Real Time Object Recognition

Progress Report in partial fulfillment of the requirement for the degree of

Bachelor of Technology
In
Computer Science and Engineering

By
Sahil Narang

To

MAHARAJA SURAJMAL INSTITUTE OF TECHNOLOGY
Affiliated to Guru Gobind Singh Indraprastha University
Janak Puri, New Delhi-110058.

August 2012

Acknowledgement

I thank my respected guide, Dr. Johannes Mohr, Neural Information Processing Group, Technischen Universität Berlin from the bottom of my heart for his excellent guidance and supervision through all the phases of the development of this project. I would also like to thank Mr. Jong-Han Park for his immense cooperation and work in this project. Most importantly, I would like to thank Dr. Klaus Obermayer for welcoming me into the NIP group.

It was due to Dr. Mohr's toiling efforts that we tackled such a complex project. He encouraged us to gather the professional knowledge and material for the completion of this project. He helped us with his invaluable and finest suggestions to give shape and finesse to the work done.

I sincerely hope that users will find this project useful. Whatever intellectual effort may be reflected from this report is the direct result of the informative and stimulating discussions that I have had in the course of this internship. Given the time constraint and continuous nature of a research activity, there is always a scope to ameliorate object recognition techniques

Note:

This project is currently in progress. Therefore, a majority of this report is not available presently.

Sincerely,

Sahil Narang

.

TECHNISCHEN UNIVERSITÄT BERLIN

The **Technische Universität Berlin** (**TUB** or **TU Berlin**) is a research university located in Berlin, Germany. The TU Berlin was founded in 1879 and is one of the largest and most prestigious technical universities in Germany. The university alumni and professor list include National Academies elections, two National Medal of Science laureates and ten Nobel Prize winners. The TU Berlin is a member of TU9, an incorporated society of the largest and most notable German institutes of technology and of the Top Industrial Managers for Europe network. It also belongs to the Conference of European Schools for Advanced Engineering Education and Research. As of 2011, TU Berlin is ranked 46th (2010: 48th) in the world in the field of Engineering & Technology according to QS World University Rankings.

The internationally renowned Technische Universität Berlin is located in Germany's capital city at the heart of Europe. Its activities focus on building a distinctive profile, exceptional performance in research and teaching, excellent qualifications for its graduates and a forward-looking administration. The TU Berlin strives to promote the dissemination of knowledge and to facilitate technological progress by adhering to the principles of excellence and quality. Strong regional, national and international networking partnerships with science and industry are an important aspect in this regard.

Its research and teaching endeavors can be characterized by a broad spectrum of academic disciplines, ranging from engineering science to natural science, planning science and economics, as well as the humanities and social sciences.

BERNSTEIN CENTER FOR COMPUATIONAL NEUROSCIENCE

The Bernstein Center Berlin addresses one of the most challenging questions in computational and cognitive neuroscience:

"How is it possible that we can react to sensory stimuli with millisecond precision if intermediate processing elements – on the level of single synapses, single neurons, small networks and even large neural systems – vary significantly in their response to the same repeated stimulus?"

The Center's interdisciplinary research is executed by groups of the Charité, Freie Universität Berlin, Humboldt-Universität zu Berlin, Max-Delbrueck-Centrum, Technische Universität Berlin and the Universität Potsdam. It is part of the National Bernstein Network Computational Neuroscience (NNCN) and funded by the Federal Ministry of Eduation and Research (BMBF).

I. Neural Information Processing Group

The NIP group is concerned with the principles underlying information processing in biological systems. On the one hand they attempt to understand how the brain computes, on the other hand they want to utilize the strategies employed by biological systems for machine learning applications.

A. Areas of Work

- ***Models of Neuronal Systems***

In collaboration with neurobiologists and clinicians we study how the visual system processes visual information. Research topics include: cortical dynamics, the representation of visual information, adaptationand plasticity, and the role of feedback. More recently we became interested in how perception is linked to cognitive function, and we began to study computational models of decision making in uncertain environments, and how those processes interact with perception and memory.

- ***Machine Learning & Neural Networks***

Here we investigate how machines can learn from examples in order to predict and (more recently) act. Research topics include the learning of proper representations, active and semisupervised learning schemes, and prototype-based methods. Motivated by the model-based analysis of decision making in humans we also became interested in reinforcement learning schemes and how these methods can be extended to cope with multi-objective cost functions. In collaboration with colleagues from the application domains, machine learning methods are applied to different problems ranging from computer vision, information retrieval, to chemoinformatics

- ***Analysis of Neural Data***

Here we are interested to apply machine learning and statistical methods to the analysis of multivariate biomedical data, in particular to data which form the basis of our computational studies of neural systems. Research topics vary and currently include spike-sorting and the analysis of multi-tetrode recordings, confocal microscopy and 3D-reconstruction techniques, and the analysis of imaging data. Recently we became interested in the analysis of multimodal data, for example, correlating anatomical, imaging, and genetic data.

Table of Contents

INTRODUCTION

I. MOTIVATION

Recognizing familiar faces in a crowd, detecting a ball when playing soccer, differentiating between cats and dogs, Classifying letters when reading. All of these are trivial tasks for human beings. Within 150 milliseconds the human visual system can detect and discriminate between an incredible diverse assortment of stimuli, in motion or not, patterned or un-patterned, 2-D or 3-D. As a result, the human brain manages the recognition of 3-D objects in an impressive way without having problems with variability in the appearance, due to viewpoint, illumination or occlusion. However, understanding the human visual processing is very complex because our subjective impressions tell us little about the way we accomplish these daily tasks. For instance, although it seems that we can recognize objects equally well from any viewing angle, experiments in cognitive vision do not confirm this assumption. Nevertheless, the goal of many scientists is to create computer vision systems which manage to work both, as fast and as accurate as the human visual system.

II. OBJECT RECOGNITION IN COMPUTER VISION

This project especially deals with recognition of 3-D objects. Therefore, first we will give a short overview of the object recognition process in general and second, commonly used methods for creating descriptions for 3-D objects will be discussed.
In recent years much research has been done in the field of object recognition.
Generally, in computer vision an object recognition system is divided into two parts - a training phase and a classification phase as illustrated in figure 1. Training is usually done from single images containing the respective object in different poses. From these images an object representation is extracted which contributes to the knowledge of the recognition system. The task of the classification phase is to identify learned objects in a given scene. Therefore initially a description of the scene is needed in order to make a classification, whether the object appears in the scene or not.

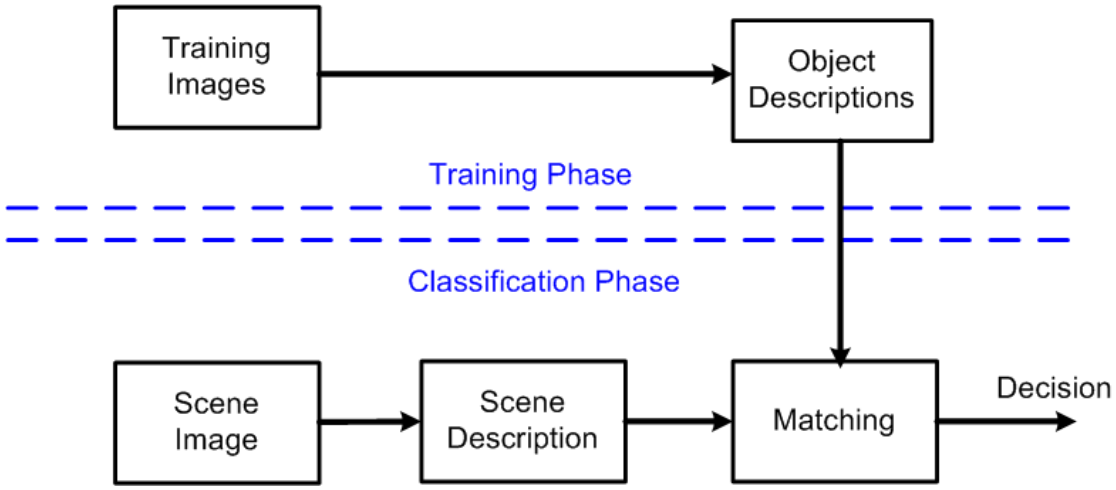

Figure 1: Scheme of common recognition application

At this point we can already imagine that object representations are of great importance for each recognition system. As previously mentioned the aim of an object recognition system is to work fast and accurately. However, in order to be fast, it is necessary to have sparse representations which save memory and in addition accelerate classification because only a small amount of data has to be processed. However storing small descriptions of objects at the same time implies, the need of distinctive descriptions to be further on resistant against the variability in appearance of objects in scenes. Therefore the challenge is to extract sparse and robust representations. Different methods have been presented for describing 3-D objects from images.

First there are shape-based methods which propose to create 3-D computer models from objects in images. However, the major difficulty of these approaches is to generate 3-D models of information available from 2-D images. Any 2-D image is always consistent with infinitely many 3-D interpretations. Therefore it is not possible to build a correct 3-D model from a single image because only information of a particular viewpoint is available. In other words, nothing is known about what is going on behind the surface. In addition, studies of numerous neuro-scientists are a further argument against shape-based methods. For instance, Logothetis has made extensive experiments with monkeys showing that learning of 3-D objects is done view-dependent. However shape-based methods would suggest to be capable of recognizing objects viewpoint-invariant which is not supported by cognitive science.

An alternative to shape-based methods are appearance-based methods which create representations based on images. In other words, the intention is to avoid creating a 3-D model but therefore use the image itself for making a description. At the same time this means that each distortion appearing on the image, e.g. reflectance, is also stored in the representation. Nevertheless the appearance-based methods have turned out to be expedient for recognition of 3-D objects because successful object recognition systems must cope with these effects. In addition, when talking about the

appearance-based approach we have to distinguish between local and global methods. The first one tries to represent the object as a collection of local features while the second one uses the entire image of an object for the representation. Since global methods use the whole image information they are very sensitive to clutter, background or occlusions. Local features are usually based on the description of patches around distinctive points in the image. Concerning 3-D object recognition this means that distinctive small pieces of the object are stored for the representation and therefore local features are a nice approach to minimize the amount of data to be processed without losing most important information. To summarize, appearance-based approaches using local features are gaining more and more importance.

