# Chapter 1

# DISTANCE DESCRIPTOR

## 1.1 Introduction

1. The ability to recognize objects enables a robotic system to interpret its environment and make intelligent decisions based on the identity of the objects around it.

2. To recognize objects from images, prior information about appearance of the object is required.

3. The appearance of the object is encoded in form of features which an object recognition system can extract and through it decode the identity of the object by the application of machine learning methods.

4. To unambiguously distinguish between multiple objects, the information encoded in the object features extracted from a single image needs to be sufficiently discriminative.

5. In cases where information available in a single image of a target object is insufficient to determine its identity, multiple-views of the target can be combined to gather information sufficient to unambiguously identify an object.

6. The idea of combining multiple sensor measurements comes under the purview of active sensing.

7. In the computer vision domain, using multiple views of an object methods are closely related to class of multi-resolution or image pyramid based techniques.

8. Since the task of separating the background from foreground pixels in an image becomes problematic in the absence of sharp edges and in the presence of noise, object recognition techniques that can extract information from an image offer a way to detect the presence of a target object in an image.

9. In this paper, we propose an object recognition method that combines information from multiple views of a target acquired from a series of heights, to build a feature descriptor that can operate on an image without segmentation of foreground and detect the presence of a target object.

## 1.2 Preliminaries

### 1.2.1 Grabcut-in-one-cut Algorithm

1. When initialized with foreground and background seed pixels, graphcut treats the segmentation problem as a binary graph partition problem to separate the foreground pixels from background.

2. To perform the binary partition of foreground and background, graphcut checks for similarity or dissimilarity of the pixels neighboring the input foreground and background seeds till a complete binary partition is achieved through labeling of all points in an image as either foreground or background.

3. If the location of an object in an image is known, graphcut segmentation offers a method to segment the image into foreground and background pixels.

Graphcut algorithm treats the segmentation problem as a graph partition problem [3]. Consider a case where each pixel $p \in \mathcal{P}$ of an image is treated as a vertex $i$ in the graph $G(\mathcal{V}, \mathcal{N})$. Two neighboring pixels $p$ and $q$ represented by vertices $i$ and $j$ respectively are connected by an edge $e_{ij}$ in the graph $G$. The task of binary partition of this graph $G$ into foreground and background pixels is cast into an optimization problem where the energy $E$ is to be minimized. The energy $E$ here is formulated as

$$E(L) = \sum_{p_i \in \mathcal{P}} D_p(L_p) + \sum_{(p,q) \in \mathcal{N}} V_{p,q}(L_p, L_q), \tag{1.1}$$

where $L_p$ corresponds to a labeling of pixel $p$, $D$ is a data penalty function that assigns a likelihood for pixel $p$ to belong to either foreground or background label class, $V_{p,q}$ is an interaction potential function that encourages spatial coherence by penalizing discontinuity in label values of pixels. A binary partition of this graph is obtained by choosing a label for each pixel $p$ such that the energy $E$ of the image is minimized. This graph-based segmentation theory, also referred to as max-cut min-flow algorithm was was adapted in [4] to suit the binary image segmentation problem. To this end, a variant of graphcut, the Grabcut-in-one-cut [4] uses pre-specified foreground and background seeds to perform a binary partition of an image. According to Grabcut-in-one-cut algorithm [4], the energy function in (1.1) is reformed as

$$E_{seeds}(S) = -\beta \|\theta^S - \theta^{\bar{S}}\|_{L_1} + \lambda |\partial S| \,. \tag{1.2}$$

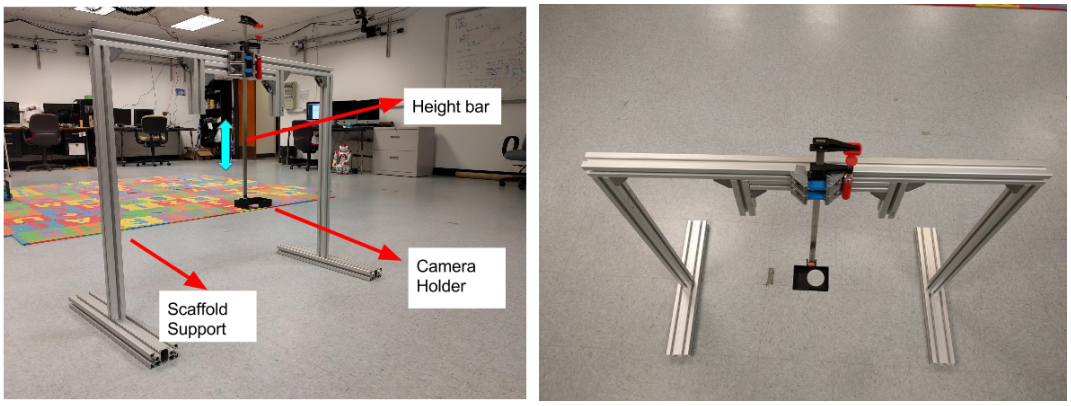
Here $\theta^S$ and $\theta^{\bar{S}}$ are the histograms of the foreground and background pixels in the image. The first term of (1.2) penalizes the pixels with intensities that overlap with both the foreground and background distributions. The second term $|\partial S|$ of (1.2) is a term analogous to $V$ in (1.1), that tries to enforce similar labeling of neighboring pixels and therefore penalizes a change in labeling across neighbors. The weights $\lambda$ and $\beta$ allow to vary the contribution of the first and second term to the value of the energy function in (1.2). Finally the $L_2$ in (1.2) refers to the $L2$-norm.

The algorithm proposed in [4] generates a binary partition of an image into background and foreground by extrapolating the input foreground and background seed pixels. If there is an automated means to find the location of foreground objects, the seeds required by this algorithm can be automatically generated hence making this guided segmentation approach fully autonomous.

### 1.2.2 Data Collection Apparatus

#### 1.2.2.1 Imaging Rig

1. The imaging rig comprises a frame with a camera holder that allows the camera to be positioned at different heights from the ground.




(a) (b)

Figure 1.1: The imaging rig allows the height bar to slide up or down, thereby letting the height of the camera from the ground to be varied. The different components of the imaging rig are labeled in Figure 1.1a. The camera holder attached to the lower end of the height bar is designed to carry a GoPro Hero 4 camera.

The imaging rig shown in Figure 1.1 allows a camera to be held at different heights from the ground for imaging experiments. The height bar slides up or down and can be locked at a specific configuration through specially designed holders on the scaffold support of the imaging rig. By varying the position of the height bar, the height of the camera holder that is attached to the lower end of the height bar can be varied. This allows the camera held on to the camera holder to capture images of targets placed on the ground from different controlled heights. The camera holder was designed to hold a GoPro Hero 4 camera [1].

#### 1.2.2.2 Imaging Environment

1. Since underwater object recognition could pose a challenging problem due to presence of noise and lighting variations, data was collected in a test tank filled with water.

Environment where object recognition experiments are performed could affect the performance of an algorithm. In case of an environment with limited to no control over the imaging parameters, the level of noise introduced into the sensor measurements is difficult to gauge for taking any corrective recourse. To avoid such difficulties, it

might be tempting to perform object recognition experiments in controlled laboratory conditions, thereby allowing the possibility to tune the environmental parameters to enhance the performance of an algorithm. However if an algorithm is intended to operate on natural images, there are often going to be unpredicatble variations in environmental factors that could adversely affect the performance of an algorithm. In such a case, to accurately validate an algorithm that is intended to operate on natural environments, it is imperative to choose a test environment where the conditions are close to the expected natural conditions. Following this line of reasoning, since this object recognition algorithm is intended to operate on noisy natural environments, an underwater environment with uncontrolled lighting was chosen as the test environment. A circular tank in the Robot Discovery Lab (RDL) at University of Delaware filled with 4 feet of fresh water was thus utilized to collect experimental data. The imaging rig described in Section 1.2.2.1 was submerged inside this test tank during the data collection process.

#### 1.2.2.3 Object Specimens

1. To validate the developed multi-image recognition method, commonly available produce like orange and strawberry were selected as they offered specimens with natural intra-class along with significant inter-class differences in appearance.

