Color Cooccurrence Histograms for Object Segmentation and Appearance-Based Pose Estimation

Author: Staffan Ekvall
Supervisor: Frank Hoffmann

26th March 2003
Abstract

CAS, an institution at KTH, owns a large mobile platform for service robotics. The robot is equipped with a camera, a manipulator arm and a hand for grasping. This thesis addresses the problem of interpreting the visual information from the robot’s camera, in order to enable the robot to grasp an object. The problem can be divided into three subproblems: object recognition, object segmentation and pose estimation. Two methods have been tested for object recognition and segmentation: color histograms and color cooccurrence histograms. The later turned out to be suitable for pose estimation as well.

The results show that the algorithm is able to estimate the object rotation with an average error of 10 degrees, which is good enough to make the robot grasp the object. The conclusion is that a color cooccurrence histogram is an efficient way of representing an object for recognition and pose estimation. Because of its invariance to scaling and translations, the algorithm is very robust, which is required if used in a real environment. The algorithm is also invariant to changing lighting conditions.

Färg histogram av pixelpar för objektsegmentering och estimering av rotation.

Sammanfattning


# Contents

1 Introduction .......................................................... 1  
  1.1 Project Goal ..................................................... 1  
  1.1.1 Object Recognition ......................................... 1  
  1.1.2 Object Segmentation ......................................... 1  
  1.1.3 Pose Estimation ............................................. 2  
  1.2 Related Work .................................................... 2  
  1.2.1 Model Based Tracking System ............................... 2  
  1.2.2 Object Recognition with Color Cooccurrence Histograms .... 2  
  1.2.3 Appearance Based Neural Image Processing For 3-D Object Recognition ........................................... 3  
  1.2.4 Neural 3-D Object Recognition Using Optimized Gabor Filters ......................................................... 3  
  1.2.5 Probabilistic Object Recognition and Localization ....... 3  

2 Using Color Histograms for Object Segmentation .............. 3  
  2.1 Color Histogram .................................................. 3  
  2.2 Matching Color Histograms ..................................... 4  
  2.3 X- and YHistograms .............................................. 4  
  2.4 Matching X- and YHistograms .................................. 5  
  2.5 Using Color Histograms Combined with X- and YHistograms for Object Segmentation ..................................... 6  
  2.5.1 Normalizing for Intensity .................................... 6  
  2.5.2 Searching the Image for a Match ......................... 6  
  2.5.3 Window Creation .............................................. 7  
  2.5.4 Choosing a Threshold ....................................... 7  
  2.5.5 Combining with X- and YHistogram Matching for Better Results ....................................................... 7  
  2.6 Results .......................................................... 8  
  2.6.1 Test Setup ..................................................... 8  
  2.6.2 Test Results ................................................... 9  

3 Using Color Cooccurrence Histograms for Object Segmentation 9  
  3.1 Color Cooccurrence Histogram .................................. 10  
  3.1.1 Distance Cooccurrence Histogram ............................ 10  
  3.1.2 Angle Cooccurrence Histogram .............................. 10  
  3.1.3 Different Methods of Making Incomplete Histograms .... 10  
  3.2 Matching Cooccurrence Histograms ............................. 11  
  3.3 Selecting Which Colors to Store: K-means Clustering ........ 11  
  3.3.1 K-means Clustering, Algorithm [Duda et al., 2000] ....... 11  
  3.4 Using Color Cooccurrence Histograms for Object Segmentation ......................................................... 12  
  3.5 Results .......................................................... 12  
  3.5.1 Test Setup ..................................................... 12  
  3.5.2 Test Results ................................................... 12  

4 Pose Estimation ......................................................... 12  
  4.1 Various Methods of Estimating the Pose of an Object ......... 13  
  4.1.1 Appearance-based Methods .................................. 13  
  4.1.2 Feature-based Methods ..................................... 13  
  4.1.3 Combining Appearance-based Methods with Feature-based Methods ...................................................... 14  
  4.2 Using Color Cooccurrence Histograms for Rotation Estimation ......................................................... 14  
  4.2.1 Algorithm for Rotation Estimation Using Color Cooccurrence Histograms ............................................. 14  
  4.2.2 Removing the Background from the Training Images ..... 15  
  4.2.3 Parameters of Relevance ................................... 15
4.2.4 Individual Colors Versus Shared Colors ................. 15
4.3 Results .................................................................. 17
  4.3.1 Test Setup ...................................................... 17
  4.3.2 Results Using Angle Cooccurrence Histograms ........ 17
  4.3.3 Results Using Distance Cooccurrence Histograms ..... 17
  4.3.4 Results Using Normal Cooccurrence Histograms ...... 17

5 Results Running the Algorithms Together ..................... 19
  5.1 Test Setup .......................................................... 19
  5.2 Test Results ........................................................ 19
  5.3 Robustness with Respect to Viewing Angle ............... 19
  5.4 Choosing the Number of Training Images ................. 20
  5.5 Segmentation Threshold Relevance ......................... 20

6 Conclusion ............................................................. 21

7 Recommendations ..................................................... 21

8 References ............................................................. 22
1 Introduction

In the future, we can expect personal robots to aid us in our daily lives. Robots will not only work in manufacturing, but operating offices and homes as well. In order to achieve a level of autonomy required in unstructured household environment many problems have to be solved. One of them is the analysis of visual information; how to enable a robot to see. If this can be done in a robust way, a big step towards household robots has been taken.

