University of Bonn

Master of Science

Multi-Scale, Categorical Object Detection and Pose Estimation using Hough Forest in RGB-D Images

Author:
Ishrat Badami

Supervisor:
Prof. Dr. Sven Behnke

A thesis submitted in fulfilment of the requirements for the degree of Master of Science

in the

Autonomous Intelligent Systems
Institute for Computer Science

May 2013

Declaration of Authorship

I, Ishrat Badami, declare that this thesis titled, ' Multi-Scale, Categorical Object Detection and Pose Estimation using Hough Forest in RGB-D Images ' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- I have acknowledged all main sources of help.

Signed: _____

Date: _____

"Use what talent you possess: the woods would be very silent if no birds sang there except those that sang best."

Henry Van Dyke

Dedicated to my mother, ***Duriya Badami****, my father,* ***Kurban Badami****, and my mathematics teacher,* ***Rasul Virani****. Thank you for your unconditional support in my studies. Thank you for believing in me. Thank you to give me chance and guidance to prove and improve myself in every walk of my life. I am honoured to have you as my guardians. . .*

Acknowledgements

I thank Jörg Stückler for his enormous help and motivation. It was quite an educational experience working with him. He was very patient and guided me throughout my thesis work. It would've been really difficult without his support.

I further would like to extend my gratitude to my supervisor Professor Sven Behnke, for providing me an opportunity to work in his research group and an encouraging research environment in his Lab.

I also wish to thank Varun Raj Kompella, Manus McElhone, Aljosa Osep, and my colleagues to help me with my questions and doubts time to time.

UNIVERSITY OF BONN

Abstract

Autonomous Intelligent Systems
Institute for Computer Science

Master of Science

Multi-Scale, Categorical Object Detection and Pose Estimation using Hough Forest in RGB-D Images

by Ishrat BADAMI

Classification and localization of objects enables a robot to plan and execute tasks in unstructured environments. Much work on the detection and pose estimation of objects in the robotics context focused on object instances. We propose here a novel approach that detects object classes and finds the canonical pose of the detected objects in RGB-D images using Hough forests. In Hough forests each random decision tree maps local image patch to one of its leaves through a cascade of binary decisions over a patch appearance, where each leaf casts probabilistic Hough vote in Hough space encoded in object location, scale and orientation. We propose depth and surfel pair-feature as an additional appearance channels to introduce scale, shape and geometric information about the object. Moreover, we exploit depth at various stages of the processing pipeline to handle variable scale efficiently.

Since obtaining large amounts of annotated training data is a cumbersome process, we use training data captured on a turn-table setup. Although the training examples from this domain do not include clutter, occlusions or varying background situations. Hence, we propose a simple but effective approach to render training images from turn-table dataset which shows the same statistical distribution in image properties as natural scenes.

We evaluate our approach on publicly available RGB-D object recognition benchmark datasets and demonstrate good performance in varying background and view poses, clutter, and occlusions.

Contents

Declaration of Authorship i

Acknowledgements iv

Abstract v

1 Introduction 1
- 1.1 Related Work . . . 2
  - 1.1.1 Instance-Level Object Detection and Pose Estimation . . . 2
  - 1.1.2 Category-Level Object Detection and Pose Estimation . . . 3
- 1.2 Contribution . . . 5
- 1.3 Outline . . . 5

2 Background 7
- 2.1 Random Forest (RF) . . . 7
- 2.2 Generalized Hough Transform (GHT) . . . 8
- 2.3 Implicit Shape Model (ISM) . . . 9
- 2.4 Summary . . . 9

3 Object Recognition in RGB Images Using Hough Forest 10
- 3.1 Hough Forest Construction . . . 11
  - 3.1.1 Training Data . . . 11
    - 3.1.1.1 Patch Appearance . . . 12
  - 3.1.2 Binary Node Tests . . . 12
  - 3.1.3 Leaf Information . . . 15
- 3.2 Detection . . . 15
  - 3.2.1 Class Probability Calculation . . . 15
  - 3.2.2 Hough voting . . . 16
    - 3.2.2.1 Handling Variable Scales . . . 18
  - 3.2.3 Back-Projection and Bounding Box Construction . . . 18
  - 3.2.4 Summary . . . 19

4 Extensions **20**
4.1 Training Using RGB-D Images . . . 21
4.1.1 Training Data . . . 21
4.1.2 Tree Construction . . . 21
4.1.3 Leaf Information . . . 23
4.2 6-DoF Object Detection . . . 24
4.3 Features . . . 26
4.3.1 Depth . . . 26
4.3.2 Surfel-pair Features . . . 27
4.4 Summary . . . 28

5 Generation of Training Dataset **29**
5.1 Motivation . . . 29
5.2 Image Rendering Pipeline . . . 30
5.2.1 Segmentation and Construction of 3D Object Model . . . 33
5.2.2 Filling Depth . . . 34
5.3 Summary . . . 34

6 Evaluation and Discussion **36**
6.1 Experiment Setup . . . 36
6.2 Appearance Channels . . . 37
6.3 Tree Depth . . . 44
6.4 Sample Density . . . 47
6.5 Tree Density . . . 49
6.6 RGB-D Turn-Table Dataset . . . 51
6.7 Orientation Uncertainty Measure . . . 51
6.8 Summary . . . 51

7 Conclusion **58**

List of Figures **59**

List of Tables **62**

Abbreviations **63**

A Derivations **64**
A.1 Parzen-Window Density Estimation . . . 64
A.2 Quternion Interpolation . . . 65

B Results **67**

Bibliography **74**

Chapter 1

Introduction

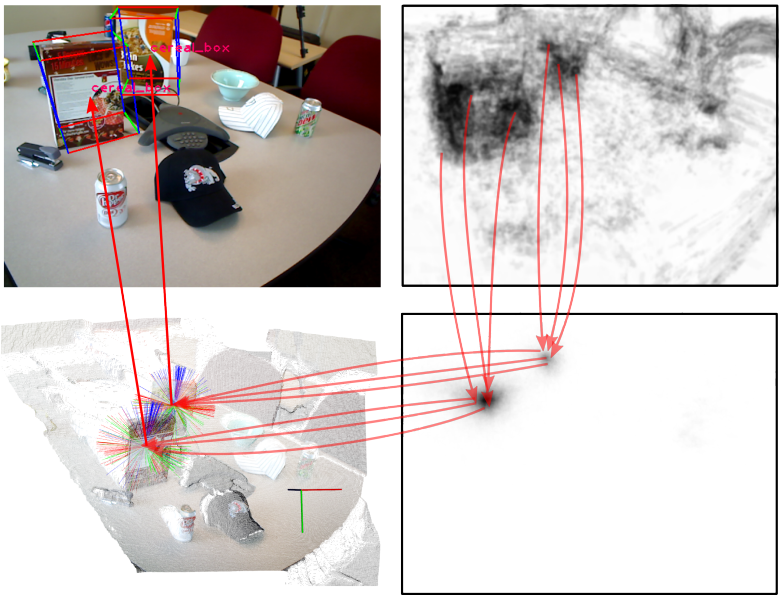


FIGURE 1.1: Object-class detection and canonical 6-DoF pose estimation in RGB-D images. We discriminatively train random decision forests to classify pixels into object classes (upper right) and to vote for the 3D position of objects (lower right). The canonical pose of the detected object class is estimated in a second stage of Hough voting for the position clusters (lower left).

With an increasing demand for automation in our lives, robots will soon be introduced to carry out our household activities. For example, cleaning, dish washing, etc.. Object detection and localization are among the core necessities for robots interacting in such unstructured environments. To do this autonomously and successfully however, poses several challenges. Much of the previous work within this context has considered the detection and pose estimation of specific object

instances. However, these shallow object instance detection pipelines do not tend to scale well for objects within classes that contain a large amount of instances. Furthermore, these methods do not generalize in detecting previously unseen instances. By grouping the instances in a class taxonomy and by further refining from coarse object categories to individual instances, efficient detection pipelines at the instance as well as category level can be obtained [1, 2]. Perceiving object classes, however, imposes challenges over object instances, since the detection must handle intra-class variation but still needs to distinguish view poses onto the objects.

1.1 Related Work

1.1.1 Instance-Level Object Detection and Pose Estimation

Scale-invariant descriptors have been extensively used for several computer-vision problems, for example view-point image matching, view-based object recognition, etc.. First introduced and developed by David Lowe [3], the SIFT (Scale-Invariant Feature Transform) descriptor is invariant to rotations, translations and scaling transformations in an image. Partially inspired from SIFT , another feature detector proposed by Herbert Bay et al. [4] called the SURF (Speeded Up Robust Features) is also used for object recognition. SURF is several times faster than SIFT and is again robust to different image transformations. However, frameworks such as MOPED [5] that are based on SIFT or SURF features require adequate texture on the objects to work well and the performance degrades if the objects are less textured. 3D object and pose estimation is particularly useful for a robot manipulating an object in space. Methods have been proposed to use various local point features that describe shape through the local constellation of points and surface normals. Johnson [6] presented an approach for 3D surface matching by using spin images and F. Tombari et al. [7] used unique signatures of histograms (SHOT) to describe the local surface orientation. Variants of point feature histograms (PFH) [8–10] have been used to describe and detect whole objects or parts within segments. Another recent approach to detect objects in 3D that makes use of Point-Pair Features (PPF) [11, 12] was proposed. It finds locally consistent

arrangements of surfel pairs between model and scene through either Hough voting [11] or RANSAC [12]. Several improvements to PPFs have been proposed that utilize visibility context [13], contours [14], or color [15]. As discussed earlier, we use a Hough forest framework to train a codebook of PPF votes discriminatively to estimate the pose of several object classes.

1.1.2 Category-Level Object Detection and Pose Estimation

In addition to the instance-level object detection, local features have also been used to recognize classes of objects, such as cups, cars, etc.. Building class specific features have been shown to be successful in detecting objects even in cluttered environments or from partially occluded images and generalise better to previously unseen object instances.

B. Leibe et al. [16] proposed a multi-view specific object recognition method called the Implicit Shape Model (ISM) combines the appearance based codebook and Hough transform. The codebook is built by clustering features with similar appearance and their spatial distribution with respect to the object center. During the recognition phase, features are matched to the codebook entries and the matched features cast probabilistic votes for the object position based on their spatial distribution with respect to the object center. This is carried out for different training images, taken from the same viewpoint, resulting in detecting novel objects. This is extended to a multi-view system by building a large set of independent single-view detectors.

P. Felzenszwalb et al. [17] presented a system that uses a mixture of multiscale deformable part models for object detection. The method uses a *latent SVM* formulation to discriminatively train the model. The method models parts star-shaped in spatial relation to a root part. The appearance of parts is encoded using Histograms of Oriented Gradients (HOG) [18]. DPMs detect mixtures of parts to further increase robustness against intra-class variability of the appearance of parts and to incorporate various view-poses of the objects.

Hough forests [19] as discussed earlier are adapted random forests that learn a dense pixel-wise codebook in a discriminative way. A forest consists of multiple random decision trees that decide on the local appearance of a pixel in binary

decision cascades. It stores the spatial distribution of relative object locations at the leaves of the trees, which are used to cast location votes during recall. The learning objective separates classes and produces Hough votes that focus well.

Estimating the pose at the category level can be broadly classified into methods along two categories: methods that estimate discrete or continuous view poses and methods that utilize only RGB images or that exploit dense depth.

Discrete vs. continuous pose estimation. Several methods for discrete view pose estimation have been proposed [2, 20, 21]. Interpolation techniques need then to be applied to obtain continuous pose estimates [22, 23]. Sun et al. [24] apply depth registration to find an accurate pose in a post-processing step. Some approaches also directly vote [25] or optimize [26, 27] for the pose of the object. These approaches extract local image features either from 3D models obtained through Structure-from-Motion [25] or from views synthesized from CAD models [26, 27], and model the 3D constellation of the features. We propose a method that is discriminatively trained to cast continuous votes for the 3D location and orientation of objects. While our method is trained from discrete views, we exploit local shape properties to transform the orientation vote of a pixel into a reproducible view-pose-invariant local coordinate frame. In this way, our method copes with continuous view pose changes.

Exploiting dense depth.

Only a few approaches in the computer vision literature make additional use of dense depth. Sun et al. [24] proposed a method called the Depth-Encoded Hough Voting (DEHV) that uses depth information and trains models using depth maps acquired with a range camera. The method jointly detects objects by depth-encoded voting and estimates their poses by registering the the inferred point cloud to a 3D model. Our approach directly votes for the pose of the object, utilizing local shape properties. Consistency of the votes is directly integrated as a discriminative training objective of the random forest. We also propose to use depth as a feature cue but scale-normalize the features using depth. In this way, the random forest is not required to capture multiple scales within its codebook.

Wang et al. [28] use depth to improve Hough forests during the training stage. In addition to 2D offset uncertainty, they also incorporate 3D offset dispersion as a split measure into the Hough forest framework. They incorporate votes from the spatial context of objects and use depth to store the relative scale of the votes with

respect to the object size. Since depth is not used during recall, the votes have to be cast across multiple scales. This approach only votes for the object location and a bounding box, while ours retrieves the full 3D position and orientation of the object.

1.2 Contribution

Recently, a state-of-the-art task-adaptive learning technique called *Hough forests* [19] was introduced, which are random forests adapted to efficiently perform a generalized *Hough transform*. The random trees model the probability distribution over class labels of the image pixels into their leaves. In addition to that during recall each leaf casts a vote in Hough space for object location and scale.

In this thesis, we propose a novel approach to object-class detection and pose estimation by extending the Hough forests to efficiently operate on RGB-D images, where D represents the depth. We use dense depth measures to normalize image features in the decision cascade of the trees for scale changes. The use of depth also allows for incorporating the right scale directly into the position and orientation votes of the pixels, making scale as an additional voting parameter obsolete. Dense estimates of the local surface orientation at each pixel allow for view-point invariant voting for the object pose. Finally, we make use of 3D information in training to render arbitrary amounts of novel training scenes that capture background variability, clutter, and occlusions in real imagery.

We test the efficacy of our approach by evaluating it on publicly available RGB-D objects and scenes datasets. We demonstrate that our approach outperforms Hough forests that operate on RGB images only and would vote for the 2D pixel location of the object. Furthermore, our method recovers the canonical pose of the objects with good accuracy.

1.3 Outline

The outline of rest of the chapters is as following. In Chapter 2, we briefly discuss related background work and in Chapter 3 we give detailed description of the hough forest implementation. In Chapter 4 we describe our method of object recognition

in RGB-D images and also explain how the estimation of pose is made. In Chapter 5 we demonstrate our novel method of rendering training dataset to increase the robustness of the system. In Chapter 6 we show our experimental results and compare it with the original implementation of Hough forest. In the end in Chapter 7 we conclude and propose some future work to improve the recognition.

Chapter 2

Background

In this chapter we briefly discuss methods for object recognition which inspired our work. In Section 2.1, object classification with random forest is summarized. In Section 2.2, we describe the generalized hough transform for instance based object detection. In the end we present details on Implicit Shape Model [29] which uses Hough voting based technique combined for object category recognition. Section 2.3.

2.1 Random Forest (RF)

Random Forests are used for various supervised or semi-supervised learning tasks such as classification, regression [30, 31], density estimation etc. A typical random forest consists a set of binary decision trees [32]. Each tree is trained in a supervised way. The root node contains all the labeled training samples. In order to construct the tree, at each non-leaf node, a set of random binary tests is generated. The test which splits the labeled training samples in an optimal way is picked and assigned to that node. Depending on the result of the test, training samples are given to the either of the children nodes (Fig. 2.1) until the leaf node is reached.

When the maximum depth limit of a tree is reached or samples per node are less than the threshold then a leaf node is generated. Each leaf node contains information about the training samples reached to it, e.g. in case of classification task, each leaf node contains information of the class distribution. At runtime,

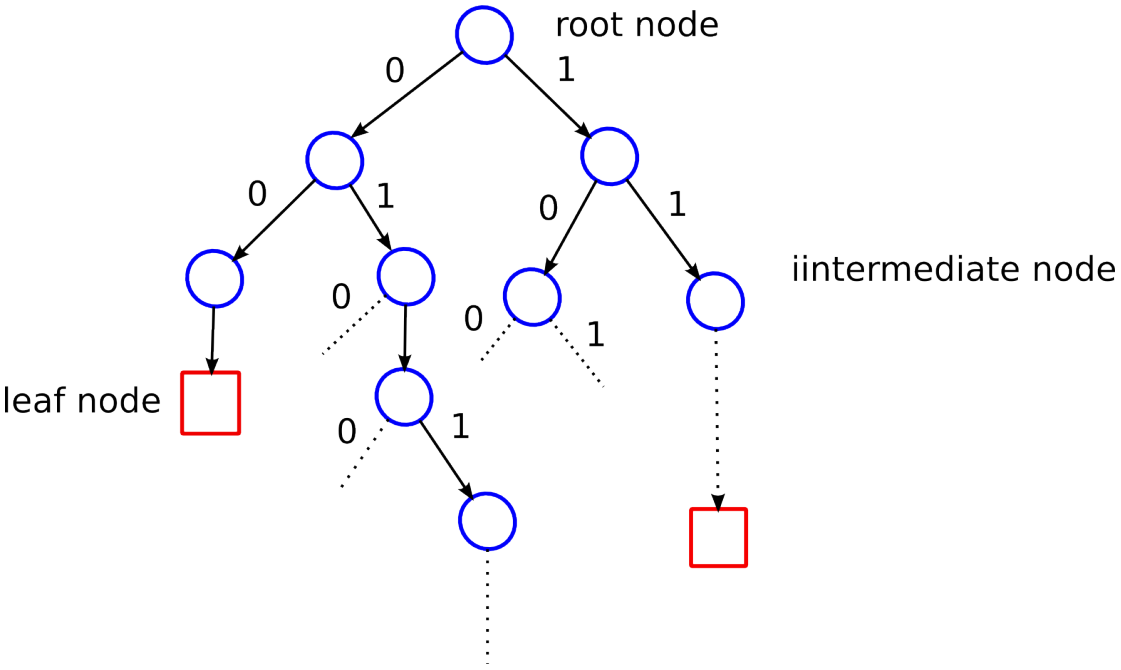


FIGURE 2.1: Figure shows a typical random decision tree. At each non-leaf node random binary test is chosen applicable to all the data reached it, to split it in an optimal way.

test sample generated traverses through all the trees in the forest and output is computed by averaging the distributions recorded at the reached leaf nodes.

2.2 Generalized Hough Transform (GHT)

The Generalized Hough Transform (GHT) [33], introduced by Dana H. Ballard, is the alteration of the Hough Transform using the principle of template matching. It adapts Hough Transform to be used for not only the detection of an object described with an analytic equation (e.g. line, circle, etc.) but also an arbitrary object described with its model. Basically GHT converts the problem of finding the model's position to a problem of finding the transformation's parameter that maps the model into the image. In case of object detection in image these parameters can be defined in terms of spatial information such as relative position of object center from local object regions.

The main drawbacks of the GHT are its high computational complexity and storage requirements that become significant when object orientation and scale have to be considered.

2.3 Implicit Shape Model (ISM)

GHT based object detection method of Liebe et.al [29] learns the Implicit shape model of the object. This model can be seen as a codebook of interest point descriptors. Once the codebook is created each entry is assigned a set of offset vectors relative to the object center. During detection interest point descriptors are matched with the codebook. Each matched entry then vote for the location of the object center in Hough space parameterized in position and scale. Each maxima in Hough space gives a hypothesis of object location in the image. All the votes that contributed to the maxima in Hough space are back-projected in the image. After refining the hypothesis a segmented mask for the object is obtained. See figure 2.2.