The focus of this work is to recognize a class of objects that can exhibit some level of intra-class variance that is characteristic to naturally occurring objects. One such example is underwater marine organisms, where each species can have several identifiers that distinguish them from a different species. However in most cases, the members of a single species also exhibit some level of variation in their appearance. To accommodate such cases, the specimens used to validate this multi-image algorithm were required to have (a) some characteristics that are unique to the class they belong to (a) small variations with regards to appearance within the same class of objects (a) easy accessibility for experimentation purposes. All these points were satisfied by natural produce like potatoes, oranges, tomatillos and strawberries. Natural produce

(a) Potatoes (b) tomatillos

Figure 1.2: An illustration of the inter-class and intra-class variance exhibited by naturally occurring objects like potatoes (Figure a) and tomatillos (Figure b)

.

exhibit significant inter-class variance, sufficient intra-class variance and are readily available in stores making them ideal candidates for object specimens. Figure 1.2 shows an assortment of potatoes and tomatillos to visually reinforce the inter-class and intra-class variance exhibited by naturally occurring objects. The underwater data on which this multi-image algorithm is validated is composed of a set of 10 oranges and 13 strawberries.

## 1.3 Definitions

### 1.3.1 Histogram Signature

As a part of this object recognition technique we define a histogram signature that is an improvement on the generic histogram, specifically designed to deal with possible noise in the pixel values. The rest of this section defines this histogram signature.

The generic histogram $H_f : b_i \rightarrow n_i$ of some colorspace $f$, with $n_i$ representing the count of bin $b_i \in \mathcal{B}$, captures the distribution of values of pixels in colorspace $f$ in an image. Any pixels affected by noise in image $I$ could corrupt the histogram $H_f$. If we assume that $e\%$ of the pixel values of an image constitutes noise, then the histogram $H_f$ can be reformulated to reject the bins containing $e\%$ of pixel values that are least likely to occur as dictated by the histogram $H_f$ itself. Before computing the refined

version of this histogram, the histogram $H_f$ is first normalized to get $\bar{H}_f : b_i \rightarrow \bar{n}_i$ as shown in (1.3).

$$\bar{H}_f(b_i) = \frac{n_i}{\sum_j n_j} = \bar{n}_i \tag{1.3}$$

Now the refined histogram signature $\mathcal{H}_f : b_i \rightarrow n'_i$ is computed, such that

$$n'_i = \begin{cases} \bar{n}_i & b_i \in \mathcal{B}_e \\ 0 & b_i \in \bar{\mathcal{B}}_e \end{cases}, \tag{1.4}$$

where $\mathcal{B}_e$ and $\bar{\mathcal{B}}_e$ constitute a binary partition of $\mathcal{B}$, i.e. $\mathcal{B}_e \cap \bar{\mathcal{B}}_e = \phi$ and $\mathcal{B}_e \cup \bar{\mathcal{B}}_e = \mathcal{B}$. The set $\bar{\mathcal{B}}_e$ is chosen such that $\bar{\mathcal{B}}_e$ is the smallest cardinality subset of $\mathcal{B}$ that satisfies the condition $\sum_{b_i \in \bar{\mathcal{B}}_e} \bar{H}_f(b_i) > 1 - e$. The formulation of set $\bar{\mathcal{B}}_e \subseteq \mathcal{B}$ can be restated as

$$\mathcal{B}_e = \underset{|\bar{\mathcal{B}}_e|}{\operatorname{argmin}} \sum_{b_i \in \bar{\mathcal{B}}_e} \bar{H}_f(b_i) > 1 - e \tag{1.5}$$

Ultimately the $|\mathcal{B}|$-element histogram signature $\mathcal{H}_f$ encodes the histogram information while attempting to prevent noise present in sensor measurements from corrupting the computed histograms of an image.

## 1.4 Methodology

1. Any machine learning technique like the one used here for object recognition comprises a learning and a testing phase.

2. The learning phase of object recognition involves extracting features that are capable of identifying a target unambiguously.

3. The testing phase is the application of the learned machine learning model to an application where the identity of an object is to be ascertained.

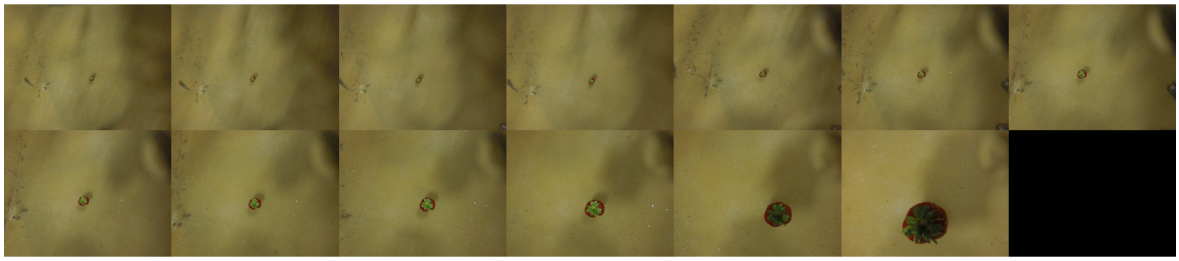
4. To validate the developed theory, an image dataset with images of objects captured from different known heights was gathered.

The algorithm developed in this paper is a form of machine learning technique. As in other machine learning techniques, there are two parts: learning and validation. In the learning phase features and other attributes of an object class are captured from labeled instances of the object available in the learning set. In the validation phase, the learned descriptor for an object class is validated against pre-labeled images to evaluate the performance of an algorithm. The data collection and annotation phases that provides data used for learning and testing parts of this algorithm is also discussed in detail.

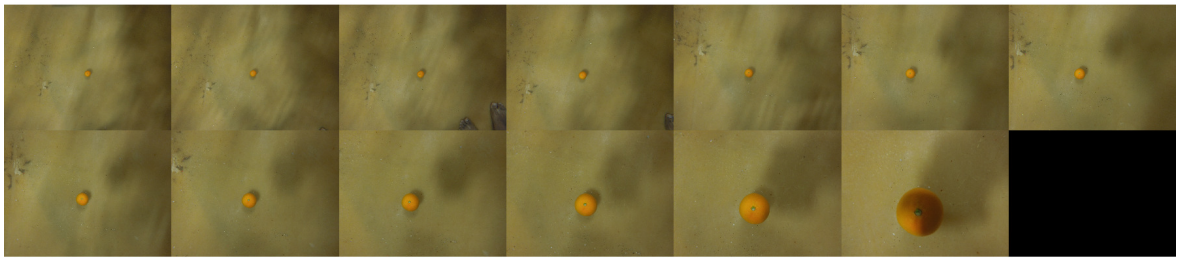
### 1.4.1 Data Collection

1. 13 Images of each target is captured between heights of 32in and 8in with an interval of 2in.

The data collection involves capturing 13 images of each object specimen. For the validation of this multi-image object recognition algorithm, data was gathered from 23 specimens (10 oranges and 13 strawberries) bringing the total images collected to 299 $(23 \times 13)$. Each specimen was first placed on the floor below the camera-holder on the imaging rig (see Figure 1.1). The height bar was moved up till the camera holder was 32 inches away from the ground. A GoPro Hero 4 camera attached to the holder was then triggered to capture an image of the specimen. This image captured the visual appearance of the specimen 32 inches away from the ground. Following capturing the first image, without disturbing the object, the next image was captured after lowering the height bar such that the camera is now 30 inches away from the ground. Another image of the specimen is now captured. This process of lowering the camera by 2 inches between subsequent images is continued till the camera gets to a height of 8 inches away from the ground. The series of images captured from a height of 32 inches to 8 inches with an interval of 2 inches between images results in 13 images per specimen. These 13 images capture the appearance of a specimen from different heights. An illustration of the 13 images captured per specimen for an instance of strawberry and orange is shown in Figure 1.3. An important point to note here is



(a) Strawberry specimen seen from different heights



(b) Orange specimen seen from different heights

Figure 1.3: A set of 13 images gathered for a specimen (strawberry in Figure a and orange in Figure b) starting from a height of 32 inches (top left) up to a height of 8 inches (bottom right) away from the target. Each subsequent image was captured 2 inches closer to the specimen

.

the imaging rig is handled manually between each image capture which sometimes leads to displacement of the imaging rig or the specimen on the ground. Additionally disturbances in the underwater environment where the specimen was placed added another source of inconsistency that sometimes lead to the specimen or the imaging rig getting displaced during the data collection process. Since natural environments are uncontrolled, these uncontrolled factors in the data collection process add some level of beneficial variation to the gathered dataset which in turn could make the machine learning algorithm robust to such variations.