1.1 Project Goal

This thesis deals with some of the subgoals of the project goal, described in this section. The project goal is to design a mobile robot able to pick up and deliver certain objects on demand. For instance, a typical scenario might be the user asking the robot to go to the living room and bring the box of raisins to the kitchen. For the task of grasping, the robot uses vision as its primary sensor modality. Therefore, this thesis is concerned with the problem of interpreting visual information. See Figure 1 for an image of the robot.

The project goal can be divided into six subgoals:

1. Navigation [Jensfelt, 2001]
2. Object Recognition [Robaert, 2001]
3. Object Segmentation [Robaert, 2001]
4. Pose Estimation [Kragic et al., 2002]
5. Control of arm and hand [Strandberg, 2002]
6. Grasping [Strandberg, 2002]

Algorithms have been developed by the authors mentioned above, and been integrated into a complete functional system [Petersson et al., 2002]. The current system shows room for improvement in the object segmentation and appearance based pose estimation part in terms of speed and robustness. Object recognition, object segmentation and pose estimation are problems related to each other. These problems are solved with a new approach in this thesis, with the aim of improving the robustness and speed of the overall system.

1.1.1 Object Recognition

Object recognition is needed to establish the presence of an object in the 2D-image. The algorithm is used as a foundation for the search executed by the object segmentation routine. As the segmentation algorithm frequently invokes the recognition algorithm it needs to be fast and efficient.

Two methods have been evaluated for object recognition. The first method involves color histograms, see Section 2, while the second method involves color cooccurrence histograms, described in Section 3.

1.1.2 Object Segmentation

Object segmentation is the problem of finding an object in an image and only to retain that part of the image that actually contains the object. This is useful because the background is removed, which simplifies the subsequent step of the appearance-based pose estimator.
Figure 1: The robot is equipped with a PUMA arm connected to a three-fingered hand for grasping, mounted on a mobile cylindric base (XR4000) that hosts the power supply and computer hardware.

1.1.3 Pose Estimation

Pose estimation is the problem of estimating the pose of an object based on its visual appearance in a 2D-image. The pose is the location and orientation of the object, and six parameters are needed to fully describe the pose; three translational parameters and three rotational parameters.

To simplify the problem, the objects are assumed to stand on a planar horizontal surface such as a board or table, rather than being arbitrarily oriented in space. The height of the table is known, and since the table can be considered a horizontal plane, two rotational parameters can be set to zero. Therefore, only two translational parameters and one rotational parameter needs to be estimated in this case, x, z and α. The pose estimation problem has been solved using color cooccurrence histograms, see Section 3.1 and Section 4.2.

1.2 Related Work

Much work has been done in the area of object recognition. Some of the related work is presented in this section.

1.2.1 Model Based Tracking System

A model based tracking system has been developed for this robot. The system uses PCA to estimate an initial pose and relies on a wire-frame model of the object, which is used to find and track features in the consequent images. An accurate estimate of the object's pose is obtained, which enabled the robot to successfully grasp the object. However, the PCA-based initial pose estimation is not very robust towards object translations and segmentation errors. [Kragic et al., 2002]

1.2.2 Object Recognition with Color Cooccurrence Histograms

A system for object recognition using distance color cooccurrence histograms (see Section 3.1.1) has been developed by Peng Chang and John Krumm. The system effectively segmentates an object from an image and is also robust towards partial object occlusion. [Chang et al., 1999]
1.2.3 Appearance Based Neural Image Processing For 3-D Object Recognition

An algorithm for recognizing objects with arbitrary pose in 2-D images has been created by Chunrong Yuan and Heinrich Niemann. Objects are represented with compact features extracted using the wavelet transform. The wavelet transform offers a technique of multi-dimensional signal decomposition, although it is not translational invariant. A three-layer feed-forward network is used for the object classification. To achieve translational invariance, two additional neural nets are used. One net estimates the x-translation, the other the y-translation. When these offsets are known, the coefficients of the wavelet transform can be modified accordingly. Results show a recognition rate of about 98 % in average. [Yuan et al., 2000]

1.2.4 Neural 3-D Object Recognition Using Optimized Gabor Filters

The authors created a feature-based object recognition system, based on Gabor filter kernels. Using these filters, the image is transformed into a low dimensional feature space. A neural network is used for classification, and is trained on the features extracted by the Gabor filtering. Since the feature space is low-dimensional, feature extraction does not require much computational effort, and the system can be used in real-time. The system is able to classify objects with a misclassification rate of 20 %. The strength of the system lies in that no geometric information about the objects are needed. [Heidemann et al., 1996]

1.2.5 Probabilistic Object Recognition and Localization

By representing 3-D objects by the probability density function of 2-D local characteristics, an effective object recognition system can be built. The authors propose Gaussian derivatives as local characteristics, but the same method can be applied to other characteristics. The probability density is stored in multidimensional histograms. The probability of an object with a certain pose can then be calculated, given a set of local characteristics. The results show that the algorithm has a 90 % recognition rate, even when the object is only visible to 13 %. When the object is visible to 62 % or more, the recognition rate is 100 %. The authors also present an expanded version of the algorithm, in which the object location also is estimated. [Schiele et al., 1999]

2 Using Color Histograms for Object Segmentation

Object segmentation is the problem of finding an object in an image and only to retain that part of the image that actually contains the object. In this Section an algorithm using color histograms is evaluated.