III. REAL TIME OBJECT RECOGNITION AT TU BERLIN

The aim of this project is to develop a recognition system that is largely scale-, illumination-, translation-, and rotation-invariant. The recognition system should be capable of being used for any object, regardless of its shape or size. Also, the system should be able to recognize the objects regardless of their surrounding environment.

This research based project has three areas of focus, described below:

- **Generating 'Proto-Objects' i.e. Potential Objects**

The task of this process is to generate proto-objects that can then be processed further. It essentially involves extracting whole objects from their surrounding environment. It is important to ensure that the objects are retrieved regardless of their surrounding environment i.e. the surrounding environment should not affect the performance of this task.

This is achieved by first applying a segmentation algorithm followed by a combination of depth based clustering and merging. Experiments have determined the remarkable accuracy of this approach in generating stable, near perfect objects.

- **Feature Computation & Matching**

The task of this process is to generate suitable features of the proto-objects which were retrieved in the previous step. It requires a thorough evaluation of feature detectors and descriptors. The features so selected must ensure the invariant nature of the system. They must also ensure that the system is capable of performing recognition in real time. Once the features have been extracted, it is required that they be matched with the previously built Object Model in order to actually determine whether the object under consideration is the same as the object model or not. It is required that this search be carried out as efficiently as possible.

A number of feature descriptors such as MSER, SURF, SIFT, Color Histogram, Opponent Color Space, Transformed Color Histogram were evaluated for this task. Experiments have determined the superiority of the SIFT Feature Descriptor, especially when it comes to 'textured' objects. Experiments on the aforementioned Color Descriptors have proven the validity of using a top-down approach i.e. using color to eliminate proto objects before applying SIFT to the remaining objects. As for

feature matching, we have implemented the randomized kd-trees approach available in FLANN. Experiments have proven the accuracy and efficiency of this approach.

We are currently exploring the use of efficient region descriptors to work with MSER, especially in case of objects that lack texture, evaluating various color descriptors, exploring various preprocessing steps to improve efficiency etc.

- **Building an Object Model**

A key task of any recognition system requires building an efficient representation of the object. Care needs to be taken to ensure that a significant number of features of the training object are extracted in such a way so as to facilitate the invariant nature of the system. It is also important to ensure the efficiency of these features in order for the system to be capable of performing in real time.

This is achieved firstly, by extracting features of the training object from multiple views and secondly, by creating a feature trajectory that ensures the selection of the best features. Experiments have proven the efficiency of this approach as the system was able to recognize objects despite changes in the orientation of the object and the viewing angle.

RESEARCH & EXPLORATION

I. INTRODUCTION

A. Purpose

The purpose of this document is to summarize the research work done at each phase of the development of the software.

B. *Scope*

The scope of this SRS document encompasses the following: -

1. It serves as a reference to study, analyze and understand methodologies to be designed, programmed and documented.
2. It acts as a tool for the development of test cases which would exercise small conditions of the program.

C. Audience

Designers, Programmers, Testers, Researchers and any fellow interested in studying computer vision, especially Object Recognition.

D. Definitions, Acronyms and Abbreviations

See Appendix

E. References & Formulae

See Appendix

II. IMAGE SEGMENTATION

A. Introduction

In computer vision, segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as superpixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images.

More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics. The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image (see edge detection). Each of the pixels in a region are similar with respect to some characteristic or computed property, such as color, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristic(s).

B. Characteristics of a Good Segmentation Algorithm

- *Capture perceptually important groupings or regions, which often reflect global aspects of the image.* While there are many approaches to image

segmentation that are highly efficient, these methods generally fail to capture perceptually important non-local properties of an image

- *Be highly efficient, running in time nearly linear in the number of image pixels.* In order to be of practical use, we believe that segmentation methods should run at speeds similar to edge detection or other low-level visual processing techniques, meaning nearly linear time and with low constant factors. For example, a segmentation technique that runs at several frames per second can be used in video processing applications.

C. Graph Based Segmentation

We take a graph-based approach to segmentation. Let G = (V;E) be an undirected graph with vertices v_i belonging to V (the set of elements to be segmented) and edges $(v_i; v_j)$ belonging to E (corresponding to pairs of neighbouring vertices). Each edge $(v_i; v_j)$ belonging to E has a corresponding weight $w((v_i; v_j))$, which is a non-negative measure of the dissimilarity between neighbouring elements v_i and v_j . In the case of image segmentation, the elements in V are pixels and the weight of an edge is some measure of the dissimilarity between the two pixels connected by that edge (e.g., the difference in intensity, color, motion, location or some other local attribute).

In the graph-based approach, a segmentation S is a partition of V into components such that each component (or region) C belonging to S corresponds to a connected component in a graph $G_0 = (V;E_0)$, where E_0 is a subset of E. In other words, any segmentation is induced by a subset of the edges in E. There are different ways to measure the quality of segmentation but in general we want the elements in a component to be similar, and elements in different components to be dissimilar. This means that edges between two vertices in the same component should have relatively low weights, and edges between vertices in different components should have higher weights.

D. Efficient Graph Based Image Segmentation [1]

A predicate D is defined for evaluating whether or not there is evidence for a boundary between two components in a segmentation (two regions of an image).This predicate is based on measuring the dissimilarity between elements along the boundary of the two components relative to a measure of the dissimilarity among neighboring elements within each of the two components. The resulting predicate compares the inter-component differences to the within component differences and is thereby adaptive with respect to the local characteristics of the data.

Int(C) is defined as the internal difference of a component $C \sqsubseteq V$ to be the largest weight in the minimum spanning tree of the component.

Dif (C1,C2) is defined as the difference between two components $C1\,, C2\ \subseteq V$ to be the minimum weight edge connecting the two components.

The region comparison predicate evaluates if there is evidence for a boundary between a pair or components by checking if the difference between the components, Dif (C_1,C_2), is large relative to the internal difference within at least one of the components, Int(C_1) and Int(C_2). A threshold function is used to control the degree to which the difference between components must be larger than minimum internal difference.
We use the following threshold function based on the size of the component :

$$\Gamma(C) = \frac{k}{|C|}$$

where |C| denotes the size of C, and k is some constant parameter.

We define the pairwise comparison predicate as,

$$D(C1\,, C2) = \begin{cases} true\ if\ Dif\ (C1, C2) > MInt(C1, C2) \\ false\ otherwise \end{cases}$$

where the minimum internal difference, MInt, is defined as,

$$MInt\ (C1, C2) = \min((Int(C1) + \Gamma(C1), Int(C2) + \Gamma(C2))$$

E. Parameters

- **Gaussian Smoothing Function σ**

A Gaussian filter is used to smooth the image slightly before computing the edge weights, in order to compensate for digitization artifacts. We always use a Gaussian with σ = 0.8, which does not produce any visible change to the image but helps remove artifacts.

- **Threshold Parameter k**

There is one runtime parameter for the algorithm, which is the value of k that is used to compute the threshold function Γ. It effectively sets a scale of observation, in that a larger k causes a preference for larger components.

F. Example

Figure 2 : Original Image

Figure 3 : Segmented Image

Image Size : 320 x 240
σ : 0.8
k : 300

G. Summary

The segmentation algorithm makes simple greedy decisions, and yet produces segmentations that obey the global properties of being not too coarse and not too fine according to a particular region comparison function. The method runs in $O(m * logm)$ time for m graph edges and is also fast in practice, generally running in a fraction of a second.

III. FEATURE EXTRACTORS AND DESCRIPTORS

In pattern recognition and in image processing, feature extraction is a special form of dimensionality reduction. When the input data to an algorithm is too large to be processed and it is suspected to be notoriously redundant (e.g. the same measurement in both feet and meters) then the input data will be transformed into a reduced representation set of features (also named features vector). Transforming the input data into the set of features is called *feature extraction*. If the features extracted are carefully chosen it is expected that the features set will extract the relevant information from the input data in order to perform the desired task using this reduced representation instead of the full size input.

In Image Processing, Feature Extraction can be thought of as a two step process :

1. **Interest Point Detector**. Reliable point-features will be detected.
2. **Local Image Descriptor**. Each point feature will be described by its local region from which a feature vector is computed.

For any object in an image, interesting points on the object can be extracted to provide a "feature description" of the object. This description, extracted from a training image,

can then be used to identify the object when attempting to locate the object in a test image containing many other objects. To perform reliable recognition, it is important that the features extracted from the training image be detectable even under changes in image scale, noise and illumination. Such points usually lie on high-contrast regions of the image, such as object edges.

A. SIFT

Scale-invariant feature transform (or **SIFT**) [2][3] is an algorithm in computer vision to detect and describe local features in images. The algorithm was published by David Lowe in 1999. SIFT can be split up into two parts - a difference-of-Gaussian point detector and a local image descriptor providing the SIFT features.
The following steps describe SIFT in detail [Web][1]

1. Constructing a scale space

In the first step of SIFT, you generate several octaves of the original image. Each octave's image size is half the previous one. Within an octave, images are progressively blurred using the Gaussian Blur operator.

2. LoG Approximation

This refers to the computation of second order derivatives (or the "laplacian") of the blurred images generated in the previous step. This is vital for locating edges and corners on the image. These edges and corners are good for finding keypoints. Blurring is important as it smoothes out the noise and stabilizes the second order derivative.
Since Laplacian of Gaussian is computationally expensive, SIFT computes the difference between consecutive scales or, the Difference of Gaussians which is approximately the same as the Laplacian of Gaussian. This effectively reduces the computation time and also leads to scale invariance.

3. Finding Keypoints

Finding key points is a two part process

- *Locate maxima/minima in DoG images*

SIFT iterates through each pixel (X) in a image and checks all its neighbors. The check is done within the current image, and also the one above and below it in the scale space. This way, a total of 26 checks are made. X is marked as a "key point" if it is the greatest or least of all 26 neighbors.

- *Find subpixel maxima/minima*

The marked points are the approximate maxima and minima. They are “approximate” because the maxima/minima almost never lies exactly on a pixel. It lies somewhere between the pixel. But we simply cannot access data “between” pixels. So, we must mathematically locate the subpixel location. Subpixel values are generated by first using Taylor expansion of the image around the approximate key point and then finding the extreme points of the resulting equation.