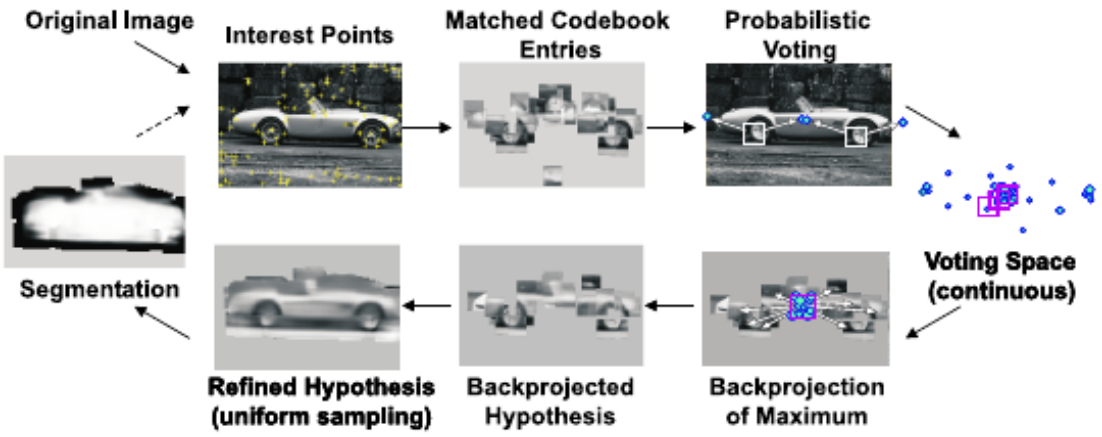


FIGURE 2.2: In implicit shape model, during training, image patches are extracted around each interest point descriptor and matched with class specific appearance codebook. Each matched entry casts a probabilistic vote in the Hough space which leads to the hypothesis of object location. Votes that contributed to the hypothesis are back projected and a segmented mask is obtained [29].

2.4 Summary

We explained random forest classifier and Generalized Hough transform in the context of object recognition. We discussed Implicit shape model which combines the two ideas of appearance codebook and and GHT. In the following chapter we will describe the Hough forest [19] framework which combines RF and Hough transform in a natural way.

Chapter 3

Object Recognition in RGB Images Using Hough Forest

This chapter discusses a class specific object recognition method using hough forest proposed by Gall and Lempitsky [19]. For convenience, we refer to the method as *Gall's* method for the rest of the chapter. Hough forests are ensembles of random decision trees. Each tree maps image patches of training images to one of its leaves through a cascade of binary decisions over local appearance. These leaves can be seen as a discriminative appearance codebook of visual words. Each leaf stores the distribution of class labels that reached it. Additionally, the leaves carry spatial information about the object, for example, the relative location of the object center. During recall, this information is used to classify test pixels into object classes and to cast votes in a hough space parametrized in object location and scale.

The rest of the chapter explains in detail two main phases of this method: *Training* (Section 3.1) and *Detection* (Section 3.2). We describe the acquisition of training data in Section 3.1.1 and show how tree construction is carried out in a supervised way (Section 3.1.2). Further, we demonstrate what information is stored in leaf nodes in Section 3.1.3. In the detection phase we provide detailed information about class probability calculation (Section 3.2.1), hough voting based object detection (Section 3.2.2) and bounding box estimation following back-projection in Section 3.2.3

3.1 Hough Forest Construction

3.1.1 Training Data

In a typical Hough forest [19], each tree $\mathcal{T}$ is constructed based on a set of local image patches. At root node, set of patches $\mathcal{S}_0$ can be written as:

$$\mathcal{S}_0 : \{(\mathcal{I}(\mathbf{y}), c(\mathbf{y}), \mathbf{d}(\mathbf{y}))\} \tag{3.1}$$

Where $\mathbf{y}$ is a patch center, $\mathcal{I}(\mathbf{y})$ is a patch appearance, $c(\mathbf{y})$ is a class label and $\mathbf{d}(\mathbf{y})$ is a displacement vector (see Fig. 3.1).

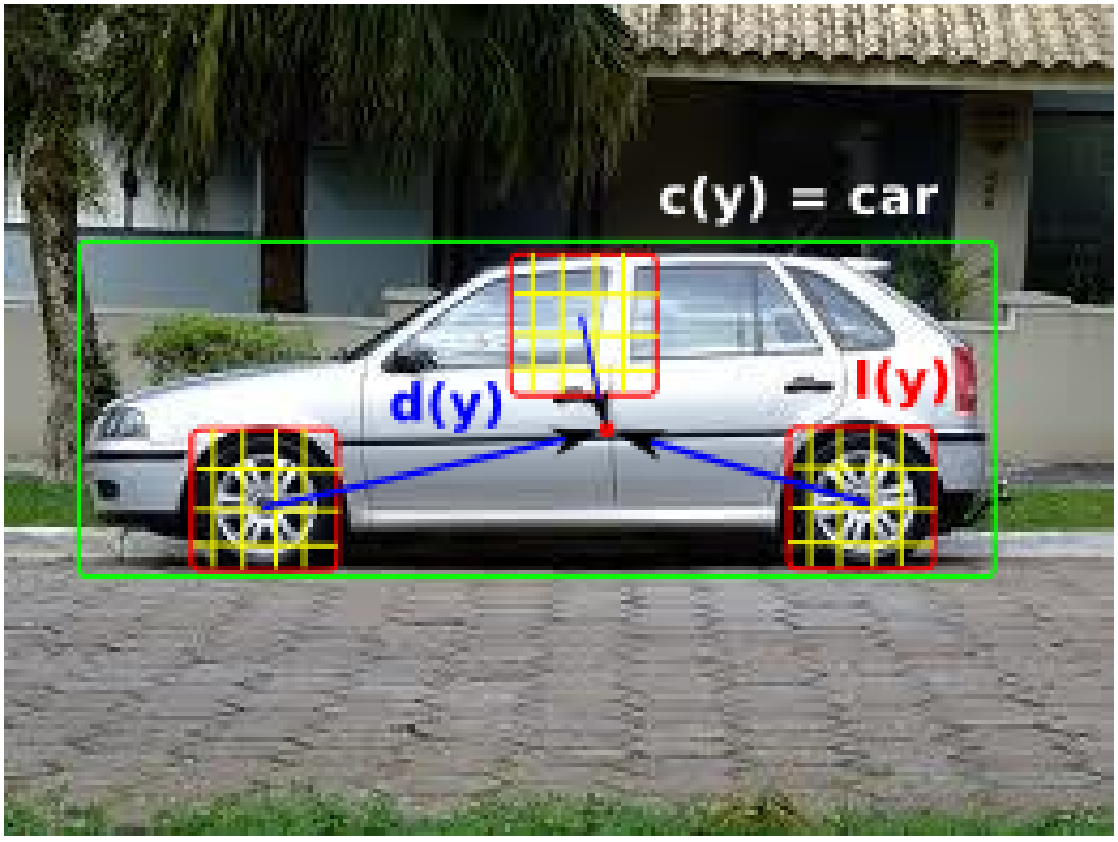


FIGURE 3.1: For an object class *car*, the bounding box is annotated in green. The fixed size patches(red) are sampled within the bounding box. The displacement vector $\mathbf{d}(\mathbf{y})$ from patch-center to the center of the car is drawn in blue.

The local image patches are sampled from positive and negative training images. Positive training images contain an instance of the object category of interest with annotated bounding boxes. A set of negative images is constituted by images in

which that category does not occur. The patches belong to object class are assigned $c(\mathbf{y}) = 1$, whereas patches sampled from background class are assigned $c(\mathbf{y}) = 0$. Each object patch is also assigned a $2D$ displacement vector $\mathbf{d}(\mathbf{y})$, corresponding to the relative location of the patch center from the object center. For background patches, $\mathbf{d}(\mathbf{y})$ is not defined. The implementation in *Gall's* method [19] does not incorporate scale invariance at the training stage, therefore fixed sized (i.e. $w \times h$) object patches are sampled from the bounding boxes, which are pre-scaled to approximately the same size. At run time, to achieve scale invariance, detection is done at several scales (refer Section 3.2.2.1 for details).

3.1.1.1 Patch Appearance

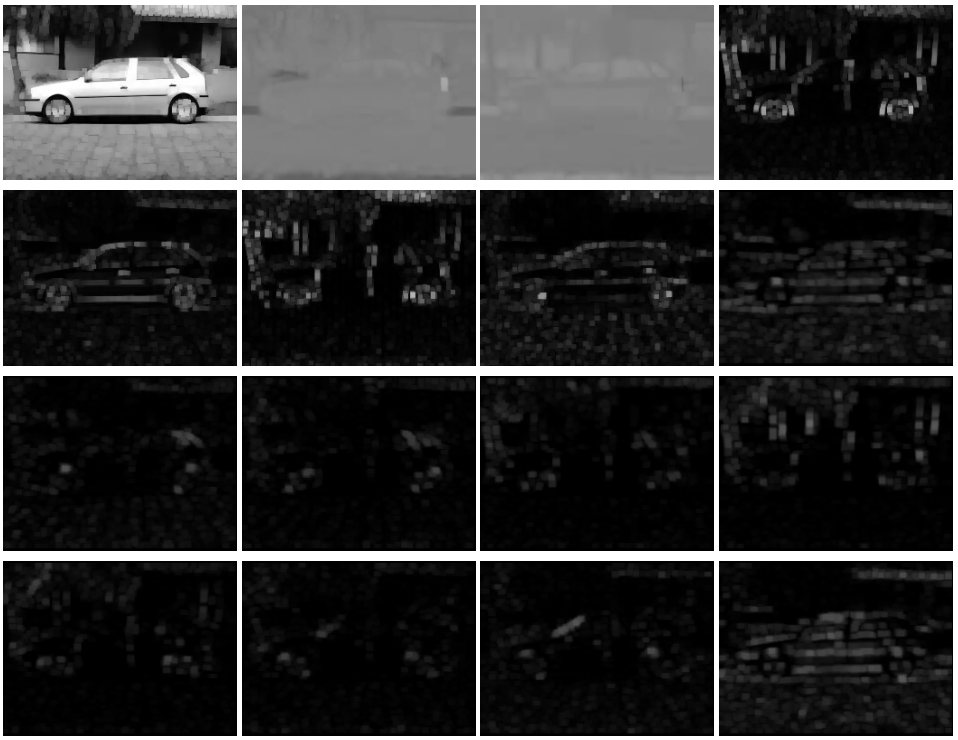
Each sampled patch carries the partial appearance of the image it is sampled from. The appearance is normally defined by the extracted appearance channels namely raw intensity and color, derivative filter responses, histogram of oriented gradients etc. Formally, the appearance of the patch can be written as $\mathcal{I} = \{I^1, I^2, ...I^N\}$, where each I^j is $w \times h$ size image and C is an index of a channel. A total 32 number of appearance channels are used in *Gall's* method [19]. They are listed as follows:

- 3 color channels in *Lab space*,
- 2 first order derivatives and 2 second order derivatives in x and y dimension respectively($|I_x|, |I_y|$, $|I_{xx}|, |I_{yy}|$) over intensity channel L,
- 9 HoG ([18]) like soft bin count of oriented gradients with 9 bins.

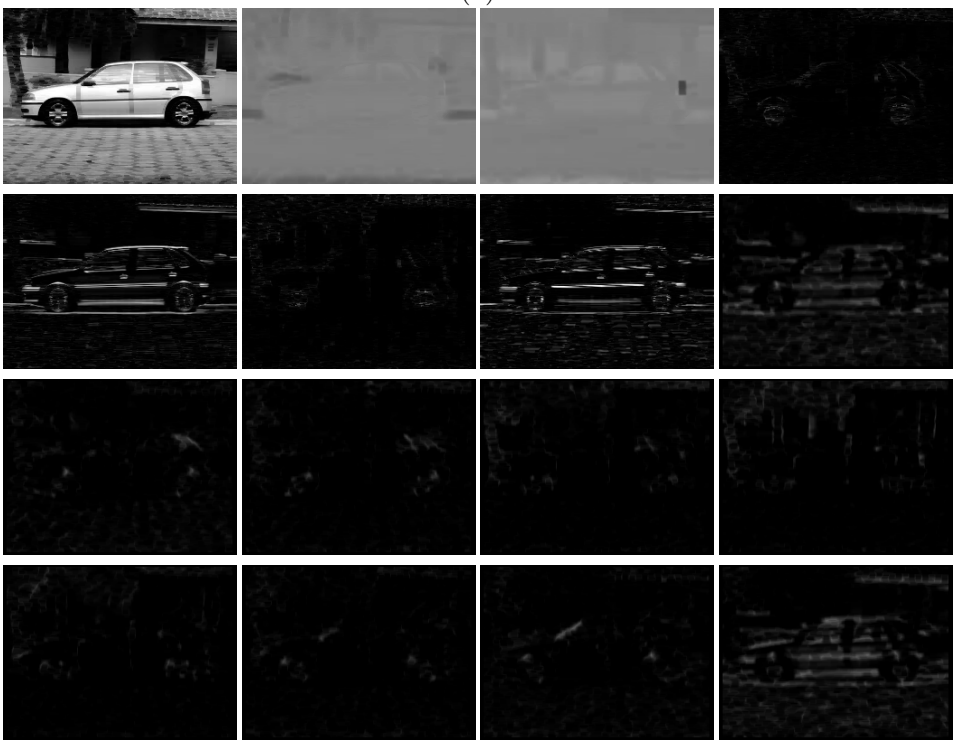
In order to make the detection robust against intensity change and noise, min and max filter responses are calculated for all the 16 channels mentioned above (Fig. 3.2).

3.1.2 Binary Node Tests

At each node a random binary test is generated during tree construction. In *Gall's* method [19], a simple pixel based test is chosen. The test ($t_{a,p,q,r,s,\tau}$) is defined by



(a)



(b)

FIGURE 3.2: Responses of (a) Max and (b) Min filter of the appearance channels of example image in Fig 3.1: (from top-left to bottom right) *Lab* color channels, 1st and 2nd derivatives in x and y direction of intensity channel L, 9 HoG bin images.

an appearance channel $a \in \{1, 2, ...C\}$, two offset positions (p, q), (r, s) within a $w \times h$ image patch, and a threshold τ:

$$t_{a,p,q,r,s,\tau} = \begin{cases} 0 & \text{if } I^a(p,q) - I^a(r,s) < \tau \\ 1 & \text{otherwise} \end{cases} \tag{3.2}$$

Such a test simply compares the difference of the pixel pair values in the same channel against a threshold.

Hough forests are trained in a supervised way. At runtime, both the class distribution and the spatial information (in this case displacement) vectors are used to cast the votes. The aim is to optimize the node split by picking the best test from a set of randomly generated tests $\{t^k\}_n$ for node n such that, the uncertainties in both class labels $\{c(\mathbf{y})\}_n$ and the displacement vectors $\{\mathbf{d}(\mathbf{y})\}_n$ decreases towards the leaves. To achieve such an optimization,at any node n, two measures of uncertainty are defined.

Class label uncertainty. This uncertainty measures the impurity of the class labels $c(\mathbf{y})$. It is defined as:

$$M_1(n) = -\sum_{l=0}^{1} log\left(\frac{|\{\mathbf{y} \in \mathcal{S}_n | c(\mathbf{y}) = l\}|}{|\mathcal{S}_n|}\right). \tag{3.3}$$

Displacement vector uncertainty. This second measure corresponds to the impurity of position vectors $\mathbf{d}(\mathbf{y})$ of the patch relative to the object center. At each node n, for all the positive class samples in the node $\mathcal{S}_n^p = \{\mathbf{y} \in \mathcal{S}_n | c(\mathbf{y}) = 1\}$, the uncertainty measure M_2 is defined as:

$$M_2(n) = \sum_{\mathbf{y} \in \mathcal{S}_n^p} \left(\mathbf{d}(\mathbf{y}) - \frac{1}{|\mathcal{S}_n^p|} \sum_{\mathbf{y}' \in \mathcal{S}_n^p} \mathbf{d}(\mathbf{y}')\right)^2. \tag{3.4}$$

Note that the displacement vector for background patches are undefined, they are ignored here.

Given the two uncertainty measures (Eqn. 3.3, Eqn. 3.4), a binary test is chosen as follows: First, a pool of node tests $\{t^k\}_n$ is generated by randomly sampling channel a, offset vectors p, q, r, s, and threshold τ. Then, with equal probability any one of the uncertainty measures (Eqn. 3.3, Eqn. 3.4) is chosen unless the number of negative patches is small ($< 5\%$), in which case only the displacement

uncertainty is chosen. Finally, the binary test with the minimal sum of the respective uncertainty measures for the two subsets is picked and assigned to that node (Eqn. 3.5).

$$\underset{k}{\operatorname{argmin}}\Big(M_\star(\{\mathbf{y}|t^k(\mathcal{I}(\mathbf{y}))=0\}) + M_\star(\{\mathbf{y}|t^k(\mathcal{I}(\mathbf{y}))=1\}\Big) \tag{3.5}$$

where $\star = 1$ or 2 (depending on the choice of uncertainty measure).

3.1.3 Leaf Information

During training, in each leaf node $\mathcal{L}$ information about the training patches reaching that node are saved. Out of all the patches that reached the leaf node, a class probability distribution $C_\mathcal{L} = \frac{\{\mathbf{y}|c(\mathbf{y})=1\}}{|\{c(\mathbf{y})\}|}, \mathbf{y} \in \mathcal{L}$ and a list of displacement vectors $\mathcal{D}_\mathcal{L} : \{\mathbf{d}(\mathbf{y})|\mathbf{y} \in \mathcal{L}\}$ are built. During runtime this information can be used to calculate the probability of a test patch to be an object class and to cast a probabilistic vote for the object centroid in XY dimensions in 2D hough space. This way, the leaves of the hough forest form a discriminative codebook.

3.2 Detection

Once the hough forest is constructed, it can be used to localize the object's centroid in test images using the generalized Hough Transform. Consider a patch $\mathcal{P}(\hat{\mathbf{y}}) = (\mathcal{I}(\hat{\mathbf{y}}), c(\hat{\mathbf{y}}), \mathbf{d}(\hat{\mathbf{y}}))$ centered at position $\hat{\mathbf{y}}$ in the test image. Here $\mathcal{I}(\hat{\mathbf{y}})$ is known, whereas $c(\hat{\mathbf{y}})$ and $\mathbf{d}(\hat{\mathbf{y}})$ are unknown. Let $E(\mathbf{x})$ denotes the random event corresponding to the existence of the object centered at $\mathbf{x}$ in the image.

