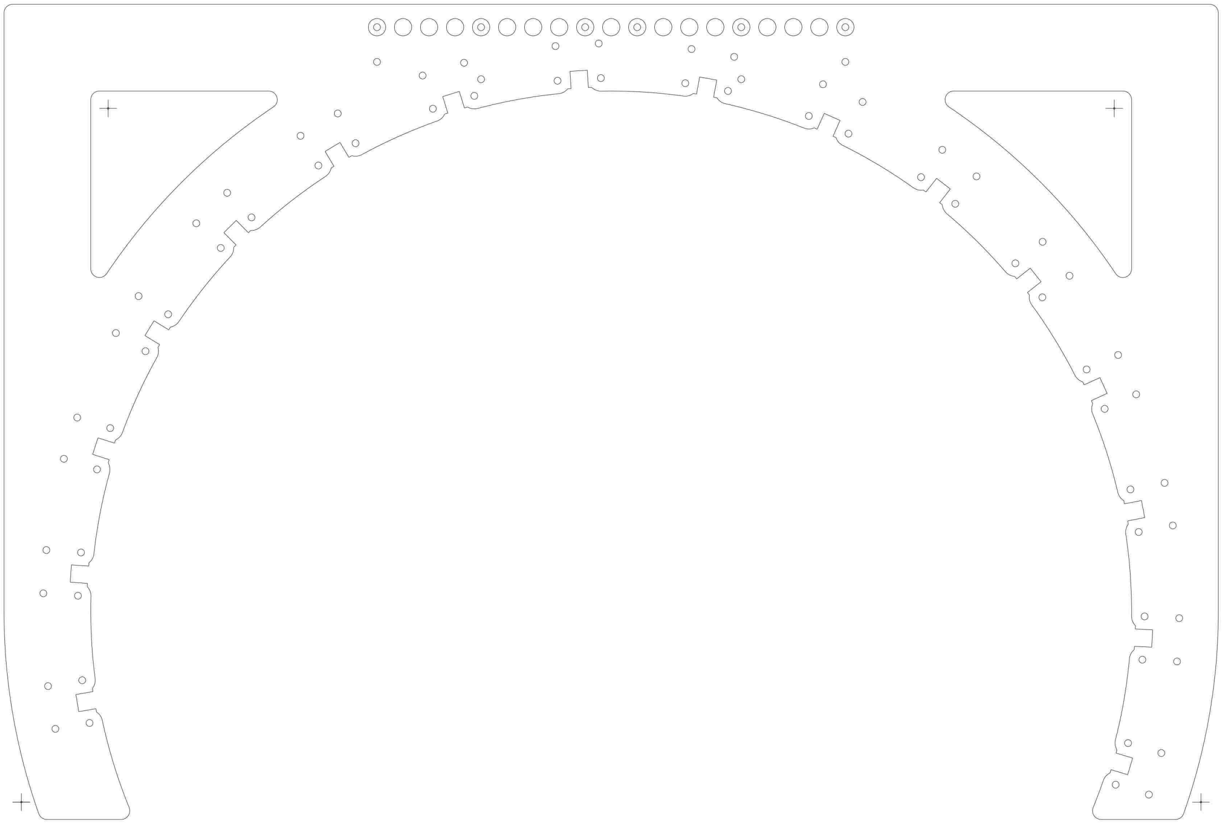


Figure 3.3: Arch drawing

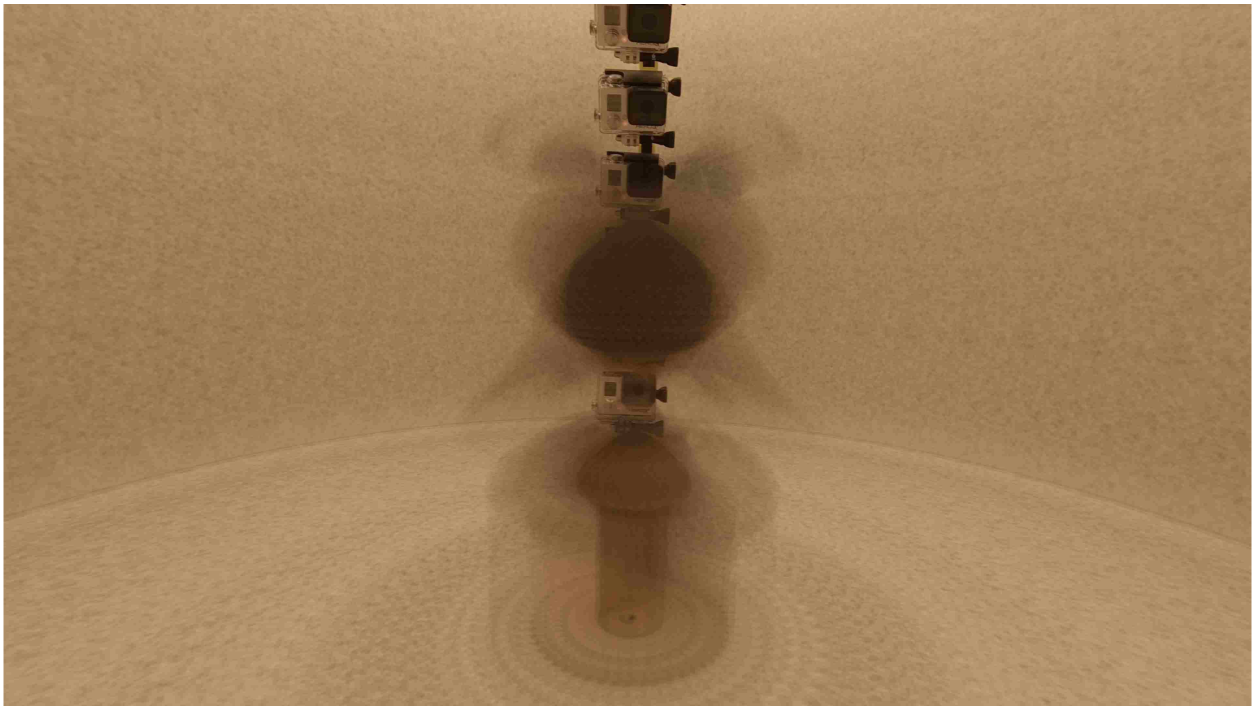


Figure 3.4: Image produced by averaging all of the images from a single camera as it spins around the arch

A drawback with the arch as it is currently designed becomes evident when doing the SfM. Because the arch is roughly semicircular, one side of the arch can see the other side of the arch in all images. This other side remains stationary within all of the pictures while the rest of the scene changes. This confuses the SfMalgorithm as the locations of points on the opposite side of the arch do not change while the arch is rotating. This throws off the matching of points. Figure 3.4 depicts the "average image" observed during its full rotation. As can be seen, the cameras on the opposite side of the arch remain constant as the rest of the image changes.

Because of this problem with observing the opposite side of the arch, a cloud of points appears above the reconstructed object, as seen in Figure 3.2b. While the locations of the cameras can still be generated accurately despite this impediment, subsequent algorithms applied to the 3D model are not as effective due to the hallucinated noise in the resulting point cloud.

Future work could include methods to remove that point cloud noise either in post-processing or in acquisition. One possibility would be to create an arch that is only a quarter of a circle in shape. This would allow for no single camera to see any other camera in any of the pictures. However, the weight of the arch would be offset and must be balanced by a counter weight. Furthermore, two quarter circles could be offset by less than 180°.

Another possibility is to use a single camera that can move along the semi-circle as well as around the hemisphere. This would provide a robust system and would be cheaper as the cameras are the most expensive part of the arch and with this system only a single camera is required. However the mechanism would be more complex to design and build, and would take far longer to run as the single camera would have to move into position and take all of the photos that are currently collected in parallel with multiple cameras.

3.3 Freehand

The arch provides a way to capture many images in a short amount of time. However, it is not free of drawbacks. As a large device, it is not very portable. Further because it uses cameras with wide angle lenses, it is best when taking pictures of relatively large objects. In addition, the noise created in the point cloud because of observing the opposite side of the arch can cause problems when working on the 3D model generated. The reconstructed models have noise in the point cloud due to observing the opposite side of the arch. (See figures 3.4 and 3.2b). Another option we have evaluated is taking the pictures freehand. The camera used can be as simple as a mobile phone camera or as complex as a DSLR camera. As before, it is most effective when the camera is calibrated to remove any distortion within the images. This method takes more time than the arch and does not allow for even spacing or numerous images to be collected. However, it does not require a large apperatus or an arch setup to be used.

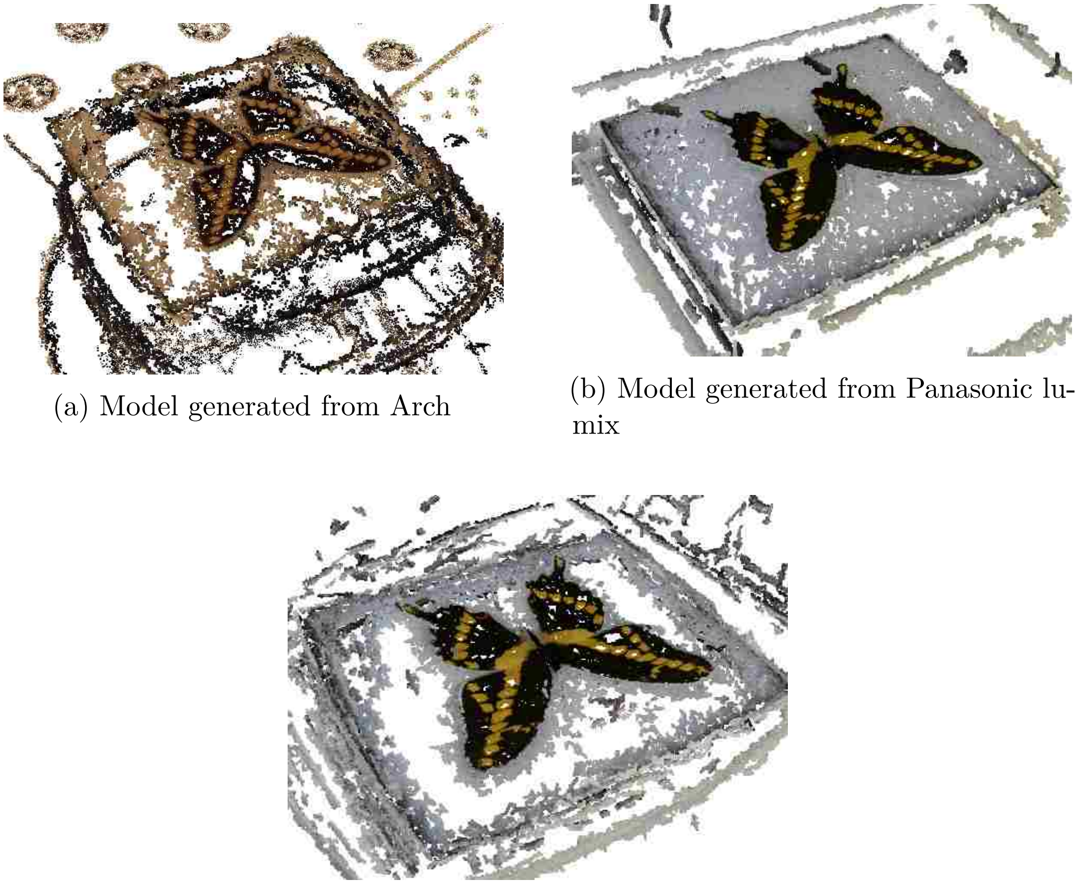
We use two different types of cameras in this process during the testing. The first type of camera used in the freehand setup is a Panasonic Lumix DMC-FZ1000. This is a professional “Bridge” camera with higher-quality optics. And therefore produces higher-quality images with little distortion. However, at approximately $800, its cost must be taken into account. The second camera used was the same camera on a Samsung Galaxy Note 4 smartphone. While this smartphone is similar in price, its sensor and optics are smaller and this inferior. As this is typical of the type of camera that most people would already have, it would not require additional expense to obtain the data.