Due to the wide-angle fish-eye lens in the GoPro camera used, the ratio of the pixels of the object specimen to the background could be very small when the camera is far away from the target. In technical terms, when the images of a target are captured by a camera that is far away from the specimen, the number of pixels that correspond to the specimen in the image could be statistically insignificant. This

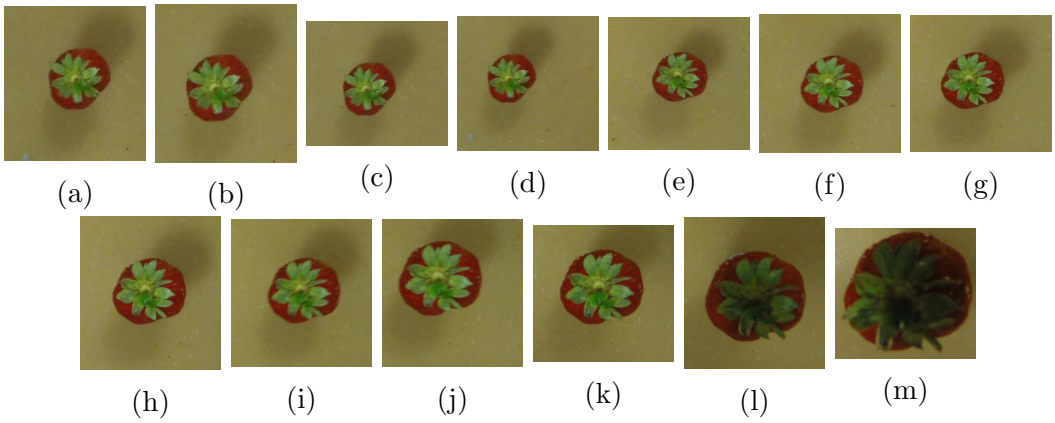



Figure 1.4: The strawberry specimen in Figure 1.3a after cropping to remove excess background

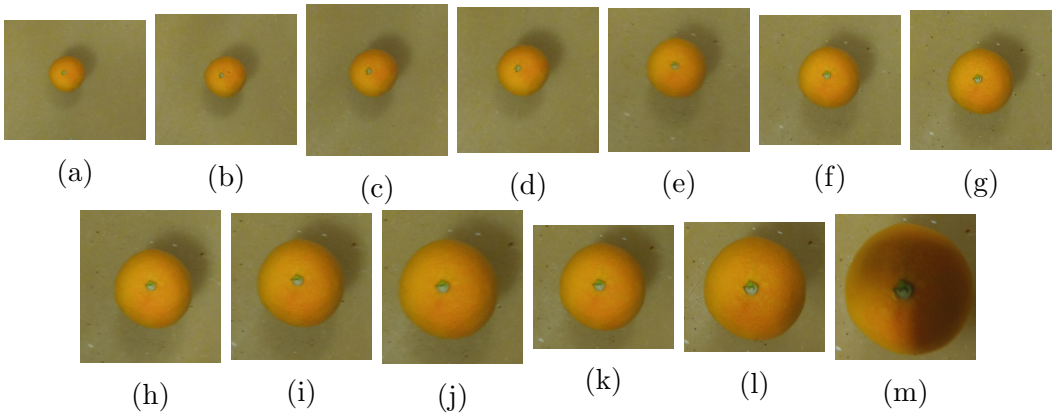



Figure 1.5: The orange specimen in Figure 1.3b after cropping to remove excess background

object recognition procedure depends on histograms of images (more details of how the histogram is relevant is explained in Section 1.4). For the histogram to accurately capture the properties of a specimen, the images of the specimen need to have a high ratio of object pixels over background pixels. To improve the ratio of object pixels in the image, the images were cropped to discard the excess background. The strawberry and orange specimen shown in Figure 1.3a and Figure 1.3b are shown again in Figure 1.4 and Figure 1.5 respectively, but after cropping of the excess background.

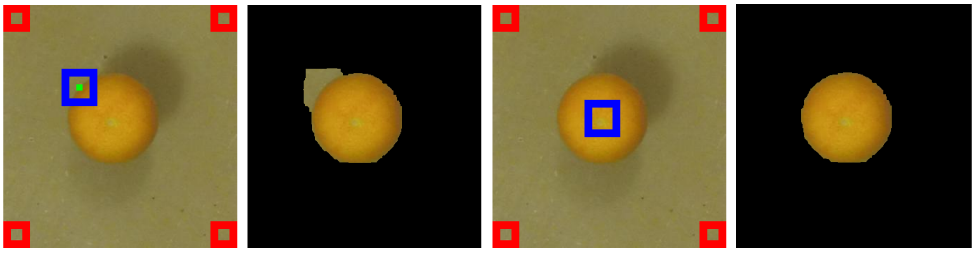
### 1.4.2 Annotation Process

Just like any other supervised machine learning algorithm, this method requires annotation of images into labeled background and foreground. The task of annotation is a costly and labor intensive process. The whole idea of developing an object recognition algorithm is often related to avoiding human annotation or any other form of manual intervention to identify objects from images. However, to accomplish this task an initial set of images need to be annotated manually to contribute to the learning set of a machine learning algorithm. In this object recognition method, each learning sample constitutes 13 images of a specimen from different heights. This increases the task of annotation 13-fold. To avoid or at least minimize the manual annotation involved, a semi-automated annotation framework was devised.

In the first stage of the annotation process, Bottom-Up Visual Attention (BUVA) technique (described in Section **??**) is used to pick the most *interesting* point in an image. According to the visual attention hypothesis, if a distribution of features can be used to characterize an image, the most interesting point or the first fixation corresponds to the point in the image which is most unlikely to belong to the distribution of features of the image. In other words, the first fixation corresponds to a point that does not belong to the background and hence most likely is a foreground object pixel. In the first stage of this annotation process, the first fixation point along with all its neighboring pixels within a rectangle of dimensions $15\%\, image\, width \times 15\%\, image\, height$ centered on the fixation are assumed to be foreground pixels. We will refer to this rectangle as the *foreground hypothesis rectangle*. Since the data collection process ensures that an object in an image is close to the center of the image rather than the edges of the image, the pixels in the boundary of the images can safely be assumed to be background pixels. Hence, rectangular blocks of pixels in the 4 corners of an image, of dimension $5\%\, image\, width \times 5\%\, image\, height$, are labeled as background seed pixels. The foreground and background pixel hypothesis thus obtained are used in the next stage of the annotation process to segment the image into foreground and background.

The foreground and background seeds obtained before can be used to initialize

a grabcut-in-one-cut algorithm [4]. This grabcut-in-one-cut algorithm, a variant of graphcut algorithm (described in Section 1.2.1) bifurcates an input image into background and foreground by labeling pixels *similar* in appearance to foreground seeds as foreground and the other pixels that appear closer to background seeds as background. The result of this graphcut algorithm is a binary labeling of an image into foreground and background pixels. However, the result from the visual attention process that supplies the foreground seeds might not produce accurate results in all cases. There are often instances when the region tagged as foreground contains edge pixels of an object. This is explained by the nature of visual attention to pick points that constitute indicators of a change in distribution which could easily be points at the edge of an object that are essentially discontinuities indicating a transition between foreground and background pixel distributions. Visual attention picking object edge pixels could result in the foreground seeds inside foreground hypothesis rectangle erroneously containing some background pixels. An illustration of this effect can be seen in Figure 1.6a where the green dot at the edge of the object is the determined visual attention fixation. The blue foreground hypothesis rectangle around the fixation point contains some wrongly labeled background pixels provided as input foreground pixels to the graphcut algorithm. Background pixels in the foreground seeds provided to the graphcut segmentation algorithm can result in inaccurate segmentation as seen in Figure 1.6b. Thus to improve the performance of the graphcut algorithm, we use a process to iteratively refine the foreground seeds. This task of iterative refinement operates by taking the result of the graphcut segmentation obtained through the foreground seeds from visual attention and computing the centroid of the foreground pixels generated after the application of graphcut algorithm. This centroid acts as the new center of the foreground hypothesis rectangle for the next iteration of graphcut segmentation. This process is repeated till the segmentation results stabilize. In other words, this iterative graphcut process is repeated till the number of foreground pixels from the previous iteration rejected as background by the current iteration is within a small threshold of 5%. In mathematical terms, if $F_i$ is the set of foreground pixels labeled



(a) (b) (c) (d)

Figure 1.6: A graphical illustration of operation of visual attention and recursive graphcut as a part of the annotation process flow is shown here. Figure a shows the visual attention fixation as a green dot. Since the visual attention fixation is on the edge of the object, the foreground hypothesis rectangle shown in blue contains background pixels. When the graphcut algorithm operates on Figure a utilizing the pixels inside the blue rectangle as the foreground seeds and the pixels inside the red rectangles in the corners of the image as background seeds, the resulting segmented foreground region is displayed in Figure b. Due to the presence of background pixels in the input foreground seeds supplied to the graphcut algorithm, the segmentation result obtained in Figure b is inaccurate. When recursive graphcut algorithm is applied, the blue foreground hypothesis rectangle eventually moves towards the center of the object, as seen in Figure c, and no longer contains background pixels. With the updated foreground hypothesis rectangle after recursive graphcut segmentation, the improved segmentation result can be seen in Figure d

.