2.1 Color Histogram

The histogram of a color image can be a useful representation of that image. A histogram counts the number of pixels of each color shade. The process is computationally efficient, since each pixel only has to be counted once. The number of bins in a histogram determines the level of detail of the histogram, and is equal to how many color shades the histogram contains. The histogram is normalized, by dividing with the total number of pixels, so that images of different sizes can be compared with each other. Equation 1 provides a mathematical explanation, where the the N-binned histogram of a gray scaled image of size X and Y, with pixel values ε [0, 1] is constructed. This equation should be applied for all n ∈ [1, N]. The histogram of a color image is constructed using the histogram from the different color channels and then combining them into a single histogram. The drawback with color histograms is that they contain no geometric information, so
objects with similar colors but completely different shapes, might have similar color histograms.

$$h[n] = \frac{1}{X \cdot Y} \sum_{x=1}^{X} \sum_{y=1}^{Y} \left\{ \begin{array}{ll} 1 & \text{if } Image[x][y] \in \left[ \frac{n-1}{N}, \frac{n}{N} \right] \\ 0 & \text{otherwise} \end{array} \right. \quad (1)$$

2.2 Matching Color Histograms

There are several ways to determine the similarity of two histograms. One of the most straightforward ways is to sum the squared distances\(^1\), see Equation 2. The smaller this value, the better the histograms match each other. The problem with this similarity measure is that background color in the test image, that is not part of the object, contributes as much to the error, as the lack of object colors. A simple way to overcome this problem is to weigh each distance with the amount of object colors in that bin, see Equation 3. That way, background colors that are not part of the object do not contribute to the overall error. Figure 2 demonstrates the effects of these two methods.

$$Error = \sum_{n=1}^{N} (h_{object}[n] - h_{image}[n])^2 \quad (2)$$

$$Error = \sum_{n=1}^{N} h_{object}[n] \cdot (h_{object}[n] - h_{image}[n])^2 \quad (3)$$

2.3 X- and Y-histograms

Another type of color histograms are the X- and Y-histograms. The idea is to preserve at least some geometric information, by counting the colors along each

---

\(^1\)By 'distance' one understands the difference between two histograms in one particular bin
row (or column). This way, geometric information such as stripes and textures can be preserved in the histogram. One way to create these histograms is to compute a separate histogram for each row and color channel, but that would require lots of calculations. A simpler way to calculate X- and Y-histogram is according to Equation 4. The average value of each color channel and row is preserved. One histogram for each color channel is created. The problem with this approach is that a striped object with high intensity colors has the same histogram as a single-colored object with low intensity colors. Figure 3 illustrates this problem. Another problem is that the method is neither scale- nor rotation-invariant.

\[
\begin{align*}
    h_x[n] &= \frac{1}{N} \sum_{y=0}^{Y-1} \text{Image}[n][y] \\
    h_y[n] &= \frac{1}{N} \sum_{x=0}^{X-1} \text{Image}[x][n]
\end{align*}
\]

(4)

2.4 Matching X- and Y-histograms

The matching of X- and Y-histograms can not be done in the same way as described in Section 2.2. This is because these type of histograms are based on an average value rather than absolute counts, so the foundation of the histogram types are completely different. Instead, Equation 5 is used to measure to what extend the histograms intersect [Shapiro et al., 2001]. As Equation 3, this equation also rewards the presence of object colors, rather than to penalize the presence of background colors. The histogram is normalized by the total sum of the entire histogram such that its match value can be used for comparison, see Equation 6 [Shapiro et al., 2001].

\[
\text{Intersection} = \sum_{n=1}^{N} \min(h_{\text{object}}[n], h_{\text{image}}[n])
\]

(5)

\[
\text{Match} = \frac{\sum_{n=1}^{N} \min(h_{\text{object}}[n], h_{\text{image}}[n])}{\sum_{n=1}^{N} h_{\text{object}}[n]}
\]

(6)
2.5 Using Color Histograms Combined with X- and Y-Histograms for Object Segmentation

Using color histograms only works sufficiently well in some cases, where the colors of the object to be recognized are distinctive. The main weakness with color histograms is that two objects with completely different appearances may have very similar color histograms (see Figure 4). In these cases the results might be improved by using X- and Y-histograms. This Section is a step by step description of how to segment an image by means of color histograms in combination with X- and Y-histograms.

2.5.1 Normalizing for Intensity

In order to deal with varying lighting conditions, it is necessary to normalize the intensity. Using Equation 7, the normalized values always sum up to one [Shapiro et al., 2001]. This way, it does not matter if the picture is dark or light, the normalized colors will still be the same.

\[
\begin{align*}
\text{intensity} &= (R + G + B)/3 \\
\text{normalized red} &= R/(R + G + B) \\
\text{normalized green} &= G/(R + G + B) \\
\text{normalized blue} &= B/(R + G + B)
\end{align*}
\]  

(7)

2.5.2 Searching the Image for a Match

Once the histogram for the object is constructed, it is used to search images that potentially contain the object. The search is performed using a small search window, that sweeps across the image. The displacement at each step is small enough so that consecutive windows overlap to at least 50%, in order assure that no object is missed. The larger the overlap, the more precise is the search, at the cost of increased computational effort. Each window position is matched with the training data according to Section 2.2. Each match returns an error \( \varepsilon \). The error is transformed into a vote value, using Equation 8, which is a measure of how likely it is that this particular search window contains the object. Once the entire image has been searched, the vote matrix provides a good estimate on the possible location of the object. An example is shown in Figure 5.

\[
f(\varepsilon) = \log(1/\varepsilon)
\]  

(8)
2.5.3 Window Creation

Possible object bounding windows are constructed with the information extracted from the vote matrix. These candidate windows are then searched at a higher resolution using X- and Y-histograms. The basic idea is to start with a small window at the local maxima, and gradually expand the borders of the window until some stopping-criterion is fulfilled. In my case, the stopping-criterion is to expand as long as the ratio of the average value of the window border points, compared to the local maxima, exceeds a certain threshold \( \Phi \). For example, a threshold of 0.7 allows the algorithm to expand each border as long as the average value of the border is higher than 0.7 times the local maxima. The obvious disadvantage of this criterion is the need to identify a suitable threshold, which might depend on the objects color distribution and texture. Cross-validation techniques can be used to determine the optimal threshold value.