4. Eliminating Bad Keypoint

Key points generated in the previous step produce a lot of key points. Some of them lie along an edge, or they don’t have enough contrast. In both cases, they are not useful as features. So SIFT gets rid of them.

5. Assigning Orientation to Keypoints

In order to achieve rotation invariance, SIFT assigns an orientation to each keypoint. To do so, it first collects gradient directions and magnitudes around each keypoint. It then generates a histogram for this data. Using the histogram, the most prominent gradient orientation(s) are identified. If there is only one peak, it is assigned to the keypoint. If there are multiple peaks above the 80% mark, they are all converted into a new keypoint (with their respective orientations).

6. Generating SIFT Features

SIFT takes a 16×16 window of pixels around the keypoint. It then splits that window into sixteen 4×4 windows. From each 4×4 window, it generates a histogram of 8 bins where each bin corresponds to 0-44 degrees, 45-89 degrees, etc. Gradient orientations from the 4×4 are put into these bins. This is done for all 4×4 blocks. Finally, it normalizes the 128 values so generated. Hence, a unique feature vector consisting of 128 values is generated for each keypoint.

Figure 4 SIFT Matching : Exhibiting Scale & Orientation Invariance

B. MSER

Maximally Stable Extremal Regions (**MSER**) was proposed by Matas et al. [4] to find correspondences between image elements from two images with different viewpoints. Extremal regions have two desirable properties. Firstly, the set is closed under continuous (and thus perspective) transformation of image coordinates and, secondly, it is closed under monotonic transformation of image intensities.

Informally, MSER can be explained as follows :

Imagine all possible thresholdings of a gray-level image I. We will refer to the pixels below a threshold as 'black' and to those above or equal as 'white'. If we were shown a movie of thresholded images, with frame t corresponding to threshold t, we would see first a white image. Subsequently black spots corresponding to local intensity minima will appear and grow. At some point regions corresponding to two local minima will merge. Finally, the last image will be black. The set of all connected components of all frames of the movie is the set of all maximal regions.
Such regions are of interest since they posses the following properties:

• Invariance to affine transformation of image intensities.
• Covariance to adjacency preserving (continuous) transformation $T : D \rightarrow D$ on the image domain.
• Stability, since only extremal regions whose support is virtually unchanged over a range of thresholds is selected.
• Multi-scale detection. Since no smoothing is involved, both very fine and very large structure is detected.
• The set of all extremal regions can be enumerated in $O(n \log \log n)$, where n is the number of pixels in the image.

Comparison to other Region Descriptors

In Mikolajczyk et al. [6], six region detectors are studied (Harris-affine, Hessian-affine, MSER, edge-based regions, intensity extrema, and salient regions). A summary of MSER performance in comparison to the other five follows.

- **Region density** - in comparison to the others MSER offers the most variety detecting about 2600 regions for a textured blur scene and 230 for a light changed scene, and variety is generally considered to be good. Also MSER had a repeatability of 92% for this test.
- **Region size** - MSER tended to detect many small regions, versus large regions which are more likely to be occluded or to not cover a planar part of the scene. Though large regions may be slightly easier to match.
- **Viewpoint change** - MSER outperforms the five other region detectors in both the original images and those with repeated texture motifs.
- **Scale change** - Following Hessian-affine detector, MSER comes in second under a scale change and in-plane rotation.

- **Blur** - MSER proved to be the most sensitive to this type of change in image, which is the only area that this type of detection is lacking in.
  Note however that this evaluation did not make use of multi-resolution detection, which has been shown to improve repeatability under blur.
- **Light change** - MSER showed the highest repeatability score for this type of scene, with all the other having good robustness as well.

MSER consistently resulted in the highest score through many tests, proving it to be a reliable region detector.

IV. COLOR

So far, intensity-based descriptors have been widely used for feature extraction at salient points. To increase illumination invariance and discriminative power, color descriptors have been proposed.

A. RGB Histogram

The RGB histogram is a combination of three 1-D histograms based on the R, G and B channels of the RGB color space. This histogram possesses no invariance properties.

B. Opponent Histogram

Opponent histogram The opponent histogram is a combination of three 1-D histograms based on the channels of the opponent color space.The channels of the opponent color space can be computed by transforming the RGB channels as shown below :

$$\begin{pmatrix} O1 \\ O2 \\ O3 \end{pmatrix} = \begin{pmatrix} \frac{R-G}{\sqrt{2}} \\ \frac{R+G-2B}{\sqrt{6}} \\ \frac{R+G+B}{\sqrt{3}} \end{pmatrix}$$

The intensity information is represented by channel O3 and the color information by O1 and O2. Due to the subtraction in O1 and O2, the offsets will cancel out if they are equal for all channels (e.g. a white light source).

O1 and O2 are shift-invariant with respect to light intensity and the intensity channel O3 has no invariance properties.

C. Transformed Color Distribution

An RGB histogram is not invariant to changes in lighting conditions. However, by normalizing the pixel value distributions, scale-invariance and shift invariance is achieved with respect to light intensity. Because each channel is normalized independently, the descriptor is also normalized against changes in light color and arbitrary offsets:

$$\begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} = \begin{pmatrix} \frac{R - \mu R}{\sigma R} \\ \frac{G - \mu G}{\sigma G} \\ \frac{B - \mu B}{\sigma B} \end{pmatrix}$$

where μC represents the mean and σC the standard deviation of the distribution in channel C computed over the area under consideration (e.g. a patch or image). This yields for every channel a distribution where μ = 0 and σ= 1.

Table 1 Invariance of Descriptors (Indicated by +)

	Light Intensity Change	***Light Intensity Shift***	***Light Intensity Change & Shift***	***Light Color Change***	***Light Color Change & Shift***
RGB Histogram	-	-	-	-	-
O1 ,O2	-	+	-	-	-
O3, Intensity	-	-	-	-	-
Transformed Color	+	+	+	+	+

V. OBJECT MODEL GENERATION

A desire of each object recognition system is that it is capable of detecting a trained object in any scene and thus be able to cope with different poses of the object or varying lighting conditions or possible occlusions. Clearly, it is impossible to keep a database that has examples of each view of an object under each possible pose and lighting condition. Therefore the challenge is to extract distinctive features in order to get a robust object representation which should overcome these requirements. Consequently object representations are of great importance.

A. Single View Based Method

Most object recognition systems determine the identity of an object on the basis of the information gathered from a single image. Typically, a set of features is extracted from an image and compared against the features of the training image. Using local features means applying an interest point detector on the training image and to extract for all detected keypoints a feature vector describing the local region around the

detected point. Finally these feature vectors fi will form the representation M of the object in the training image.
M = [f_1, f_2, f_3, ..., f_n]

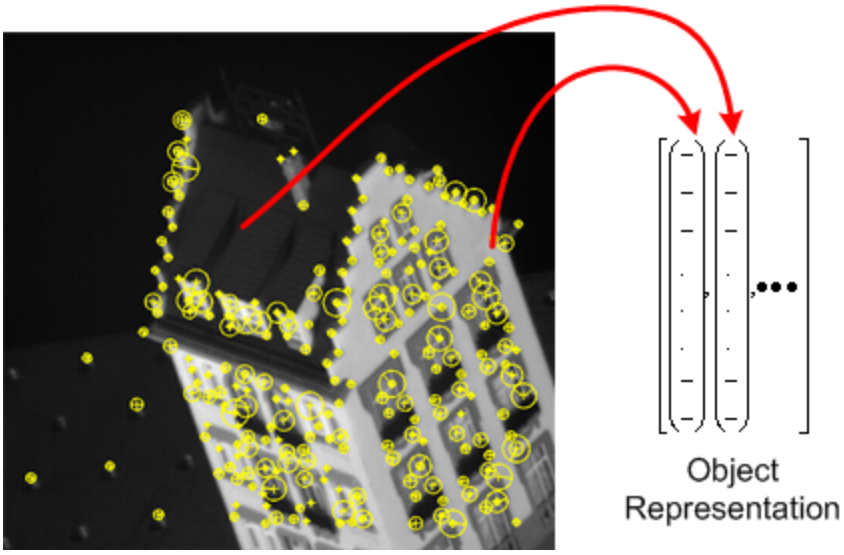

Figure 5 The Single View method uses features extracted from a single image of the desired object for forming object representations

B. Multiple View Based Method

When thinking of 3D objects you can easily imagine that it is not sufficient to learn from a single image in order to recognize it in all possible poses. Furthermore a common problem is the sensitivity of local image descriptors against viewpoint changes. As a result multiple images from different views of the same object will overcome these problems. However, it will also result in a representation where the number of features will increase proportional to the number of gathered views. For each training image you will apply the Single View method and all extracted feature vectors will be combined to a representation M of the object as shown in figure 3.2. Formally,
M = [f11, f12, ..., f1n1, f21, f22, ..., f2n2, ..., fm1, f12, ..., fmnm],
where fij represents the j-th feature vector of the i-th image.