3.2.1 Class Probability Calculation

The probabilistic evidence $p(E(\mathbf{x})|\mathcal{I}(\hat{\mathbf{y}}))$ of the object centered at $\mathbf{x}$ given the patch appearance $\mathcal{I}(\hat{\mathbf{y}})$ centered at $\hat{\mathbf{y}}$ can be calculated. Assuming the evidence results from the patches belong to the bounding box, it implies that $c(\hat{\mathbf{y}}) = 1$. As

a result we get:

$$\begin{aligned} p(E(\mathbf{x})|\mathcal{I}(\hat{\mathbf{y}})) &= p(E(\mathbf{x}), c(\hat{\mathbf{y}}) = 1|\mathcal{I}(\hat{\mathbf{y}})) \\ &= p(E(\mathbf{x})|c(\hat{\mathbf{y}}) = 1, \mathcal{I}(\hat{\mathbf{y}})) \cdot p(c(\hat{\mathbf{y}}) = 1|\mathcal{I}(\hat{\mathbf{y}})) \\ &= p(\mathbf{d}(\hat{\mathbf{y}}) = \hat{\mathbf{y}} - \mathbf{x}|c(\hat{\mathbf{y}}) = 1, \mathcal{I}(\hat{\mathbf{y}})) \cdot p(c(\hat{\mathbf{y}}) = 1|\mathcal{I}(\hat{\mathbf{y}})) \end{aligned} \quad (3.6)$$

Both factors in Eqn. 3.6, can be estimated by passing the patch appearance $\mathcal{I}(\hat{\mathbf{y}})$ through the trees in the class-specific hough forest. Let us assume that for a tree $\mathcal{T}$ the patch appearance ends up in a leaf $\mathcal{L}$. The first factor can then be approximated using the Parzen-window estimate based on the displacement vectors D_L collected in the leaf during training. While the second factor can be straightforwardly estimated as the proportion of object patches C_L at train time. For a single tree $\mathcal{T}$, the probability estimate can be written as:

$$p(E(\mathbf{x})|\mathcal{I}(\hat{\mathbf{y}}); \mathcal{T}) = \left[\frac{1}{|D_L|} \sum_{\mathbf{d} \in D_L} \frac{1}{2\pi\sigma^2} exp\left(- \frac{||(\hat{\mathbf{y}} - \mathbf{x}) - \mathbf{d}||^2}{2\sigma^2} \right) \right] \cdot C_L \quad (3.7)$$

Where $\sigma^2_{2\times 2}$ is the covariance of the Gaussian Parzen window. For the entire forest, the probabilities resulting from different trees are averaged:

$$p(E(\mathbf{x})|\mathcal{I}(\hat{\mathbf{y}}); \{\mathcal{T}_t\}_{t=1}^{T}) = \frac{1}{T} \sum_{t=1}^{T} p(E(\mathbf{x})|\mathcal{I}(\hat{\mathbf{y}}); \mathcal{T}_t) \quad (3.8)$$

3.2.2 Hough voting

Equations (3.7) and (3.8) define a probabilistic vote cast by a single image patch for the location of the object. All the votes coming from different patches in the image are accumulated in a 2D Hough image $\mathcal{V}(x)$ constructed in XY dimension.

$$\mathcal{V}(x) = \sum_{\hat{y} \in V(x)} p(E(\mathbf{x})|\mathcal{I}(\hat{\mathbf{y}}); \{\mathcal{T}_t\}_{t=1}^{T}) \quad (3.9)$$

Detection of object class can be achieved by searching for maxima locations and values $\{\mathbf{x}', \mathcal{V}(\mathbf{x}')\}$ in hough space. The $\mathcal{V}(\mathbf{x}')$ values serves as the confidence measures for each hypothesis.

The computation of the hough image using the order of operations as suggested by equations (3.7)-(3.8) would be inefficient. Instead, the same image can be

computed by first, going through each pixel location $\hat{\mathbf{y}}$ in the test image, passing the patch appearance $\mathcal{I}(\hat{\mathbf{y}})$ through every tree in the hough forest, and saving the class probability of a pixel averaged over all trees into a probability image $\mathcal{P}$ as in Eqn. 3.10.

$$\mathcal{P}(\hat{\mathbf{y}}) = \frac{1}{T}\sum_{t=1}^{T} C_{L_t}(\hat{\mathbf{y}}) \tag{3.10}$$

Subsequently, the probabilistic votes are cast from each pixel $\hat{\mathbf{y}}$ weighed with $\mathcal{P}(\hat{\mathbf{y}})$ to all pixels $\{\hat{\mathbf{y}} - \mathbf{d} | \mathbf{d} \in D_L\}$. The hough image $\mathcal{V}(\mathbf{x})$ is then obtained by Gaussian-filtering the vote counts accumulated in each pixel.

For two dimensions x_1 and x_2 of hough image $\mathcal{V}$ and weight w, each vote at location $\mathbf{x}(x_1, x_2) \in \mathcal{V}$ coming from any test image pixels $\hat{\mathbf{y}}(y_1, y_2)$ and one of the displacement vectors $\mathbf{d}(d_1, d_2)$ stored at that test pixels is given by:

$$\begin{aligned} x_1 &= y_1 - d_1 \\ x_2 &= y_2 - d_2 \\ w &= \mathcal{P}(\hat{\mathbf{y}}) \end{aligned} \tag{3.11}$$

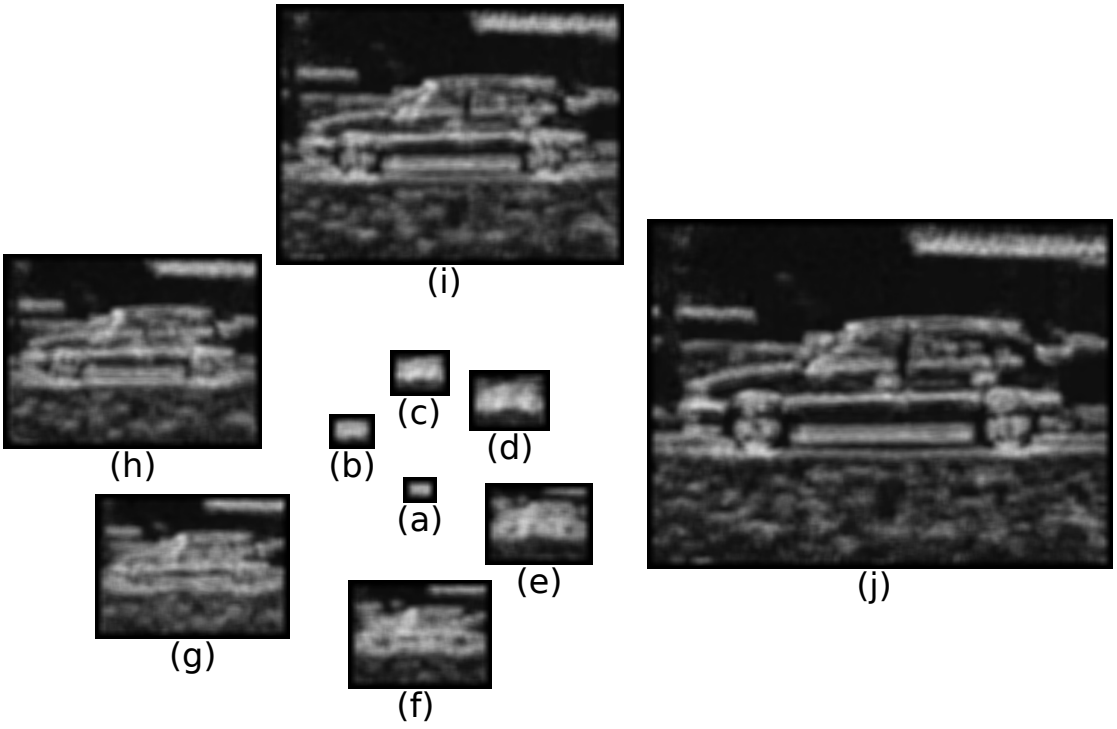


FIGURE 3.3: (a)-(j) Class probability calculation of example image in Fig. 3.1 in increasing scales.

3.2.2.1 Handling Variable Scales

To achieve scale invariance, the test image is resized by a set of scale factors $s_1, s_2, ...s_S$. The class probability images $\mathcal{P}_s$ for each scale are generated (Fig. 3.3). The Hough images $\mathcal{V}_1, \mathcal{V}_2, ...\mathcal{V}_S$ are then computed independently at each scale. For $\mathbf{x} \in \mathcal{V}_s$ Equ. 3.11 changes to as follows:

$$\begin{aligned} x_1 &= \frac{y_1}{s} - d_1 \\ x_2 &= \frac{y_2}{s} - d_2 \\ w &= \mathcal{P}_s(\frac{\hat{\mathbf{y}}}{s}) \end{aligned} \tag{3.12}$$

All the hough images are then stacked in a 3D scale-space frustum (Fig. 3.4), the Gaussian-filtration is performed across the third (scale) dimension, and the maxima of the resulting function are localized in 3D. The resulting detection hypotheses have the form $(\mathbf{x}', s', \mathcal{V}^{s'}(\mathbf{x}'))$.

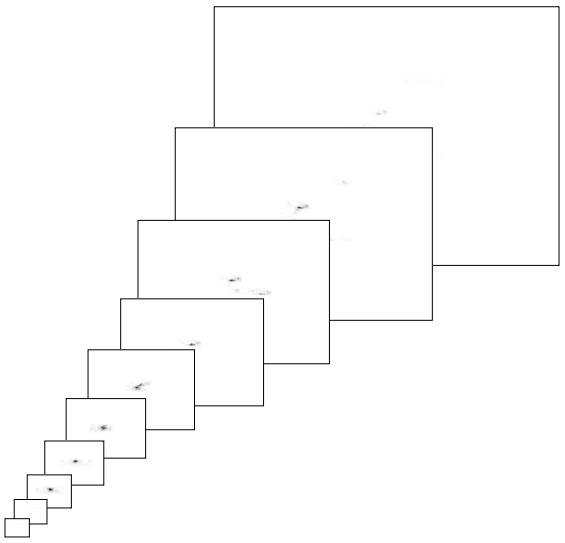
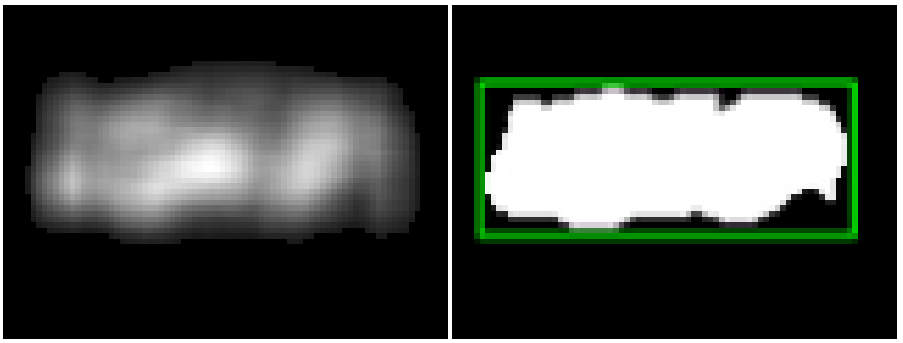


FIGURE 3.4: For each scale a Hough space in object location is generated. The hypothesis corresponding to the maxima in entire scale space is sought.

3.2.3 Back-Projection and Bounding Box Construction

Once the local maxima $(\mathbf{x}', s')$ are sought in $\{\mathcal{V}_s\}$, all the votes contributing to the hypothesis are collected. Support S_h for hypothesis h at location $\mathbf{x}'$ can be



(a) (b)

FIGURE 3.5: Votes that contributed to maxima in Hough space are then again back projected in the (a) back-projection mask and thresholded with adaptive threshold to generate (b) binary mask. Bounding box is estimated tightly encompassing the binary mask

written as:

$$S_h = \{\mathbf{x} \in \mathcal{V}'_s | K(\mathbf{x} - \mathbf{x}') > 0\}. \tag{3.13}$$

where K is a kernel (generally Gaussian kernel) with only local support such that the set S_h contains only votes in the local neighborhood of $\mathbf{x}'$. The votes collected at $\mathbf{x}$ are back projected to location $\hat{\mathbf{y}}$ in a back-projection mask.

This back-projection mask is simply thresholded by an adaptive threshold (set to half the value range) to form a binary mask. The tightest bounding box encompassing this binary mask is used as bounding box estimate (Fig.3.5).

3.2.4 Summary

The chapter explained in detail the hough forest framework for class specific object recognition proposed by Gall and Lempitsky [19]. The framework uses RGB images as input data and constructs a random forest classifier in a supervised way. It uses appearance based pixel pair test for node splitting criterion while minimizing uncertainty in a class label as well as in displacement vectors towards leaf nodes. At the leaf node it stores the information about training samples reaching it and during recall uses the same information to cast a probabilistic hough vote about object location and scale.

Chapter 4

Extensions

The previous chapter (Chap. 3) discussed class specific object recognition using Hough forest. This technique mainly exploits visual cues such as raw intensity, color, derivatives etc to train the classifier. These cues are obtained from 2D RGB images. In this chapter, we extend Hough forest by introducing shape cues such as *depth* and *surfel point pair* features [34, 35]. Using this additional information we not only localize object position but also detect its orientation and estimate full 6-DoF pose.

In Section 4.1, we explain modified training of forest with additional shape cues. We derive the relative orientation of query point with respect to canonical object orientation and save this additional information in leaf nodes, in a memory efficient way. In Section 4.2, we propose a simple voting system for object location and orientation. Further, we describe how we adapted Hough forest framework to include shape cues and discuss their advantages in Section 4.3.

4.1 Training Using RGB-D Images

4.1.1 Training Data

Using depth image I_d, image pixel coordinates (r, c) (r^{th} row and c^{th} column) can be transformed to real 3D coordinates (x, y, z) as follows:

$$\begin{aligned} x &= \frac{(c - c_{center}) I_d(r, c)}{f} \\ y &= \frac{(r - r_{center}) I_d(r, c)}{f} \\ z &= I_d(r, c) \end{aligned} \tag{4.1}$$

Where f is the focal length of the camera and (r_{center}, c_{center}) is the image center.

In a modified version of Hough forest $\mathcal{F}$, instead of fixed sized local image patches, each of the decision trees $\mathcal{T}$ is built using a set of sampled image pixels $\mathbf{q}$. At the root node the training set $\mathcal{S}_0$ can be written as:

$$\mathcal{S}_0 = \{(\mathcal{I}(\mathbf{q}), c(\mathbf{q}), \mathbf{p}(\mathbf{q}), \mathbf{d}(\mathbf{q}), \mathbf{n_q})\} \tag{4.2}$$

where $\mathcal{I} = \{I^1, I^2, \ldots, I^N\}$ is the appearance of the training image from where the pixel is sampled, I^j is the j^{th} appearance channel, $c(\mathbf{q}) \in \mathcal{C} : \{0, 1\}$ is a class label, $\mathbf{p}(\mathbf{q})$ is a $3D$ point corresponding to the pixel $\mathbf{q}$, $\mathbf{d}(\mathbf{q})$ is the relative 3D location of an object center to the $\mathbf{p}(\mathbf{q})$ and $\mathbf{n_q}$ is a normal vector at the 3D point $\mathbf{p}(\mathbf{q})$.

4.1.2 Tree Construction

As explained in previous chapter (Chap. 3), each node n during training is ascribed a pixel-pair-based binary test $t : \mathbf{q} \to \{0, 1\}$ over an appearance channel of the image to separate the training samples $\mathbf{q} \in \mathcal{S}_n$ that reach the node. For j^{th} appearance channel I^j and $2D$ offset vectors $\mathbf{u}_1$, $\mathbf{u}_2$, the test $t_{j,\mathbf{u}_1,\mathbf{u}_2,\tau}(\mathbf{q})$ is then defined as:

$$t_{j,\mathbf{u}_1,\mathbf{u}_2,\tau}(\mathbf{q}) = \begin{cases} 0, & \text{if } I^j(\mathbf{q} + \mathbf{u}_1) - I^j(\mathbf{q} + \mathbf{u}_2) < \tau \\ 1, & \text{otherwise.} \end{cases} \tag{4.3}$$

The offset vectors $\mathbf{u}_k$ are chosen within a local region of query pixel $\mathbf{q}$.

Similar to relative position vector $\mathbf{d}(\mathbf{q})$, at each node n and for all the positive class samples $\mathcal{S}_n^p = \{\mathbf{q} \in \mathcal{S}_n | c(\mathbf{q}) = 1\}$ we also compute a relative transformation matrix from the local frame of the query point $\mathbf{p}(\mathbf{q})$ to a canonical object class frame. Let $\mathbf{C}$ be the camera frame and $\mathbf{Q}$ be the local frame at query point $\mathbf{p}(\mathbf{q})$. If we define ${}^{M}\mathbf{T}_{N}$ as a relative transformation from frame N to frame M then the local frame at point $\mathbf{p}(\mathbf{q})$ (in camera frame) can be denoted as ${}^{C}\mathbf{T}_{\mathbf{Q}}$. The local frame at point $\mathbf{p}(\mathbf{q})$ is computed using the normal $\mathbf{n}(\mathbf{q})$ and one of the 3D offset vectors $\mathbf{p}(\mathbf{u}_k)$, $k \in \{1, 2\}$. Hence at each query point $\mathbf{p}(\mathbf{q})$, two local frames $\{{}^{C}\mathbf{T}_{\mathbf{Q}_1}, {}^{C}\mathbf{T}_{\mathbf{Q}_2}\}$ are generated. Where, ${}^{C}\mathbf{T}_{\mathbf{Q}_k} = [{}^{C}\mathbf{R}_{\mathbf{Q}_k}, {}^{C}\mathbf{t}_{\mathbf{Q}_k}] = [\mathbf{l}_k, \mathbf{m}_k, \mathbf{n}_k, {}^{C}\mathbf{t}_{\mathbf{q}}]$ and

$$\begin{aligned} \mathbf{l}_k &= \frac{\mathbf{n}_\mathbf{q} \times \mathbf{u}_k}{||\mathbf{n}_\mathbf{q} \times \mathbf{u}_k||_2}, \\ \mathbf{m}_k &= \mathbf{l}_k \times \mathbf{n}_\mathbf{q}, \\ \mathbf{n}_k &= \mathbf{n}_\mathbf{q}, \\ {}^{C}\mathbf{t}_\mathbf{q} &= \mathbf{d}(\mathbf{q}). \end{aligned} \tag{4.4}$$

If we denote ${}^{C}\mathbf{T}_{O}$ as a canonical frame of an object in the camera frame, then the relative transformation from the object frame to the query pixel frame ${}^{\mathbf{Q}_k}\mathbf{T}_{O}$ (that corresponds to k^{th} offset vector) is:

$${}^{\mathbf{Q}_k}\mathbf{T}_{O} = ({}^{C}\mathbf{T}_{\mathbf{Q}_k})^{-1} \times {}^{C}\mathbf{T}_{O}, \tag{4.5}$$

During recall, class labels $\{c(\mathbf{q})\}$, displacement vectors $\{\mathbf{d}(\mathbf{q})\}$ and rotation quaternions $\{{}^{\mathbf{Q}_k}\mathbf{R}_{O}\}$, saved at the reached leaf node, cast a probabilistic vote for object location and orientation. Hence it is important to minimize the uncertainty of votes coming from a single leaf node. In order to minimize the class label uncertainty and displacement vector uncertainty we use uncertainty measures M_1(Eqn. 3.3) and M_2 (Eqn. 3.4) as shown in Chapter 3.

Additionally, we propose an uncertainty measure for object orientation votes as below:

$$M_3(n) = \sum_{k=1}^{2} \sum_{\mathbf{q} \in \mathcal{S}_n^p} A\left(({}^{\mathbf{Q}_k}\mathbf{R}_{O})^{-1} \times \underset{\mathbf{q} \in \mathcal{S}_n^p}{\text{quatinterp}}({}^{\mathbf{Q}_k}\mathbf{R}_{O}) \right), \tag{4.6}$$

where quatinterp performs generalized quaternion interpolation to compute mean rotation and $A(\circ)$ denotes rotation angle of the quaternion. During training, if

negative patches are $> 5\%$, M_1 is chosen with 50% probability whereas, M_2 or M_3 is chosen with 25% probability. If the number of negative patches is small ($< 5\%$), only M_2 and M_3 is chosen with equal probability.

4.1.3 Leaf Information

During training each leaf-node $\mathcal{L}$ saves information about the training samples reaches to that node. In this case, relative class frequencies $C_{\mathcal{L}}$, displacement vectors $\mathcal{D}_{\mathcal{L}}$ and relative orientations $\mathcal{R}_{\mathcal{L}}$ are saved.