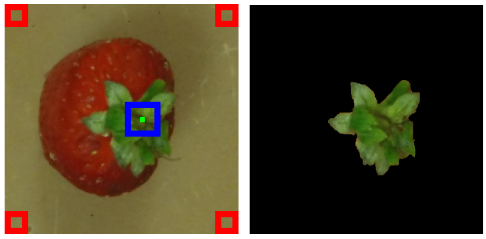
by the graphcut algorithm in iteration $i$, then the iterative process is continued till the condition in (1.6) is met. The change in the blue foreground hypothesis rectangle after recursive graphcut segmentation and the resulting improved foreground labeling can be seen in Figure 1.6c and Figure 1.6d respectively.

$$\frac{|F_{i-1} - F_i|}{|F_i|} \leq 0.05 \tag{1.6}$$

When the foreground region labels from the graphcut algorithm stabilize, its unlikely for the foreground region to change with additional iterations. This stabilization condition captured in (1.6), indicates that the foreground region label hypothesis generated by graphcut has converged. In case the convergence condition in (1.6) is not met after $N = 5$ iterations, further recursion is abandoned and the labeling results available after $N$ iterations is taken.

Though this process appears to be fully automated, there are cases where this automated segmentation fails. A case of segmentation failure could be due to foreground seeds identified by visual attention not belonging to the specimen in the image. Visual attention not picking up the specimen could be due to the presence of other *interesting* artifacts in the image that bias visual attention away from the specimen. There are also cases where the texture of the specimen is not uniform, in which case visual attention could be biased towards a specific part of the object. This could result in graphcut segmentation only identifying the sub-region of the specimen that biased the visual attention fixation. An instance of only segmenting a sub-region of the specimen is often seen in the case of strawberry which is characterized by a green stalk along with a reddish pulp. Visual attention and graphcut could end up mistakenly segmenting either the green stalk or the red pulp only. This case is graphically described in Figure 1.7. The visual attention fixation shown by a green dot in Figure 1.7a is biased towards the green stalk region of the strawberry specimen. An application of recursive graphcut on this example only ends up segmenting the green stalk region as seen in Figure 1.7b. Such cases of failure of automated annotation process are rectified here through human verification and inclusion of foreground seeds that are representative of the complete specimen, i.e. both the green stalk and the red pulp sub-regions of strawberry in this case. Therefore, to prevent failed cases of segmentation degrading the learning set, a manual verification process is added as the last step of this annotation process. During this process, for all failed instances of automated segmentation, the foreground and background seed pixels are manually set.

This final manual verification and correction process requires human effort. However, during experiments it was noted that a very small percentage of cases needed corrective action. In summary, this semi-automated annotation process is relatively less cumbersome than a fully manual annotation effort. An illustration of the output of the annotation process for a single specimen is shown in Figure 1.8. Figure 1.8 represents a montage of 13 images of an orange specimen with the top-left component showing the segmented foreground from the image captured 32 inches away from the



(a) (b)

Figure 1.7: A graphical illustration of a failure case of automated annotation process which requires human verification and correction. The visual attention fixation shown as a green dot in Figure a is biased towards the green stalk region of the strawberry. Since this strawberry specimen exhibits binary texture–green stalk and red pulp, the combined visual attention and graphcut automated annotation approach ends up segmenting only the stalk sub-region of this strawberry specimen as the foreground, as portrayed in Figure b. Such cases of failure of the automated annotation approach calls for human verification and correction of foreground seeds to enable graphcut algorithm to segment the whole strawberry specimen

.

orange specimen. Likewise the bottom-most row shows the image that was captured 8 inches away from the specimen. The montage in Figure 1.8 is arranged such that the 13 images, from left to right, are in the decreasing order of the heights from which the images were capture. This sequence of images in the montage wraps around after every 4 images resulting in four rows to represent 13 images.

### 1.4.3 Learning

1. Learning phase of a machine learning method involves parsing labeled data to identify features that can reliably indicate the identity of an object.

2. In this method, we extract features from multiple images of a target acquired from different heights.

3. Manual intervention is minimized by using visual attention to aid the detection of interesting regions in the image that could be targets.

4. Graphcut is then segments the detected interesting regions using the the visual attention fixations as foreground seeds.

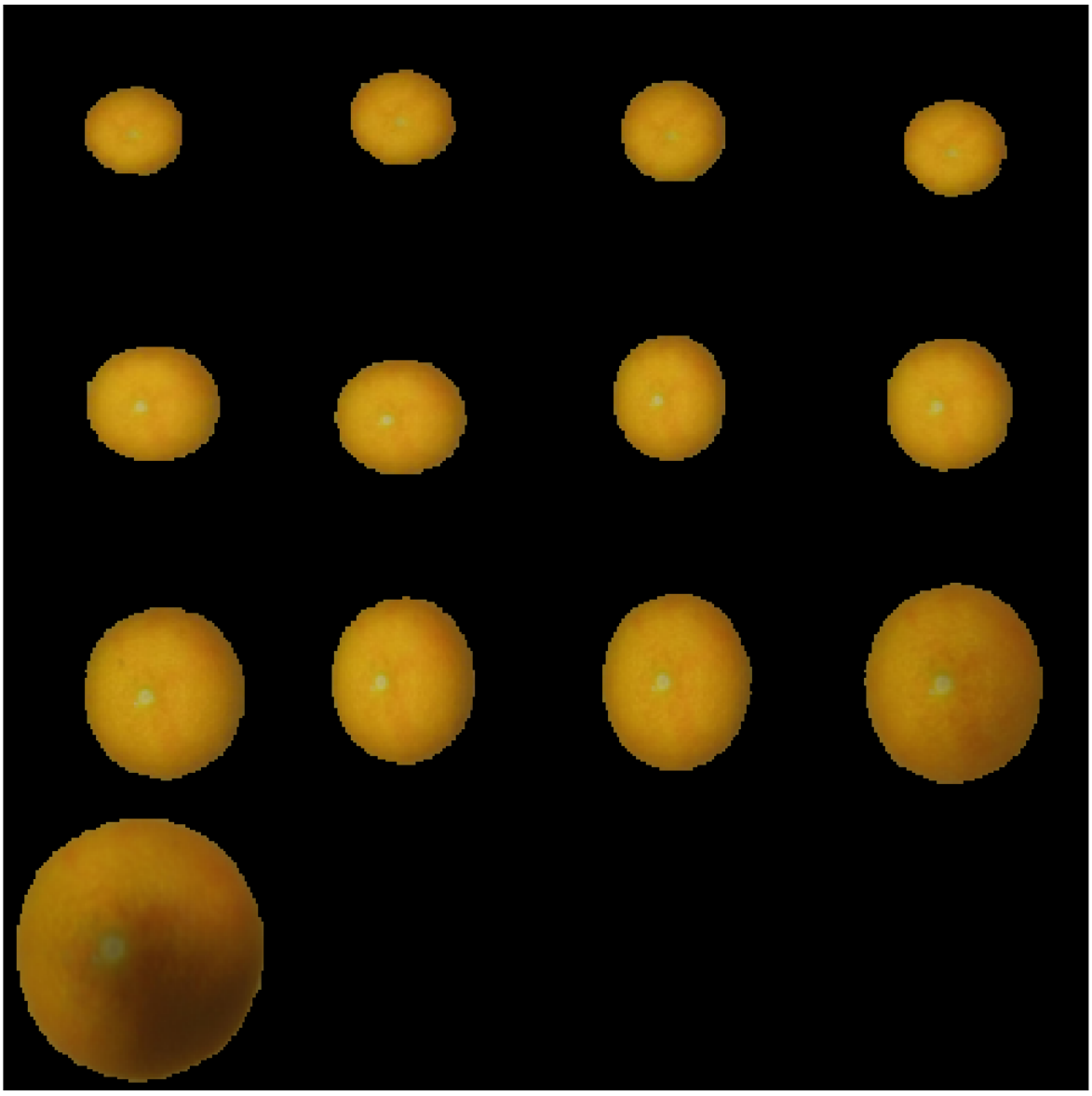


Figure 1.8: The output of the annotation process showing a montage of the foreground segmented from a set of 13 images that belong to a single orange specimen. The left-top montage component is the image of the orange specimen captured 32 inches from the ground. The montage components arranged from left to right to progressively show images that were captured closer to the orange specimen. The closest image that was captured is 8 inches away from the specimen and is shown in the bottom-most row.