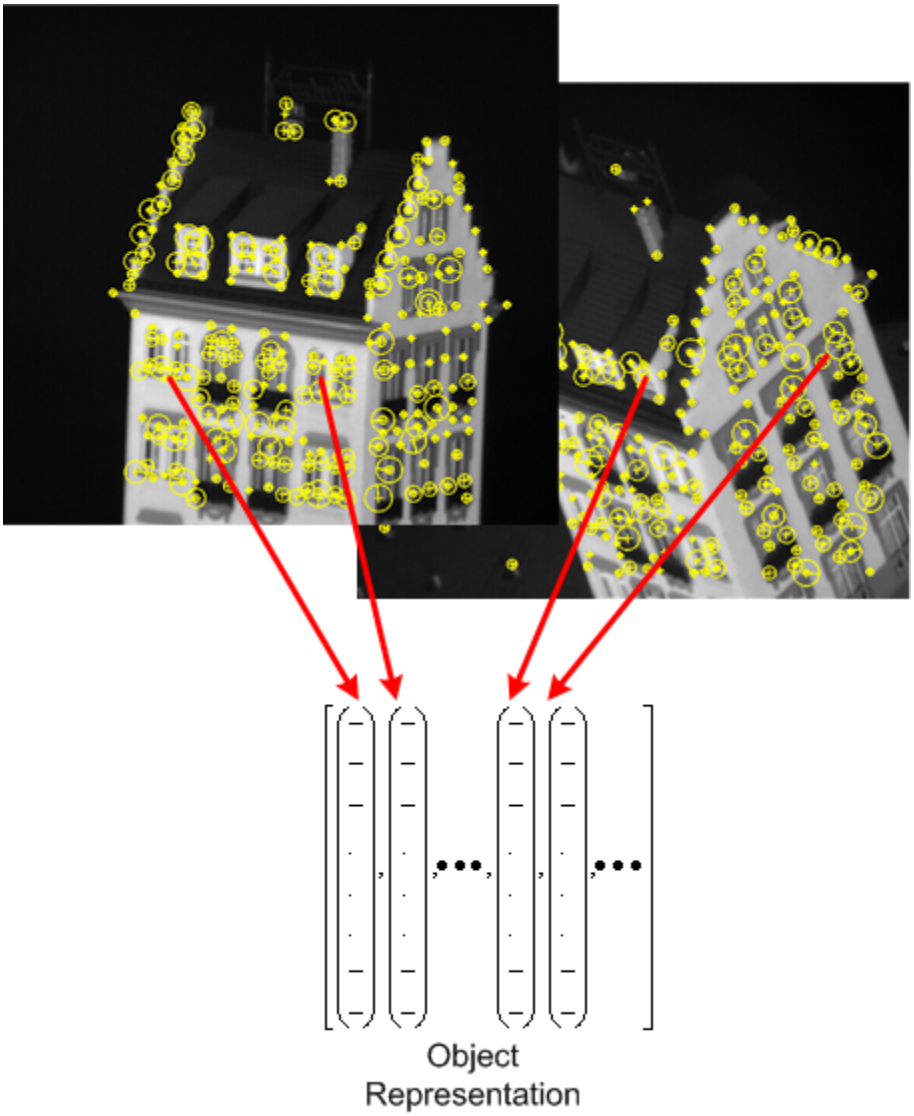

Figure 6 Multiple View method uses features extracted from multiple images depicting multiple views of the desired object for forming object descriptions

C. Continuous View Based Method

In computer vision we have to work with discrete signals and therefore we even have a discrete image sequence that consists of a finite number of frames. We can find correspondence between successive image frames and thereby form a feature trajectory in order to get a robust description of the object. So our approach for creating an object description out of an image sequence
can be split up into 3 steps :

1. Extraction of local feature trajectories. Tracking of local features will give correspondences between keypoints of successive frames which will finally form trajectories.
2. Selection of robust feature trajectories. Decide which trajectories are useful for object recognition.
3. Combining descriptors of trajectories to an object representation. A trajectory consists of many continuously changing feature vectors and

therefore contain much redundant information. So the challenge is find representatives for all trajectories which will finally form the object representation.

1. Setup for Extraction of Local Feature Trajectories

The training object is rotated in front of a camera and the local feature detector is applied at regular intervals. Features of the object are continuously tracked using a Tracking Algorithm. These Features are either added to existing trajectories or a new trajectory is created for them depending on certain conditions. This method finally provides trajectories ti and moreover the set of all trajectories T of a given image sequence. Formally this can be expressed as

$T = \{t1, t2, ..., tn\}$

with,

$ti = [f1, f2, ..., fni\ ]$

2. Algorithm for new detected keypoints:

1. For each new keypoint do:

(a) Search trajectories which haven't been allotted a keypoint from the current set of newly detected keypoints. Compute the
Euclidean distance of the last feature vector of each trajectory to the current keypoint and take the one with closest distance.

(b) If the Euclidean distance is smaller than a defined threshold, discard this keypoint from the set of newly detected keypoints and go to step 1(d). Else go to step 1(d)

(c) Add the newly detected keypoints to the trajectory and repeat for next keypoint.

(d) Create a new trajectory for this keypoint and repeat for next keypoint.

3. Selection of Robust Feature Trajectories

Discard trajectories that haven't been allotted a keypoint for k consecutive frames. This is to ensure that only features that have been tracked for a significant number of frames are used to represent the object and outliers are removed.

4. Combining descriptors of trajectories to an object representation.

As already described, a feature trajectory consists of feature vectors which have been computed at each frame of a single tracked keypoint. However using all feature vectors of each robust trajectory will lead to a very huge object representation containing much redundant information. Therefore we seek to find an appropriate method for getting good representatives for a single trajectory and finally for the whole object. Primitive approaches for summarizing trajectories would be using standard methods like the following ones:

- *Mean.*

The simplest method for summarizing data is to compute the arithmetic mean feature vector ɬ for a trajectory j = 1, 2, ...,m,

$$\not{f}j = \frac{1}{n}\sum_{i=1}^{n} fi$$

where f is the i^{th} feature vector of the current trajectory and n corresponds to the length of the current trajectory. All mean feature vectors will finally form the object representation M,

M = [f1, f2, ..., fm].

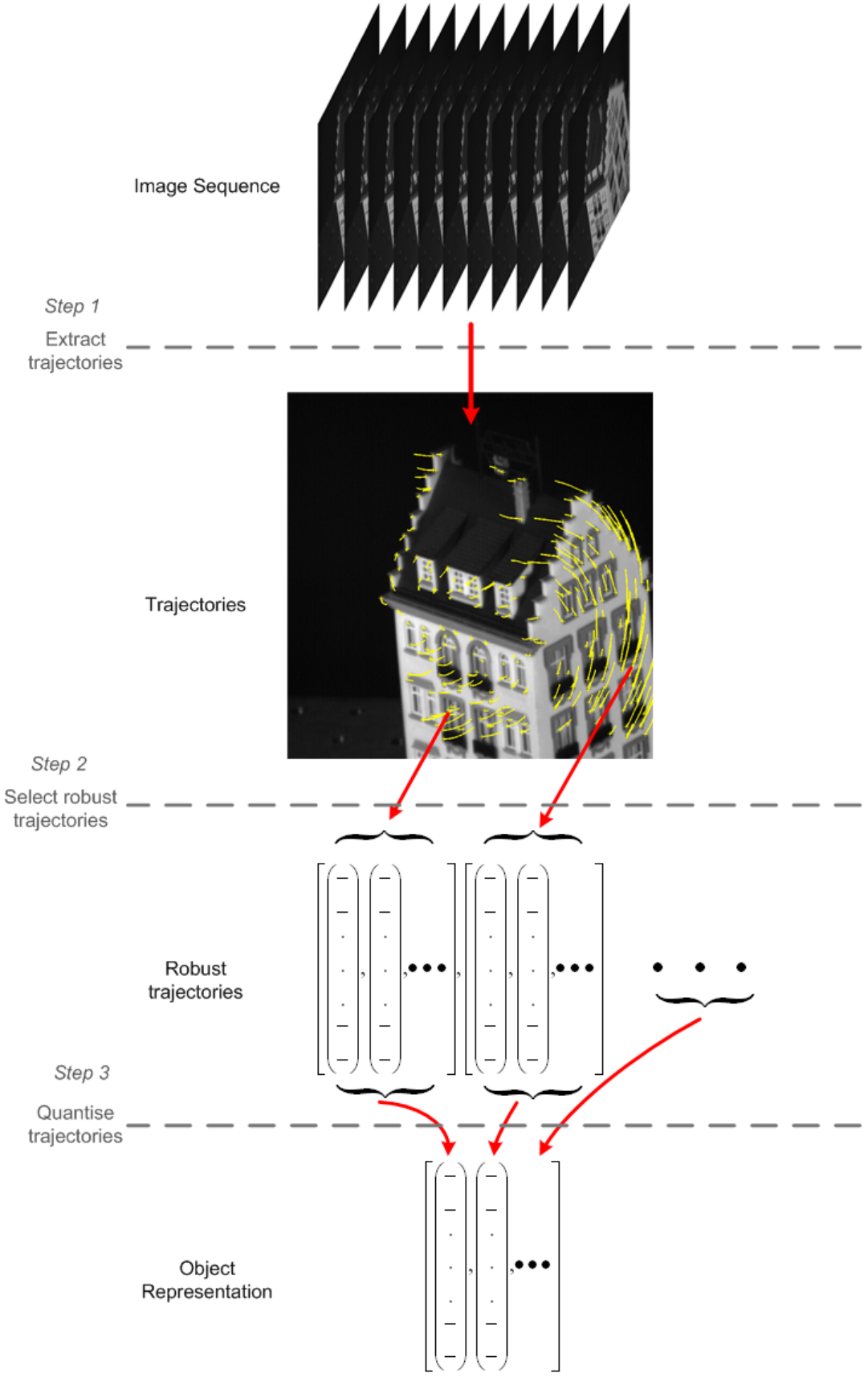

Figure 7 The Continuous View method can extract an object representation from an image sequence. Trajectories are formed which will finallybe summarized to a representation of the object

VI. SIMILARITY COMPUTATION USING FLANN

Local Feature Detectors are applied on each proto object to extract a feature vector. This feature vector is then used to find the closest nearest neighbor from the object recognition. An object is said to 'found' or recognized only if the nearest neighbor distance is less than a certain threshold. FLANN (Fast Library for Approximate Nearest Neighbor) [8] is used to find the nearest neighbor.

A. Introduction

FLANN is a library for performing fast approximate nearest neighbor searches in high dimensional spaces. It contains a collection of algorithms we found to work best for nearest neighbor search and a system for automatically choosing the best algorithm and optimum parameters depending on the dataset. FLANN introduces an algorithm which modifies the previous method of using hierarchical k-means trees. While previous methods for searching k-means trees have used a branch-and-bound approach that searches in depth-first order, FLANN uses a priority queue to expand the search in order according to the distance of each k-means domain from the query. In addition, FLANN is able to reduce the tree construction time by about an order of magnitude by limiting the number of iterations for which the k-means clustering is performed. FLANN also introduces another approach that uses multiple randomized kd-trees which is a modification of the widely used kd tree approach.