3.4 Structure From Motion

Structure from motion (SfM) [10, 25] can be used to generate a 3D point cloud of a static target object. The algorithm requires many images of the object and numerous points that are visible in multiple images to simultanously estimate both the 3D locations of these points and the camera locations and orientations. Recovering the extrinsic parameters of each image is critical for automating the annotation process.

Figure 3.5 shows the model generated for a Giant Swallowtail Butterfly (*Papilio cresphontes*) reconstructed from images using each of the capturing methods mentioned above. As can be seen, the model generated using the Panasonic bridge camera produced the best results. Specific 3D points common to multiple images were found in the Panasonic model that were missed in the other two. As a result, points are filled in where they are missing from the other two models, producing a higher-resolution and better-quality model. The Panasonic camera has the highest resolution and the highest quality optics among the three cameras used.

The model generated using the arch method produced the lowest-quality model. This is most likely due to the optics of the GoPro cameras used and fewer scene points found during reconstruction. GoPro cameras have a high radial distortion, a wide angle lens and a relatively low resolution. GoPro cameras are, in addition, expensive and using 16 of them for the arch cost far more than the bridge camera or smartphone would. However, using the arch provides the advantage that little time is required to capture the images needed for



(a) Model generated from Arch

(b) Model generated from Panasonic lumix

(c) Model generated from Smartphone

Figure 3.5: 3D Models of a swallowtail butterfly specimen, each generated using a different method of image capture

the model; roughly 30 seconds per object. The smartphone method provides the additional benefit that it is effectively free. Many people have smartphones with cameras, so no extra materials are needed. The models produced with the smartphone camera were right between the other two approximately in terms of quality.

It is important to note that the quality of the reconstructed model is secondary in importance. What is critical is the accurate recovery of the location and orientation the of the camera for each image collected. This provides the necessary information to do multiview annotation with very few manual annotations. Using this method, we can do segmentation and keypoints with potential for other kinds of annotation.

Chapter 4

Multiview Annotation

A system can learn better when provided with images that are segmented [18]. To help facilitate this desire, we want to be able to provide the segmentation of the object as well as images of the object from multiple angles. However, providing this extra data manually can become cumbersome when dealing with a large dataset. We have created a method to simultaneously annotate many images of an object without having to manually go through each image one-by-one. This multiview annotation process can provide various annotations including segmentation, object parts, pose information and other information.

While annotating a large list of images typically requires a user to annotate each image independently, using the proposed method requires marking only a few images and the rest of the images are then annotated automatically. We focus on two types of annotation. The first is keypoint annotation, which consists of locating specific points of interest on the object in a few images. The location of this point is then calculated in the rest of the images. The second is segmentation, which consists of outlining the entire object within a few images. This outlining is then calculated in the rest of the images.

An image of an object depicts what is visible from a camera at a given location and in a particular orientation. A point in the image corresponds to a ray in three-dimensional space, extending from the optical center of the camera out through that point in the image plane. The following equation finds that three-dimensional line based off of a point of interest clicked in the camera. The location, orientation and focal length of the camera must be known. The first two are more often referred to with the geometric terms translation and

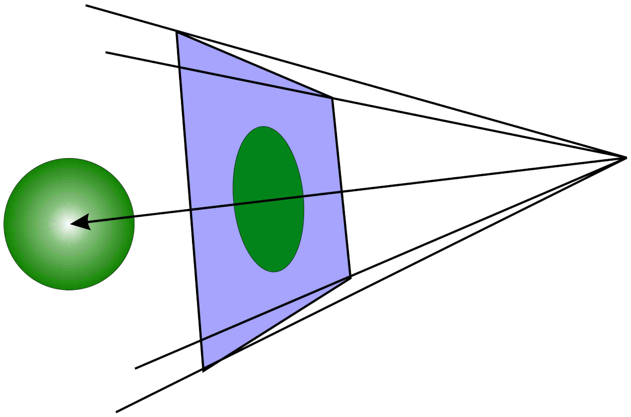


Figure 4.1: 3D line created by clicking point in camera

rotation and are commonly denoted as T and R. Also, the point of interest is located relative to the center of the image. This point of interest is identified as x and y.

This calculated point is as follows:

$$\begin{aligned} \mathbf{p} &= [x, y, f] \\ \mathbf{d} &= R^{-1}\mathbf{p}^{T} \\ l &= (T, \mathbf{d}) \end{aligned} \tag{4.1}$$

The resulting 3 Dimensional line that is represented by the click on the viewing plane of the image is identified by l. This line is infinite in length running in both directions containing the location of the camera(T) and point(D). This calculated point is a distance of the focal point away from the camera. This is shown in Figure 4.1

4.1 Multiview Keypoint Annotation

Annotating the data allows for easier localization of the parts of an object within the image. If the system knows where each part of the object is within each image, it can better isolate that part and focus on what it looks like from different angles. There is a problem with focusing on parts however, as a given part may be visible in one image, but obscured in another. Even if the point is occluded, the system will calculate where it would be if it were not occluded. Therefore, there may be some parts that are included within the machine learning samples that are not part of the object they are calculated to be. However, the location of the point projected onto the image could potentially tell whether the point is occluded or not. It is possible in that case that occluded three-dimensional points can be used to help with the learning process as long as the system knows the points are occluded and takes that into account.

Nearest Point to Lines

All the images taken of a given object are collectively referred to as a batch. After a batch is taken, the three-dimensional geometry of the object is recovered using SfM [10]. After their reconstruction, the location of each camera is known to the system. As a result, the translation and rotation of each camera relative to any other camera is also known. The user is able to click on a point of interest in an image. Because an image is a two-dimensional projection of a three-dimensional world, a point on the two-dimensional image plane corresponds to a ray in three-dimensions extending from the cameras optical center and piercing thee image plane at that point. When viewed from another camera, the projection of this ray onto that camera is called an epipolar line. Figure 4.2 shows the epipolar lines shown on one image from the point specified in the other image. This ray can be calculated as a three-dimensional ray in world space and then, using the translation and rotation relationship between cameras, it can be projected as an epipolar line onto the other images within the batch. The point of interest the user clicked on is somewhere on this ray. However, because it is a ray, it has an infinite number of possibilities as to the location of the actual point of interest the user has specified. The three-dimensional location of this selected point of interest is found somewhere on this epipolar line in the other images within the same batch. The user is then able to click on the same point of interest in a second image, producing a different epipolar line in each image. Where these two epipolar lines cross is the location of the three-dimensional point of

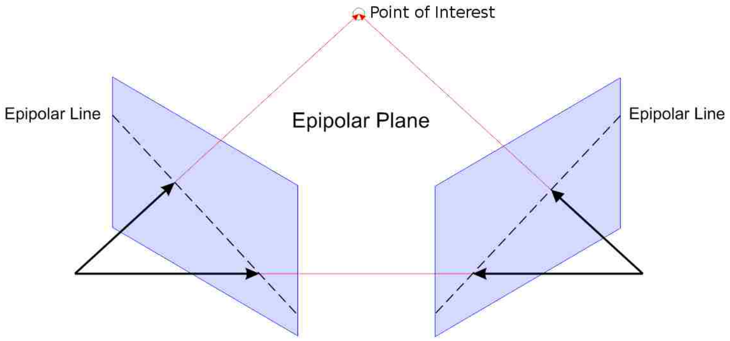


Figure 4.2: Epipolar geometry

interest the user has clicked on. While only two images need to be specified to calculate a specifc three-dimensional point, this process can be repeated for other images, increasing the reliability of the estimated three-dimensional location. These points will continue to cross on all other images at the same location. The three-dimensional location of the point of interest can be calculated from the intersection of or closest point to these lines. This point can then be projected back onto all of the other images. Therefore, by specifying a single point on a few images, that same point can easily be calculated in all of the other images within that batch.

A difficulty with this method however is imprecision in input. Because of these errors, the rays specified will not generally intersect. Therefore, simply calculating where the three-dimensional rays intersect will not work most of the time. However, a three-dimensional point can still be calculated by minimizing the distance between the rays. While only two images are needed, the more images that are annotated, the more accurate the position of that point can be calculated by minimizing the distance of the calculated point to each line provided. This nearest point, projected back onto all of the images will indicate the location of the part of interest within the images without the user having to go through each one. Minimizing the calculated three-dimensional point will in turn minimize the error of the point projected back onto the images. This method saves great amounts of time. In the following equation the rotation, translation and focal length of each camera is needed. These are identified as (R_1, $\mathbf{t_1}$, f_1), (R_2, $\mathbf{t_2}$, f_2) respectivly. In addition, the point of interest identified by the user is needed in camera 1. This point of interest is located in pixel values relative to the center of the image.

The calculation done for this is as follows:

$$\begin{aligned}
\mathbf{p_1} &= [x, y, f] \\
\mathbf{d_1} &= R_1^{-1}\mathbf{p_1}^T \\
\mathbf{t} &= R_1 * (\mathbf{t_2} - \mathbf{t_1}) \\
\mathbf{n} &= R_2 R_3^{-1}(\mathbf{t} \times \mathbf{p_1}) \\
\mathbf{p_{2L}} &= [-h_y, 0, f_2] \\
\mathbf{p_{2R}} &= [h_y, 0, f_2] \\
y_L &= \frac{-(\mathbf{n_x p_{2L}} + \mathbf{n_z p_{2L}})}{\mathbf{n_y}} \\
y_R &= \frac{-(\mathbf{n_x p_{2R}} + \mathbf{n_z p_{2R}})}{\mathbf{n_y}}
\end{aligned} \tag{4.2}$$

4.2 Segmentation

Segmentation is done using a method called voxel carving [13]. If an object has been segmented within an image, every pixel within that image will either be in or outside that segmentation. Voxel carving consists of making a three-dimensional block made up of smaller stacked three-dimensional blocks called voxels. These voxels are stacked next to each other on all sides to produce a block. Figure 4.3 shows a stacked block of 4x4 voxels. The voxels in this figure are not stacked up right next to each other in order to show the separation of the voxels themselves. However, in the multiview geometry algorithm, all of the voxels are stacked right next to each other to form a large three-dimensional hyperrectangle.