For each training pixel $\mathbf{q}$, we save relative orientations corresponding to offset vectors chosen at each node along the path from the root node to the reached leaf node. The local frames corresponding to each node at the query point differ only by an angle around the normal $\mathbf{n_q}$ at $\mathbf{q}$ (Fig. 4.1). If the leaf depth in tree is d, we can store these $2 \times d$ relative rotations for each training pixel memory-efficiently by one reference rotation ${}^{\mathbf{Q}_k,0}\mathbf{R}_O$ for the orientation at the root node and angular differences $\alpha_{\mathbf{Q_k},n}$ around $\mathbf{n_q}$ for every other point-pair in the decision cascade, i.e.,

$$ {}^{\mathbf{Q}_k,n}\mathbf{R}_O = {}^{\mathbf{Q}_k,n}\mathbf{R}_{\mathbf{Q}_k,0} \times {}^{\mathbf{Q}_k,0}\mathbf{R}_O, \tag{4.7} $$

where,

$$ {}^{\mathbf{Q}_k,n}\mathbf{R}_{\mathbf{Q}_k,0} = \mathbf{R}\left(\mathbf{n_q}, \alpha_{\mathbf{Q}_k,n}\right). \tag{4.8} $$

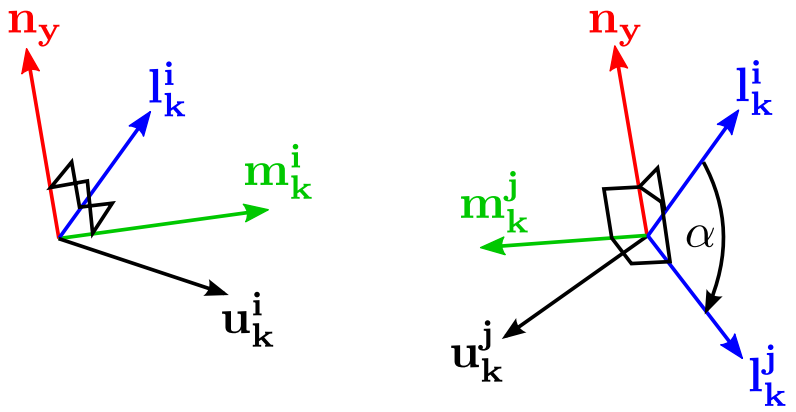


FIGURE 4.1: All local frames differ only by an angle around the normal. This property allows for efficiently saving relative orientations towards the object rotation by angles for training examples.

4.2 6-DoF Object Detection

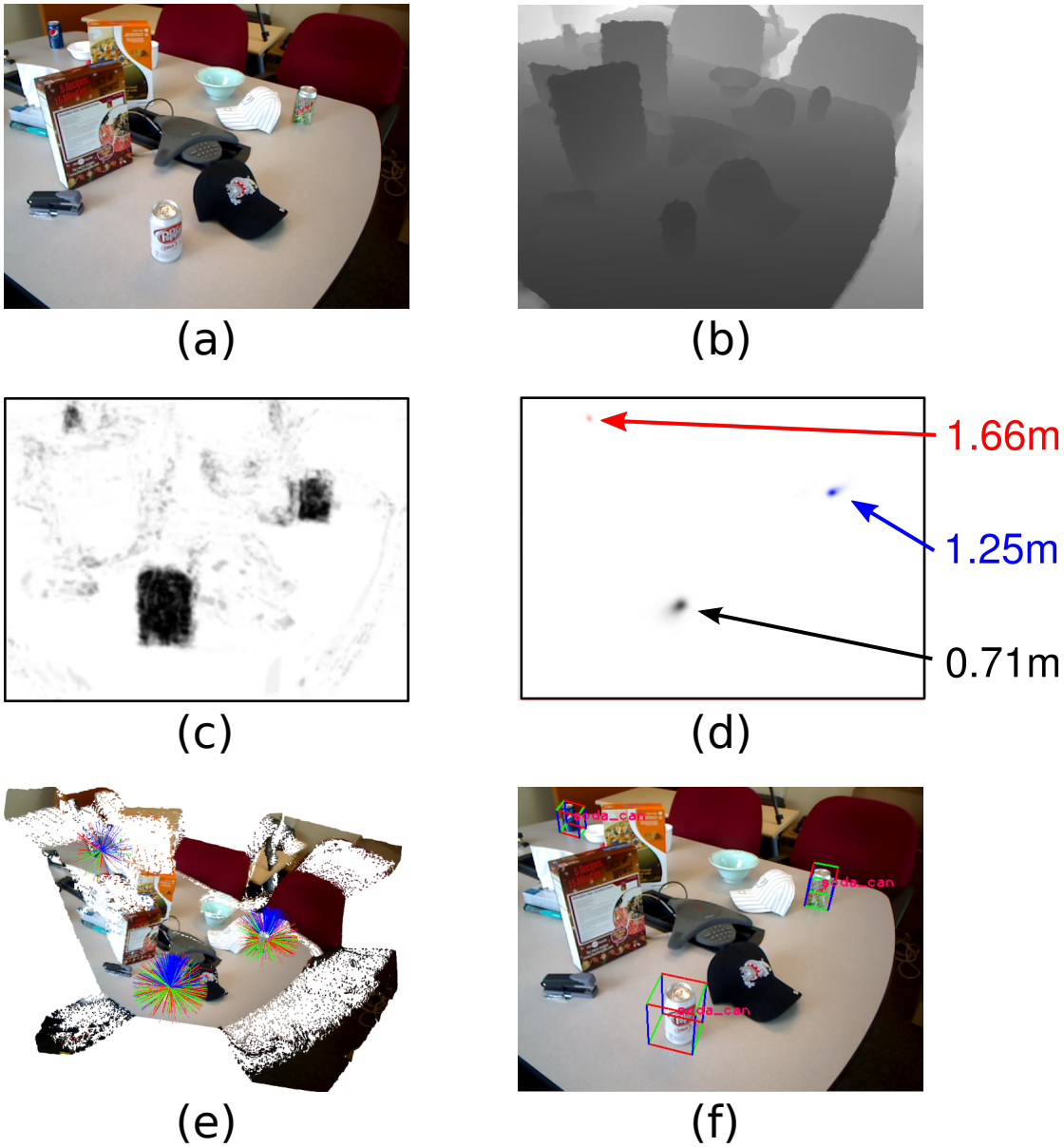


FIGURE 4.2: During detection, for each test pixel in a (a) test image, the (b) class probability is computed. Pixels labeled as object class then vote for the object location in 3D Hough space (c) and maxima are sought. Votes that contributed in finding the maxima then again vote for object orientation in the 4D Hough space corresponding to each maxima (d). Similarly maxima in orientation Hough space are sought. Once a complete 6-DOF object pose is found, the pre-computed bounding boxes are projected (e).

During recall, each test image pixel $\hat{\mathbf{q}}$ traverses through all the trees $\{\mathcal{T}_t\}$ and the class probability of the image pixel is computed by averaging the frequency of class

labels at the reached leaf nodes $\mathcal{L}_t(\hat{\mathbf{q}})$ recorded during training, i.e.

$$p(c \mid \mathcal{F}, \hat{\mathbf{q}}) \;=\; \frac{1}{T}\sum_{t=1}^{T} p(c \mid \mathcal{L}_t(\hat{\mathbf{q}})), \tag{4.9}$$

where $p(c \mid \mathcal{L}_t(\hat{\mathbf{q}})) = C_{\mathcal{L}_t}(c)$ is the class frequency in leaf $\mathcal{L}_t$ of tree t.

Detection is done in two passes: First the object location in $3D$ Hough space is estimated. Training pixels contributing to each location hypothesis then vote for object orientation in corresponding 4D Hough space.

Hypothesis search for object location. In the first pass, each reached leaf node casts probabilistic votes for the object position in a 3D Hough space $\mathcal{V}(x, y, s)$ and maxima are sought. We discretize the 3D Hough space into image locations and scales. The latter is represented in inverse depth to model higher position accuracy at closer distances. For any location (x, y, s) in Hough space $\mathcal{V}$ votes coming from each point $\mathbf{p}(\mathbf{q})$ and 3D displacement vector $\mathbf{d}(\mathbf{q})$ with weight w can be written as:

$$\begin{aligned} w &= p(c \mid \mathcal{F}, \hat{\mathbf{q}}) \cdot \mathrm{dist}(\hat{\mathbf{q}}) \\ x &= \mathbf{p}(\hat{\mathbf{q}})_x - \mathbf{d}(\mathbf{q})_x \\ y &= \mathbf{p}(\hat{\mathbf{q}})_y - \mathbf{d}(\mathbf{q})_y \\ s &= \frac{1}{\mathbf{p}(\hat{\mathbf{q}})_z - \mathbf{d}(\mathbf{q})_z} \end{aligned} \tag{4.10}$$

where $\mathrm{dist}(\hat{\mathbf{q}}) = \mathbf{p}(\hat{\mathbf{q}})_z$ is a distance of $\mathbf{p}(\hat{\mathbf{q}})$ from the camera. Note that w is proportional to both the relative class frequency and the distance of the test point from the camera. The latter is used to normalize the weight against scale change in the projected size of the object in the image. The above voting scheme directly votes for the appropriate scale of the object in the $3D$ Hough space, this obviates the need for calculating separate probability images for each scale as in *Gall's* method [19].

Object orientation estimation. In the second pass, all the training pixels that contributed to each maxima in object location Hough space, then vote for the orientation of the object (parameterized in quaternions) in a $4D$ Hough space.

The orientation votes are computed for all nodes that pixel $\hat{\mathbf{q}}$ passed on its path from root to leaf in each tree. At the root node, the orientation votes

$$^{C}\mathbf{R}_{O}^{k} = {}^{C}\mathbf{R}_{\hat{\mathbf{q}},0}^{k} \times {}^{\mathbf{q},0}\mathbf{R}_{O}^{k}, \tag{4.11}$$

are computed from the local frames for the offsets $k \in \{1, 2\}$. All other nodes vote for the orientations

$$^{C}\mathbf{R}_{O} \;=\; {}^{C}\mathbf{R}_{\hat{\mathbf{q}},n}^{k} \times \mathbf{R}\left(\mathbf{n}_{\mathbf{y}}, \alpha_{\mathbf{q},n}^{k}\right) \times {}^{\mathbf{q},0}\mathbf{R}_{O}^{k} \tag{4.12}$$

which are recovered from the relative rotations $\mathbf{R}\left(\mathbf{n}_{\mathbf{q}}, \alpha_{\mathbf{Q}_k,n}\right)$ towards the reference orientation in the root.

Once a full 6-DoF pose is detected, a pre-measured 3D bounding box is placed and aligned according to the location and orientation of the object (see Fig. 4.2).

4.3 Features

We train our binary decision functions at each node on different appearance channels namely *color* in Lab space, 1^{st}- and 2^{nd}-order *gradients* in x and y dimensions on the intensity channel, *depth*, *surfel-pair features*, and *HoG*. HoG channels are produced as a soft bin count of gradient orientation in a depth-normalized window around each pixel. To boost invariance against noise and disturbance, we further perform *min* and *max* filtration with depth-normalized kernel-size in a local neighborhood.

4.3.1 Depth

It has been observed that depth cues can improve object detection tremendously [36, 37]. It enriches the information about the object in terms of geometry, shape, contour etc. We thus use depth as an additional appearance channel.

Unlike *Gall's* method [19], we use depth-normalized offset vectors in binary node tests. This way, the size of the offset vectors is automatically adjusted according to the scale of objects in the image, which obviates the need for presenting object class

FIGURE 4.3: At every query pixel $\mathbf{q}$ offset vectors are scaled according to the depth $I_d(\mathbf{q})$

training images at multiple scales and handles variable scales efficiently during recall. Using the depth information $I_d(\mathbf{q})$ at training pixel $\mathbf{q}$, the binary test function Eq. (4.3) is changed as below:

$$t_{j,\mathbf{u}_1,\mathbf{u}_2,\tau}(\mathbf{q}) = \begin{cases} 0, & \text{if } I^j\left(\mathbf{q} + \frac{\mathbf{u}_1}{I_d(\mathbf{q})}\right) - I^j\left(\mathbf{q} - \frac{\mathbf{u}_2}{I_d(\mathbf{q})}\right) < \tau \\ 1, & \text{otherwise.} \end{cases} \tag{4.13}$$

4.3.2 Surfel-pair Features

Each object is characterized by a specific geometry, which provides the basic visual hint for recognition. For example, a *soda can* has a cylindrical shape whereas *cereal boxes* are cuboid. The availability of dense depth images allow incorporation geometric features into the decision cascade.

In order to capture such shape characteristics, we include 4 dimensional surfel-pair features [8] as an additional node splitting criteria. These features characterize the

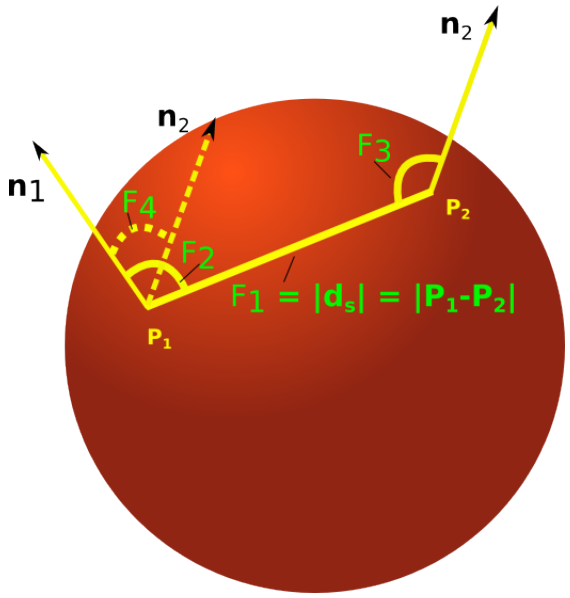


FIGURE 4.4: Surfel is a 4 dimensional point pair-feature comprising of distance between two points, angles between each of the normals and the vector connecting two points, and angle between the two normals.

relative position and local surface orientation of two points in the scene. For any two points $\mathbf{p_1}$ and $\mathbf{p_2}$ on the object, if their corresponding normals are denoted by $\mathbf{n}_1$, and $\mathbf{n}_2$, the surfel-pair feature vector is computed as:

$$S(\mathbf{p}_1, \mathbf{p}_2) = \left(\|\mathbf{d}\|_2, \angle(\mathbf{n}_1, \mathbf{d}), \angle(\mathbf{n}_2, \mathbf{d}), \angle(\mathbf{n}_1, \mathbf{n}_2)\right), \tag{4.14}$$

where $d := \mathbf{p}_2 - \mathbf{p}_1$. If any one of the channels of the surfel-pair features is chosen for the test function, the function thresholds on the value of the feature directly by:

$$t_{j,\mathbf{u}_1,\mathbf{u}_2,\tau}(\mathbf{q}) = \begin{cases} 0, & \text{if } S^j\left(\mathbf{p}\left(\mathbf{q} + \frac{\mathbf{u}_1}{I_d(\mathbf{q})}\right), \mathbf{p}\left(\mathbf{q} - \frac{\mathbf{u}_2}{I_d(\mathbf{q})}\right)\right) < \tau \\ 1, & \text{otherwise.} \end{cases} \tag{4.15}$$

4.4 Summary

In this chapter we described a Hough forest framework for object orientation estimate in addition to object detection. We demonstrated the use of depth information at various levels of the detection pipeline. We also introduced shape and geometry cues to incorporate characteristic information about object class into the classifier.

Chapter 5

Generation of Training Dataset

Visual object detection in unstructured environments is challenging due to background variability, clutter, occlusions, changes in illumination, different viewpoints and variable scales. It is important to capture these variations during training to achieve robustness in the system. However, manually annotating a rich training dataset with large variety in scenes with ground truth object location and orientation is infeasible. Instead we propose to make use of a simple controlled training setup that provides ground truth conveniently and that artificially renders a variety of scenes from this data.

Later in this chapter, we provide a motivation for the use of an artificially rendered new dataset in Section 5.1. In Section 5.2 we explain our rendering pipeline in detail and show some results. We further exploit the controlled turn-table setup to compose full 3D object model and discuss its benefits in the context of object recognition in Section 5.2.1.

5.1 Motivation

RGB-D Object dataset [2] is obtained from several different view-points of objects on a turn-table setup with varying pitch and yaw angles (Fig. 5.1). Using this camera setup with a fixed distance from the object, a video sequence is recorded from different heights for each object spun around on the turntable at a constant speed.

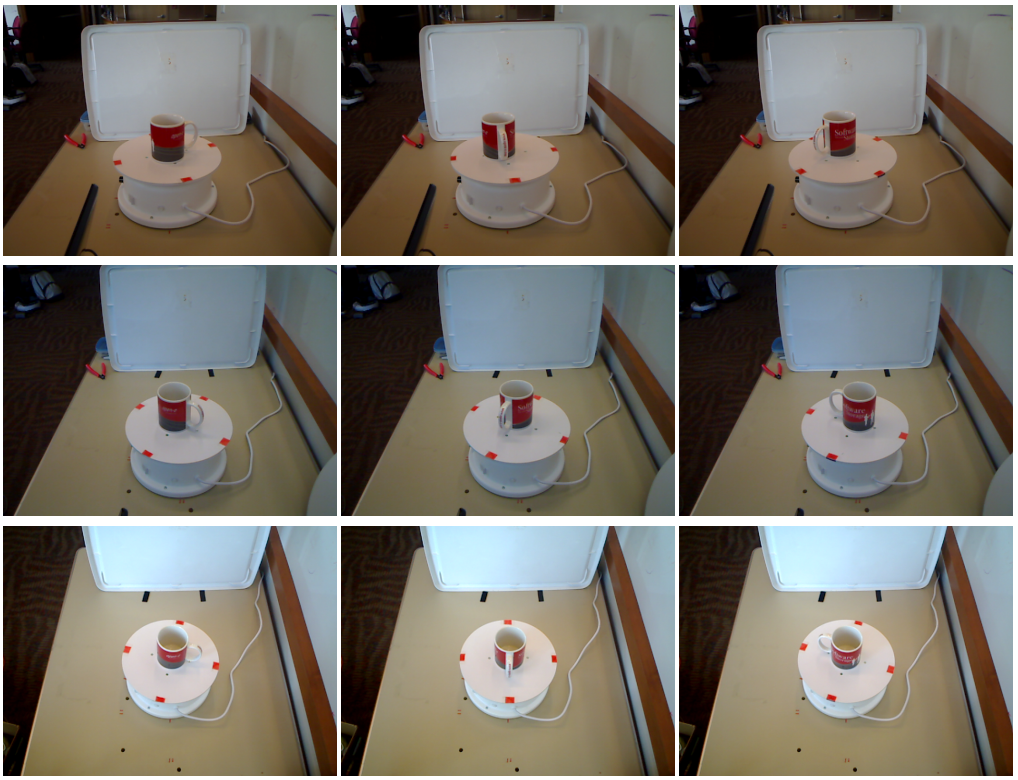


FIGURE 5.1: Training images captured using a turn-table contains all the views of the object and 3 different pitch angles from the fixed distance of the camera.

With this setup, capturing training data can be accomplished very fast. By utilizing the knowledge of the turntable rotation and distance of the camera from the object, 2D bounding box and object orientation can be annotated efficiently. A major short coming of this approach in the context of hough Forest is that it fails to capture the variety in real world scenes and systematically learns the fixed background as a part of the object. This limits the detection of objects to a simple scenario without background clutter and occlusions.

5.2 Image Rendering Pipeline

To generate new training scenes, we extract RGB-D segments of different objects from the turn-table views. In order to create positive training examples, we first render a table plane with varying texture and color. The object to be trained is placed at its original location and orientation on the turn-table to take advantage of the annotations. To simulate clutter and occlusions, we place object views of other classes around the positive object class instance.