B. Problem Description

We can define the nearest neighbor search problem as follows: given a set of points P = {p1,..., pn} in a vector space X, these points must be preprocessed in such a way that given a new query point $q \in X$, finding the points in P that are nearest to q can be performed efficiently. It is assumed that X is an Euclidean vector space, which is appropriate for most problems in computer vision.

For high-dimensional spaces, there are often no known algorithms for nearest neighbor search that are more efficient than simple linear search. As linear search is too costly for many applications, this has generated an interest in algorithms that perform approximate nearest neighbor search, in which non optimal neighbors are sometimes returned. Such approximate algorithms can be orders of magnitude faster than exact search, while still providing near optimal accuracy.

C. Randomized KD-Trees Approach

The classical kd-tree algorithm (Freidman et al.,1977) is efficient in low dimensions, but in high dimensions the performance rapidly degrades. To obtain a speedup over linear search it becomes necessary to settle for an approximate nearest-neighbor. This improves the search speed at the cost of the algorithm though not always returning the exact nearest neighbors. Silpa-Anan and Hartley (Silpa-Anan and Hartley, 2008) have recently proposed an improved version of the kd-tree algorithm in which multiple randomized kd-trees are created. The original kd-tree algorithm splits the data in half

at each level of the tree on the dimension for which the data exhibits the greatest variance. By comparison, the randomized trees are built by choosing the split dimension randomly from the first D dimensions on which data has the greatest variance. FLANN uses the fixed value D = 5 in its implementation.

When searching the trees, a single priority queue is maintained across all the randomized trees so that

search can be ordered by increasing distance to each bin boundary. The degree of approximation is determined by examining a fixed number of leaf nodes, at which point the search is terminated and the best candidates returned. The user specifies only the desired search precision, which is used during training to select the number of leaf nodes that will be examined in order to achieve this precision.

EXPERIMENTAL RESULTS

I. OFFLINE TRAINING PHASE

A. Setup

The test object is placed on a rotating table with a black background. The object is then rotated a little more than 360. The rotation video is captured using the Microsoft Kinect.

B. Creating Object Model

- Frames at regular intervals are extracted from the video thereby generating a stack of images which depict the object at varying rotations.
- The region of interest (ROI) from these images is manually selected for processing.
- Feature Trajectories are created using SIFT Features.
- The end of rotation is identified by finding the highest number of nearest neighbor matches between the SIFT descriptors in the 1st frame and the following frames. The graph in Fig 8 depicts this wherein the y-axis depicts the number of nearest neighbor matches and x-axis the frame numbers. The second peak in the graph signifies the end of rotation.

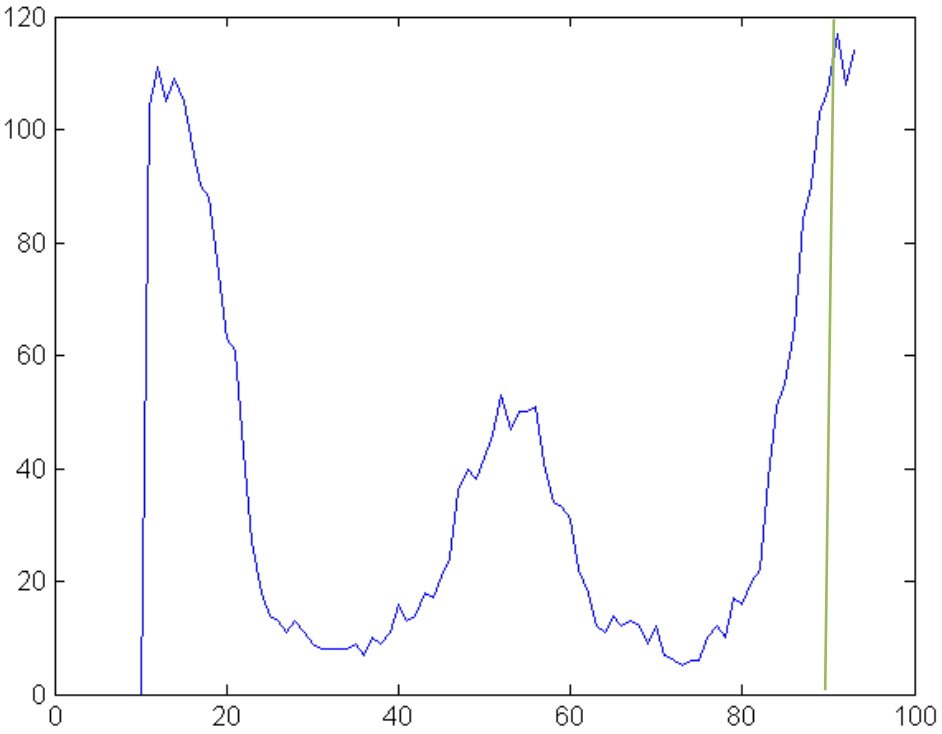

Figure 8

- The feature trajectories are then stored in a Yaml file.

II. ONLINE CLASSIFICATION PHASE

A. Input

1. Object Model (yaml file)

2. Depth Stream from Kinect (PCL)

3. Video Stream from Kinect (@ 15Hz with resolution 1240*1080

B. Output

Identified Objects labelled as found

C. Parameters

See appendix for a list of runtime parameters

D. Hardware Configuration

Memory: 11.8 GiB

Processor: Intel 12-core i7 CPU X980@ 3.33 GHz

Operating System: Ubuntue Release 11.10 (oneiric) Kernal Linuz 3.0.0.36-generic

E. Object 1

Timing:

Procedure	**Time Taken (in ms)**
Segmentation	245.28
NN Clustering	149.39
Merging	1101.85
Recognition	3363.31
SIFT computation	3280.38
FLANN Search	43.75
Total Time	**4859.83**