5. In case of failure of this automated detection and segmentation approach seeds are selected manually for the graphcut algorithm to operate on.

6. A histogram based distance metric is used to compare the images with segmented object and the unsegmented image corresponding to each of the 13 samples available for each target object.

7. The information available from the histogram distance metric for data collected from different heights is used to construct a single feature descriptor for each instance of the target.

8. The feature descriptor from all target instances are then combined to produce a feature distribution for the target.

The learning phase of a machine learning based object recognition method involves identifying patterns in the data that represent the presence of an object. The annotated information fed to the algorithm is a pre-specified labeling of the data into foreground and background. The responsibility of the learning method here is to identify patterns in designated foreground regions, meanwhile also using information from labeled background to reject false positives. The learning algorithm then uses the patterns it identifies from the annotated foreground and background pixels to generate a learned model that is capable of identifying foreground pixels from an image.

In the machine learning technique developed in this paper, instead of using individual images with annotations of foreground and background a collection of 13 images each featuring the same specimen from a known height is utilized. This adds an additional height dimension to the feature space. The driving goal here is to encode the variation in appearance of a specimen from different heights to build a robust object recognition classifier. Height here offers an additional dimension in the feature space for the learning algorithm to capitalize on.

Since this object recognition algorithm is positioned to operate on images without prior segmentation. The features used in this algorithm should be sufficiently

generic to capture the appearance of an object directly from an image without foreground segmentation. To achieve this, we use the histogram signatures (described in Section 1.3.1) to extract information from an image in HSI-colorspace. This information from multiple specimens belonging to the same object class is then combined to generate a feature distribution. The details of this process is described in the rest of this section.

#### 1.4.3.1 Feature Distribution

The first part of the feature distribution generation process is the computation of the hue, saturation and intensity histogram signatures on the labeled foreground pixels only in the learning dataset. The parameters chosen for the histogram signature computation: the number of equally spaced bins in the histogram $|\mathcal{B}| = 256$ and the upper bound on the number of pixels affected by noise $e = 5\%$. Using these parameters the histogram signatures for all three components of the HSI-colorspace (hue, saturation and intensity) is calculated on all 13 height-tagged images of a specimen, only on the labeled foreground pixels in the image. Lets represent the histogram signature of the labeled foreground for specimen $k$ with height tag $h$ belonging to class $c$ as on the color component $f = \{H \text{ hue}, S \text{ saturation}, I \text{ intensity}\}$ be represented as $\mathcal{H}_f^{c_{hk}}$. Since there are 3 color components, we realize 3 histogram signatures per height-tagged image of a specimen. If we consider all the 13 height-tagged images available for a specimen, the total histogram signatures now available is 39 ($13 \times 3$).

In the next stage of the learning process, the histogram signatures $\mathcal{H}_f^{c_{hk}}$ from all specimens belonging to a single object class are combined to generate a generalized histogram signature for the object class. In order to achieve this, the mean of all histogram signatures corresponding to a specific height-tag from all specimens of a class are combined. In other words, the mean $\bar{\mathcal{H}}_f^{c_h}$ of all histogram signatures with height-tag $h$ are computed by taking the mean of individual components of the 256-D vectors of sequence of histogram signatures $\mathcal{H}_f^{c_{h1}}, \mathcal{H}_f^{c_{h2}}, \mathcal{H}_f^{c_{h3}}, \ldots, \mathcal{H}_f^{c_{hm}}$, where $m$ is the number of specimens of class $c$ available in the learning dataset. Mathematically

speaking the computation of the $j$-th component of the 256-D mean histogram $\bar{\mathcal{H}}_f^{c_h}$ is given as

$$\left[\bar{\mathcal{H}}_f^{c_h}\right]_j = \sum_{k=1}^{m} \left[\mathcal{H}_f^{c_{hk}}\right]_j . \tag{1.7}$$

Since there are 13 different height-tags from which data was collected along with 3 different color components, each object class $c$ is represented by a sequence of 39 ($13\times3$) histogram signatures which will be collectively referred to as $\bar{\mathcal{H}}^c$.

The sequence of 39 histogram signatures $\bar{\mathcal{H}}^c$ captures the generalized appearance of the objects of class $c$. Its important to note that the data used to compute this generalized histogram signature is composed of foreground pixels of object specimens only without any background information. If we are to use such a generalized histogram signature generated only from foreground information for classifying objects, it will necessitate prior processing of an image to detect and segment objects present in the images before this generalized histogram signature $\bar{\mathcal{H}}^c$ can be used for classification purposes. However one of the goals of this paper is to avoid segmentation while attempting classification of objects from images. To achieve this, we revisit the learning dataset to use the background information and build a representation that captures how inclusion of background pixels to different individual histogram signatures affects the generalized histogram signature $\bar{mathcal H}^c$ of class $c$.

In order to use background information for building a classified we first generate another set of histogram signatures. For specimen $k$ of class $c$ from the learning dataset, we compute a sequence of 39 histogram signatures similar to the computation involved in $\mathcal{H}_f^{c_{hk}}$ but now utilizing all pixels in the image instead of just the foreground pixels. This set of 39 histograms $\ddot{\mathcal{H}}^{c_k}$ for specimen $k$ of class $c$ is given by

$$\ddot{\mathcal{H}}^{c_k} = \ddot{\mathcal{H}}_f^{c_{hk}}\Big|_{h\times f}, \tag{1.8}$$

where $h = \{8, 10, 12, \ldots, 32\}$ and $f = \{H, S, I\}$.

Once we have the sequence of histogram signatures $\ddot{\mathcal{H}}^{c_k}$ of specimen $k$ belonging to class $c$ along with the generalized histogram signature $\bar{\mathcal{H}}^c$ of class $c$, a numeric

distance measure $D$ between the histogram signatures can be computed. If we have a series of such distance measures evaluated for a collection of specimens of class $c$, a distribution of these distance measures can be generated. This distribution of distance measures encodes the variability in generalized histogram signature $\bar{\mathcal{H}}^c$ induced by the presence of background pixels along with foreground pixels. In effect this offers an avenue to check images for the presence of an object of class $c$ without any prior segmentation of foreground pixels. The distance measure in (1.9) used for this purpose is the $L_2$-*norm* between two vectors.

$$d_{chfk} = D(\bar{\mathcal{H}}_f^{c_{hk}}, \ddot{\mathcal{H}}_f^{c_{hk}}) = \sqrt{\sum_{j=1}^{|B|} \left( \left[\bar{\mathcal{H}}_f^{c_{hk}}\right]_j - \left[\ddot{\mathcal{H}}_f^{c_{hk}}\right]_j \right)^2} \tag{1.9}$$

Since there are 39 histograms in total, we also get a sequence of 39 $d_{chfk}$ values for the $k$-th specimen of class $c$ which will be represented as $d_{ck}$,

$$d_{ck} = d_{chf}\Big|_{h \times f}, \tag{1.10}$$

where $h = \{8, 10, 12, \ldots, 32\}$ and $f = \{H, S, I\}$.