A normal vote image contains many local maxima. To minimize the number of windows, only those local maxima that are high enough compared to the global maxima are taken into account. The same threshold as above determines which windows become candidates.

2.5.4 Choosing a Threshold

Determining a unique threshold for all objects is difficult. Different objects and different scenes may require different thresholds. If the threshold is too high, parts of the object might be cut out, see Figure 6. On the other hand, if threshold is too low, the window contains too much background, and more background means more noise in the subsequent image processing steps. In principle, one could use cross-validation techniques to automatically adjust the threshold depending on the scene and object. However, the results show that the choice of threshold is not that important. Therefore, no such system was created.

2.5.5 Combining with X- and Y-histogram Matching for Better Results

X- and Y-histograms can be used to improve the results. After creating the windows described in Section 2.5.3, there are a number of candidate windows that might contain the object. Each window has a vote value that describes how well it matches the color histogram training data. Still, the absolute vote values are not fully reliable for the purpose of object segmentation but only provide qualitative information on possible object locations. As can be seen in Figure 4 objects that clearly look different nevertheless have similar color histograms. X- and Y-histograms provide
one way to gain additional information on which window is most likely to contain the object. Due to the additional geometric information contained in X-and Y-
histograms, the resulting match between training and image data is more reliable than for pure color histograms.

The X- and Y-histogram matching is done as described in Section 2.5.2, but since these histograms are sensitive to translations, the search-window is only shifted one pixel at a time, resulting in a slow search process. However, the total computational effort is still acceptable, as this search is only applied to the candidate windows, rather than the entire image. After searching a window, the vote from the best match is added to the vote value of that window. When all windows have been searched, the window’s vote values are compared to each other, and the window with the highest vote value is selected as the winner.

2.6 Results

The algorithm is able to segment some objects, but does not work reliably for all objects. Therefore it is not used in the overall system.

2.6.1 Test Setup

Five objects were included in the test. The system was trained for each object using one or two training images, with the background removed. The objects in the tests are shown in Figure 7. Ten test images were used, with varying lighting conditions. All test images were taken in an office environment, and all five test objects were included in each scene. Additionally, other objects with similar colors were present, to further test the robustness of the algorithm.

For each object, four system properties were tested:

1. Localization success. Is the system able to surround the object with a bounding window or not? In how many percent of the test images is the object within one of the candidate bounding windows?

2. Window number. On average, according to the ranking of vote values which window actually contains the object? The system returns several candidate windows that may contain the object. The windows are ranked according to their vote values, such that the first one is the most likely candidate. If the object is not located among the first candidate windows, the segmentation algorithm is not very robust.

3. Window size. Compared to the entire test image, how large is the bounding window on average? This value is a measure of how much background remains after segmentation. Of course, this value depends on the size of the object in the window. If the object is large enough to occupy half of the image, a value of 50 % is good. Therefore, this value
Figure 7: The objects that participated in the test.

<table>
<thead>
<tr>
<th>Object</th>
<th>Localization success (%)</th>
<th>Window number</th>
<th>Window size (%)</th>
<th>Object integrity (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rice</td>
<td>100</td>
<td>1,3</td>
<td>5,8</td>
<td>83</td>
</tr>
<tr>
<td>Mug</td>
<td>90</td>
<td>8,9</td>
<td>4,7</td>
<td>100</td>
</tr>
<tr>
<td>Raisins</td>
<td>100</td>
<td>3,3</td>
<td>9,7</td>
<td>97</td>
</tr>
<tr>
<td>Fanta</td>
<td>100</td>
<td>10,0</td>
<td>12,0</td>
<td>88</td>
</tr>
<tr>
<td>Cleaner</td>
<td>100</td>
<td>1,1</td>
<td>6,6</td>
<td>89</td>
</tr>
</tbody>
</table>

Table 1: Four parameters described in Section 2.6.1 were tested. The performance of the algorithm is good for the rice and the cleaner, which have distinctive colors. For the raisins the results are acceptable, but the algorithm fails on the mug and the fanta bottle. Even though the algorithm achieves a localization success of 90-100% on the mug and the fanta bottle, the average window numbers are so high and the windows so large, that the algorithm cannot be used to segment these objects.

is only used as a comparison among different segmentation algorithms tested on the same set of images.

4. Object integrity. On average, how much of the object is included in the surrounding window? This value is closely related to the segmentation threshold. Intuitively small window size and high object integrity are conflicting objectives. The threshold value for segmentation specifies the trade-off between both criteria. For example, better object integrity can be achieved at the cost of additional background in the segmented image.

2.6.2 Test Results
The segmentation algorithm works well for some objects, but fails on other objects such as the fanta bottle and the mug. The cleaner bottle is easily recognized, mainly because of its distinctive blue color. Most test scenes contained no other blue object. For objects of less distinctive colors, such as the mug, the results are worse. The conclusion is that the algorithm based on color histograms works well if the scene contains no other object of the same colors. Table 1 shows the results for all objects.