Table 5 Average elapsed time over 20 frames

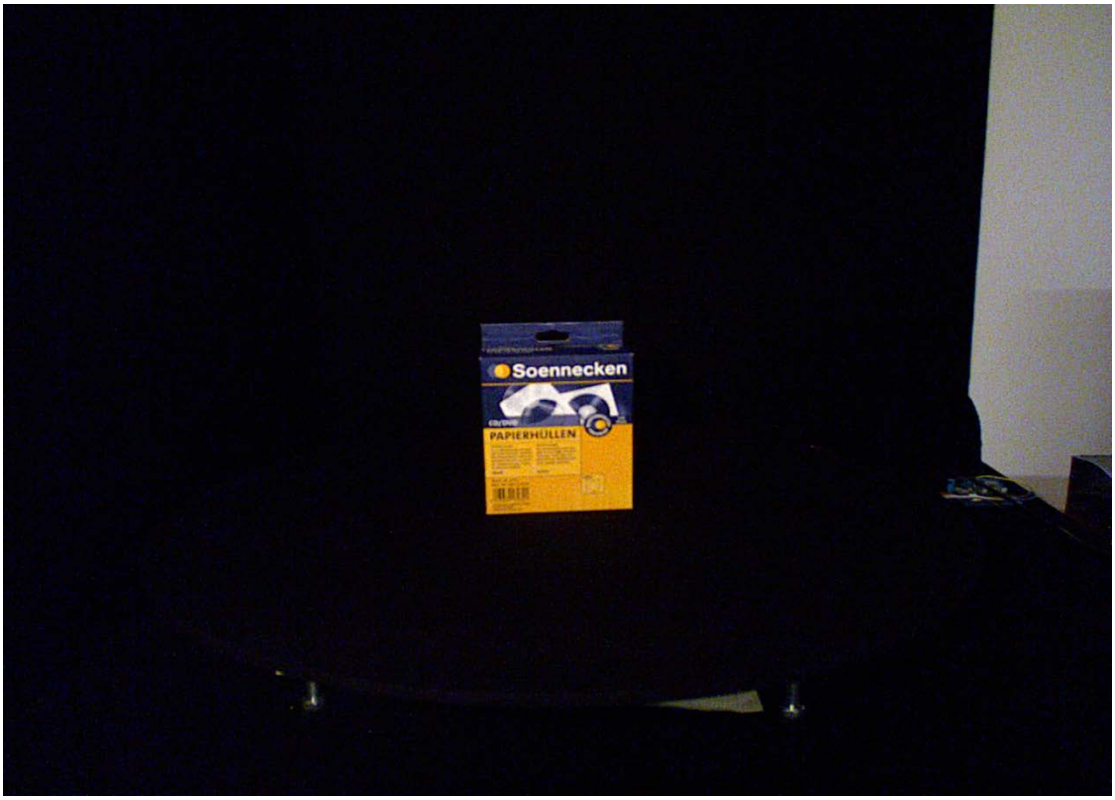

Figure 9: Object in testing environment

Figure 10: Object in Real Environment

Figure 11: Segmented Image

Figure 12: Image after Nearest Neighbor Clustering

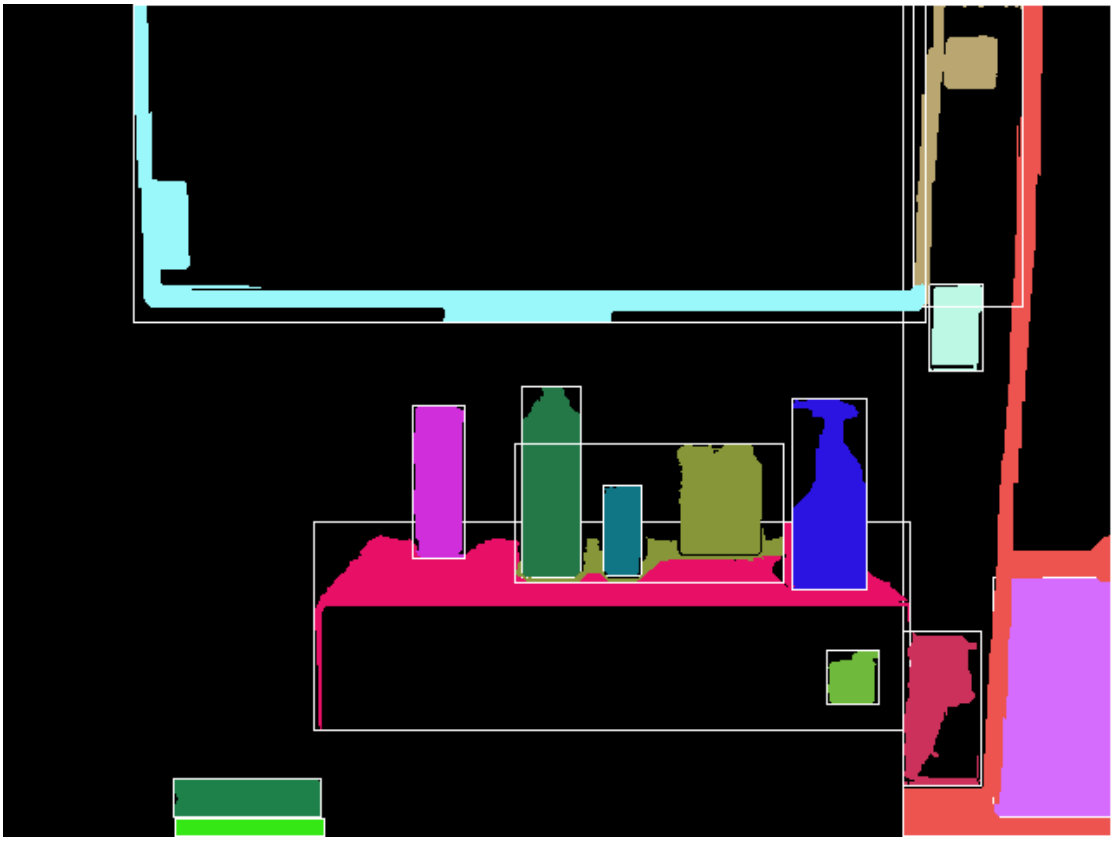

Figure 13: Image after Merging

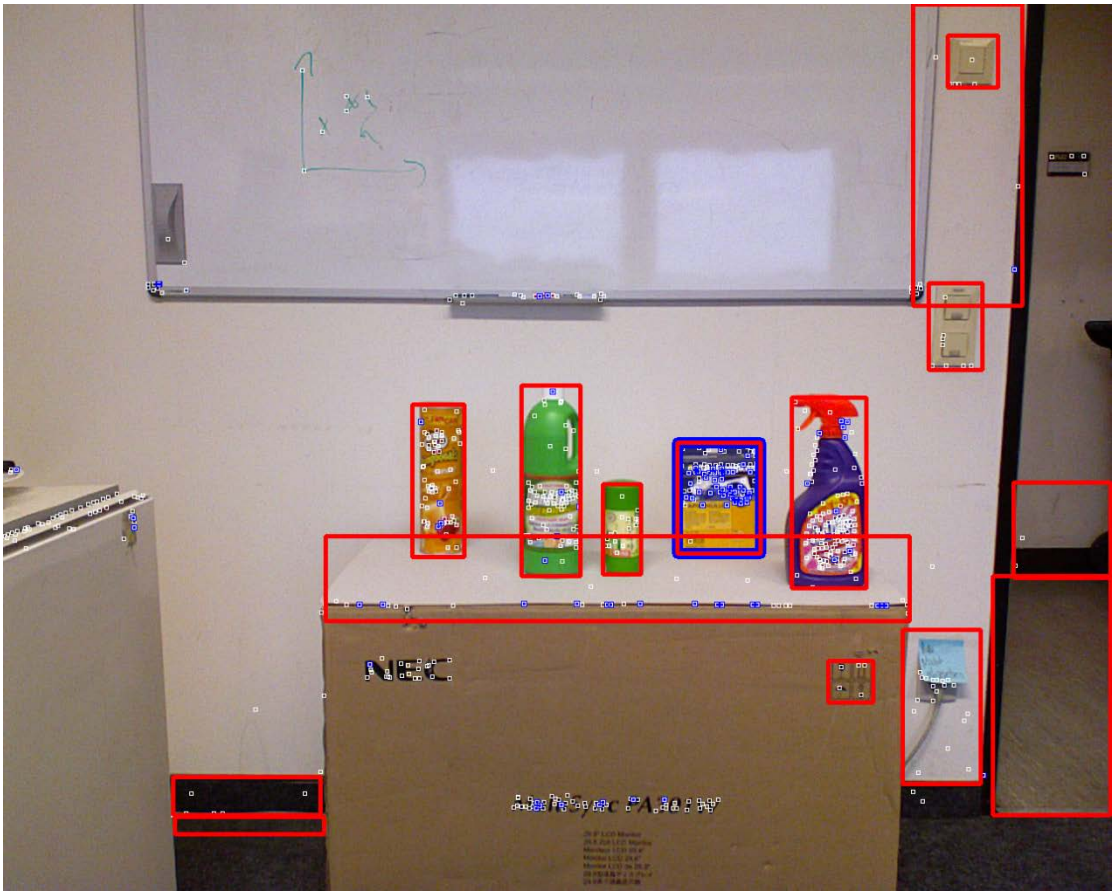

Figure 14: Recognition Results

F. Object 2

Timing:

Procedure	**Time Taken (in ms)**
Segmentation	248.14
NN Clustering	138.75
Merging	1206.47
Recognition	3185.24
SIFT computation	3107.44
FLANN Search	41.79
Total Time	**4778.6**