A sequence of distance measures $d_{ck}$ for each specimen $k$ among the $m$ specimens of class $c$ provides a representation of a feature vector of size 39 that describes the appearance of an image that contains a specimen of class $c$. If we consider the feature vectors of all the $m$ available specimens $d_{c1}, d_{c2}, \ldots, d_{cm}$, a feature distribution sequence $F_c$ that represents the appearance of images containing objects of class $c$ can be formulated. Lets assume that the $m$ values $d_{chf1}, d_{chf2}, \ldots, d_{chfm}$ are drawn from a normal distribution

$$F_{chf} \sim \mathcal{N}(\mu_{chf}, \sigma^2_{chf}) \tag{1.11}$$

$$\mu_{chf} = \frac{\sum_{k=1}^{m} d_{chfk}}{m} \tag{1.12}$$

$$\sigma^2_{chf} = \frac{\sum_{k=1}^{m} (d_{chfk} - \mu_{chf})^2}{m-1}. \tag{1.13}$$

Then the feature distribution sequence $F_c$ can be represented as a sequence of 39 distributions,

$$F_c = F_{chf}\Big|_{h \times f} \tag{1.14}$$

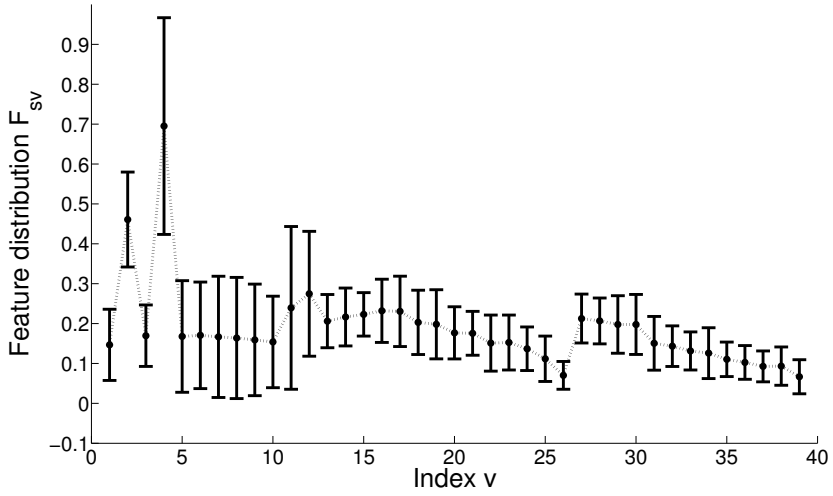
where $h = \{1, 2, \ldots, 13\}$ and $f = \{H, S, I\}$.

For the two classes of objects, Strawberry with 13 specimens and Orange with 11 specimens, their respective feature distributions $F_s$ and $F_o$ can be computed using the procedure discussed in this section. The feature distributions $F_s$ and $F_o$ are shown in Figure 1.9. The 39 distributions (see (1.11)) in the distribution sequence are shown along the $x$-axis. Each distribution here is represented by a dot and a bar adjoining it. The dot corresponds the mean and bar corresponds to the 95% confidence interval or the $2\sigma$ standard deviation interval of the distribution shown in (1.11).

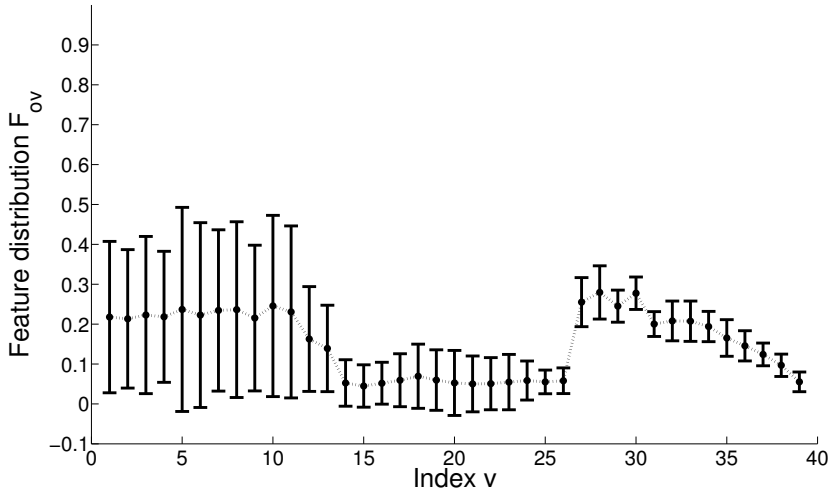
This feature distribution $F_c$ which comprises a sequence of 39 distributions offers a way to test for the presence of an object of class $c$ in an image without prior segmentation. In essence the testing for the presence of an object can be broken down into 39 individual hypothesis tests to see if an image containing an object satisfies each of the 39 distributions in $F_c$. Not all of the 39 hypothesis tests might be equally informative– some of them could be more relevant than others. To this end, a weighting factor $w_v$ can be attached to the $v$-th element in feature distribution sequence $F_c$. The details involved in the computation of these weight factors is described in the Section 1.4.3.2.

#### 1.4.3.2 Feature Weights

The feature weights offer a way to weight each distribution in the feature distribution sequence $F_c$. In case of a binary classification problem between objects of class $p$ and $q$, we choose the feature weights such that the distributions in the sequence with lower intra-class variance and minimal overlap between the distributions of classes $p$ and $q$ are weighted higher than other distributions in the feature distribution sequence. In the rest of this section the formulation of these feature distribution sequence weights is discussed.




(a)




(b)

Figure 1.9: Strawberry feature distribution sequence $F_s$ and Orange feature distribution sequence $F_o$ generated from 13 strawberry specimens and 11 orange specimens are shown in a and b respectively. The $x$-axis corresponds to the index of the distributions in feature distribution sequence. All 39 distributions (1.11) are shown along the $x$-axis. The mean of each distribution (see (1.11)) shown as a dot along with a bar that corresponds to the 95% confidence interval is also shown.

For the object class $p$ and $q$, the corresponding feature distribution sequence $F_p$ and $F_q$ are both a sequence of 39 normal distributions. Let the $v$-th component of sequence $F_p$ be $F_{pv} = \mathcal{N}(\mu_{pv}, \sigma_{pv}^2)$. The the 95% confidence interval of $F_{pv}$ is given by

$$C_{F_{pv}}^{0.95} = [\mu_{pv} - 2 \times \sigma_{pv},\ \mu_{pv} + 2 \times \sigma_{pv}]. \tag{1.15}$$

The weight $w_{pv}$ of the $v$ component of $F_p$ is given by

$$w'_{pv} = \frac{1}{(1 + |C_{F_{pv}} \cap C_{F_{qv}}|) \times \sigma_{pv}} \tag{1.16}$$

$$w_{pv} = \frac{w'_{pv}}{\sum_j w'_{pj}}, \tag{1.17}$$

where $|C_{F_{pv}} \cap C_{F_{qv}}|$ corresponds to the length of the interval obtained after the intersection of intervals $C_{F_{pv}}$ and $C_{F_{qv}}$. The first term in the denominator of (1.16) penalizes distributions whose confidence intervals share common extents with the other class in this binary classification problem. In other words, this term penalizes distributions in $F_p$ whose confidence intervals overlap with similar distribution in $F_q$. The second term in the denominator of (1.16) penalizes distributions with high intra-class variance. Ultimately the weights are normalized as shown in (1.17). The 39 feature weight sequence thus computed for a class will collectively be referred to as $W_c$. For the binary classification problem, between Strawberry and Orange specimens, the weight factors computed using (1.17) are shown in the bar plot in Figure 1.10. The black bars show the Strawberry class weights and the red bars whow the corresponding orange class weights.

The distribution components in $F_p$ with higher values of $w_p$ are weighted higher than the other distributions while performing hypothesis tests that will be discussed in the validation phase of this machine learning algorithm (Section 1.4.4).

### 1.4.4 Validation

1. To detect the presence of a target in an image, multiple images of a target are used to compute the feature descriptor.