3 Using Color Cooccurrence Histograms for Object Segmentation
The weakness of standard color histograms is that they store no or in case of X- and Y-histograms only little geometric information about the object. Color cooccurrence histograms store pixel relations and are therefore able to capture the geometric properties of the object in a better way.
3.1 Color Cooccurrence Histogram

A color cooccurrence histogram is a statistical representation of an image. Such a histogram counts how often pairs of pixels with certain colors occur in the image [Chang et al., 1999]. It is possible to either compute the complete histogram or for computational efficiency to only subsample it.

A complete histogram stores all pixel relations in the entire image. If the image is large, calculating the complete histogram becomes very time consuming. This is because the computational complexity is proportional to the square of the number of pixels.

An incomplete histogram only stores a subset of the pixel relations. Since pixels close to each other are more likely to be found in another image of the same object, compared to pixels far from each other, it makes sense to only store pixel relations with small distances, in particular if the computational resources are limited.

By normalizing the histogram with the total number of relations, complete and incomplete histogram can be compared to each other. Also, normalization allows the algorithm to compare histograms computed from images of different sizes.

In addition to the large computational demands, cooccurrence histograms possess another disadvantage. The number of entries in the histogram is much larger than for the regular histograms. Reducing the number of colors is necessary in order to restrict the amount of data to a reasonable size. With fewer colors, it is more important to build the histogram based on the most representative set of colors. Section 3.3 discusses the problem of finding the optimal color scheme.

3.1.1 Distance Cooccurrence Histogram

A distance cooccurrence histogram not only stores the pixel pairs, but also the distance between the pixels. The idea is to make the histogram contain even more geometric information, providing better data to be used for matching. The obvious disadvantage is that the histogram is scale sensitive, and to in order to use the extra information, the different images probably would have to correspond to the same scale.

3.1.2 Angle Cooccurrence Histogram

An angle cooccurrence histogram is like the distance cooccurrence histogram, but instead of the distance, the angle between the line connecting the two pixels, and a fixed direction\(^2\) (e.g. the x-axis), that is stored instead. This approach solves the problem of different scales, but if the object is rotated, the angle changes and the histograms no longer match. If the purpose of the matching is to estimate the rotation, this actually is a desirable feature, since the histograms should not match for different rotations. On the other hand, if the purpose is to find an object, it is better if the histograms match even if the object is rotated. To summarize: In theory, distance cooccurrence histograms should be the best choice for object segmentation, and angle cooccurrence histograms should be the best choice for rotation estimation.

3.1.3 Different Methods of Making Incomplete Histograms

Incomplete histograms can be computed in several ways. A simple method is to set a maximum distance \(d\), and store all pixel relations for which the distance between the pixels is less than \(d\). The advantage of this method is that it gives perfect histograms at low distances, for which the same pixel relations are most likely to be found in another image. The disadvantage is that no information about pixel

\(^{2}\)In my case, the objects are assumed to be standing on a table. Therefore, there is no significant rotation around the camera axis, and the x-axis can be used as the fixed line. If the objects are allowed to be rotated around the camera axis, no fixed line can be used. Instead, the angle between two lines connecting three pixels is used, but this makes the histogram more complex.
relations at large distances is stored. For objects with little texture, this might become a problem: Most color relations stored are pairs of colors that are the same, while calculating a complete histogram also stores all the color relations where the colors are not identical. There is a risk that the incomplete histogram does not match the reality very well.

Another way of computing incomplete histograms is to sample from a Gaussian distribution. For each pixel, a number of samples from the surrounding pixels are taken. Since a Gaussian sample distribution is used, pixels close to each other are more likely to be sampled than remote pixels. The advantage with this method is that it is easy to adjust the complexity of the histogram by specifying the number of samples per pixel. The disadvantage is that the same pixel may be sampled several times. This can be avoided by keeping track of which pixels already have been sampled, but that would be too time consuming.

3.2 Matching Cooccurrence Histograms

The matching of cooccurrence histograms is done in the same way as the matching of the X- and Y- histograms. If both histograms are normalized, Equation 9 provides a good measure of the histogram match [Chang et al., 1999]. \( D \) is the total number of distances and \( C \) is the number of colors. The same equation is used when storing angles instead of distances.

\[
\sum_{d=1}^{D} \sum_{a=1}^{C} \sum_{b=1}^{C} \min(H_1(d, a, b), H_2(d, a, b))
\]  

3.3 Selecting Which Colors to Store: K-means Clustering

In a perfect histogram, all colors are stored as their true RGB value. Such a large histogram is not suitable for object recognition though, as most entries in the histogram remain empty. If the same object has slightly different color tones in different images, the histograms of those two images no longer match. Instead, the colors are clustered, for example all orange tones are assigned to one color cluster, so that the histogram becomes more robust towards small variations in color.

The best way to cluster the colors is to locate cluster centers according to pixel density in color space. If, for example, the image is mainly blue and green, there is no need of having clusters covering the red area. A simple algorithm to partition a data set into \( C \) clusters is K-means clustering. The algorithm is described in Section 3.3.1.

A problem that one faces in clustering is that data points far away from their closest cluster center still contribute to the histogram count of that cluster. This means that the color red actually might be counted in the blue cluster, if red region does not contain a cluster. An easy way to deal with this problem is simply to ignore data points whose distance to the nearest cluster center is larger than a certain distance \( d \). The problem is how to determine an optimal values for the cluster radius. I used cross-validation to determine an optimal value for \( d \). Of course, \( d \) depends on the color scale. If a color scale of \([0, 255]\) is used, the optimal \( d \) is much larger than if a color scale of \([0, 1]\) is used.