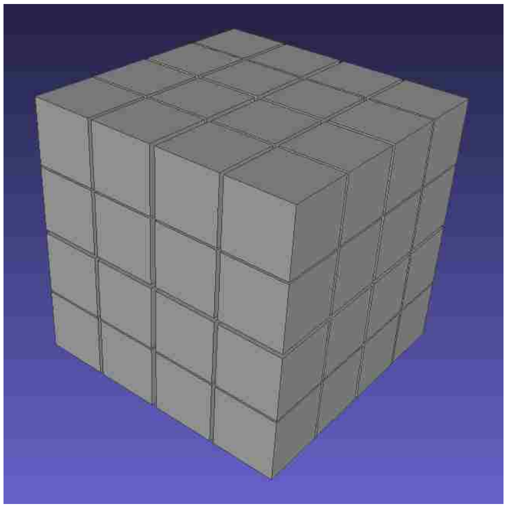


Figure 4.3: A cube stacked with voxels

Each voxel in that hyperrectangle is projected onto the images that have been segemented. If the point of that voxel occurs within the segmentation of all images that have been segmented, then that voxel is considered within the object. The location of that voxel when projected onto the rest of the images is defined as foreground to indicate it is part of the segmentation of the object. If however, that point does not occur within the segmentation of one of the images, it is then considered outside of the object and the location of that voxel is then defined as background within the rest of the images.

The size of the original hyperrectangle encompasses the entire model created from structure from motion to allow for the entire segementation to take place. If the voxels did not take up the entire model, then there is potential for part of the butterfly to not be involved in the carving process and therefore missing in the segmentation. The size of the voxels represents the resolution of the model. The higher the resolution, the smaller the voxel must be. The smaller the voxel, the more there must be of them to make up the same space of the larger cube. Each of the voxels gets tested one by one to see if it is part of the model and therefore must be included in the segmentation. The more voxels there are, the more higher the resolution of the segmentations will be. However, the more voxels there are, the more time it takes to calculate the segmentations. For these models, we chose to create a cube of 500 × 500 × 500 producing a total of 125,000,000 voxels that will be carved down and used to produce the segmentations for the other images.

This happens by choosing a group of images within the set and then segmenting them manually. The location of each voxel is then backprojected onto each manually segmented image. If the location of the voxel falls within the segmentation of all of the manually segmented images, then we can assume it is part of the object and will keep it. If however it does not occur within the segmentation of at least one of the manually segmentated images, we can assume that it does not occur within the object and can be removed from the model. Once all of the points have been tested and only those that remain within the segmentation of the manually done images remain, the location of the voxels represent the model in three-dimensional space. A second run of projection occurs next using the same method as before but instead of projecting a single point location of each voxel onto the images, the full voxel is projected and rendered back onto each image one by one. This is done by projecting each point of all six rectangular sides of the voxel and filling in those rectangles as part of the segmentation. Because the sides of the cube share common edges and points, only eight unique points need to be projected to produce the rendering of the voxel. This produces a segmentation mask onto each image showing where in the image the model shows up.

The following equation is used to backproject the voxels back onto the images in question. In the following $\mathbf{p}$ represents the point in three-dimensional space of the model. The cameras rotation matrix is represented by R and the translation is represented by T. The focal length is represented by f. The center of the image is located at point $\mathbf{c}$. The result is identified by $\mathbf{s}$.

$$\begin{aligned} n &= R(\mathbf{p} - T) \\ s &= (\mathbf{c} + n_x/n_z)f \end{aligned} \tag{4.3}$$

By allowing for segmentation of many images to be done quickly and easily by segmenting only a handfull of images manually, this method greatly reduces the time and effort needed to segment all of the images. The segmentated images then help in the learning process of a machine learning model to identify the classes of a given subject. This is evaluated further in the next chapter.

Chapter 5

Validation

Recognition algorithms can only perform well if the data they have been given to train on is representative of the images they will eventually need to recognize. A training set is called noisy if it has many inaccuracies such as mislabelled images within the set. It is much more difficult for a system to be able learn correctly from a noisy set of data. On the other hand, a dataset with low noise leads to higher accuracy and allows the system to learn more easily. Current datasets are often built using novice users to provide the image labels.

Amazon's Mechanical Turk is a global marketplace used to buy and sell tasks. Often times it is tedious tasks that most people can do, but it may take a long time to do. When a task is required, the person who needs the task done (called the requestor) may post the needed task on Amazon Mechanical Turk. Workers, from all over the world will accomplish these needed tasks for a small fee. The service does have its drawbacks though. It is generally less accurate than if experts were labelling the data. Welinder et al. [24] notes various problems that occur when using non experts to annotate images. Such problems consist of annotators not knowing the distinctions between classes that look similar, but are different. Other problems are with annotators applying different levels of an acceptable error rate when annotating and annotators producing inconsistent labeling. As a result, many researchers using Mechanical Turk use multiple annotators to produce the annotation needed. This in turn makes it more costly when annotating at a large scale. However, it will usually produce better results as it averages the labeling of multiple annotators as opposed to relying on a single one. Welinder et al. came up with a Bayesian model to help reduce the number of annotators, while keeping the accuracy up. This produced positive results, but still required a multiple number of annotators to annotate each image.

Because this was a tedious task that needed to be done but would take a long time to do, we used Mechanical Turk to have the images within the dataset segmented. We then take a subset of those segmentations in varying degrees and generate a model using those segmentations. We used 5 annotators for each image within our dataset and averaged the segmentations together to produce an overall baseline for the images within the model. The average was calculated by taking each pixel and within the image and determining if most of the annotators felt that pixel was part of the segmentation or not part of the segmentation. If the majority felt it was part the segmentation, it was declared part of the segmentation for the baseline.

5.1 Generating Segmentation through Machine Learning

Machine learning using Convolutional Neural Networks (CNN) [12] are one of the most commonly used methods of object detection within images. Images are run through layers within the network where pixel values are multiplied and added by a given set of numbers. These numbers are learned by the system as it goes through the training to generate the differentiator between different types of objects. These learned values are then used during testing to determine the class of a given unknown object.

Long et al. [14] produced a method to use Fully Convolutional Networks to do segmentation on an image. Fully Convolutional Networks are used because they do not have any fully connected layers. This gives them the ability to learn to do segmentation on an entire image as opposed to identifying the class of an object within the image. The method we use is based on the work by Newell et al. [16]. A stacked hourglass shaped model was used in training the model and then segmenting the images. The stacked hourglass and pretrained model were found off the developers site.[1]

[1] http://www-personal.umich.edu/∼alnewell/pose/ and https://github.com/anewell/pose-hg-train

5.2 Identifying Class Through Machine Learning

The manual and multiview segmentations were done on the images in the arch and panasonic datasets. A CNN was trained on the existing dataset and then run on the test set to create a baseline accuracy. Then subsequent training and tests were run for the other sets using the same setup. The accuracy for the training on each set of segmentations is calculated by comparing it on the testing set for that dataset as well as the testing set for the other datasets it was not tested on. A VGG16 network[20] was used to train and run the classification. The network and pretrained model were found off of the developers github repository.[2]

5.3 Types of Experiments

The core experiments are to identify how accurate the automatic segmentation process is. For testing purposes all images in the model were segmented by Mechanical Turkers to establish a baseline accuracy of the segmentations. Tests were then run using different number of manual segmentations to generate the multiview segmentations. These generated multiview segmentations were then compared to the equivelant baseline images to calculate an accuracy of the multiview segmentations with each set of tests. The segmentation accuracy was calculated using the same equations found in PASCALS VOC challenge [7].

$$\text{SegmentationAccuracy} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive} + \text{False Negative}}$$

A number of different experiments were created to produce the best likely scenario of producing the segmentations. We wanted to minimize the number of images used as seed images while maximizing the efficiency of the model. The fewer the number of images, used as seeds, the less human work that needs to be done and the more we can rely on the calculations. The flip side however, is that the fewer number of seed images, the less

[2] https://gist.github.com/baraldilorenzo/07d7802847aaad0a35d3

Number of Images	Accuracy
3 images	62.18%
5 images	72.85%
7 images	77.12%
10 images	77.85%
15 images	72.36%
20 images	73.10%

Table 5.1: Accuracy of Segmentation

carving that is done on the voxel hyperrectangle which means the less accurate the projected segmentations are.

Thirteen different tests were done with varying degrees of number of seed images and varying angles. The different models were three images taken solely from the top, three images taken solely from the side, three images taken from all over, five images solely from the top, five images solely from the side, five images from all over, seven images taken solely from the top, seven from only the side, seven from all over. The rest of the segmentation models created were all mixed, they are in groups of ten, fifteen, twenty, and twenty-five.

The average accuracy for the models generated is shown in Figure 5.1 and the values shown in Table 5.1. As can be seen, from the chart, ten images produces the best overall quality for the segmentations. The reason for this is unknown as it would seem that more images would produce progressively higher accuracies. However, this could be due to the idea that more than ten images takes out too many voxels and so there is not enough left over to produce as accurate of a segmentation in the rest of them.

The segmentation for the some of images is shown in Figures 5.2 and 5.3. Figure 5.2 shows the images of the Swallowtail and Buckeye butterflies using three different cameras to capture. Images 5.2a and 5.2d is taken using the arch. Images 5.2b and 5.2e were taken using the cell phone and Images 5.2c and 5.2f were taken using the panasonic. As can be seen the cell phone and the Panasonic produce the best images. The color is better in the cell phone and Panasonic images and the resolution is higher for the butterfly itself. This is mostly