Table 6 Average elapsed time over 20 frames

Figure 15: Object in training environment

Figure 16: Object in Real Environment

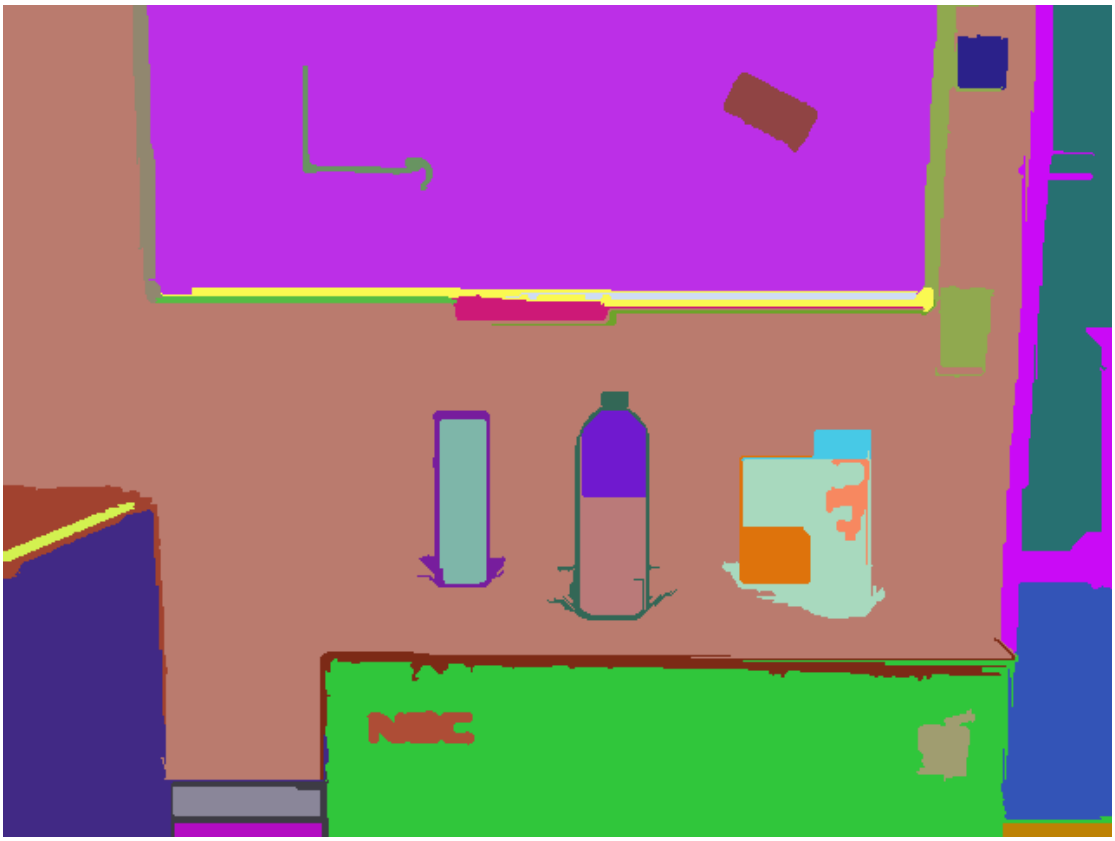

Figure 17: Segmented Image

Figure 18: Image after Nearest Neighbor Clustering

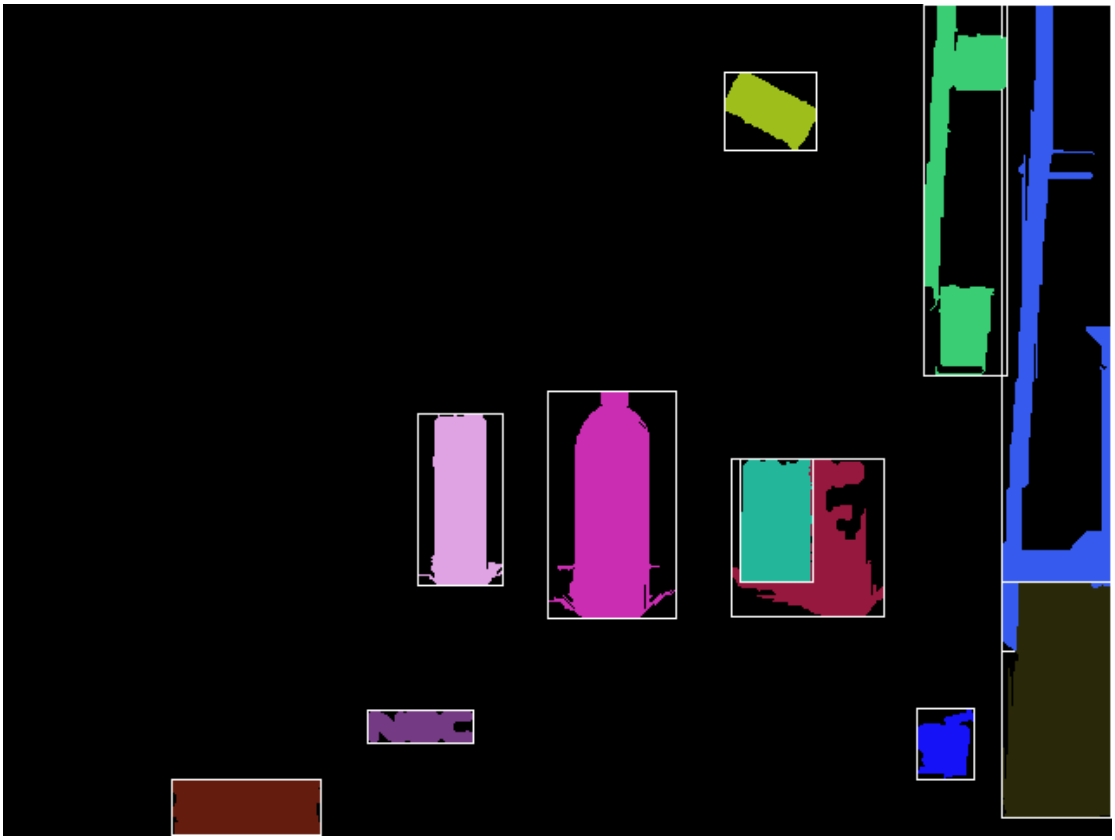

Figure 19: Image after Merging

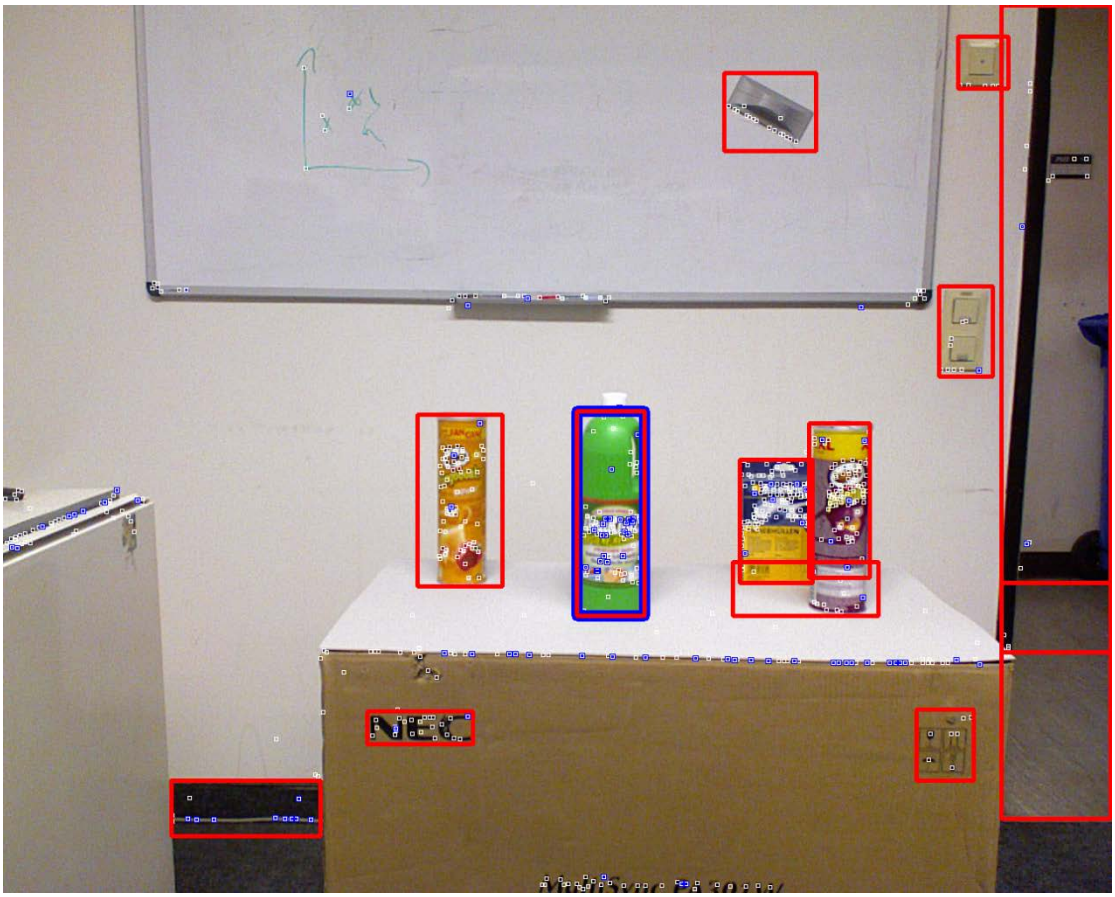

Figure 20: Recognition Results

G. Object 3

Timing:

Procedure	**Time Taken (in ms)**
Segmentation	250.98
NN Clustering	153.24
Merging	1166.00
Recognition	3448.82
SIFT computation	3363.46
FLANN Search	45.29
Total Time	**5018.22**