3.3.1 K-means Clustering, Algorithm [Duda et al., 2000]

1. Randomly initialize all clusters.
2. For each data point, assign it to the closest cluster center lying, according to some distance metric, e.g. Euclidean distance.
3. Recompute each cluster center as the mean value of all data points assigned to it.
<table>
<thead>
<tr>
<th>Object</th>
<th>Localization success (%)</th>
<th>Window number</th>
<th>Window size (%)</th>
<th>Object integrity (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rice</td>
<td>100</td>
<td>1.2</td>
<td>3.6</td>
<td>76</td>
</tr>
<tr>
<td>Mug</td>
<td>90</td>
<td>1.0</td>
<td>2.4</td>
<td>72</td>
</tr>
<tr>
<td>Raisins</td>
<td>100</td>
<td>1.3</td>
<td>2.6</td>
<td>84</td>
</tr>
<tr>
<td>Fanta</td>
<td>100</td>
<td>1.2</td>
<td>3.1</td>
<td>69</td>
</tr>
<tr>
<td>Cleaner</td>
<td>100</td>
<td>1.0</td>
<td>2.6</td>
<td>59</td>
</tr>
</tbody>
</table>

Table 2: *Four parameters described in Section 2.6.1 were tested. The system only* failed once, where it missed the mug.

4. If a cluster center does not have any data points assigned to it, randomly reinitialize it.

5. If any cluster center has moved or been reinitialized, go to step 2.

3.4 Using Color Cooccurrence Histograms for Object Segmentation

Once the histogram is created from the training images, the same search algorithm for image segmentation as described in Section 2.5.1 to 2.5.4 is used.

3.5 Results

The algorithm works better than the one using color histograms combined with X- and Y-histograms (see Section 2.6.2). It is able to reliably segment all objects, and is therefore implemented as a part of the overall system.

3.5.1 Test Setup

The test setup was identical to the experiment described in Section 2.6.1.

3.5.2 Test Results

The algorithm works well for all objects, as can be seen from the results shown in Table 2. In most cases the object is bounded by the first window, and in the remaining cases the object is bounded by the window ranked second. The size of the window is in all cases smaller compared to the window size returned by the normal color histogram segmentation algorithm (Section 2.6.2). Sometimes the size is too small, preserving only 40% of the object. This is why the object integrity values are lower in Table 2 compared to Table 1. It is possible to increase the object integrity by reducing the segmentation threshold, see Section 2.5.4. A qualitative analysis indicates that the system is at least partially robust to object occlusion. Figure 8 illustrates a typical segmentation result from this algorithm.

4 Pose Estimation

Pose estimation is the problem of estimating the pose of an object, from a 2D-image of the object. The pose is the location and orientation of the object, and six parameters are needed to fully describe the pose; three translational parameters and three rotational parameters.

To achieve the project goal described in Section 1.1, the pose of the object is needed. The mobile manipulator needs this information for the controller to properly align the arm and hand with the object. The full six-degree pose estimation problem is a very difficult task to solve. To simplify the problem, the objects are assumed to be standing on a table. The height of the table is known, so only two translational parameters and one rotational parameter need to be estimated in this case, x, z and y.
4.1 Various Methods of Estimating the Pose of an Object

There are several methods to estimate the pose of an object. Each has its pros and cons. The methods can be divided into two groups: appearance-based and feature-based methods. Appearance based methods are, as the name implies, based on the overall appearance of the object, while the feature based methods are based on specific geometric features of the object.

4.1.1 Appearance-based Methods

An example of an appearance-based method is the PCA algorithm. PCA is a mathematical method that, for a collection of data, finds the directions for which the data set shows the most variance. These axes are called eigenvectors. The data is represented as the projection along the eigenvectors. This way, the number of dimensions is reduced while most of the information in the image is still preserved.

The idea of how to estimate the pose of an object is to compare the PCA of the image with previously stored PCAs of the same object with known poses. The advantages of using PCA are that the data stored during training, tends to be small, and the construction of the PCAs is relatively fast. The main disadvantage is that this method is highly sensitive to translations.

4.1.2 Feature-based Methods

Features are a way of representing objects by calculating values from the image in the form of descriptors. Direct pixel values, derivatives, contours and curvature measures are some examples of features. A way of estimating the pose of an object is to try and match features found in the image with features found either in the training images or a geometric model of the object. However, this so called correspondence problem is often difficult to solve.

A feature-based method is used to calculate the pose of the training and test images used in the tests (described in Section 4.2). In this case, the correspondence problem is solved by manually matching the corresponding points. The features are the corner points of the object and the user manually selects corresponding corners in the image and the model. The method produces accurate estimates in a robust way, with an average estimation error of about 5 degrees.

In general, feature-based methods are more accurate than appearance-based methods. The main disadvantage of feature-based methods is that they are not as robust as appearance-based methods.
4.1.3 Combining Appearance-based Methods with Feature-based Methods

If the accuracy of feature-based methods could be combined with the robustness of appearance-based methods, a robust and accurate pose estimation algorithm could be developed. The key idea of the integrated algorithm is to obtain an initial pose with the appearance-based method, and use this estimate to project features of the object model into the image. These projected features are used to initialize the local search and match of corresponding features in the image. This approach reduces the global correspondence problem of feature based matching to a local tracking problem.

4.2 Using Color Cooccurrence Histograms for Rotation Estimation

Using color cooccurrence histograms is an appearance-based method that is used for rotation estimation. The idea is to learn the histograms of the object in different poses, and then match the histogram of the object with unknown pose, to these learned histograms. Color cooccurrence histograms are described in Section 3.1.