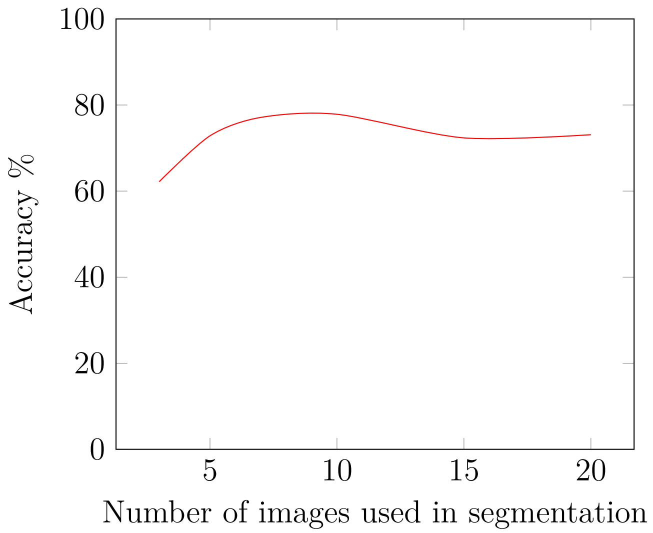
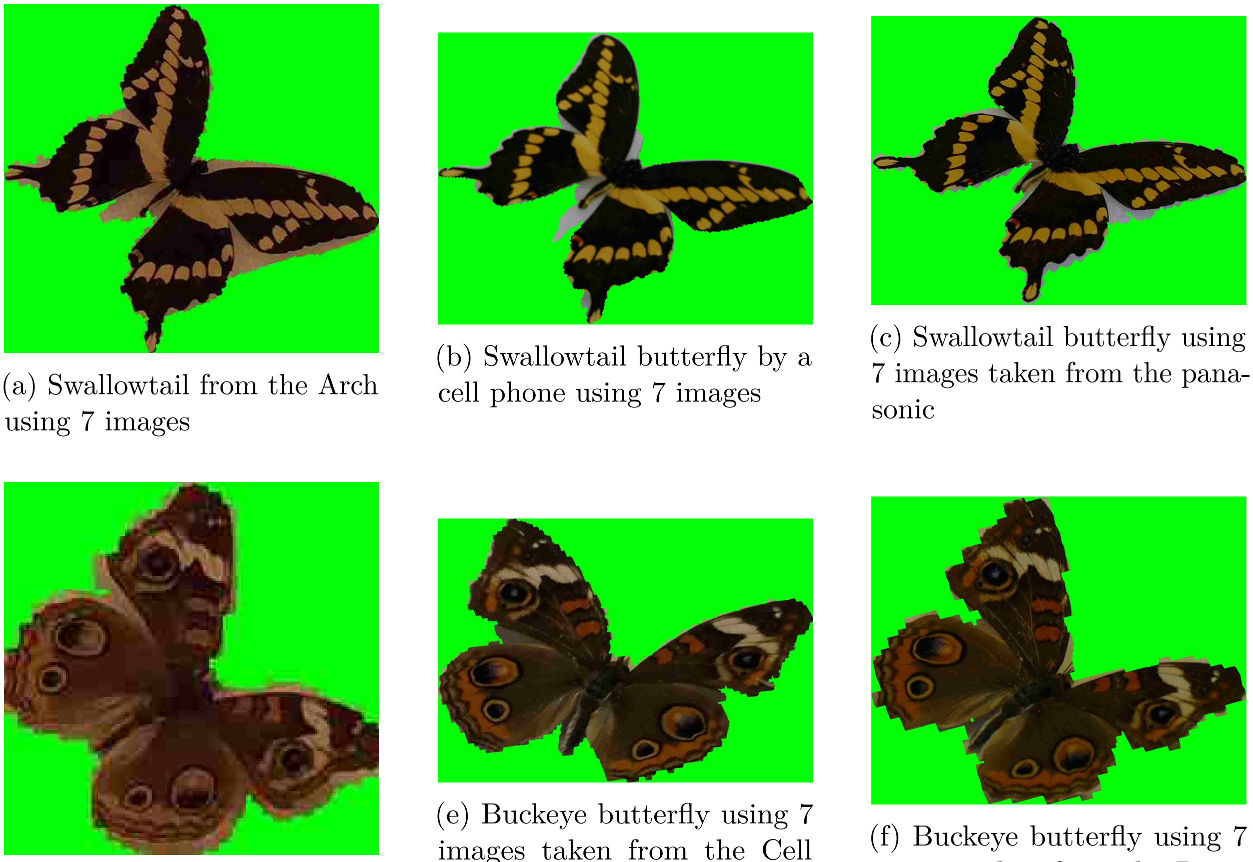


Figure 5.1: Accuracy of Segmentation as plot

because GoPro cameras have a wide viewing angle and while they will take high resolution images, those images taken only a small area within the image that is of the actual butterfly.

Figure 5.3 shows the segmentation of the swallowtail butterfly calculated using different sets of images for the seed for the calculations. Figure 5.3a shows the segmentation created using only three images as the seeds. All three images are taken from an angle above the butterfly. Figure 5.3b shows the segmentation created also using only three images, but these three are taken only from the side of the butterfly. Figure 5.3c shows the segmentation calculated also using three images as the seed. However, these images are taken from both above and the side of the butterfly. As can be seen, out of the first three segmentation calculations, the one taken from only the sides produces the best image. This is probably because the one with the images taken from the side provides the largest angle between them so the model is cut the most, producing a more accurate 3D model and therefore a more accurate segmentation onto the rest of the images.

The last three images used a larger number of seeds as the base. Figure 5.3d is taken using 7 seeds from multiple angles. Figure 5.3e is taken using 10 images and Figure 5.3f is shown using 20 images taken from multiple angles as the seeds. Figures 5.3e and 5.3f produce



(a) Swallowtail from the Arch using 7 images

(b) Swallowtail butterfly by a cell phone using 7 images

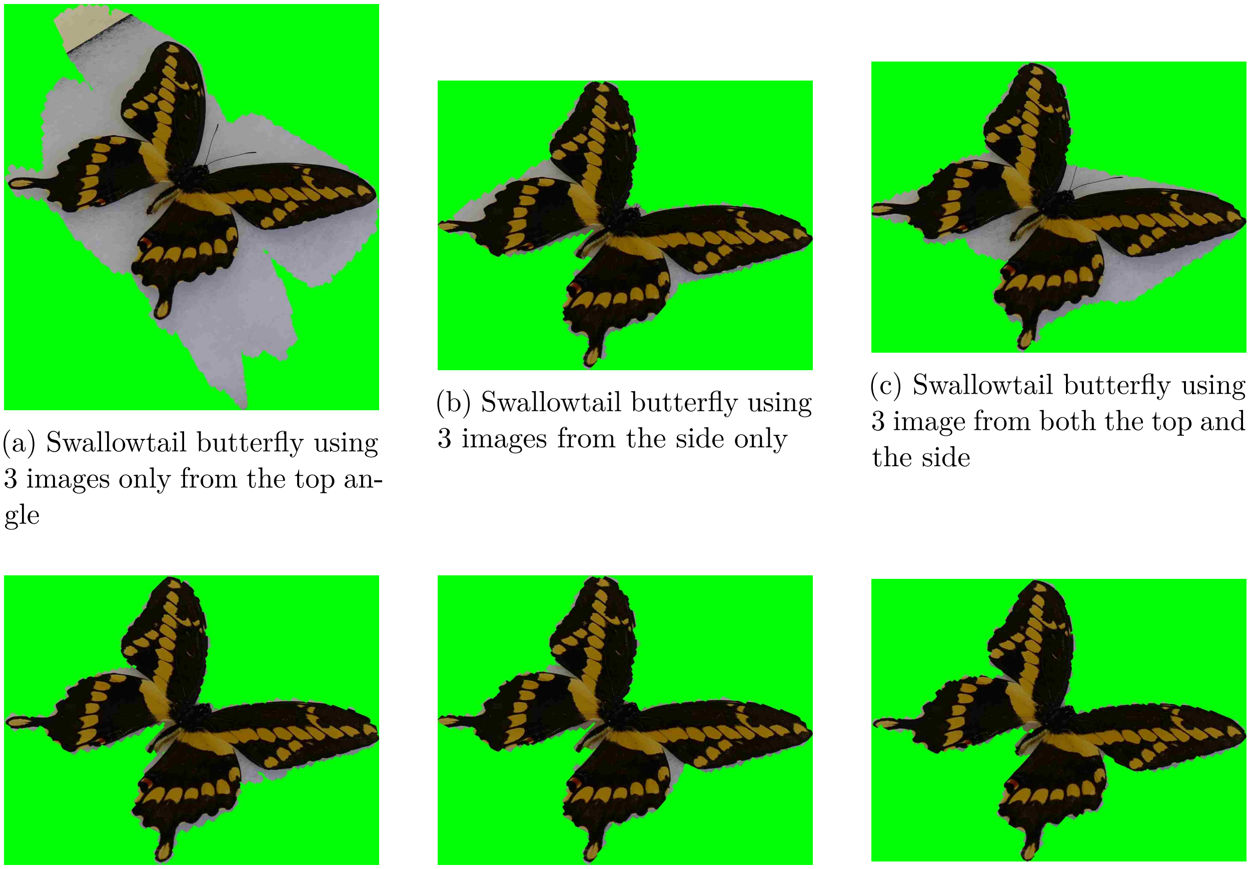
(c) Swallowtail butterfly using 7 images taken from the panasonic

(d) Buckeye butterfly using 7 images taken from the Arch

(e) Buckeye butterfly using 7 images taken from the Cell Phone

(f) Buckeye butterfly using 7 images taken from the Panasonic

Figure 5.2: Segmentation calculated on the Swallowtail and Buckeye butterflies using different methods of capture



(a) Swallowtail butterfly using 3 images only from the top angle

(b) Swallowtail butterfly using 3 images from the side only

(c) Swallowtail butterfly using 3 image from both the top and the side

(d) Swallowtail butterfly using 7 images

(e) Swallowtail butterfly using 10 images

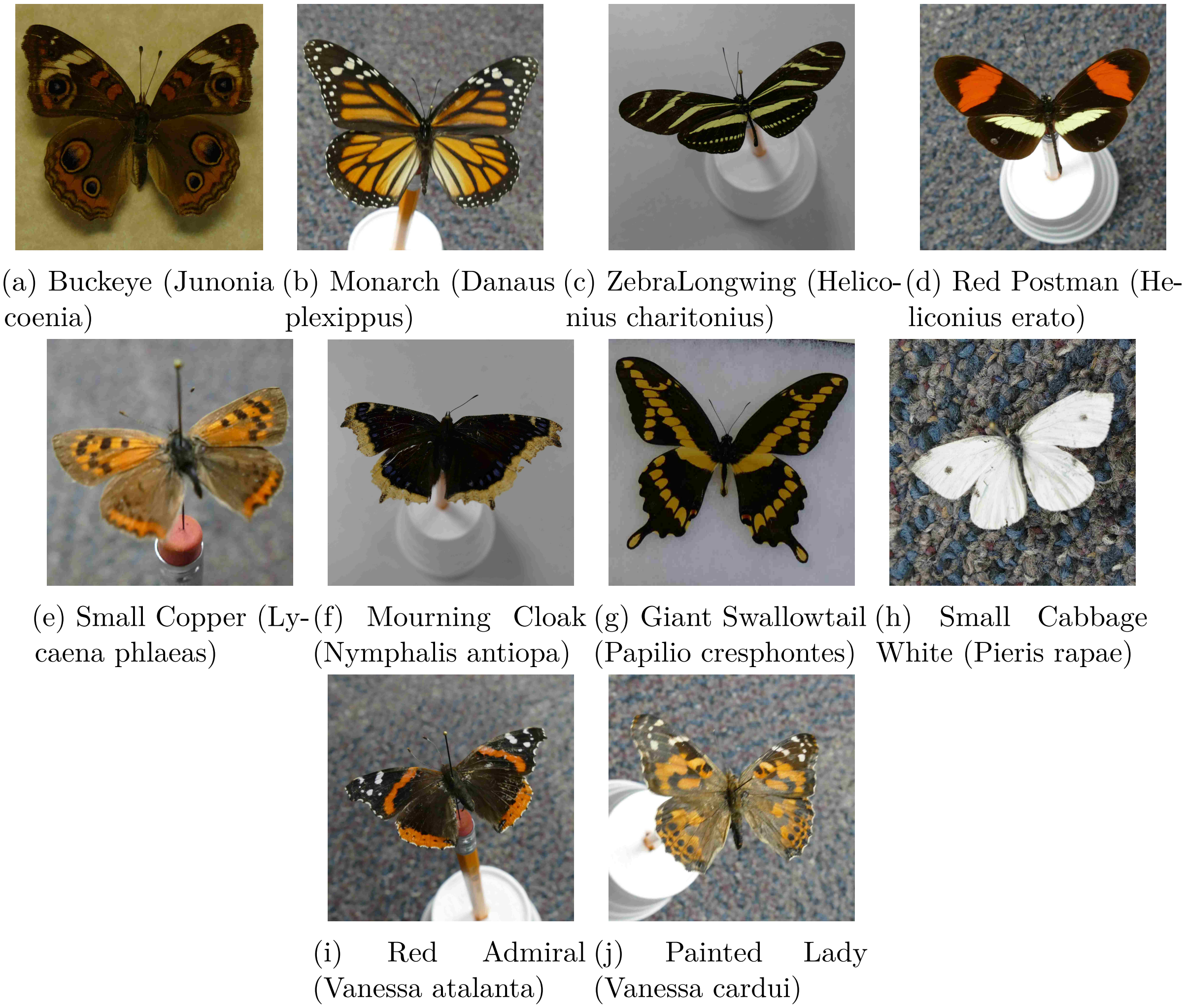
(f) Swallowtail butterfly using 20 images

Figure 5.3: Segmentation calculated on the Swallowtail butterfly captured with the panasonic using different number of images

the best results. There isn't much difference visually that can be seen from the 10 and the 20 images. Both of them produce a pretty good segmentation automatically.