Table 6 Average elapsed time over 20 frames

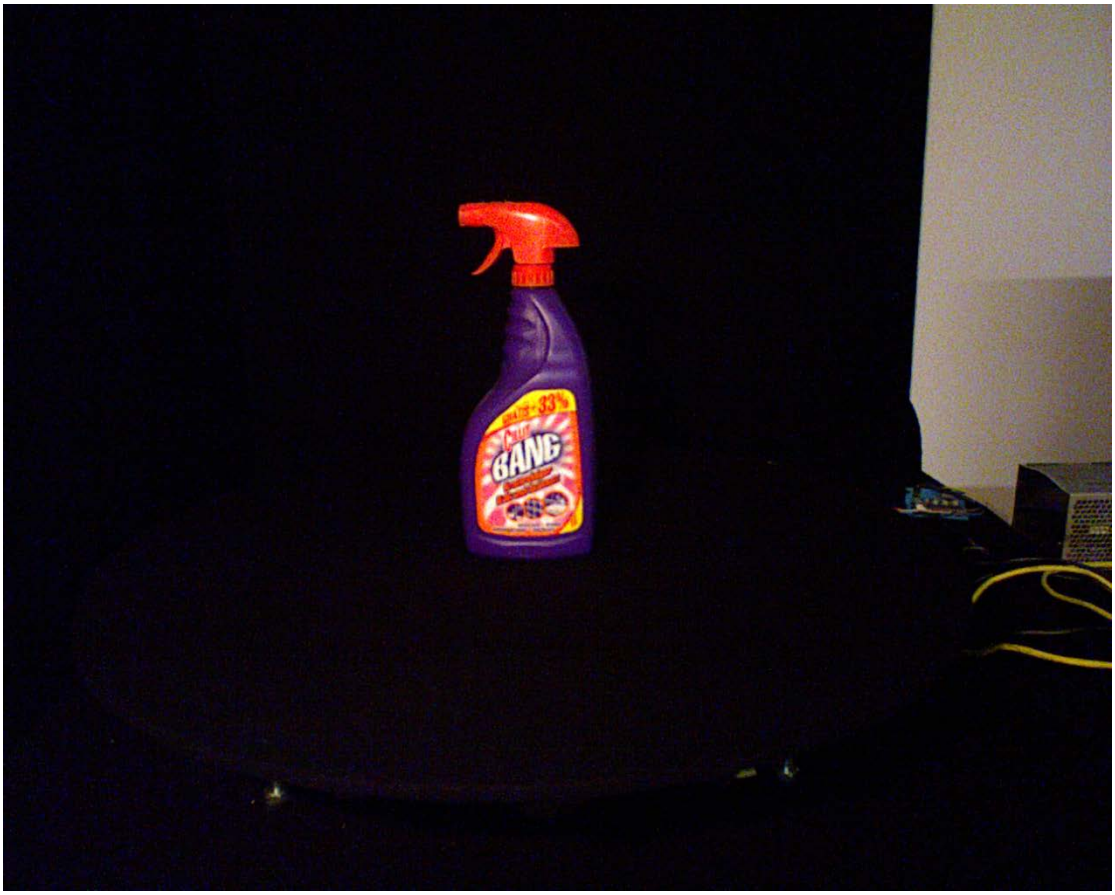

Figure 21: Object in training environment

Figure 22: Object in Real Environment

Figure 23: Segmented Image

Figure 24: Image after Nearest Neighbor Clustering

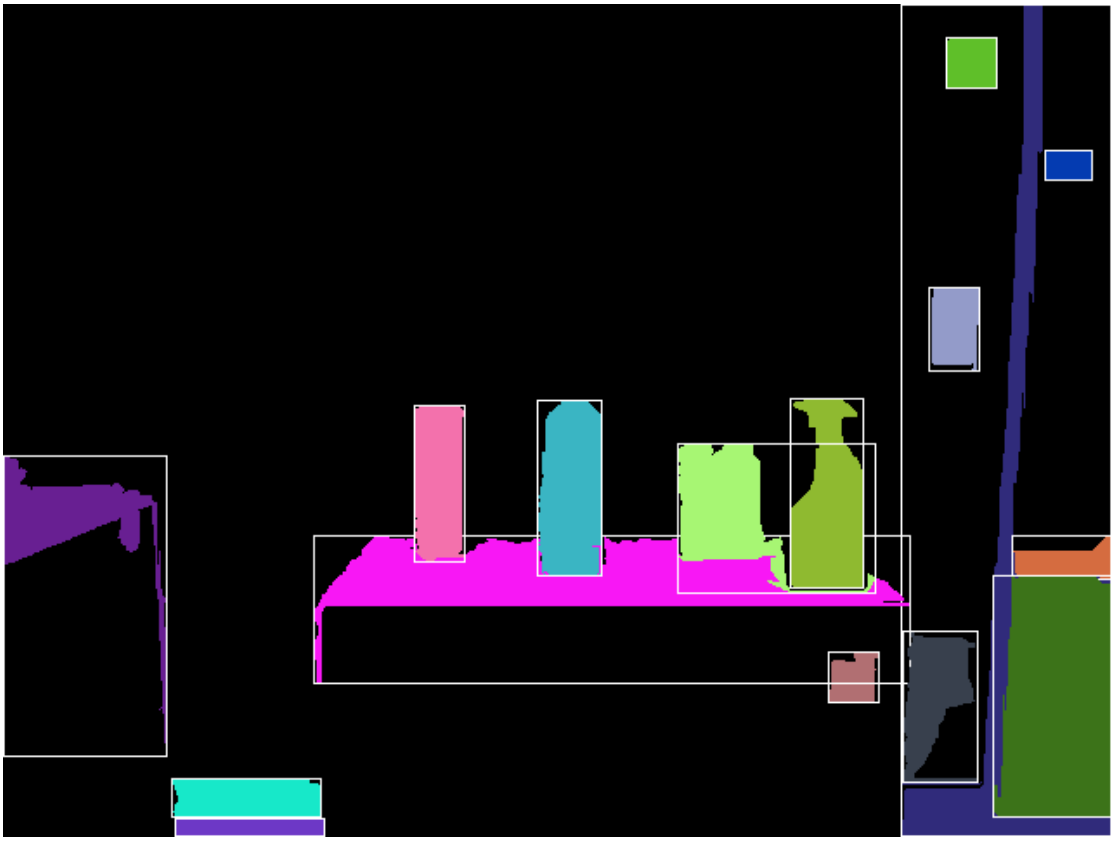

Figure 25: Image after Merging

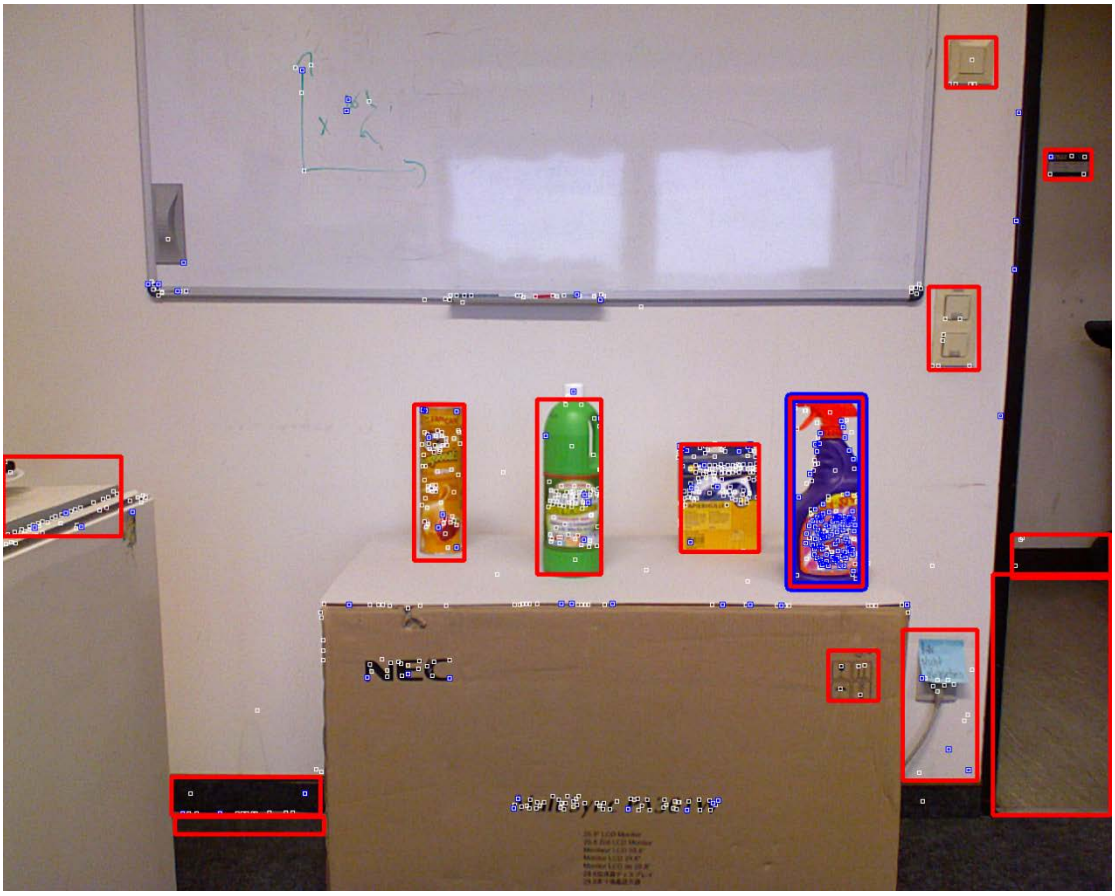

Figure 26: Recognition Results

CURRENT WORK

- **Feature Descriptors for Texture Less Objects**

  One of the key drawbacks of SIFT is its inefficiency in describing texture less objects.We plan to investigate the use of MSER for such cases.

- **Integration of Texture and Color in a top down approach**

  Color can be used to as a preprocessing step to eliminate a number of proto-objects. Only objects that satisfy the color criterion can be further processed for recognition using SIFT. The motivation behind this approach is the significant computation time associated with SIFT.

- **Extending the system to recognize multiple objects**

  As of now, the system is capable of identifying only one object as specified by the object model. We aim to extend the system to allow multiple objects to be recognized at the same time.

- **Filtering False Positives**

  We are currently working on different approaches to filter out the SIFT matches returned from the FLANN module. The idea is to use the orientation, scale and relative location parameters associated with SIFT keypoints to eliminate false positives.

- **Optimizing the Merge Step**

  Timing analysis has shown the significant computation time of the merging step. We are currently exploring other methods to reduce this inefficiency.

The preciseness of our object detection technique is better than anything we have come across in the published literature. We hope to enhance our recognition framework in the next few months and then test our system on standardized datasbases in order to present a thorough evaluation.

APPENDIX

RUNTIME PARAMETERS

S.no	Command Line Argument	Type (Default)	Description
Depth Parameters			
1.	dlim	Double (6)	Depth Limit
2.	dmode	Bool (false)	Depth Mode
3.	hisbw	Double (0.02)	Depth Histogram Bin Width
SIFT Parameters			
4.	siftsc	Int (3)	Scales
5.	siftpt	Double (0.04)	Peak Threshold
6.	siftis	Double (1.6)	Sigma
MSER Parameters			
7.	msdelta	Int(5)	Delta
8.	msminarea	Int(60)	Minimum Area
9.	msmaxarea	Int (14400)	Maximum Area
10.	Msmaxvar	Double (0.25)	Maximum Variation
11.	msmindiv	Double (0.2)	Minimum Diversity
12.	msmaxevol	Int (200)	Maximum Evolution
13.	msareathresh	Double (1.01)	Area Threshold
14.	msminmargin	Double (0.003)	Minimum Margin
15.	msblur	Int (3)	Edge Blur Size
16.	mincontarea	Int (10)	Minimum Contour Area
Graph Based Segmentation Parameters			
17.	sigma	Double (1.2)	Gaussian Smoothing function
18.	gsegk	Int (500)	k causes a preference for larger components
19.	minblob	Int (450)	
RGB Color Histogram			
20.	ncolors	Int (4)	Divides each channel into ‘ncolors’ equal sized bins. Hence, no of bins for each channel=ncolors.
Transformed Color Histogram			
21.	tbinstart	Double (-1)	Histogram range is defined by tbinstart and tbinend
22.	tbinend	Double (5)	
23.	tbinwidth	Double (0.1)	This is used to define the width of the bins and thereby the number of bins between tbinstart and tbinend
Optimization Parameters			
24.	histdist	Double (0)	Sets a threshold for distance between histograms of two segments under consideration for merging
25.	meanthresh	Double (5)	
26.	boundary-thresh	Double (0.04)	Used to remove Edges
27.	neighbour-thresh	Int (0)	Two pixels are considered neighbours as long as they are at a distance less than this threshold. Used while checking the neighbour criteria for merging.
28.	nneighbour	Int (0)	The number of nearest neighbours returned in a FLANN search.
29.	jaccardthresh	Int (0)	Sets a minimum value while computing Jaccard Index between corresponding bins of two histograms. Used to eliminate noise.
30.	segsize	Int (20000)	Sets a threshold for segments
31.	maxsizedif	Int (5)	
32.	nnthresh	Double (0)	Distance threshold for the nearest neighbors found

			using FLANN
33.	matchcnt	Int (1)	The minimum number of matches required for an object to be set as 'FOUND'
34.	matchfactor	Double (6)	Used to set a threshold for the ratio of closest distance to second closest distance in SIFT feature mode. The current object has one match if : 100*distsq1<matchfactor*matchfactor*distsq2
Program Flow Parameters			
35.	featuremode	Int (0)	0: SIFT 1:RGB Color Histogram 2:Reserved 3: Transformed Color Histogram
36.	expmode	Int (0)	0: Display All Windows 1:Display only merged segments window
37.	clusterseg	Int (0)	0: Histogram Clustering + Fine Clustering (Jong-Han's old clustering method) 1: NN Clustering
38.	libfile	String	Object Library file name (including extension)
39.	libpath	String	Object Library file path

REFERENCES

I. BOOKS

[1] Rafael C. Gonzalez, Richard E. Woods, *Digital Image Processing*, 3rd ed., Pearson Education, 2008.

[2] Walter G. Kropatsch, Horst Bischof, *Digital Image Analysis*, Springer, 2001.

II. RESEARCH PAPERS

[1] Pedro F. Felzenszwalb, Daniel P. Huttenlocher. Efficient Graph-Based Image Segmentation.

[2] Lowe, David G. (1999). Object recognition from local scale-invariant features. Proc. 7th International Conference on Computer Vision (ICCV'99) (Corfu, Greece): 1150-1157.

[3] Lowe, David G. (2004). Distinctive image features from scale-invariant key points. International Journal of Computer Vision 60(2): 91-110.

[4] J. Matas, O. Chum, M. Urban, T. Pajdla. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. Proc. of British Machine Vision Conference, pages 384-396, 2002

[5] Forssen, P-E. and Lowe, D.G. "Shape Descriptors for Maximally Stable Extremal Regions" ICCV, 2007.

[6] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, T. Kadir and L. Van Gool: "A Comparison of Affine Region Detectors"; International Journal of Computer Vision, Volume 65, Numbers 1-2 / November, 2005, pp 43-72

[7] Michael Grabner. Object Recognition with Local Feature Trajectories.

[8] Marius Muja and David G. Lowe, “Fast Nearest Neighbors with Automatic Algorithm Configuration”, in International Conference on Computer Vision Theory and Applications (VISAPP'09)