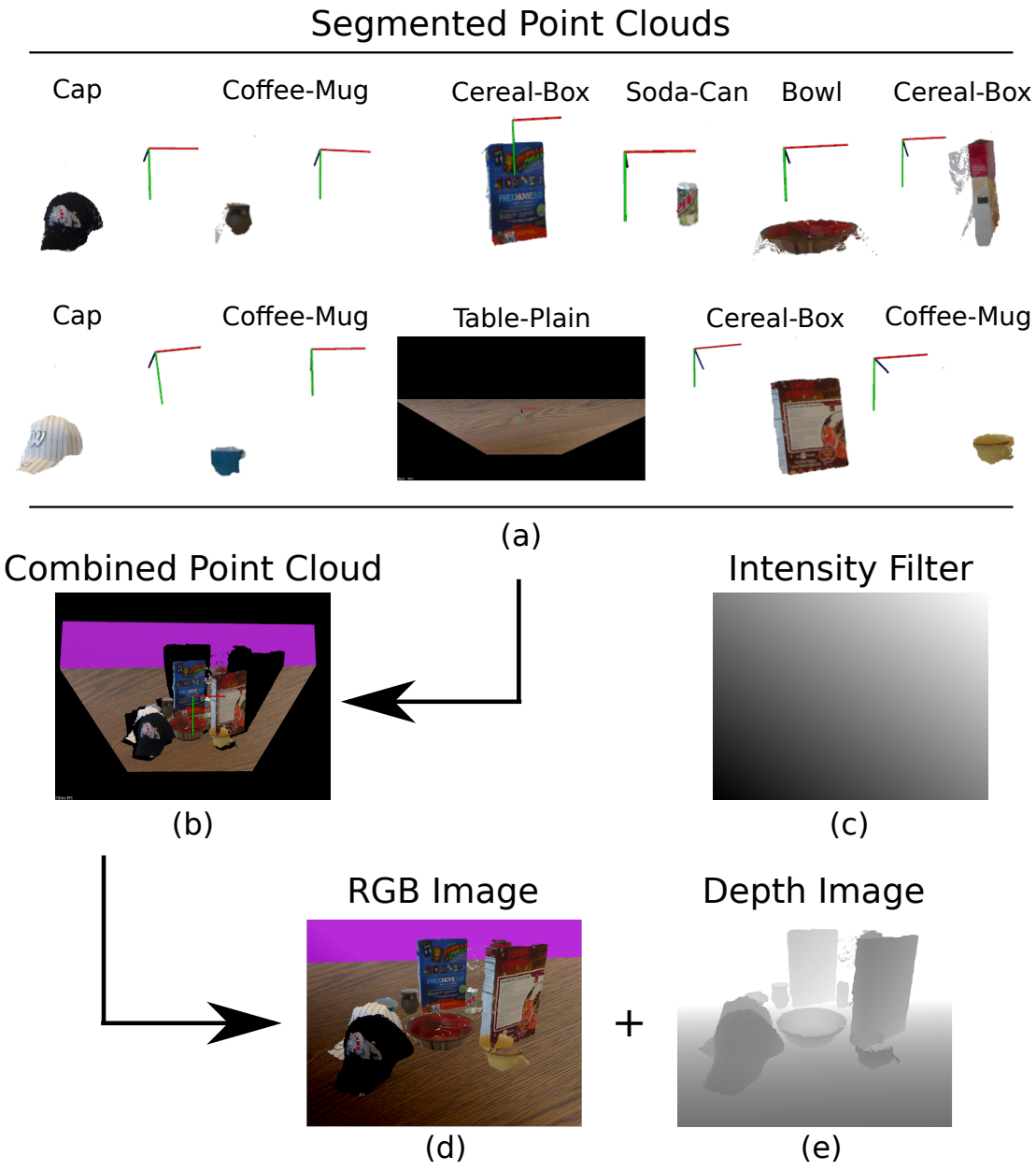


FIGURE 5.2: Figure shows the construction of a positive training example for *bowl*. The class instance is placed in a same pose from the turntable on an artificially generated table plane. Background clutter is introduced by negative object classes surrounding the bowl. Gamma of the image is changed by adding the (c) gradient image to the final RGB image.

Example images for the background class are simply populated with views of all other object classes. We then change the gamma of the image systematically to introduce intensity-changes in the image. Although our approach does not provide a photo-realistic rendering, our goal is to achieve a similar statistical distribution in intensity, color and depth as in natural scenes. Figures 5.3, 5.4 show training images rendered with our approach using the RGB-D Objects Dataset [38].

FIGURE 5.3: Positive training images are obtained by placing positive object class instances at the same place and orientation as in RGB-D Object dataset, in order to take advantage of annotations. To simulate clutter and occlusion, object class instance of other classes are are placed around the object.

FIGURE 5.4: Negative training images are generated by populating rendered table plane with negative object class.

5.2.1 Segmentation and Construction of 3D Object Model

Object segmentation is easily obtained through segmenting the turn table from the scene and then segmenting object above the turn table plane. The object's 3D center and bounding box is found through overlaying object segments from 360° viewing directions into a single point cloud and measuring extents of the points.

Once a 3D bounding box is obtained we extract the real-object's size and its center. This information can be used to learn 3D displacement vector of object point relative to the object's center, which eventually helps decreasing the smearing of the votes during recall (Fig. 5.6) and effectively increases the overall recognition rate.

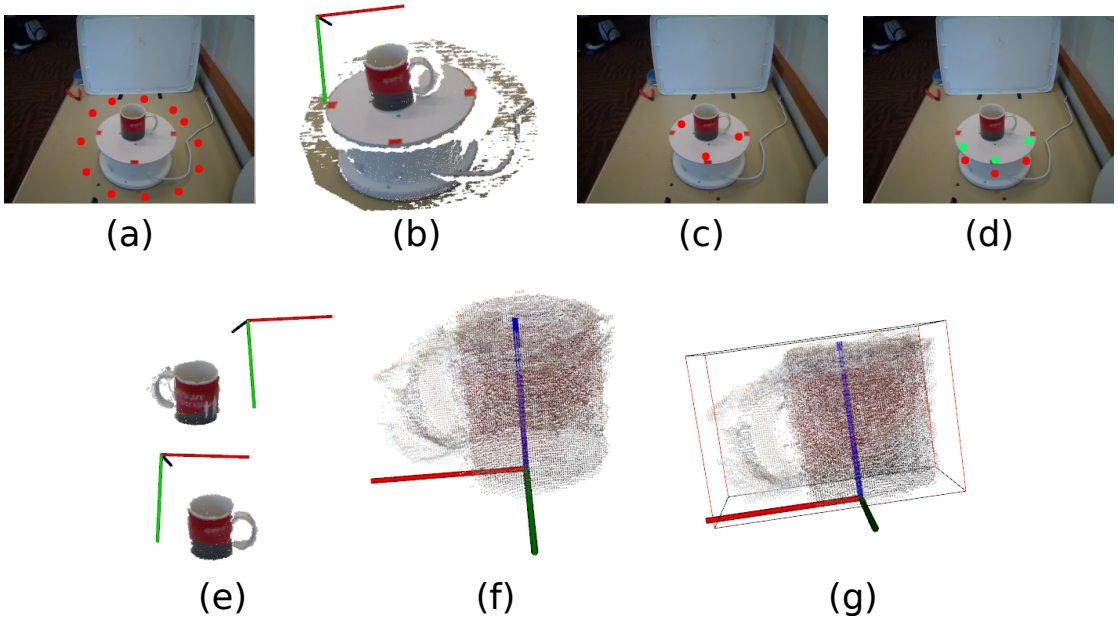


FIGURE 5.5: First the convex hull is constructed by selecting points surrounding the turntable, such that all the points in the cloud outside the hull are discarded. Then turntable plane is selected. All the points below turn table plane are once again filtered. After removing noise from the cloud, segmented object cloud can be acquired from a single view. By overlapping the views from different angles, a full 3D object model is constructed.

5.2.2 Filling Depth

RGB-D object dataset is obtained by the Microsoft's Kinect camera. The Kinect captures the depth map by projecting infra-red patterns on the scene and measuring their displacement. Due to the limitations of the depth sensing technology, data captured by Kinect contains lot of missing depth values especially for shiny object surfaces e.g. *soda-can*, *bowl*, *coffee-mug*.

We preprocessed all the depth images to fill missing depth value using join bilateral filtering [39]. It basically applies iterative smoothing at several scales in depth images while maintaining edges in the intensity image.

5.3 Summary

We discussed the short comings of the training data captured with turn-table setup and provided the solution to overcome the problem. We illustrated the image

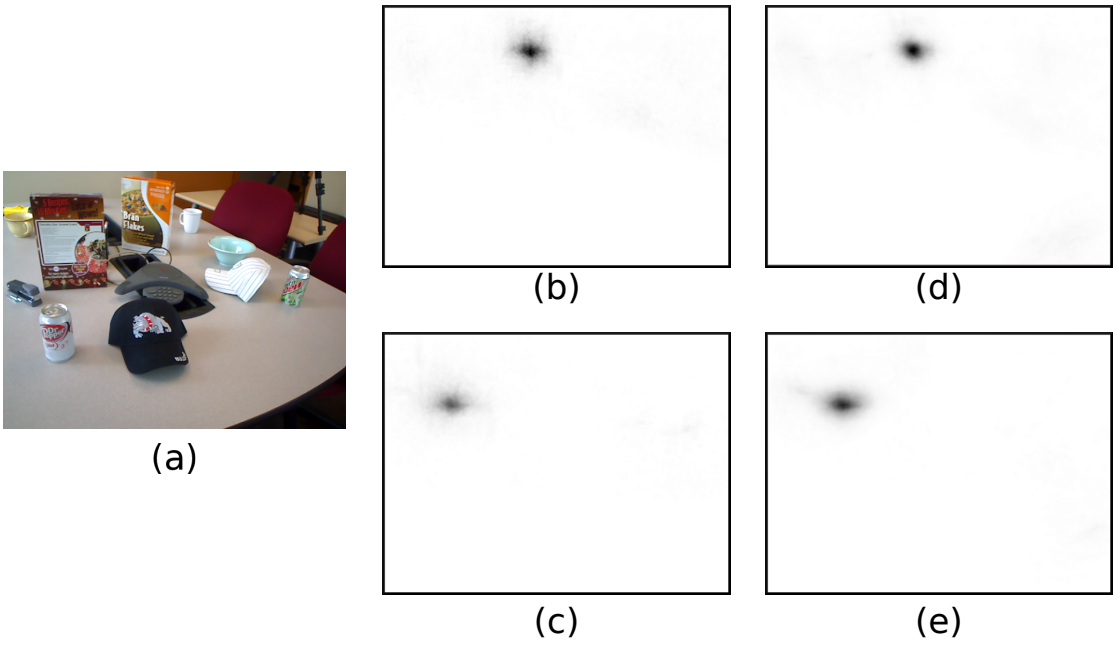


FIGURE 5.6: By voting in 3D, smearing of votes can be reduced. (a) A test image showing an instance of *cereal-box*. (b)-(c) Hough images constructed by 2D displacement vector votes, and (d)-(e) hough images constructed by 3D displacement vector votes for two instances of *cereal-box*.

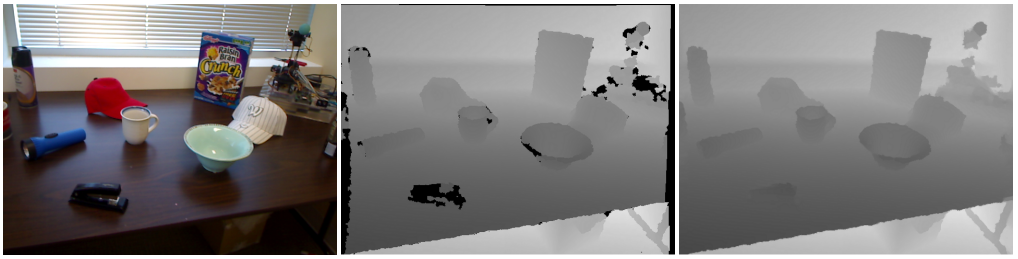


FIGURE 5.7: Missing depth values are filled using Cross Bilateral Filer [39]. (a) RGB image, (b) Depth image with missing depth values (c) Smoothed depth image while preserving edges from RGB image.

rendering pipeline to generate new training dataset which models the statistical distribution of image properties same as in natural scene.

Chapter 6

Evaluation and Discussion

In this chapter we present our experimental results and show comparison[1] with the original implementation of Hough Forest presented in [19]. We examine influence of different parameters such as combination of appearance channels (Section 6.2), tree depth (Section 6.3), sampling density (Section 6.4), tree density (Section 6.5) on the recognition rate and justify their effects. In Section 6.6, we also evaluate and compare the original RGB-D Object dataset [38] on turn-table against our artificially rendered training data. In Section 6.7, we show recognition results obtained with orientation uncertainty measure described in Chapter 4.

6.1 Experiment Setup

We evaluate our approach on the publicly available RGB-D Objects and Scenes Datasets [38]. The datasets contain RGB-D images of annotated objects in 6 classes namely bowl, coffee mug, cap, cereal box, flashlight and soda can. In the RGB-D Objects Dataset, the objects are placed on a turn-table and viewed in 3 pitch angles (30°, 45°, 60°) and approx. 10° steps in yaw. The same object instances have been placed in scene imagery for the Scenes Dataset. It comprises video sequences of common indoor environments, including office workspace, kitchens, and meeting rooms.

[1] Note that our results are not comparable to [2] since we do not evaluate on the turn-table scenes.

Our training settings are as follows (unless specified otherwise): Every class-specific Hough forest consists of 5 decision trees. For each tree we randomly choose 250 images of the object class and 250 images of background. The background images are mixed from rendered scenes (Chapter 5) and real scene imagery from the datasets that do not contain the objects to be classified. We choose 1000 and 1000 random pixels from each image, respectively. At every node, 2000 tests are generated, while the trees are trained up to a maximum depth of 20.

During detection we discretize scale range of 0 to 2 into 10 bins for the 3D location Hough space. For 4D orientation Hough space parametrized in quaternions we divide each dimension ranging from -1 to 1 into 50 discrete steps.

We count detections with confidence above set threshold as true positives, if its bounding box overlaps by at least 50% with the ground truth. Each ground truth bounding box (provided with dataset [38]) may only be associated once with a detection. For our 3D approach, we determine an enclosing 2D bounding box on the projected corner points of the found 3D bounding box. We compute precision recall and accuracy for range of thresholds 0 to 10 with step size 0.1.

For evaluation of orientation estimate, we manually annotated upward orientation of all the objects in RGB-D Scene dataset by estimating the table plane normal in through table plane segmentation.

6.2 Appearance Channels

We examine performance of various combination of appearance channels. We evaluate precision and recall for following combinations:

- color[2] + depth (Fig 6.1)
- color + depth + surfel (Fig 6.2)
- color + depth + surfel + HoG (Fig 6.3)
- color + HoG[3] [19]

[2]Note: color includes raw color channels in *Lab color space* and first and second derivative of intensity channel in x and y direction

[3]To generate a fair comparison with the method in [19], we augmented their approach to suppress local sub-maxima within the bounding boxes of strongest maxima detections.

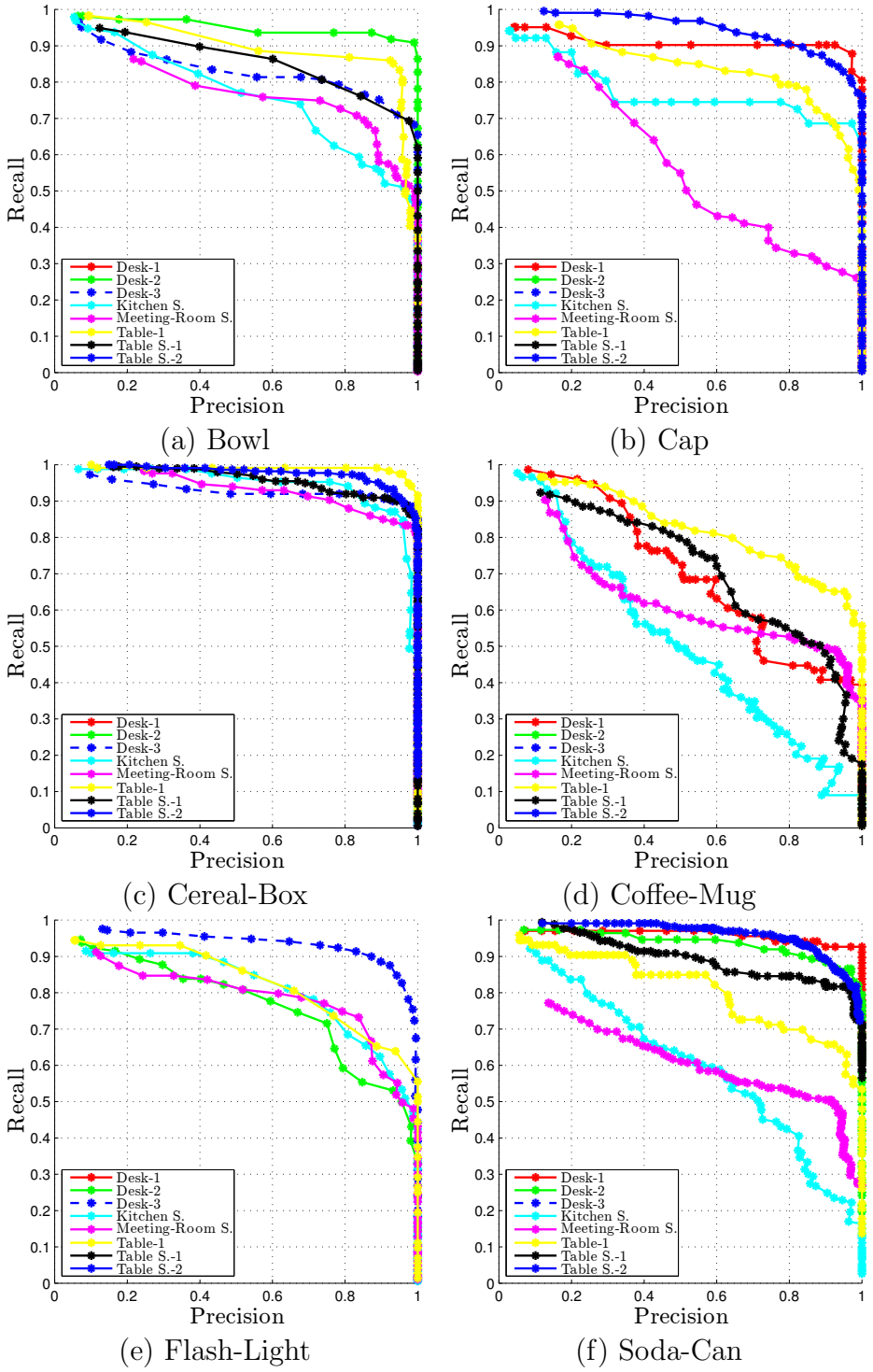


FIGURE 6.1: Precision recall curves for channel combination : color + depth, for all the objects: (a) Bowl, (b) Cap, (c) Cereal-Box, (d) Coffee-Mug, (e) Flash-Light and (f) Soda-Can, computed for all the eight *RGB-D Scene Datasets* [38].