5.4 Dataset

The focused dataset are of butterflies. Multiple classes of butterflies are mounted and imaged from different angles. A set of the same classes of butterflies are downloaded from the Leeds butterfly dataset [23]. This produces three distinct datasets. Two are under a controlled environment and one is taken from the internet in an uncontrolled environment. These datasets were compared to each other in different ways. There are ten kinds of butterflies. These butterflies are: Buckeye, Monarch, Mourningcloak, Painted Lady, Red Admiral, Red



(a) Buckeye (Junonia coenia)

(b) Monarch (Danaus plexippus)

(c) ZebraLongwing (Heliconius charitonius)

(d) Red Postman (Heliconius erato)

(e) Small Copper (Lycaena phlaeas)

(f) Mourning Cloak (Nymphalis antiopa)

(g) Giant Swallowtail (Papilio cresphontes)

(h) Small Cabbage White (Pieris rapae)

(i) Red Admiral (Vanessa atalanta)

(j) Painted Lady (Vanessa cardui)

Figure 5.4: Different Butterflies used in the dataset

Postman, Small Cabbage White, Small Copper, Swallowtail, and Finally the Zebra Longwing. An image of each model is provided in Figure 5.4

The first is a set of images of each type of butterfly class taken from the Leeds dataset. The amount of images within each class vary within the dataset. The Leeds dataset comes with the segmentation mask for all the images. This set of images that are segmented are identified by (F manual). This represents the traditional method of chosing and segmenting images.

The second set of images are 100 images of each butterfly class taken in a controlled environment. This is using the GoPro cameras attached to an arch that spins around and takes pictures of the subject. This is referred to as the Arch dataset. Because this dataset was taken within a controlled environment, the multiview method above can be used for segmentation. The images in this environment are of many different angles of the same subject. These images are also run through Mechanical Turk where workers from different backgrounds and experience segment the images. The same images are then segmented using our multivew segmentation method. A subset of images taken from the Mechanical Turk group is used to segment the rest of the images automatically. A different number of images are used in multiple tests to determine the accuracy of these models generated.

The third set of images are 100 images of each butterfly class taken in a controlled environment using Panasonic cameras. They were taken by hand around a stationary object. This set also has the ability to do multiivew segmentation.

The three different datasets of images were then split further into a training and a testing set within each dataset. The training set consisted of 50 images of each class. The testing set consisted of the rest of the images within that dataset. This produced 50 images of each class with the Arch and Panasonic group and a varying amount of each class with the Leeds dataset. This was done in order to keep the size of the training sets consistant within each dataset.

In order to test the accuracy of models that are generated using segmentation, the test images themselves must be segmented. However, in order for them to be a true test image, they cannot be segmented manually. This is due to the idea that the tests need to be completely automated. If an image was taken of a butterfly, the program should be able to identify the butterfly without the user having to segment the object beforehand. Therefore a machine learning model using semantic segmentation was created that focused on segmenting the testing images before they were tested against the training sets of each type. The machine learning algorithm used on this is based on the stacked hourglass algorithm [16]. The training images and the segmentation taken from those training images and their equivelant manual segmentation masks were reduced down to 256 × 256 images. This method trained on the training set of the three different datasets and their segmentation masks. The generated training models are then applied that to the testing set of that dataset to predict the segmentation mask of the testing images. This training and testing is done on each dataset independently of the other sets in order to generate the segmentation mask of the testing set that is used to determine the accuracy of the models generated using the different methods.

5.5 Results

5.5.1 Learned Segmentation vs Multiview Segmentation

Learned segmentation allows for the segmentation of an object to be calculated automatically using machine learning. This process requires a large number of images that are already segmentated by hand in order to be able to segment other images not seen before. We did not have a dataset large enough to train the model completely on. We also did not have the amount of time it would take to train a large model from scratch. It is common in this situation to take a model that has already been trained on a different set of classes and use that as a starting point to train on their own set of classes. This model which has been previously trained does not have to go through the same amount of testing to accurately train on a new set of classes. The machine learning model we used was originally trained on an image dataset of thousands of pictures of humans and then we fine tuned it on the butterflies.

In order to allow the segmentation to be trained at a relatively good speed and a decent amount of images, the resolution of the images used must be reduced dramatically. By contrast, multiview segmentation can be run on full resolution images with very few hand segmented images as the learning seed. We ran a test comparing the multiview segmented images and the segmented images produced using the machine learning model mentioned previously. The multiview segmented images used ten randomly chosen images out of the set to produce the segmentation for the rest of the images which produced the best model as shown above in Figure 5.1.

Figure 5.5 compares the accuracy of the segmentations using the multiview method compared to the learned segmentation method using the images obtained by the Panasonic camera. As can be seen, the accuracy of the multiview segmented images produces an accuracy comparable to the images segmented using the learned segmmentation. The learned segmentation produced an average accuracy of 81.32% accuracy while the multiview segmentation produced an average accuracy of 82.19% accuracy.

The accuracy of the segmentations goes down when using the dataset prepared with the GoPro cameras on the arch. Figure 5.6 shows the results of segmenting using learned segmentation vs multiview segmentation. The average accuracy of the learned segmentation was 63.43%. By comparison, the average accuracy of the multiview segmentation was higher at 69.27%. This is considerably lower than the segmentation using the panasonic images. In both situations using the Panasonic cameras as well as the GoPro cameras, the multiview segmentation method produced a higher accuracy to the true segmentation than using the machine learning method. It also takes fewer images in order to train the model.

5.5.2 Multiview Keypoint Annotation

Multiview keypoint annotation allows for identifying certain keypoints in the object. For a butterfly, this could mean the antennas, specific locations on the wings, the abdomen, or any other specific area on the butterfly that can be identified from the images. Keypoint annotation allows for specific points on an object to be identified. These keypoints can be used to help classify the object. A test was done to identify the accuracy of multiview keypoint annotation. A user was asked to identify the same keypoint in all of the images within a model. After the user has identified the same keypoint in a specified number of images, then that point is projected back onto the other images within the set. The distance from the location of the calculated keypoint and the true location of the keypoint within each image is calculated and averaged over all of the images. This was done on three different butterflies. Table 5.2 and Figure 5.7 show the distance between the calculated points using multiview

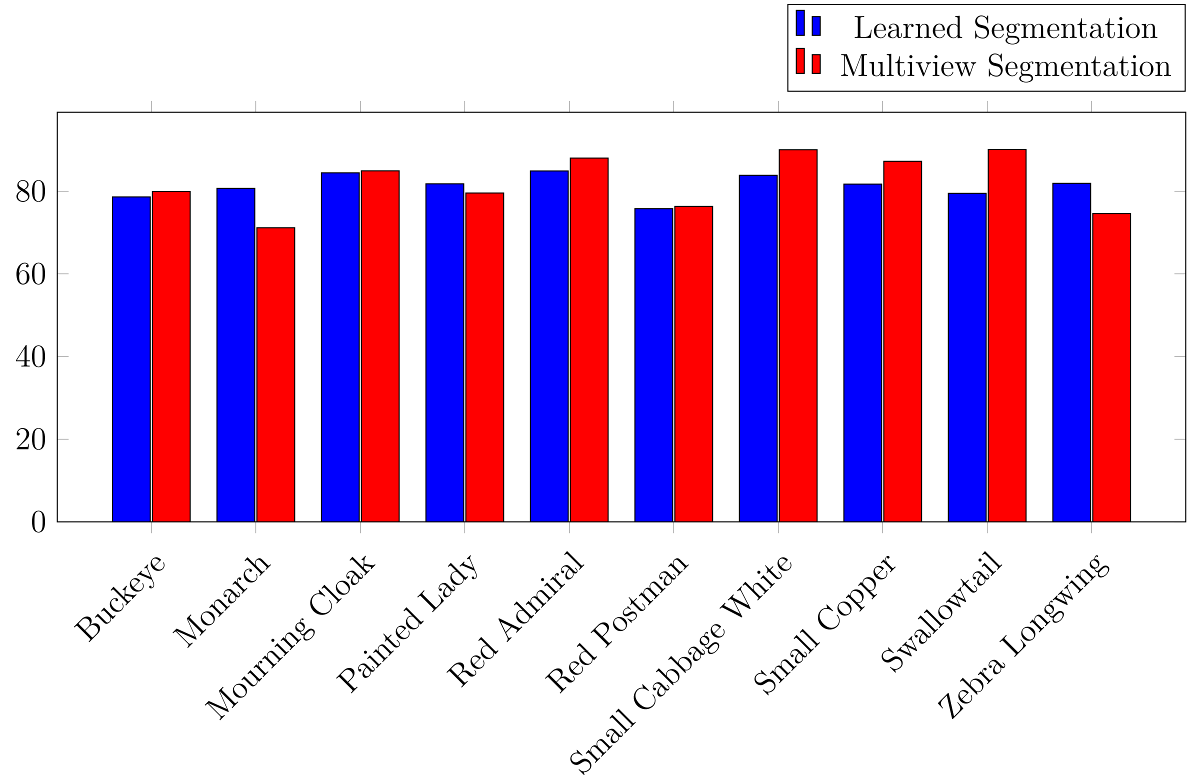


Figure 5.5: Panasonic Segmentation

keypoint annotation and the baseline true values of that point. The average distance at the best number of images is 46.67 pixels at 15 images. While 46 pixels might seem like a lot, considering the size of the image is 5472px by 3648px then 46 pixels is less than 1 percent of the image.