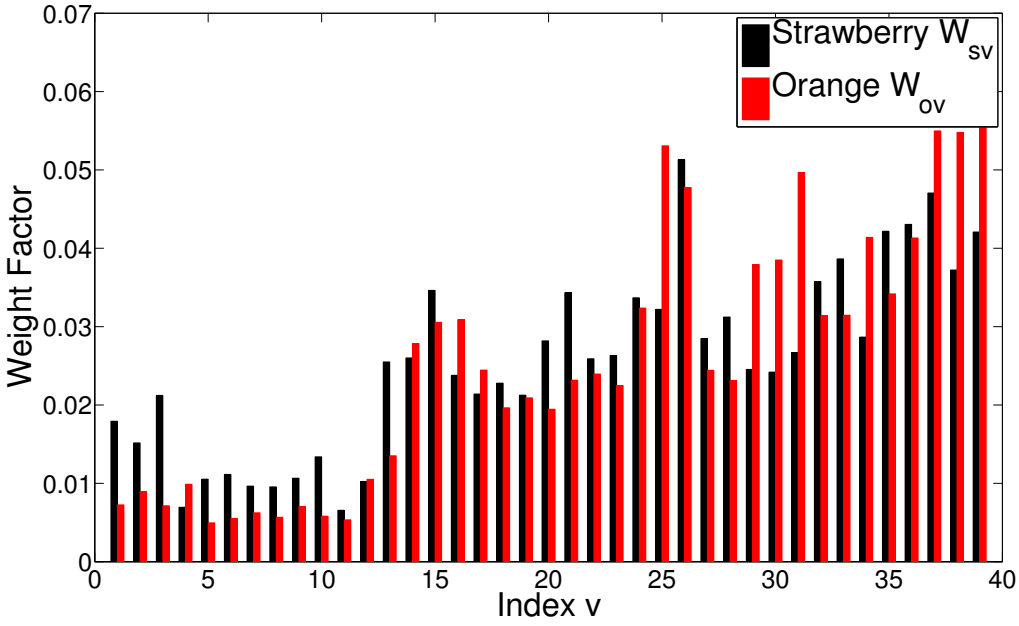



Figure 1.10: The feature weight sequences for Strawberry class $F_s$ and Orange class $F_o$ are shown as black and red bars respectively.

2. The feature descriptor is then compared against the known classes of feature distributions.

3. If a feature descriptor agrees with a feature distribution for a certain target object class, then the images currently under examination are considered to have sufficient evidence for the presence of an instance from the target object class.

In the object recognition method developed in this paper, a series of height-tagged images of an object specimen can be used to perform a binary classification task and determine the class of the object. In other words, the 13 images of 13 height-tagged images will be used to decide if the object belongs to one of the two classes $p$ and $q$. In order to achieve this, the histogram signatures (described in Section 1.3.1) are computed on each of the 13 height tagged-images available for an object specimen. Note the full image is used here as we do not have any pre-specified label information. The computation performed is identical to the one in (1.8) that results in a sequence of 39 histogram signatures $\ddot{\mathcal{H}}^k$.

The next step in the validation procedure is computing the distance measure in (1.9) between $\ddot{\mathcal{H}}^k$ and the generalized histogram signatures $\bar{\mathcal{H}}^p$ and $\bar{\mathcal{H}}^q$ of class $p$ and $q$ respectively. This results in 2 sequences of 39 distance values, $D_k^p$ against class $p$ and $D_k^q$ against class $q$, the computation of each of this sequence is identical to (1.10). The sequences $D_k^p$ and $D_k^q$ provide information about the *closeness* of the sample specimen $k$ to each of the classes $p$ and $q$. The information in these sequences need to be further distilled down to eventually attribute specimen $k$ to either class $p$ or class $q$.

Lets assume that specimen $k$ belongs to class $p$. According to this hypothesis, the $v$-th component of $D_k^p$ should satisfy the $v$-th distibution of the feature distribution sequence $F_p$. In other words, using (1.15), we can state that in 95% of the cases, the condition in (1.18) is satisfied. If we call this hypothesis test $h_{kv}^p$, then passing of this hypothesis test ($h_{kv}^p = 1$) as shown in (1.19) determines that specimen $k$ belongs to class $p$.

$$d_{kv}^p \in C_{F_{pv}}^{0.95} = [\mu_{pv} - 2 \times \sigma_{pv},\ \mu_{pv} + 2 \times \sigma_{pv}] \tag{1.18}$$

$$h_{kv}^p = \begin{cases} 1 & d_{kv}^p \in C_{F_{pv}}^{0.95} \\ 0 & otherwise \end{cases} \tag{1.19}$$

There are 39 such hypothesis tests $h_{k1}^p, h_{k2}^p, \ldots, h_{k39}^p$ that can be performed to determine if specimen $k$ belongs to class $p$. The relevance of each of these hypothesis tests in attributing the identity of the specimen $k$ to class $p$ is dictated by the corresponding weight $w_{pv}$. The information available in form of hypothesis tests along with their associated weights can be combined to get a single numeric metric or *class confidence* $H_k^p$ (shown in (1.20)) that measures the confidence in specimen $k$ belonging to class $p$.

$$H_k^p = \sum_{v=1}^{39} h_{kv}^p w_{pv} \tag{1.20}$$

Its important to note that based on the construction in (1.17) that $H_c^p \in [0, 1]$. If $H_k^p = 1$, then we can say with 100% confidence that specimen $k$ belongs to class $p$. On the fli side, the value $H_k^p$ indicates that its unlikely that speciamne $k$ belongs to

class $p$. A similar metric $H_k^q$ can be computed to determine confidence in specimen $k$ belonging to class $q$. Finally the binary classification task that classifies specimen $k$ into class $p$ or class $q$ is decided based on the magnitude of $H_k^p$ and $H_k^q$ as described in (1.21).

$$\begin{aligned} H_k^p > H_k^q \qquad & \text{specimen } k \in \text{class } p \\ H_k^p < H_k^q \qquad & \text{specimen } k \in \text{class } q \\ H_k^p = H_k^q \qquad & \text{Undetermined} \end{aligned} \tag{1.21}$$

## 1.5 Results

1. The data collected for this experiment consisted of 11 specimens of oranges and 13 specimens of strawberries each from 13 different heights in an underwater settings.

2. The feature distribution for orange and strawberry is shown in the figure.

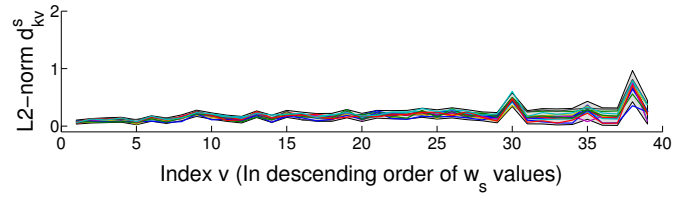
3. A leave-1-out cross-validation was performed to validate the developed machine learning technique.

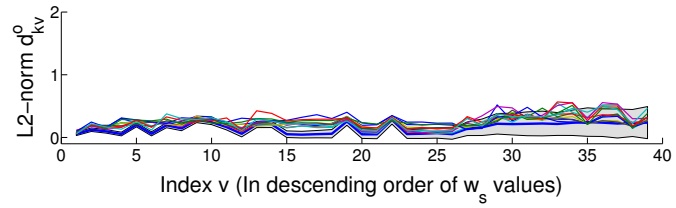
4. The feature descriptor plot for each specimen is compared against the feature distributions of the strawberry and orange class.

The data collected during the experiments consisted of images from 11 specimens from Orange class and 13 specimens from Strawberry class. Data on each specimen included images from 13 different heights, starting at 32 inches from the ground upto 8 inches away from the ground with the specimens placed below the camera on the ground. The imaging experiment was conducted in a test tank filled with water using the imaging rig shown in Figure 1.1. More details of the data collection process can be found in Section 1.4.1. Following the data collection process, the feature distribution sequence $F_s$ and $F_o$ for Strawberry and Orange classes respectively are computed using the procedure discussed in Section 1.4.3.1 and the results are shown in Figure 1.9.