The appearance of an object is represented by a color cooccurrence histogram. From this information the matching algorithm is expected to extract the rotation \( \alpha \) of the object. In the specific context of this problem, the translational parameters \( x \) and \( z \) can not be easily estimated by appearance-based methods, because the appearance of the object does not change significantly for comparatively small translations of the object on the table. The signal to noise ratio in the resulting cooccurrence histogram turns out to be too low for reliable translational estimation. Therefore, the \( x \) and \( z \) parameters will have to be estimated by other means.

The segmentation algorithm described in Section 3 could be used to estimate \( x \) and \( z \). If the center of the window returned from the segmentation algorithm is assumed to be the center of the object, it is possible to transform the pixel coordinates \( x \) and \( y \) in the image into the world coordinates \( x \) and \( z \). This transform can be done using a closed form equation, if all the parameters such as table height and camera lens focal width are known. Another way of solving the problem is to train a back-propagating neural network to learn this transformation, namely the mapping from \( x \) and \( y \) in the image and the relative height between camera and object as input to the 3D object position relative to the camera coordinate frame.

4.2.1 Algorithm for Rotation Estimation Using Color Cooccurrence Histograms

As mentioned in Section 3.1.2, one would assume angle cooccurrence histograms to give the best results. However, normal cooccurrence histograms turned out to be the representation that provides more accurate estimates. Therefore, this and the following sections only cover the rotation estimation using normal color cooccurrence histograms. A way to perform rotation estimation using color cooccurrence histograms is presented in this Section in a step by step manner.

The first step is to train the system. More training images tend to improve the performance of the algorithm, however after about 50 training images no significant improvement was observed, see Section 5.4. For each training image, compute the complete cooccurrence histogram and store it together with the rotation of the object in the image.

The next step is to compute the cooccurrence histogram of the test image. Depending on the amount of time available, this histogram is computed based on a complete or incomplete sample.

The last step is to match this histogram to each of the training histograms, according to Equation 9. Each match generates a match value. The higher the
value, the more probable it is that the object is rotated by the same angle as the object in the training image.

A nearest-neighbor approach would be just to select the histogram with the highest match, and take the stored angle of that histogram as the most likely estimate of the unknown angle. However, taking advantage of all the match data improves the robustness towards outliers and coincidental matches. The contribution of each histogram to overall estimate is weighted by a Gaussian kernel according to the similarity of the match. If the training set consists of N images, and each image with an actual angle of α generates a match value µ, the output for angle k is calculated according to Equation 10. In this equation, g(k, α) is the Gaussian overlap calculated as Equation 11, where std is the standard deviation of the Gaussian function. This equation is then applied to all angles k ∈ (-180, 180]. Figure 9 illustrates the distribution of the match data before and after this step. Note that the second graph is covering all points, even those between the training images. This is because of the Gaussian function, which is calculated for each angle. The maximum value in the new match distribution corresponds to the most likely angle. Because of the normalization according to Equation 10, the training images do not need to be uniformly distributed across the angle. Still, if training images are missing in some angular region, objects rotations with an angle in that interval will probably be estimated incorrectly.

\[
Output(k) = \frac{\sum_{i=0}^{N} (\mu_i \cdot g(k, \alpha_i))}{\sum_{i=0}^{N} g(k, \alpha_i)}
\]

(10)

\[
g(k, \alpha) = \exp\left(\frac{-\min(k-\alpha, |k+360-\alpha|, |k-360-\alpha|)^2}{2 \cdot std^2}\right)
\]

(11)

4.2.2 Removing the Background from the Training Images

To minimize the noise in the training images, the background has been removed from the images before training. This can be done either manually or using some form of color analysis or other suitable algorithm. I have chosen to remove the background manually to ensure that the objects remain intact. The training algorithm ignores pixels with background color when computing the histograms. Removing the background in the training images improves the results a great deal. Even though background removal is not applicable to the test images, the accuracy increased due to the less noisy training data.

4.2.3 Parameters of Relevance

Several parameters can be varied for optimal results. These are:

1. C: The number of colors used (Section 3.3)
2. d: The minimum distance to the nearest cluster (Section 3.3)
3. std: The standard deviation of the Gaussian kernel (Section 4.2.1)

The optimal values of these parameters were determined using cross-validation. Of course, the optimal values of these parameters varies with the object and its environment. For the rice packet, the optimal values were found to be C = 50, d = 0.08 and std = 1, when using a color scale [0, 1].

4.2.4 Individual Colors Versus Shared Colors

When clustering the colors, two methods can be used. Either each training image get its own cluster centers, or the entire training set shares the cluster centers.
Figure 9: The match data before and after filtering with a Gaussian kernel.
Colors are better represented if each training image relies uses its own cluster centers. However, for each training image that is to be matched with the test image, a new histogram needs to be created. This means, that the computational effort for calculating the histograms of one test image grows linearly with the number of training images. If instead shared clusters centers are used, only one histogram needs to be generated for the test image. Of course, the number of matching operations still grows linearly in the number of training images. However as matching is much faster than histogram calculation, shared clusters centers are to be preferred. Test results have shown that the loss in accuracy is quite small, compared to the significant gain in computational efficiency.

4.3 Results

The algorithm works well and robust, and is capable of estimating the object rotation with a small error. Therefore, the algorithm is integrated as a part into the overall system.

4.3.1 Test Setup

The test was done using 70 training images, with their backgrounds removed. For testing, 30 unmodified, not previously seen test images were used. See Figure 10 for an example.