Number of images	Buckeye	Swallowtail	Monarch
3	12.1	69.7	127.9
5	12.0	62.4	120.2
7	13.0	33.9	118.5
10	11.6	33.0	96.2
15	11.6	32.8	95.7
20	11.9	30.9	98.6

Table 5.2: Distance in pixels from calculated to real

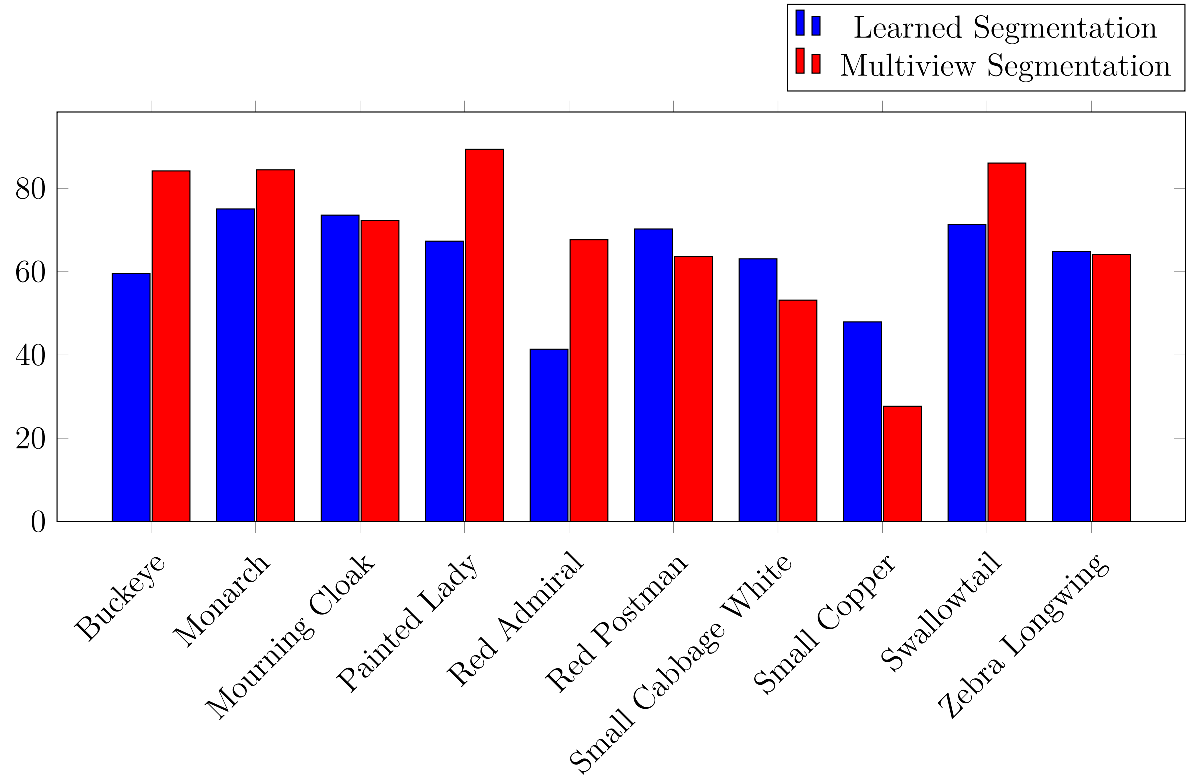


Figure 5.6: Arch Segmentation

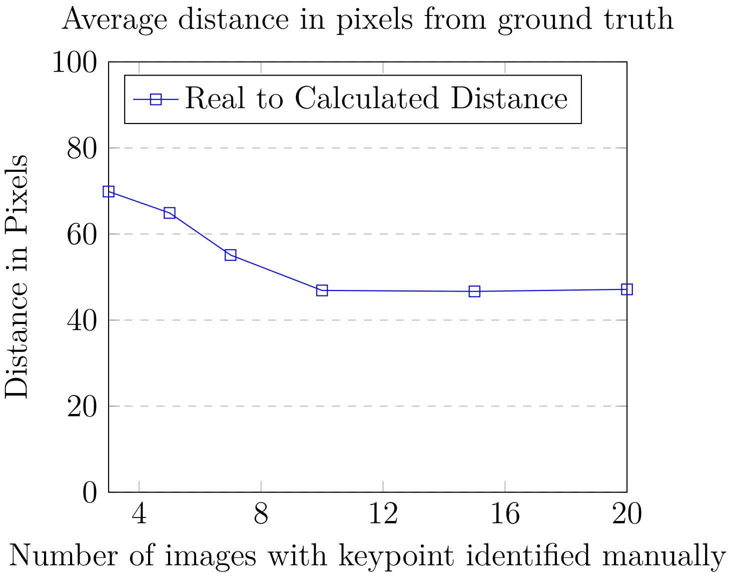


Figure 5.7: Average distance between the real location of the keypoint and the calculated location from the Multiview keypoint annotation algorithm

5.5.3 Machine Learning

The machine learning model consists of classifying the objects. Unless otherwise noted, the tests were run using one of the segmentations built using ten images as the seed as ten images produced the best result overall. There are 3 different sets of images used in the experiment. The Leeds dataset is a dataset of butterfly images created by the University of Leeds. The images came segmented when downloaded. No multivew segmenation occured on this set, as this set was not taken in a controlled environment and could not be segmented. The Panasonic dataset is the set using images taken from a Panasonic Lumix camera and the manual segmentation was done using Mechanical Turk. The images were taken by hand. Multiview segmentation was done using the process stated previously. The GoPro dataset is the set using GoPro cameras and taken with the arch system stated earlier. Multiview was also done on this dataset using the process stated previously. All sets use the same ten types of butterflies that are trained and tested on.

The tests are as follows:

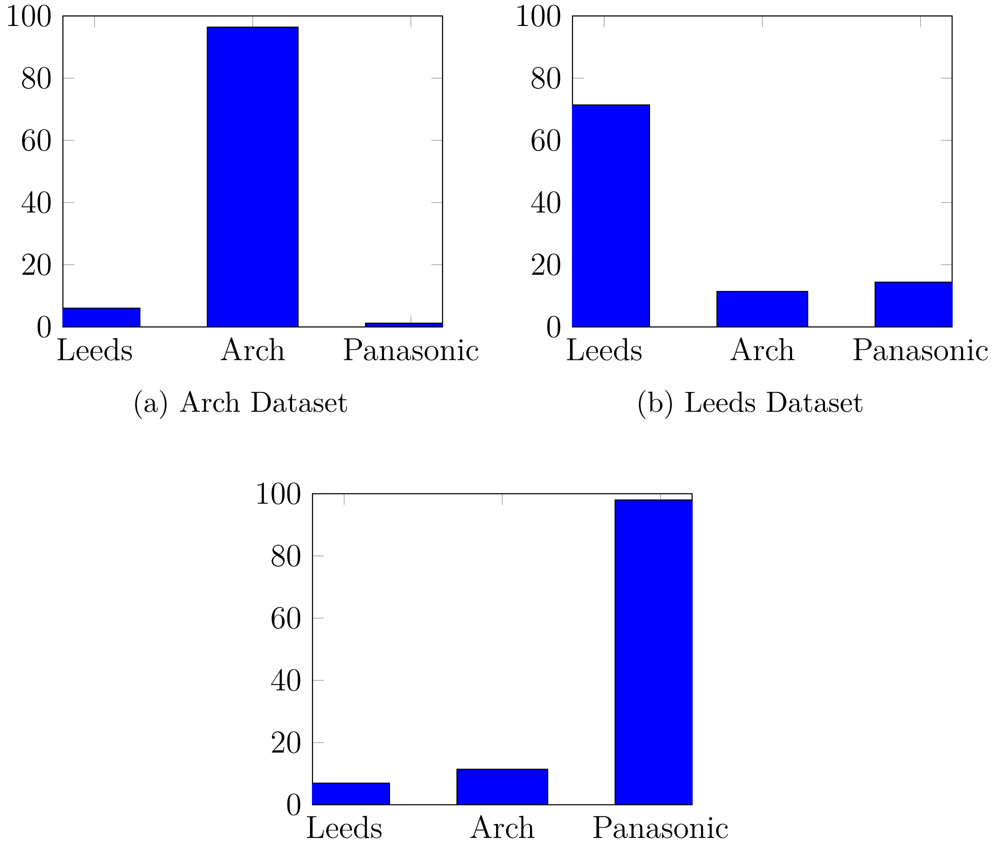
1. Unannotated Controlled Environment vs Unannotated Leeds
2. Manually Segmented Controlled Environment vs Manually Segmented Leeds
3. Multiview Segmented Controlled Environment vs Unannotated Leeds
4. Multiview Segmented Controlled Environment vs Unannotated Controlled Enviornment
5. Multiview Segmented Controlled Enviornment vs Manually Segmented Controlled Environment

Unannotated Controlled Environment vs Unannotated Leeds

This experiment establishes a baseline accuracy. We compared the accuracy of models generated with unannotated images from the Arch dataset, the Panasonic dataset and the Leeds dataset. Since the data was of the entire image and includes all of the background noise as well as does not zoom in on the butterfly images, the accuracy of the models was relativley low on all of the models for all of the models outside of the set type that it was trained on. Figure 5.8a shows the accuracy of the model generated using those images and testing against the testing images of all three sets. When training with the Arch dataset, it produced a relativly low accuracy testing agains the Panasonic and Leeds set, but high for the Arch dataset. Figure 5.8b shows the accuracy received when training the model with the Leeds dataset and then applying them to all three datasets for testing. This produced a 71% accuracy against the Leeds dataset, but really low accuracy against the Arch and Panasonic dataset. However, testing against the Arch and Panasonic datasets produces 11% and 14% accuracy respectively. Figure 5.8c shows the result of training on the Panasonic dataset. Between all of the models, there is very little accuracy when training on one dataset and testing against the others. This is not too big of a deal when dealing with unannotated images as there is a lot of noise within the image that could be confusing the model.