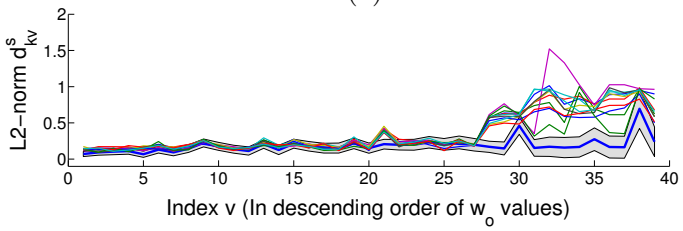
As an initial validation step both learning and testing is done on the entire dataset of 24 specimens. The learning procedure results in the feature distributions for Strawberry and Orange shown in Figure 1.9. Then each specimen is first evaluated against both Strawberry and Orange feature distributions as described in Section 1.4.4 to get distance metric $D_k^s$ and $D_k^o$. The 13 strawberry specimens evaluated against Strawberry and Orange feature distribution sequence $F_s$ and $F_o$ is shown in Figure 1.11a and respectively Figure 1.11b. The solid blue line in Figure 1.11a-*top* along with the grey band is an alternate representation of Figure 1.9a with the indices $v$ arranged in the decreasing order of their importance as dictated by the Strawberry feature weights $W_s$. In other words, the first distribution in a-top corresponds to the feature distribution with the highest feature weight in $W_s$ and the last distribution is the feature distribution with the lowest feature weight in $W_s$. Similarly the feature distribution sequence shown in Figure 1.11a-*bottom* is an alternate form of Figure 1.9b with the indices $v$ again arranged in the decreasing order of Strawberry feature weights $W_s$. The colored lines in a are the $D_k^s$ values (in a-top) and $D_k^o$ (in a-bottom) respectively for the 13 Strawberry specimens. Similarly, the two plots in Figure 1.11b-textittop and Figure 1.11b-textitbottom provide a representation of Figure 1.9a and Figure 1.9b respectively with the indices in both cases arranged in the decreasing order of Orange feature weights $W_o$. The colored lines in Figure 1.11b-textittop and Figure 1.11b - textitbottom show the $D_k^s$ and $D_k^o$ values of the 11 Orange specimens. In Figure 1.11a the colored lines mostly lie inside the 95% confidence interval band of the Strawberry feature distribution sequence $F_s$ shown in Figure 1.11a-textittop when compared against Figure 1.11a-textitbottom indicating that the Strawberry specimens match better with their own class. The same can be said from Figure 1.11a-textitbottom which shows that the orange specimens match better with the Orange distribution sequence $F_o$ rather than the Strawberry distribution sequence $F_s$.

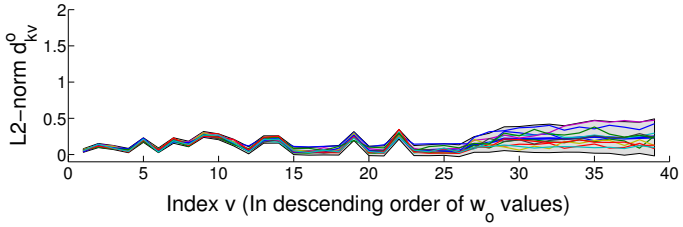
A more concrete validation of the machine learning approach discussed in this paper is by computing the *class confidence* $H_k^s$ and $H_k^o$ of each specimen against the Strawberry and Orange classes respectively and deciding on the class of the specimen as




(a)




(b)

Figure 1.11: Results obtained by using the learning dataset for testing. The plots in a shows the 13 starwberry specimens evaluated against the feature distribution sequences of Strawberry $F_s$ (top plot of a or a-top) and Orange (bottom plot of a or a-bottom). The feature distribution represented here with mean shown as a solid blue line with the 95% confidence interval shown as a grey band around the mean is an alternate representation of Figure 1.9a. The sequence of feature distributions indexed by $v$ here is arranged in the decreasing order of importance as dictated by the Strawberry feature weights $W_s$. Since the results shown in a are only for Strawberry specimens both the Strawberry distribution (a-top) and Orange distribution (a-bottom) are ordered using the Strawberry feature weights $W_s$. The colored lines in a are the $D^s_k$ values (in a-top) and $D^o_k$ (in a-bottom) for the different Strawberry specimens. The plots in b are similar to a, except that they are the evaluation results of Orange specimens against Strawberry feature sequence $F_s$ (b-top) and Orange distribution sequence $F_o$ (b-bottom). The indices $v$ in b are arranged in the decreasing order of Orange feature weights $W_o$. Both Orange and Strawberry specimens matching with most distributions of their corresponding class is a visual indicator validating the machine learning solution developed in this chapter.

described in Section 1.4.4. One thing to note here is that the class confidence measure is a comparison of a specimen against the Strawberry and Orange feature distribution sequences one of which utilized the specimen images while it was evaluated. It is a general rule to separate the learning and testing datasets before validating a machine learning approach. In this case, since the data set consists of only 24 specimens, a leave-one-out cross-validation[1] [2] was implemented. The class confidence values $H_k^s$ and $H_k^o$ of the 13 Strawberry specimens are shown as black and red bars respectively in Figure 1.12a. Its clear that the black bars are taller than the red bars in all cases. According to hypothesis test (1.19), this impies that that all Strawberry specimens are classified correctly. The same can be said for the results from Orange specimens in Figure 1.12b, since all the red bars are taller than the back bars, all the orange specimens are also classified correctly.

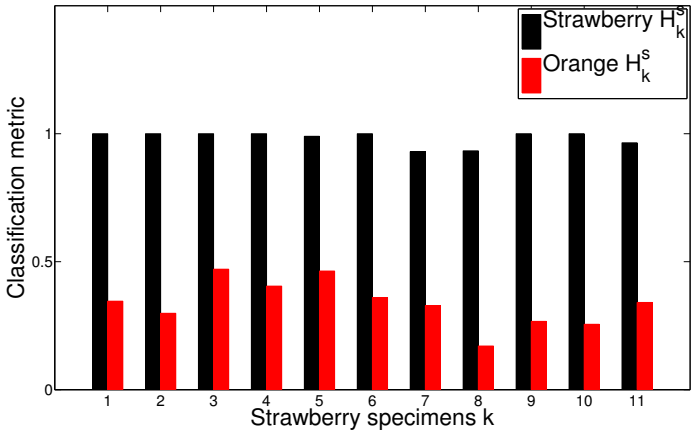
The validation procedure confirms the ability of this machine learning technique to combine data from multiple images of a specimen to do a binary classification task. The 24 specimens collectively belonging to Starwberry and Orange classes were correctly classified by the machine learning algorithm developed here.

## 1.6 Discussion

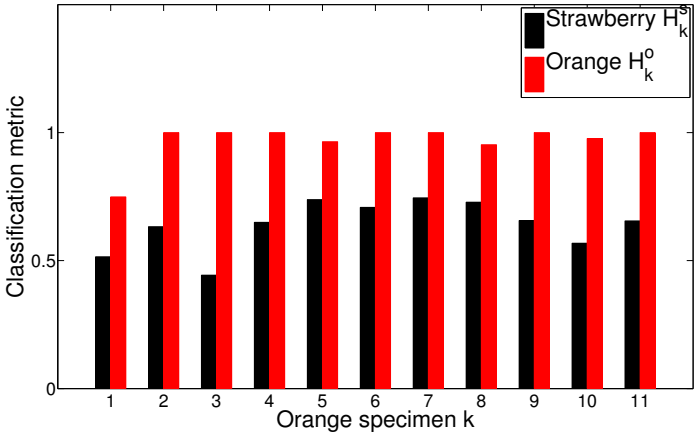
1. The plots of the feature descriptor clearly show their conformance only to the feature distribution of the target class they belong to.

2. Though capturing multiple images of a target object from different height could slowdown the imaging process and impose additional constrains on the object recognition process, detecting critical targets from noisy images justifies this end.

[1] Leave-one-out cross-validation is used in cases where the number of specimens available is very small which makes if ineffective to split the available labeled dataset into learning and testing sets. In leave-one-out cross-validation, during each iteration 1 specimen is chosen as the testing set and the all other specimens are chosen as the learning set resulting in repeatedly performing the learning procedure for each iteration.




(a)




(b)

Figure 1.12

## 1.7 Conclusion

1. The developed technique offers a way to detect objects from noisy images without any segmentation using images of a target captures from different height.

## Acronyms

**BUVA** Bottom-Up Visual Attention. 11

## BIBLIOGRAPHY


[1] Gopro hero 4. http://cbcdn1.gp-static.com/uploads/product_manual/file/256/UM_H4Silver_ENG_REVA_WEB.pdf. Accessed: 2016-10-06.

[2] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2014.

[3] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:359–374, 2001.

[4] M. Meng, L. Gorelick, O. Veksler, and Y. Boykov. Grabcut in one cut. In *International Conference on Computer Vision*, December 2013.