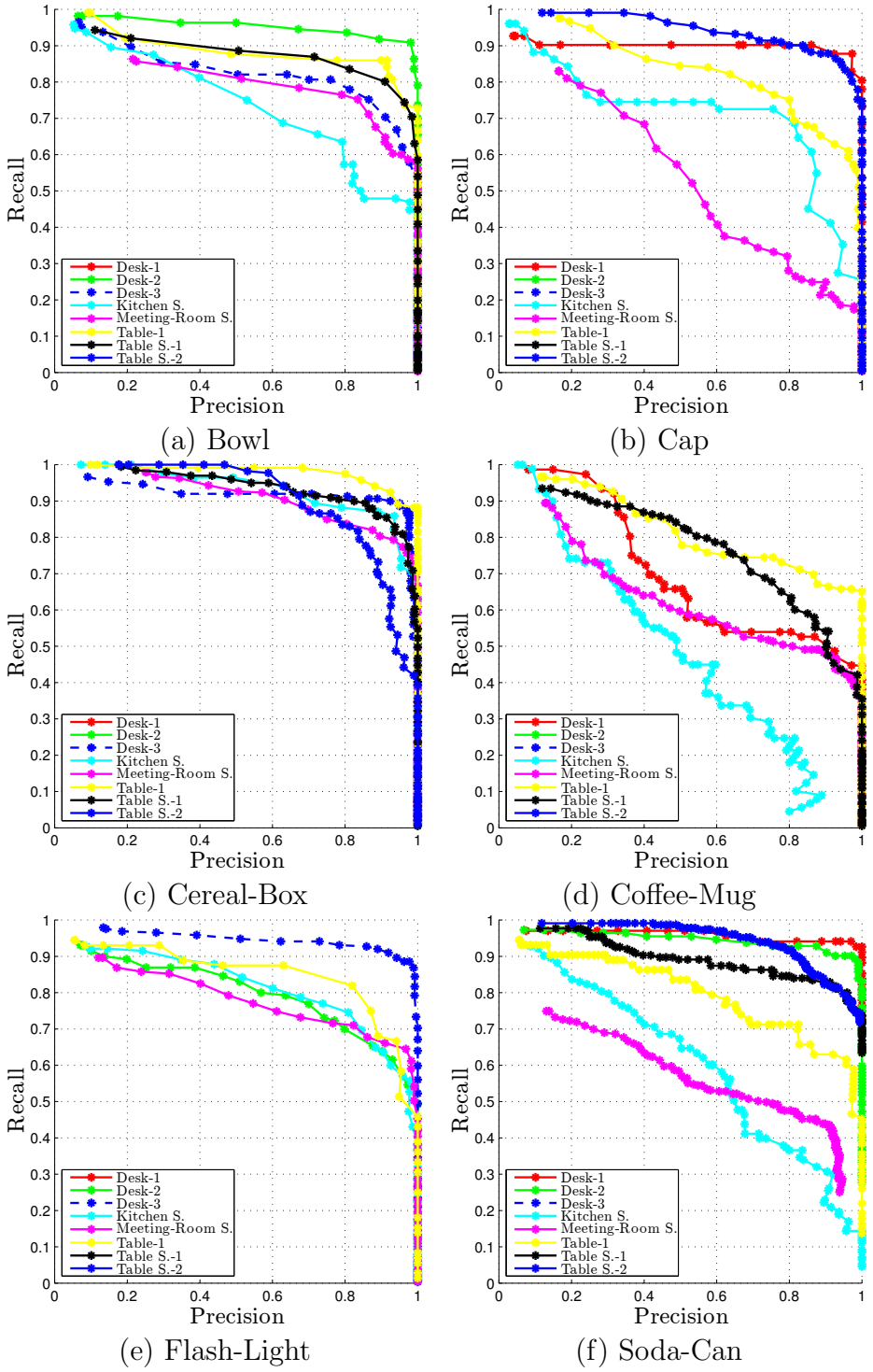


FIGURE 6.2: Precision recall curves for channel combination : color +depth +surfel, for all the objects: (a) Bowl, (b) Cap, (c) Cereal-Box, (d) Coffee-Mug, (e) Flash-Light and (f) Soda-Can, computed for all the eight *RGB-D Scene Datasets* [38].

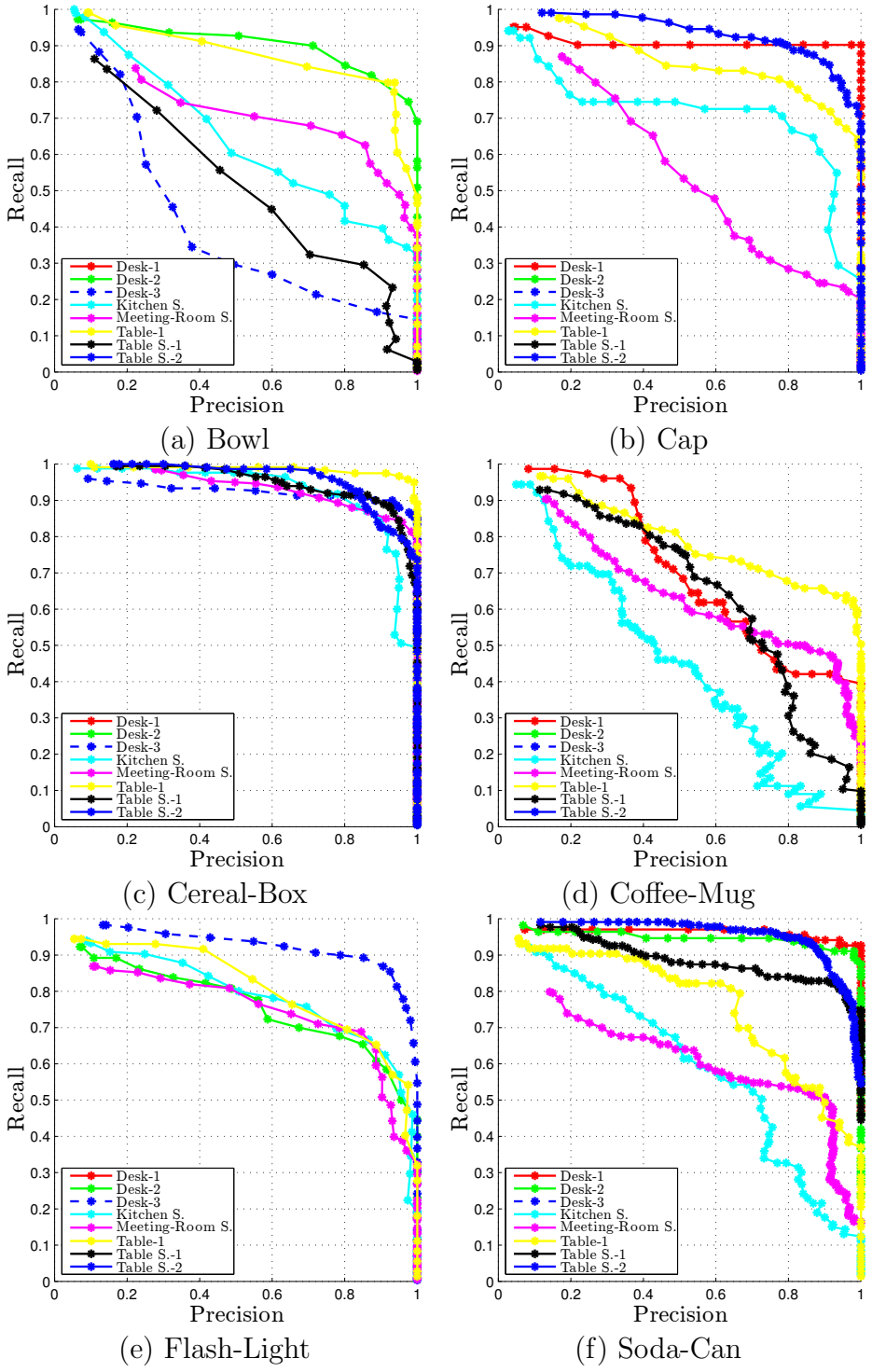


FIGURE 6.3: Precision recall curves for channel combination : color + depth + surfel +HoG, for all the objects: (a) Bowl, (b) Cap, (c) Cereal-Box, (d) Coffee-Mug, (e) Flash-Light and (f) Soda-Can, computed for all the eight *RGB-D Scene Datasets* [38].

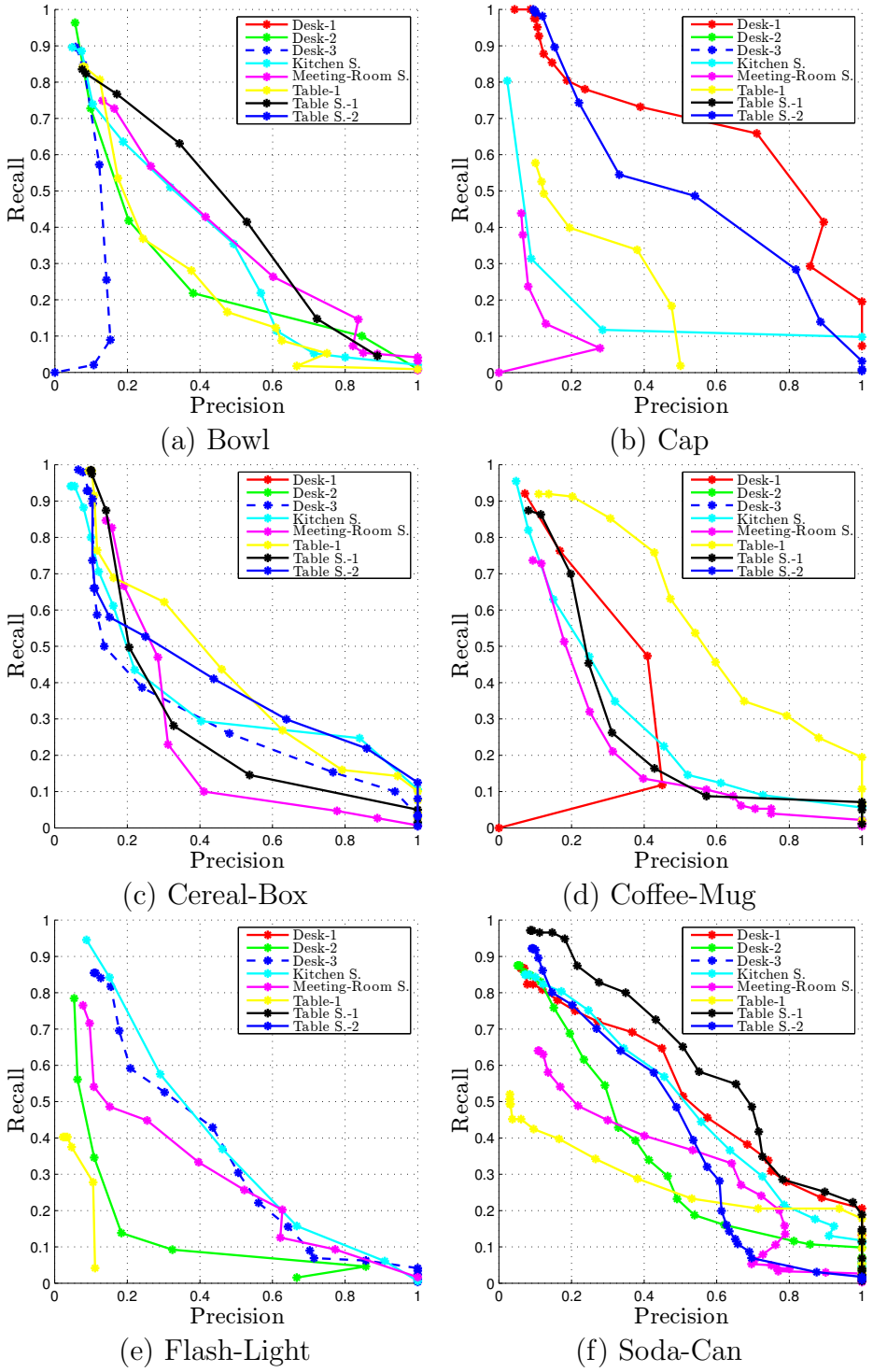


FIGURE 6.4: Precision recall curves for channel combination : color + HoG, for all the objects: (a) Bowl, (b) Cap, (c) Cereal-Box, (d) Coffee-Mug, (e) Flash-Light and (f) Soda-Can, computed for all the eight *RGB-D Scene Datasets* [38].

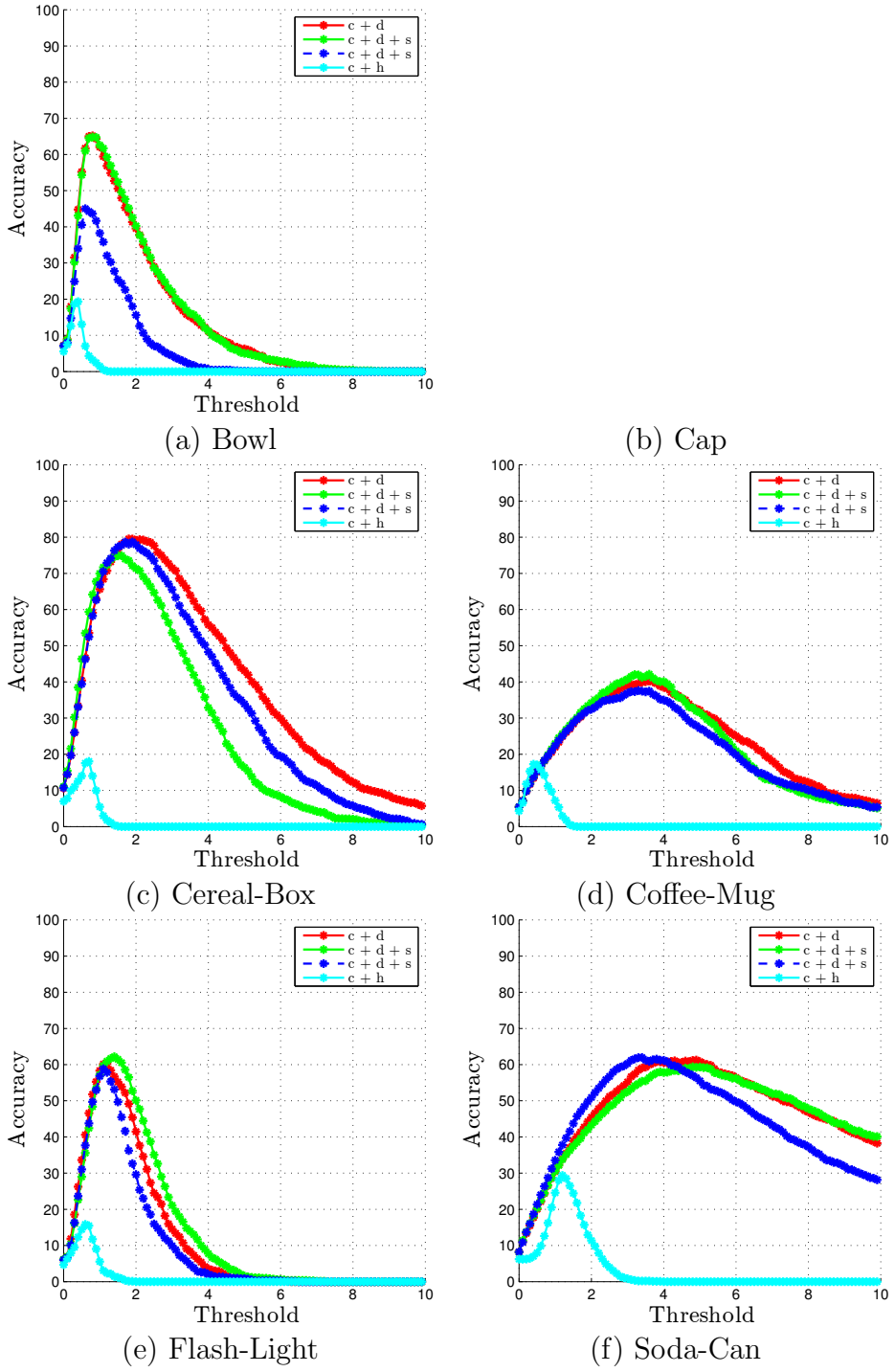


FIGURE 6.5: Detection performance comparison among appearance channel combinations for each object: (a) Bowl, (b) Cap, (c) Cereal-Box, (d) Coffee-Mug, (e) Flash-Light and (f) Soda-Can, computed for all the eight *RGB-D Scene Datasets* [38]. Where c:color, d:depth, s:surfel, h:HoG.

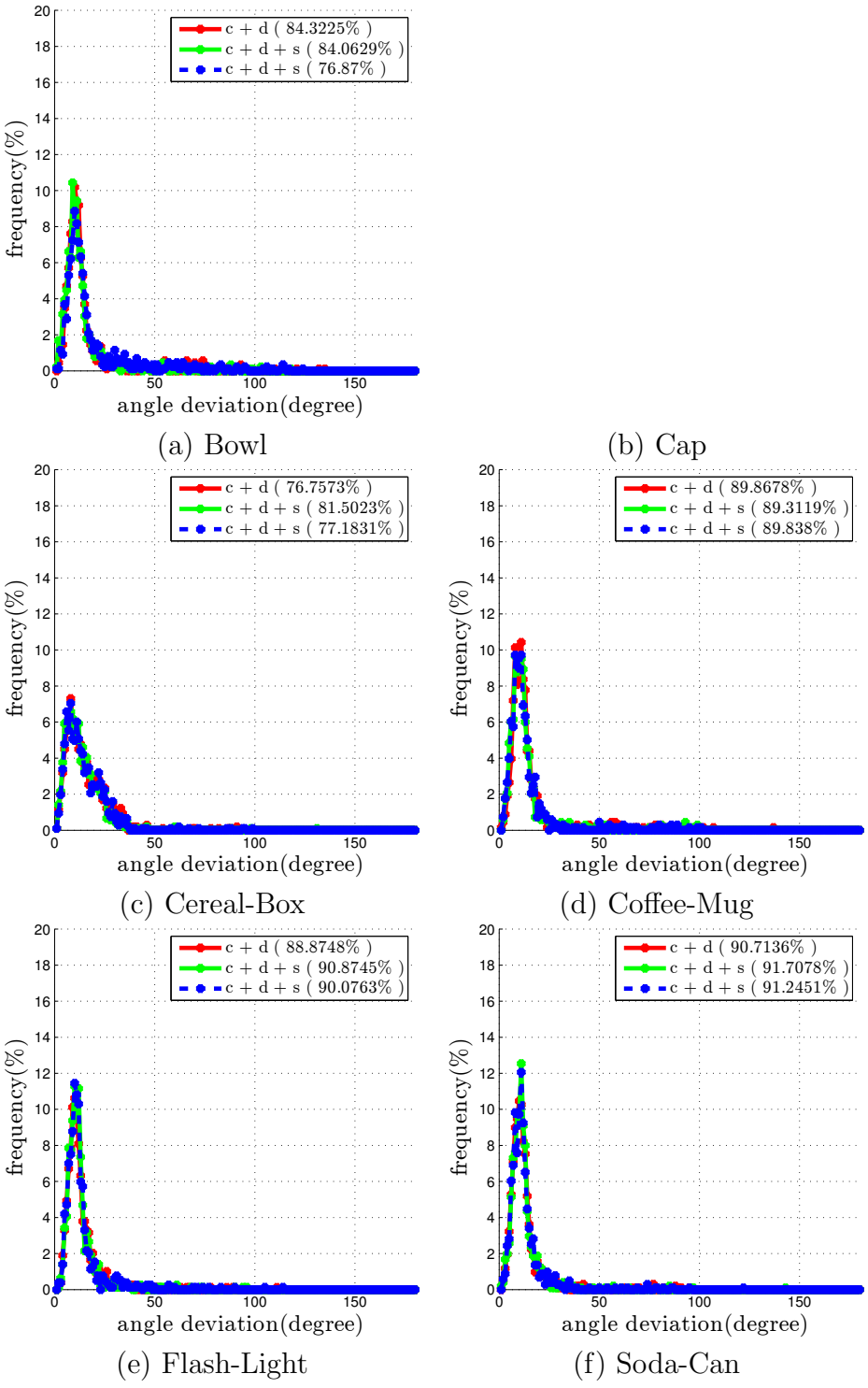


FIGURE 6.6: Comparison of orientation estimate for channel combinations for all objects (a) Bowl, (b) Cap, (c) Cereal-Box, (d) Coffee-Mug, (e) Flash-Light and (f) Soda-Can, computed for all the eight *RGB-D Scene Datasets* [38]. Percentage of observations fall within 20° is shown in bracket.