Manually Segmented Controlled Environment vs Manually Segmented Leeds

This experiment used the same set of images above, but focused on the accuracy of the model generated from the images which are annotated manually. The segmentation built using the segmentation machine learning model was used to as the testing set. This allowed the machine learning model to test on the same type of data, but not the same images, that it was trained on. As can be seen in Figure 5.9, the overall quality of the learning model grows considerably when using the manually segmented images. While the unannotated images produced high accuracy classification on the same dataset it was trained on, it was considerably lower on the other two datasets producing an accuracy of less than 20%. The manually annotated images on the Arch and Panasonic datasets produced a lower accuracy when testing on their own dataset compared with the unannotated model tested previously. However, they were more accurate when testing against the other datasets than the unannotated model tested previously. This is most likely due to the background of the images becomes a solid color and the images are cropped to fit the butterfly making them the center of the image and fitting

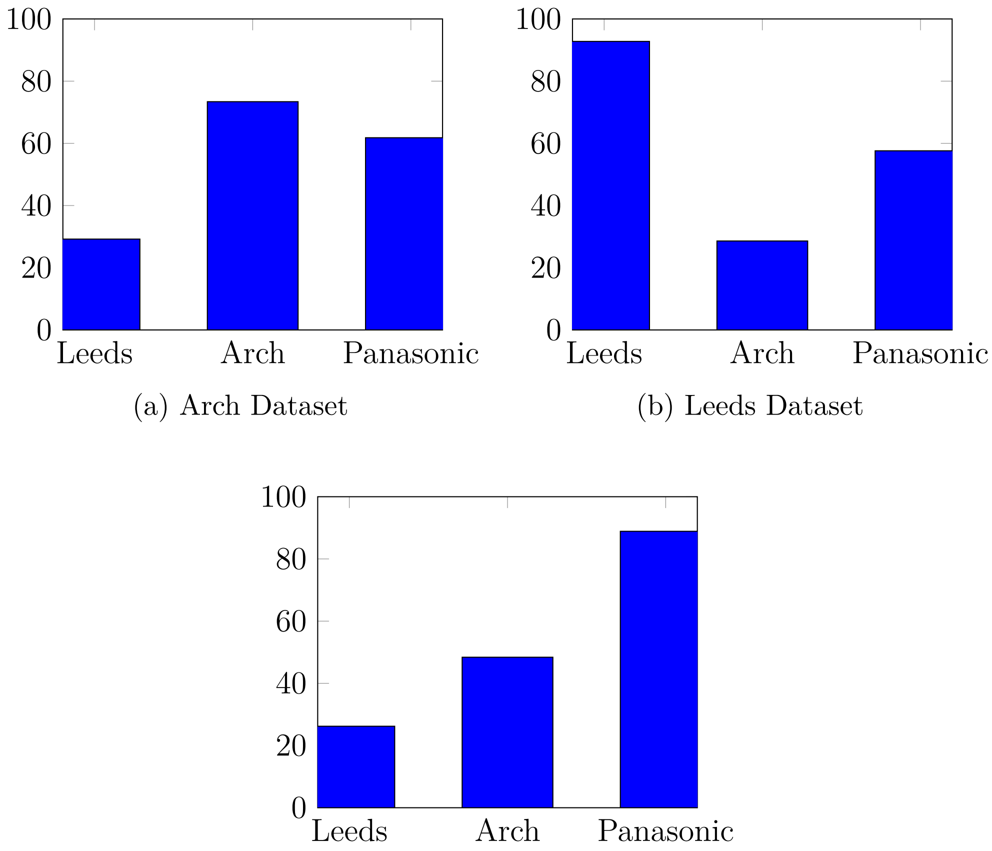


(a) Arch Dataset

(b) Leeds Dataset

(c) Panasonic Dataset

Figure 5.8: The baseline dataset run on the original images without any segmentation



(a) Arch Dataset

(b) Leeds Dataset

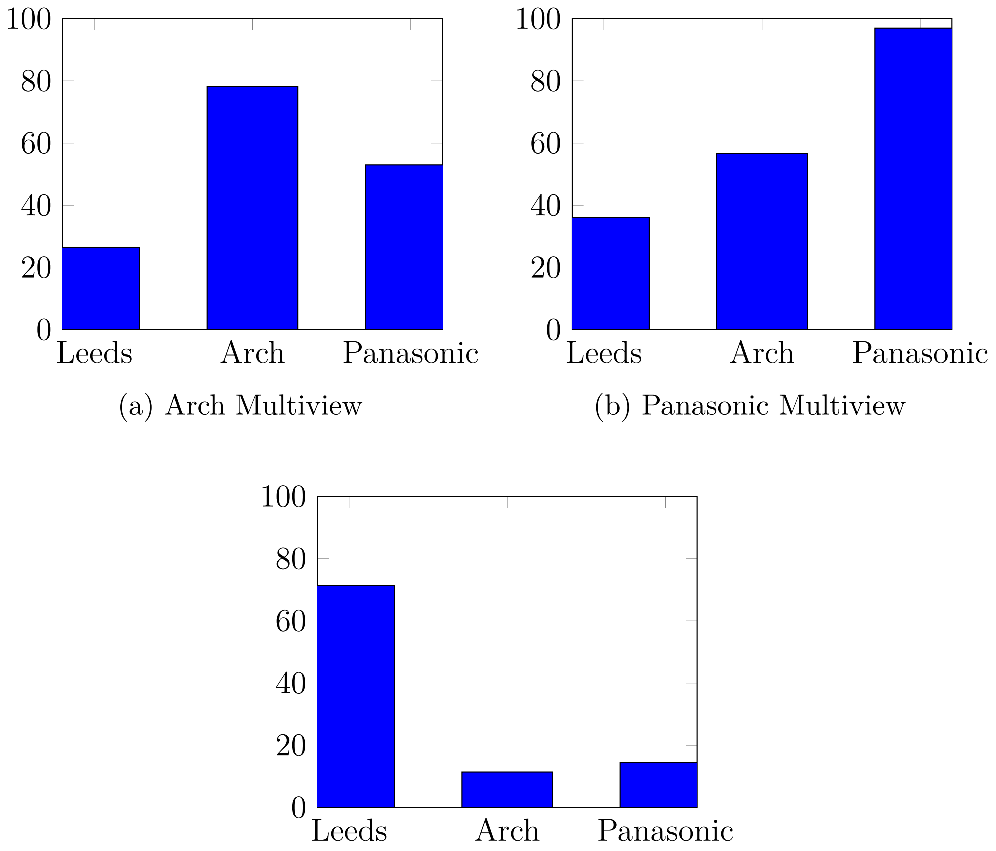
(c) Panasonic Dataset

Figure 5.9: The accuracy of the datasets trained on manually segmented images

the entire butterfly within the image. This way, the learning model only has to concentrate on the butterfly itself and not any of the background which doesn't provide any information as to the class of the butterfly.

Multiview Segmented Controlled Environment vs Unannotated Leeds

This experiment compares the images annotated using the multiview method with images from the Leeds dataset that are unannotated. This really is not a fair comparison as we are comparing two different types of images. However, it does give us some good data points as to how well images that are annotated do well against those that are not. As can be seen in Figure 5.10 the multiview models produced significantly better results across the images that



(a) Arch Multiview

(b) Panasonic Multiview

(c) Leeds Unannotated

Figure 5.10: The accuracy of the models generated using multiview annotation compared to the unannotated leeds dataset

are not part of the same dataset as the model was trained on. However, it produced about the same accuracy on the dataset it was trained on.

Multiview Segmented Controlled Environment vs Unannotated Controlled Environment

This experment is the same as above except instead of comparing the Leeds dataset to the annotated Arch and Panasonic datasets, we are comparing the Arch and Panasonic unannotated images with the Arch and Panasonic segmented images. Therefore, it is a much more fair comparison as we are comparing the same datasets, one annotated and one unannotated. This gives a basis for how well images are learned when annotated verses the

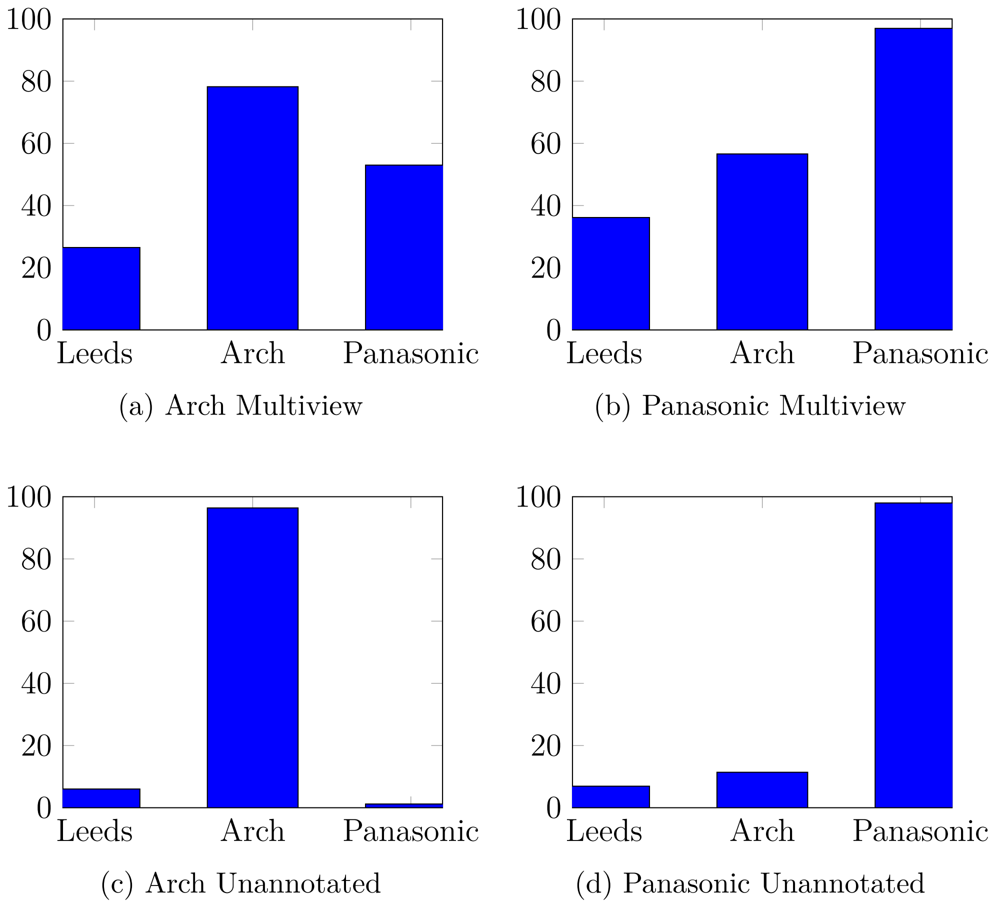


Figure 5.11: Comparing the accuracy of the model generated using unannotated images with the model generated using annotated images

same images that are unannotated. It also justifies the reason for annotating images in the first place. As can be seen in Figure 5.11 the results are significantly better for the multiview annotated images than for the unannotated images. When testing against the same dataset it was trained on, the results are a little worse in the multiview images than the unannotated images. However, when testing against other sets, the results are significantly better. This is probably due to the training model over fitting to the dataset it was being trained on, so wouldn't be able to test against other types of images.