4.3.2 Results Using Angle Cooccurrence Histograms

First, angle cooccurrence histograms were tried, since they were assumed to give the best results. Figure 11 shows the results with respect to the number of angle bins used. The reason that the variance of the error with respect to the number of bins is so large, is that Gaussian samples were used during histogram generation, see Section 3.1.3. The graph indicates that there is a relationship between the number of angle bins and the mean angle error. The error decreases up until about 20 bins, after which it remains relatively stable.

4.3.3 Results Using Distance Cooccurrence Histograms

The results using distance cooccurrence histograms turned out to be even better than those obtained for the angle cooccurrence histograms. In addition, the number of distance bins used does not seem to have any effect on the results, as Figure 12 shows. These results imply that using normal cooccurrence histograms, that is, neither storing distance or angle, is the best solution.

4.3.4 Results Using Normal Cooccurrence Histograms

As mentioned earlier, the results were best when using normal cooccurrence histograms. As Figure 13 shows, the average angle error was 18 degrees when a maximum radius of 10 pixels was used in the generation of the cooccurrence histograms.
Rotation estimation using angle cooccurrence histograms

Figure 11: Rotation estimation using angle cooccurrence histograms. The figure shows how the results depend on how many angle bins used in the histogram.

Average angle error, using distance cooccurrence histograms with maximum distance 10

Figure 12: The number of distance bins used does not seem to have any effect on the results.
5 Results Running the Algorithms Together

Finally, the segmentation algorithm and the rotation estimation algorithm were tested together. The cropped image from the segmentation algorithm was fed into the rotation estimation algorithm. The results were surprisingly good.

5.1 Test Setup

The same training images used in the experiments described in Section 4.3.1, were used in these experiments. The test images were large, uncropped images, such as the one in Figure 14. For this test, 30 test images were used.

5.2 Test Results

Test results were even better than those for the manually cropped images in the previous Section. An average angular error of 10 degrees was achieved, which is quite remarkable, considering that the correct angle computed by the feature based method using manual matching of features carries an uncertainty of about 5 degrees (see Section 4.1.2). An explanation for these good results is that as the segmentation algorithm efficiently extracts mostly object pixels, the cropped images contain much less background (noise), compared to the cropped images used in the previous tests (see Figure 10).

5.3 Robustness with Respect to Viewing Angle

To test the algorithm’s robustness with respect to viewing angle, additional cross-validation tests were performed on another object, the box of raisins. The camera tilt angle was varied between 0 and about 30 degrees for the test- and training images. Figure 15 illustrates the difference in object appearance. An average angular error of 24.8 degrees was obtained. Because of the similarity in visual appearance
of the different sides of the packet, some angular errors of 180 degrees contributed to increase the mean angle error. For the purpose of grasping an object it does not matter whether the packet is rotated -90 degrees or +90 degrees, so these 180-degree errors are somewhat irrelevant. Due to this, the errors were modified according to Equation 12. The new test resulted in an average angle error of 14.2 degrees. From this, the conclusion is drawn that the algorithm is robust with respect to viewing angle.

\[
\text{Error } \varepsilon = \min(\varepsilon, 180 - \varepsilon)
\]  

(12)

5.4 Choosing the Number of Training Images

The question of how the number of training images affects the results is answered in Figure 16. Based on the derivative of the learning curve, the conclusion is that using more than 50 training images does not improve the results.

5.5 Segmentation Threshold Relevance

As described in Section 2.5.4, the segmentation threshold determines how much of the object is kept after the segmentation. For a low threshold, the entire object remains intact but a larger part of the background becomes included as well. For a high threshold, only the core region of the object remains after segmentation. This appears to cause a serious problem in robustness, since different scenes and objects require specifically adjusted thresholds. Therefore, the influence of the threshold on the performance was investigated. Figure 17 shows the relationship between the threshold and the mean angular error. As expected, using a too low threshold
Figure 16: The learning curve shows that adding more training images would probably not improve the results.

includes too much background and makes rotation estimation difficult. In contrast, when using a high threshold, the algorithm still seems to be working quite well. The conclusion is that one prefers a high threshold to a lower one.

6 Conclusion

A color cooccurrence histogram is an efficient way of representing the appearance of an object in the context of object recognition and pose estimation. Because of its invariance to scaling and translations, the algorithm is very robust, a property that is essential for using these methods in a real world environment. Color cooccurrence histograms are quite sensitive to object rotations. That feature makes it possible to use them for rotation estimation. Background clutter and noise disturbs the algorithm, but it still performs well. The algorithm is also invariant towards changing lighting conditions, because of the color normalization.

7 Recommendations

The system should be used in combination with another independent object recognition algorithm. The segmentation algorithm produces several candidate windows, and an additional, possibly computationally demanding algorithm is needed to determine which of the first few candidate windows actually contains the object. This algorithm does not need to have the same speed as the segmentation algorithm, since it only considers the preselected candidates rather than searching the entire image.

Once the object is segmented, and the pose is estimated, a feature-based pose estimator such as the model based tracking algorithm proposed in [Kragic et al., 2002] can be used to further improve the accuracy of the estimated pose (see Section 4.1.3). Additionally, this pose estimation algorithm could be combined with another appearance-based pose estimator (e.g. a PCA based system) to provide even
more robustness.

8 References


[Robaert, 2001] 

[Schiele et al., 1999] 

[Shapiro et al., 2001] 
**Computer Vision**; L. Shapiro and G. Stockman; Prentice Hall; 2001.

[Strandberg, 2002] 

[Yuan et al., 2000] 