In Table 6.1, the average accuracy at equal precision/recall error rate (EER) over all sequences is tabulated for different channel combinations and compared to the method in [19]. We observe that our method outperforms pure RGB based object recognition as in [19] with significant margin (Fig 6.5 , Fig 6.6). The depth channel introduces essential information about object shape. It allows spatial information to be represented in 3D, which reduces the smearing of votes in Hough space and increases the overall recognition rate. Among all the objects we used for testing, performance of *coffee-mug* is lowest mainly due to its high shape resemblance with other object categories such as soda-can and bowl. The size of the objects also influences detection rate as for large objects, detection is achievable even at further distances, e.g. *cereal-box*.

TABLE 6.1: Average accuracy at EER for different channel combinations. See text for details.

Accuracy				
category	Appearance Channel Combinations			
	color+depth (%)	color+depth+surfel (%)	color+depth +surfel+HoG (%)	color+HoG [19] (%)
bowl	66	**68**	49	21
cap	**64**	62	64	8
cereal box	**83**	77	80	20
coffee mug	44	**45**	43	17
flashlight	64	**66**	60	16
soda can	65	65	**66**	27

In Table 6.2[4] the mean error and standard deviation for orientation estimate is tabulated. Our method provides good estimates of object orientation with an average mean error (for color + depth + surfel channel combination) of ca. 14°. Naturally, objects with spherical shapes, such as the caps or bowls, yield higher angular deviation.

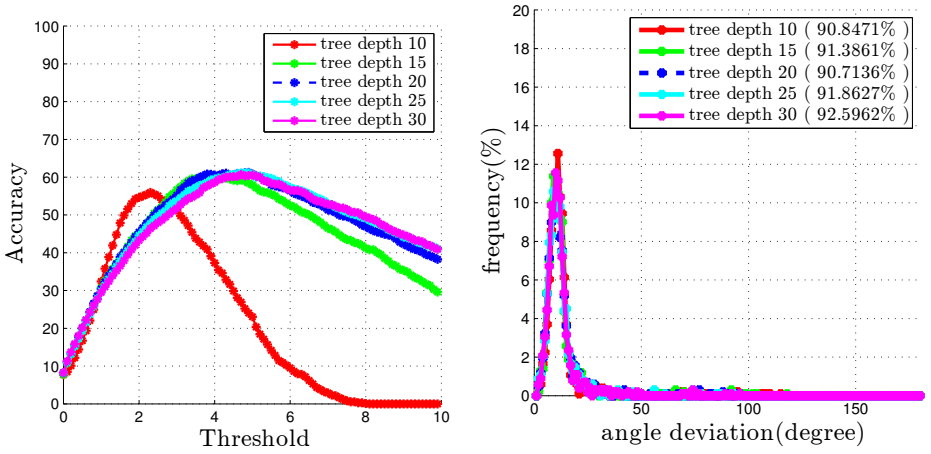
6.3 Tree Depth

We examine the effect of depth of trees in the forest over recognition rate. Other parameters are as follows

[4]For *Gall's* method orientation estimate is not possible.

TABLE 6.2: Mean and standard deviation of error in orientation estimate. See text for details.

Angle deviation			
category	Appearance Channel Combinations		
	color+depth ($\mu \pm \sigma$)	color+depth+surfel ($\mu \pm \sigma$)	color+depth +surfel+HoG ($\mu \pm \sigma$)
bowl	15.98 ± 18.55	14.51 ± 15.88	18.43 ± 19.92
cap	17.86 ± 17.15	18.82 ± 17.16	18.54 ± 17.0
cereal box	13.91 ± 10.34	12.78 ± 9.71	13.60 ± 10.17
coffee mug	13.22 ± 13.79	12.88 ± 14.13	12.73 ± 13.34
flashlight	12.35 ± 9.86	12.02 ± 9.78	12.31 ± 10.70
soda can	12.0 ± 11.21	11.60 ± 10.05	11.60 ± 9.92

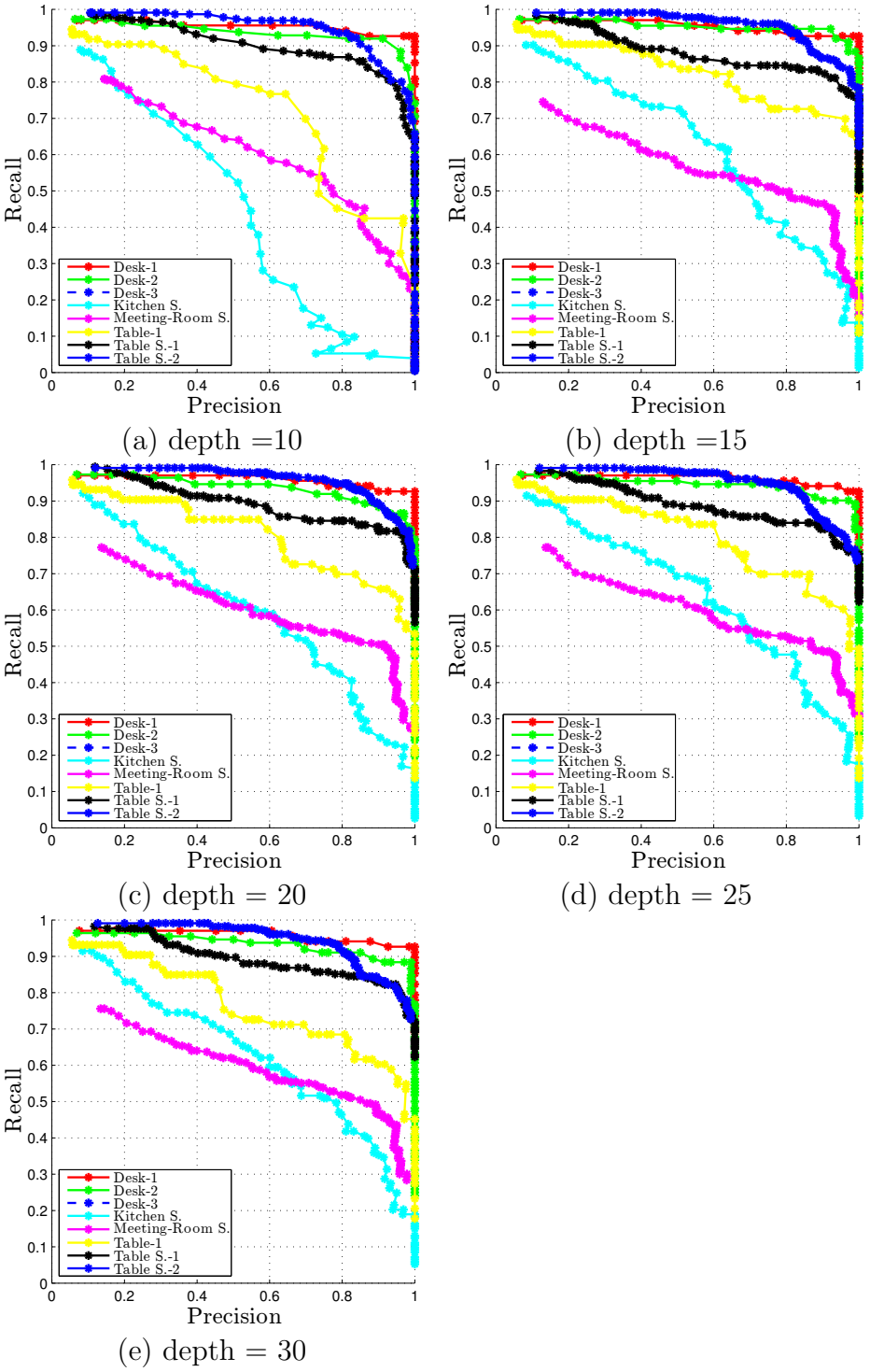


(a) accuracy (b) orientations

FIGURE 6.7: Performance comparison for tree depth for object *soda-can* (a) accuracy , (b) angle deviation histogram. Percentage of observations fall within 20° is shown in bracket.

- channel combination: color + depth + surfel,
- number of trees in the forest : 5,
- samples per training image : 1000

We observe that with increment in tree depth, accuracy in detection and orientation estimate increases (Fig. 6.8, 6.7). Note that with sparse sampling in training pixels, leaves are generated at lower depth and performance gain will be saturated at lower tree depth.



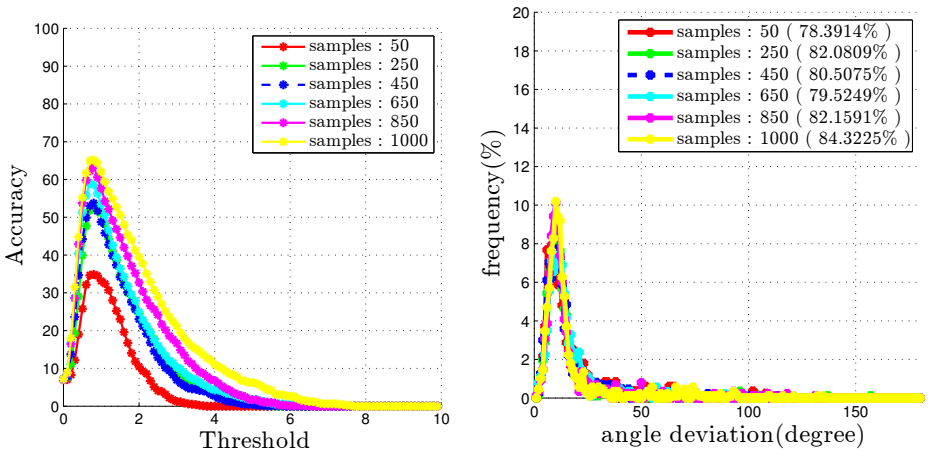
(a) depth =10 (b) depth =15

(c) depth = 20 (d) depth = 25

(e) depth = 30

FIGURE 6.8: Precision recall curves for the object *soda-can* at various tree depths: (a) depth = 10, (b) depth = 15, (c) depth = 20, (d) depth = 25, and (e) depth = 30. computed for all the eight *RGB-D Scene Datasets* [38]. All plots are computed for color + depth + surfel channel combination

6.4 Sample Density



(a) accuracy (b) orientations

FIGURE 6.9: Performance comparison for sample density for object *bowl* (a) accuracy, (b) angle deviation histogram. Percentage of observations fall within 20° is shown in bracket.

We also investigate training sample density parameter. Other parameters are kept constant as follows:

- channel combination: color + depth + surfel,
- number of trees in the forest : 5,
- depth of tree : 20

Results show that with higher sampling density, performance increases (Fig. 6.10, Fig. 6.9). Naturally, with higher number of samples more variety of visual and shape information of the object category can be captured. Although for every object category, the object instances we used for training occur in RGB-D scene data set, hence there is a possibility of inducing over fitting in the forest by increasing the sample density.

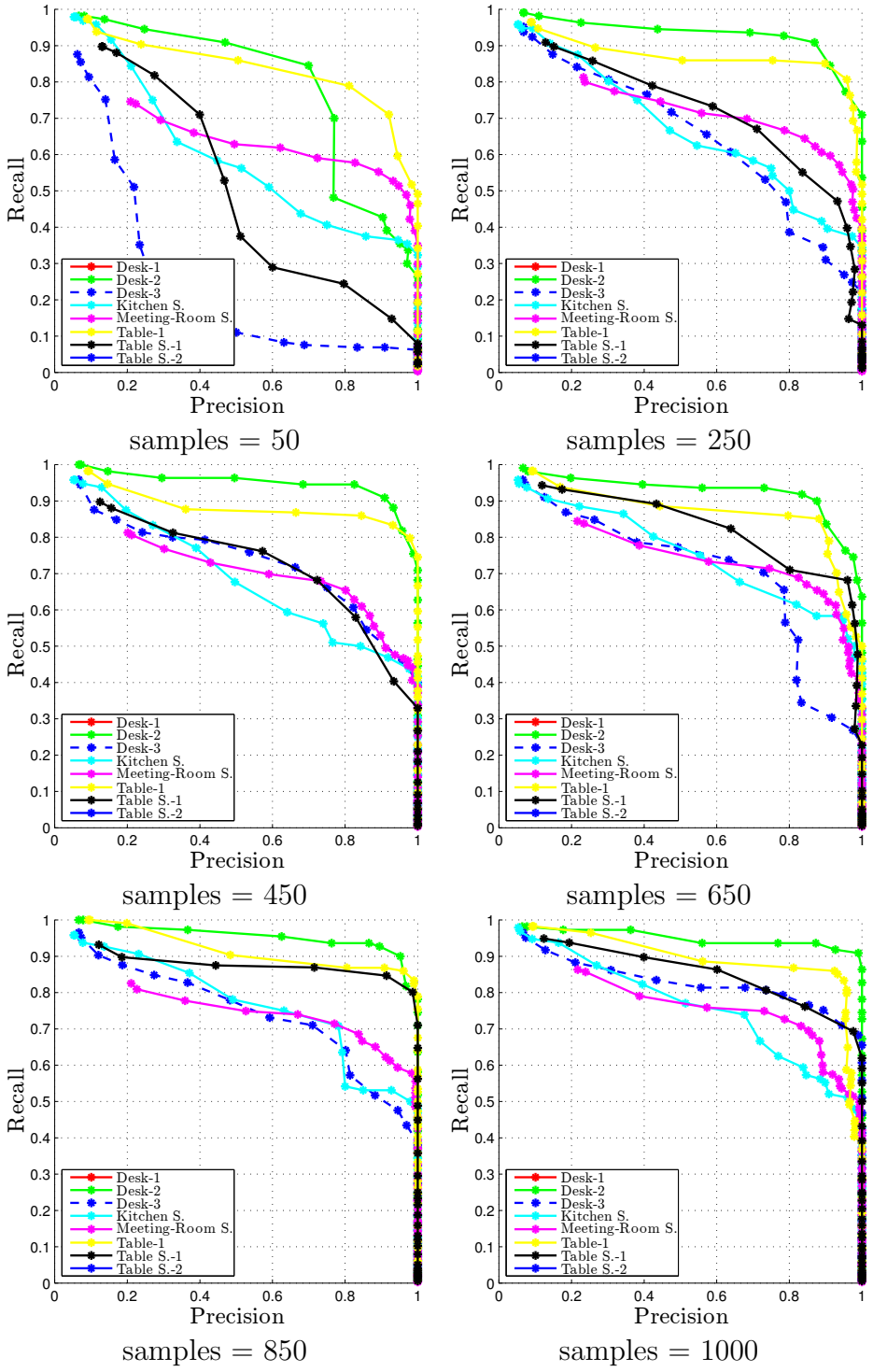
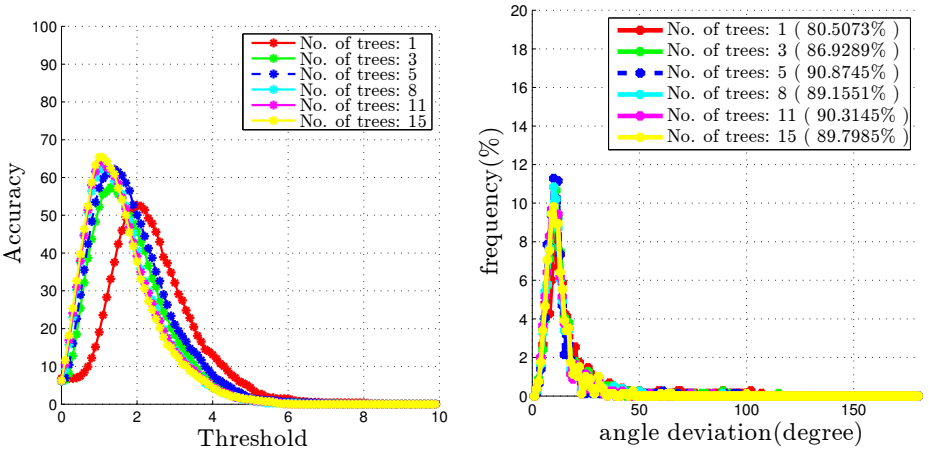


FIGURE 6.10: Precision recall curves for the object *bowl* for different pixel density per image: (a) samples = 50, (b) samples = 250, (c)samples = 450, (d) samples = 650, (e) samples = 850 and (f) samples = 1000, computed for all the eight *RGB-D Scene Datasets* [38]. All plots are computed for color + depth + surfel channel combination

6.5 Tree Density



(a) accuracy (b) orientations

FIGURE 6.11: Performance comparison for tree density for object *flashlight* (a) accuracy , (b) angle deviation histogram. Percentage of observations fall within 20° is shown in bracket.

We observe the performance change with tree density in forest. During this test other parameters, as mentioned below, are kept unchanged.

- channel combination: color + depth + surfel,
- depth of tree : 20,
- samples per training image : 1000

We find that assembling several trees, each trained with a random set of training data, achieves superior results as compared to single deterministic decision tree (Fig 6.12, 6.11). During training, randomization is introduced by selection of random training subset and random set of binary test at non-leaf nodes for each tree in the forest. Through which high generalization and stability against noise are achieved.