Multiview Segmented Controlled Enviornment vs Manually Segmented Controlled Environment

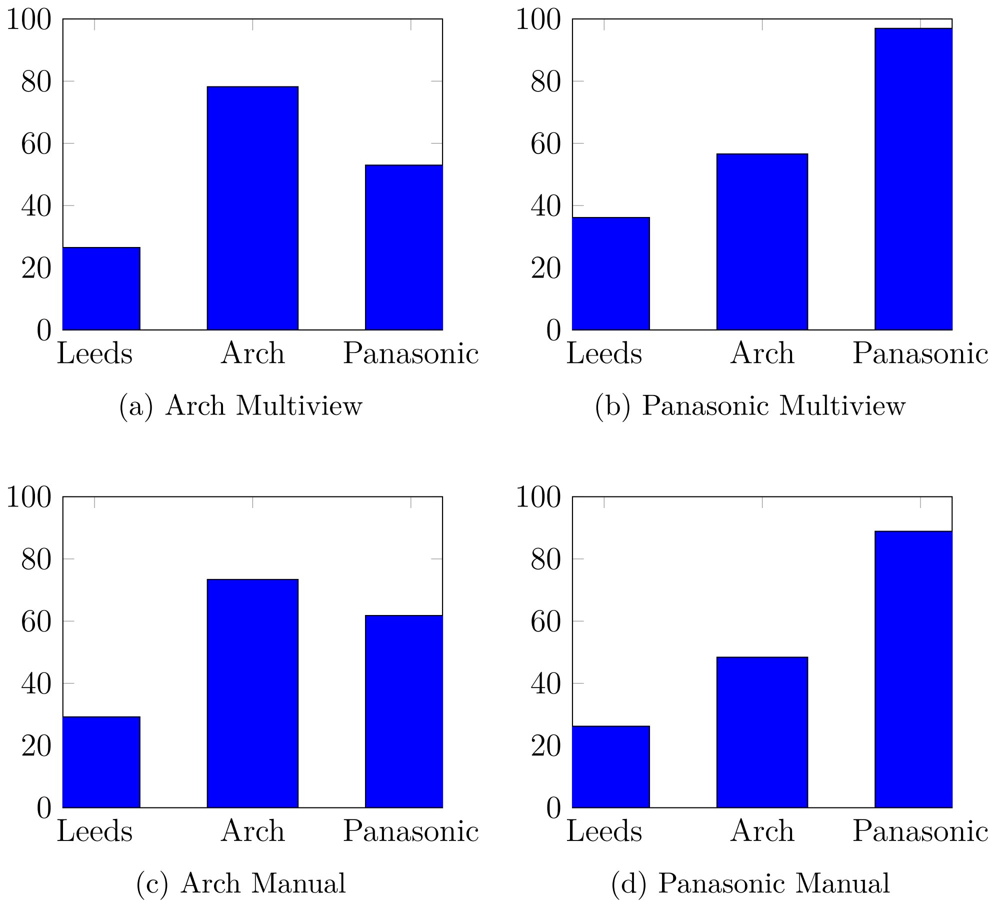
This experiment is really the heart of the thesis. Since the models trained on annotated images tend to do better when testing against other datatsets than the models generated on their unannotated counterparts, it is in the best interest of programmers to annotate images before they build models. However, the traditional method for annotating images is slow and costly. This experiment compares our method of annotating images which is faster and cheaper with the traditional method to show that we get just as good if not better results faster with a smaller cost.

We took the different images that were segmentated using the multiview method and generated a machine learning model with these images trained to learn the class of the butterflies. The accuracy of these models were then tested against the images generated using the learned segmentation model.

We compared the accuracy of the model generated by the multiview segmentation with those generated by the manual segmentation. The results are in Figure 5.12. The accuracy of those using the Panasonic images with the multiview segmentation came out a little more accurate than those with the manual segmentation. The opposite is true for the Arch images.

The Arch multiview images produced a 52.56% accuracy while the manually segmented images received an average of 71.46% accuracy. On the other hand, the panasonic images using multiview produced an accuracy of 63.23% while the manually segmented images produced an accuracy of 54.49%.

These accuracies are not great across the board. However, the accuracy of the models tested on the same capture type as the ones trained on is significantly higher. 78.2% on the arch multiview images, and 96.96% accuracy on the Panasonic multiview images compared to 73% accuracy on the Arch manual images and 88% accuracy on the Panasonic.



(a) Arch Multiview

(b) Panasonic Multiview

(c) Arch Manual

(d) Panasonic Manual

Figure 5.12: The multiview segmentation models compared to the manually segmented models

The models generated using the multiview segmented images are comparable to those using the manually segmented images showing that multiview segmented images can replace the manually segmented reducing the time needed to segment the data without affecting the quality of the learning process to a great degree.

Chapter 6

Conclusion and Future Work

We found that multiview segmentation from these ten images produced about the same accuracy as the manually segmented images, but greatly reduced the effort needed to produce those segmentations. This greatly reduced the time and effort needed to hand segment images needed for training.

Multiview segmentation can reduce the amount of time and money used to create the needed segmentation for training images. While machine learning models for segmentation must be trained using many thousands of pictures that are manually segmented, multiview segmentation only requires a few pictures to be hand segmented in order to automatically segment hundreds more.

We found that using only ten manually segmented images, we were able to extend the segmentations to many more images accurately. The downside is that multiview segmentation must be done using pictures taken in a controlled environment with the target object not moving locations or changing positions.

Keypoint annotation had a similar effect. We only needed to identify the keypoint in about 10 images in order to get the accuracy of the model to a high degree. That keypoint was then able to be projected back to the rest of the images without any other human intervention. This can greatly save time and effort in annotating images.

We found that while the unannotated images produced a high accuracy for the same type of capture data, it produced a very low accuracy for a different type of capture data even though the classes of butterflies were the same. On the other hand, a model trained using annotated images produced a higher accuracy among images across multiple classes even though it was lower testing against its own dataset compared to the unannotated model, thereby reducing the overfitting of the model. This shows that segmentated images can produce a better trained model with less overfitting. We found that neither the Arch nor the Panasonic trained model was able to do very well on the Leeds test dataset. We did only use one physical example of each class when capturing and training the models. This probably caused some overfitting within the model trained which is why the Arch and Panasonic dataset did not do well when testing against the Leeds dataset. The Leeds dataset would probably be the most appropriate dataset to use for testing as it is the one that is the most realistic representation of what the model would be predicting on. With future work, more physical examples of each class and revision of the models and capturing technique, the models generated using multiview segmentation can probably increase the accuracy of prediction.

6.1 Future Work

We would like to test this setup using multiple types of cameras and structures. A new arch is being built that will allow for more powerful cameras to be installed on the system instead of using GoPros. This setup will allow for better images to be taken of the model more atonomously. In addition, within the model that was built with the arch, over 500 images could be taken with the system and merged together in a 3D model. However, a problem came up when attempting to merge all 500 images together. A ghost effect in the 3d model gets built above the object because the cameras on the other side of the arch can always be seen in the same location as the arch is circulating around the model. When this happens the SfMalgorithm gets confused with the cameras on the other side and manifests it as a structure in the center. The new arch being built turns the model instead of the cameras and sets the cameras up so they do not see each other at any time. This should reduce the amount of external artifacts seen and therefore built as part of the model.

As the goal is to reduce the amount of work needed, we want to maximize the number of segmentations that can be produced with a single object sample. However, one issue with using only one example of an object with multiple angles is that while the model gets to train using multiple angles on the target object, it only gets trained on a single pose of the object. This can cause problems when attempting to classify an object that is in a different pose. We would like to test out this system using multiple poses of the object and see if it increases the learning ability of generated models to meet or excede the current accuracy. As the machine learning was not the focus of this thesis, there was only one pretrained machine learning model that was used. There are multiple ones out there and it would be interesting to train on different kinds of models to see if the accuracy can be increased on outside datasets.

References

[1] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, V. Loumos, and E. Kayafas. A license plate-recognition algorithm for intelligent transportation system applications. *Institute of Electrical and Electronics Engineers Transactions on Intelligent Transportation Systems*, 7(3):377–392, 2006.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *Institute of Electrical and Electronics Engineers Transactions on Pattern Analysis and Machine Intelligence*, 2017. 10.1109/TPAMI.2016.2644615.

[3] T. L. Berg, A. Sorokin, D. A. Forsyth, D. Hoiem, I. Endres, and A. Farhadi. It's All About the Data. *Institute of Electrical and Electronics Engineers*, 98(8):1434–1452, 2010.

[4] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3D human pose annotations. *Computer Vision and Pattern Recognition, Institute of Electrical and Electronics Engineers Computer Society Conference on*, pages 2–9, 2009.

[5] S. Branson, C. Wah, F. Schroff, B. Babenko, P. Welinder, P. Perona, and S. Belongie. Visual Recognition with Humans in the Loop. *European Conference on Computer Vision*, pages 438–451, 2010.

[6] J. Deng, W. Dong, R. Socher, and L. Li. Imagenet: A large-scale hierarchical image database. *Computer Vision and Pattern Recognition, Institute of Electrical and Electronics Engineers Conference on*, pages 248–255, 2009.

[7] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2014.

[8] R. Farrell, O. Oza, V. I. Morariu, T. Darrell, and L. S. Davis. Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance. *Computer Vision, Institute of Electrical and Electronics Engineers International Conference on*, pages 161–168, 2011.

[9] M. Faundez-Zanuy. Biometric security technology. *Institute of Electrical and Electronics Engineers Aerospace and Electronic Systems Magazine*, 21(6):15–26, 2006.

[10] J. J. Koenderink and a. J. van Doorn. Affine structure from motion. *Journal of the Optical Society of America. A, Optics and image science*, 8(2):377–385, 1991.

[11] A. Kovashka, S. Vijayanarasimhan, and K. Grauman. Actively selecting annotations among objects and attributes. *Computer Vision, Institute of Electrical and Electronics Engineers International Conference on*, pages 1403–1410, 2011.

[12] A. Krizhevsky, I. Sulskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information and Processing Systems*, pages 1–9, 2012.

[13] K. N. Kutulakos and S. M. Seitz. A Theory of Shape by Space Carving. *International Journal of Computer Vision*, 38(3):199–218, 2000.

[14] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. 39(4):640–651, 2015.

[15] S. Mori, C. Y. Suen, and K. Yamamoto. Historical Review of OCR Research and Development. *Institute of Electrical and Electronics Engineers*, 80(7):1029–1058, 1992.

[16] A. Newell, K. Yang, and J. Deng. Stacked Hourglass Networks for Human Pose Estimation. *European Conference on Computer Vision*, pages 483–499, 2016.

[17] N. M. Oliver, B. Rosario, and A. P. Pentland. A bayesian computer vision system for modeling human interactions. *Institute of Electrical and Electronics Engineers Transactions on Pattern Analysis and Machine Intelligence*, 22(8):831–843, 2000.

[18] A. Rabinovich, A. Vedaldi, and S. Belongie. Does Image Segmentation Improve Object Categorization ? *UCSD CSE Technical Report*, 2007.

[19] J. Seo, S. Han, S. Lee, and H. Kim. Computer vision techniques for construction safety and health monitoring. *Advanced Engineering Informatics*, 29(2):239–251, 2015.

[20] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *Computing Research Repository*, 2014. https://arxiv.org/pdf/1409.1556.pdf.

[21] A. Torralba, B. C. Russell, and J. Yuen. LabelMe: Online image annotation and applications. *Institute of Electrical and Electronics Engineers*, 98(8):1467–1484, 2010.

[22] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. 2011.

[23] J. Wang, K. Markert, and M. Everingham. Learning models for object recognition from natural language descriptions. *British Machine Vision Conference*, 1:2.1–2.11, 2009.

[24] P. Welinder, S. Branson, S. Belongie, and P. Perona. The Multidimensional Wisdom of Crowds. *Most*, 6:1–9, 2010.

[25] M. J. Westoby, J. Brasington, N. F. Glasser, M. J. Hambrey, and J. M. Reynolds. âĂŸStructure- from- Motion’ photogrammetry: A low- cost, effective tool for geoscience applications. *Geomorphology*, 179:300–314, 2012.

[26] J. Xiao, A. Owens, and A. Torralba. SUN3D: A database of big spaces reconstructed using SfM and object labels. *Institute of Electrical and Electronics Engineers International Conference on Computer Vision*, pages 1625–1632, 2013.