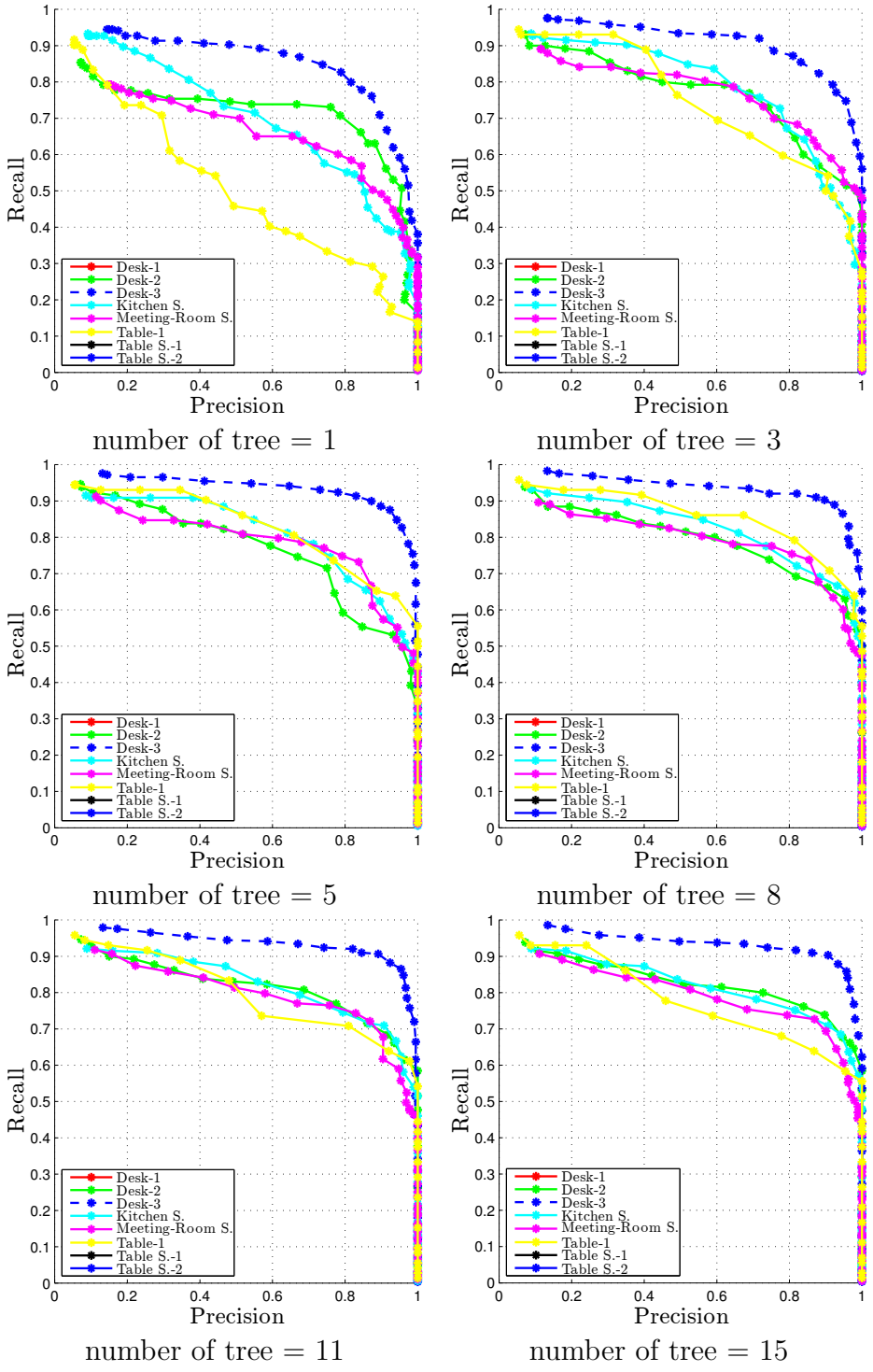


FIGURE 6.12: Precision recall curves for the object *flashlight* for different pixel density per image: (a) number of tree = 1, (b) number of tree = 3, (c) number of tree = 5 (d) number of tree = 8, (e) number of tree = 11 and (f) number of tree = 15, computed for all the eight *RGB-D Scene Datasets* [38]. All plots are computed for color + depth + surfel channel combination

6.6 RGB-D Turn-Table Dataset

We discussed in Chapter 5 regarding the short comings of the turn-table training dataset. To justify that, we train our system with original RGB-D Object dataset [38] and compare the results with Hough forest trained over artificially rendered training images proposed in Chapter 5. Results show that we achieve considerable performance boost on our artificial training dataset carefully rendered from [38]. Figure 6.13 shows precision-recall curve for each object. Figure 6.14 and Figure 6.15 shows accuracy and orientation performance comparison between original turn-table dataset and rendered image dataset.

6.7 Orientation Uncertainty Measure

Finally, we analyze the response of an additional orientation uncertainty measure included during training. Results show that overall recognition performance decreases with this measure (Fig 6.16, Fig 6.18, Fig 6.17). The possible reason behind it is, local orientation and position of the object point are not independent from each other but are constrained according to the shape of the object.

6.8 Summary

We discussed our recognition results with change in variety of parameters. We showed that our method out performs results with *Gall's* method [19] by considerably higher margin. We also justify the generation of rendered training image dataset by showing superior performance of the classifier trained with it as compared to the one train with original turn-table dataset.

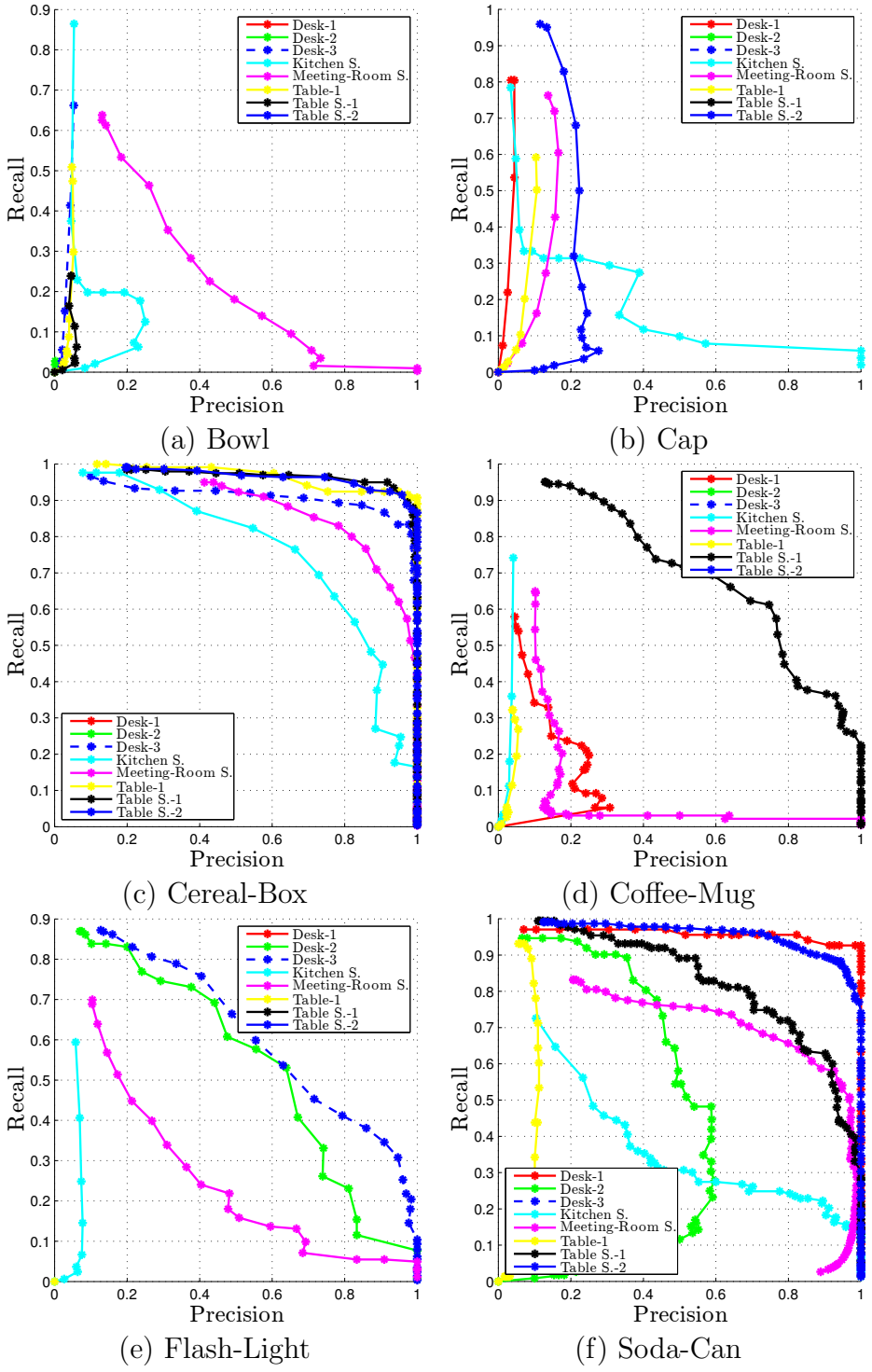


FIGURE 6.13: Precision-Recall curves for all the objects trained by turn-table training dataset: (a) Bowl, (b) Cap, (c) Cereal-Box, (d) Coffee-Mug, (e) Flash-Light and (f) Soda-Can, computed for all the eight *RGB-D Scene Datasets* [38]. All plots are computed for color + depth + surfel channel combination

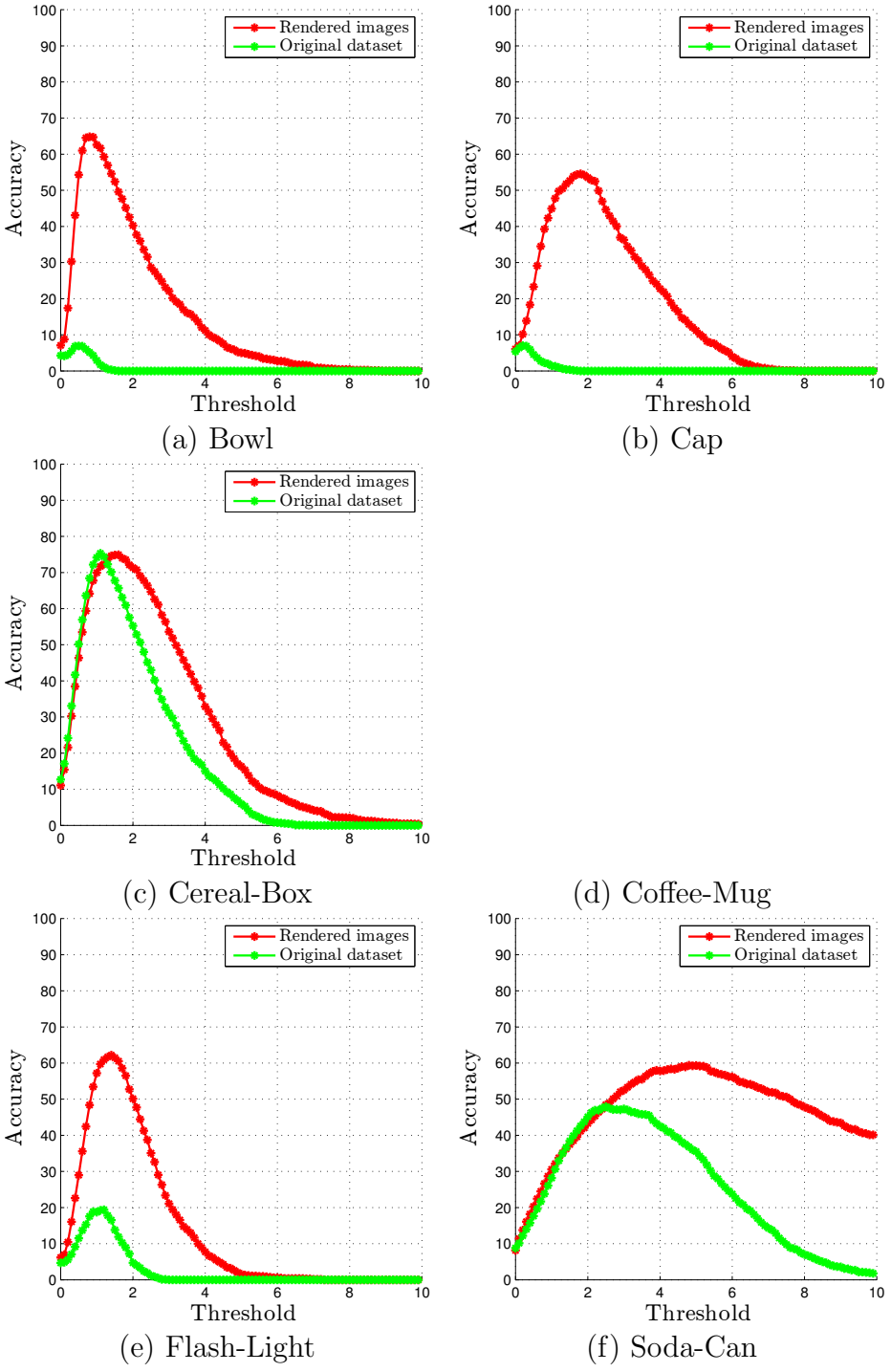


FIGURE 6.14: Detection performance comparison between classifiers trained by rendered and original turn-table training images for the objects: (a) Bowl, (b) Cap, (c) Cereal-Box, (d) Coffee-Mug, (e) Flash-Light and (f) Soda-Can,computed for all the eight *RGB-D Scene Datasets* [38]. All plots are computed for color + depth + surfel channel combination.

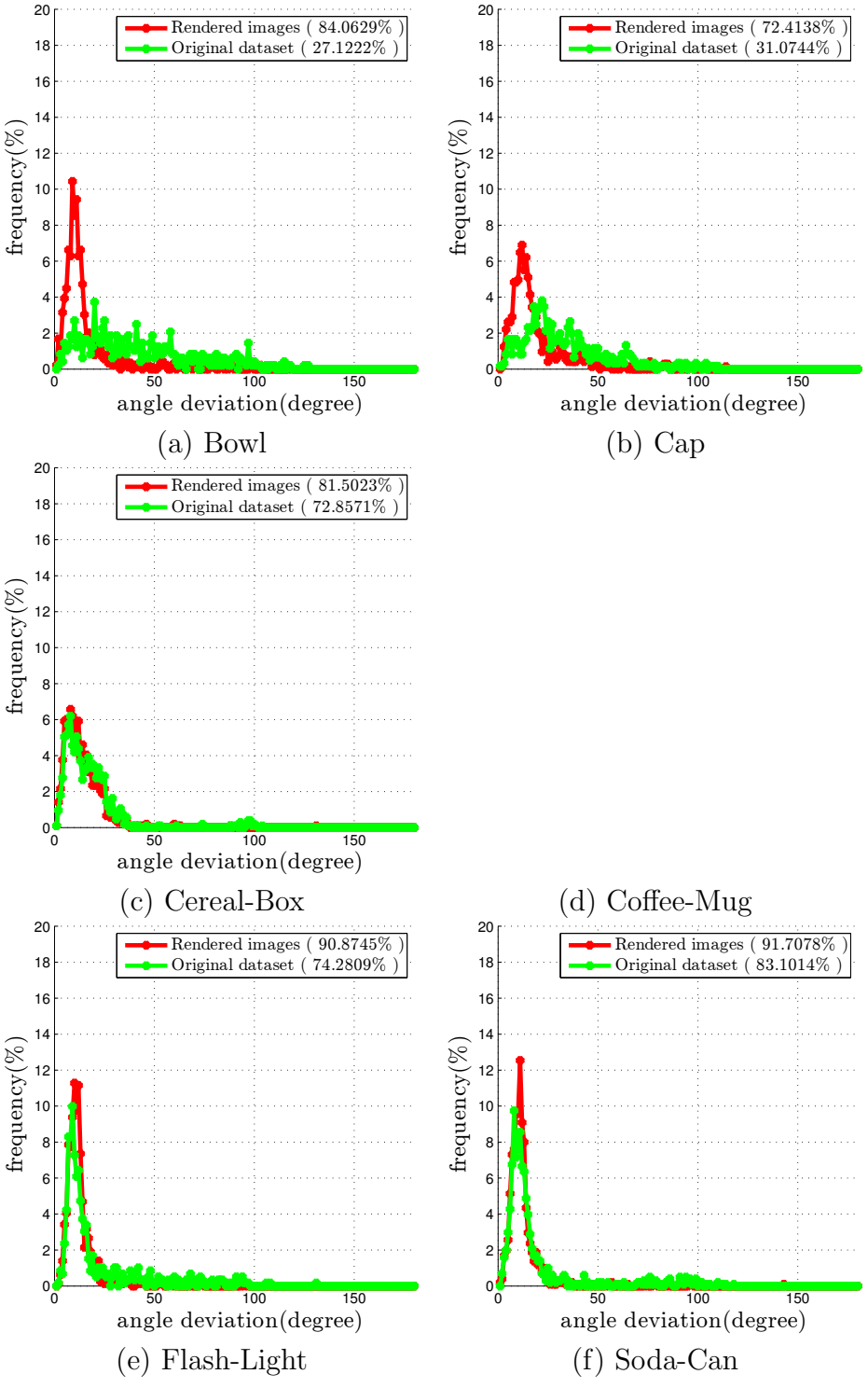


FIGURE 6.15: Comparison of orientation estimate between classifiers trained by rendered and original turn-table training images for all objects (a) Bowl, (b) Cap, (c) Cereal-Box, (d) Coffee-Mug, (e) Flash-Light and (f) Soda-Can, computed for all the eight *RGB-D Scene Datasets* [38]. Percentage of observations fall within 20° is shown in bracket.

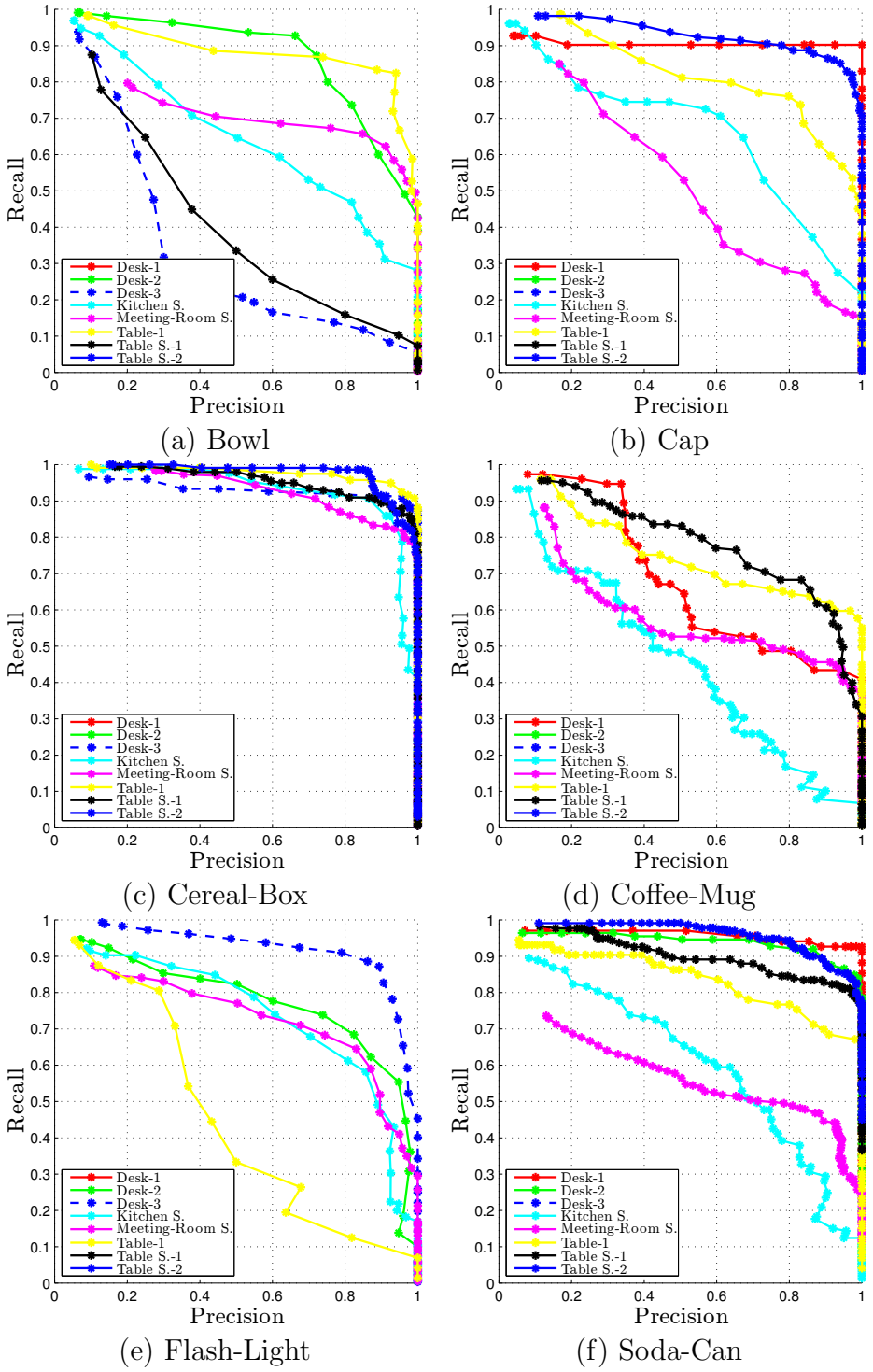


FIGURE 6.16: Precision-Recall curves for all the objects trained with orientation uncertainty measure for all the objects: (a) Bowl, (b) Cap, (c) Cereal-Box, (d) Coffee-Mug, (e) Flash-Light and (f) Soda-Can, computed for all the eight *RGB-D Scene Datasets* [38]. All plots are computed for color + depth + surfel channel combination

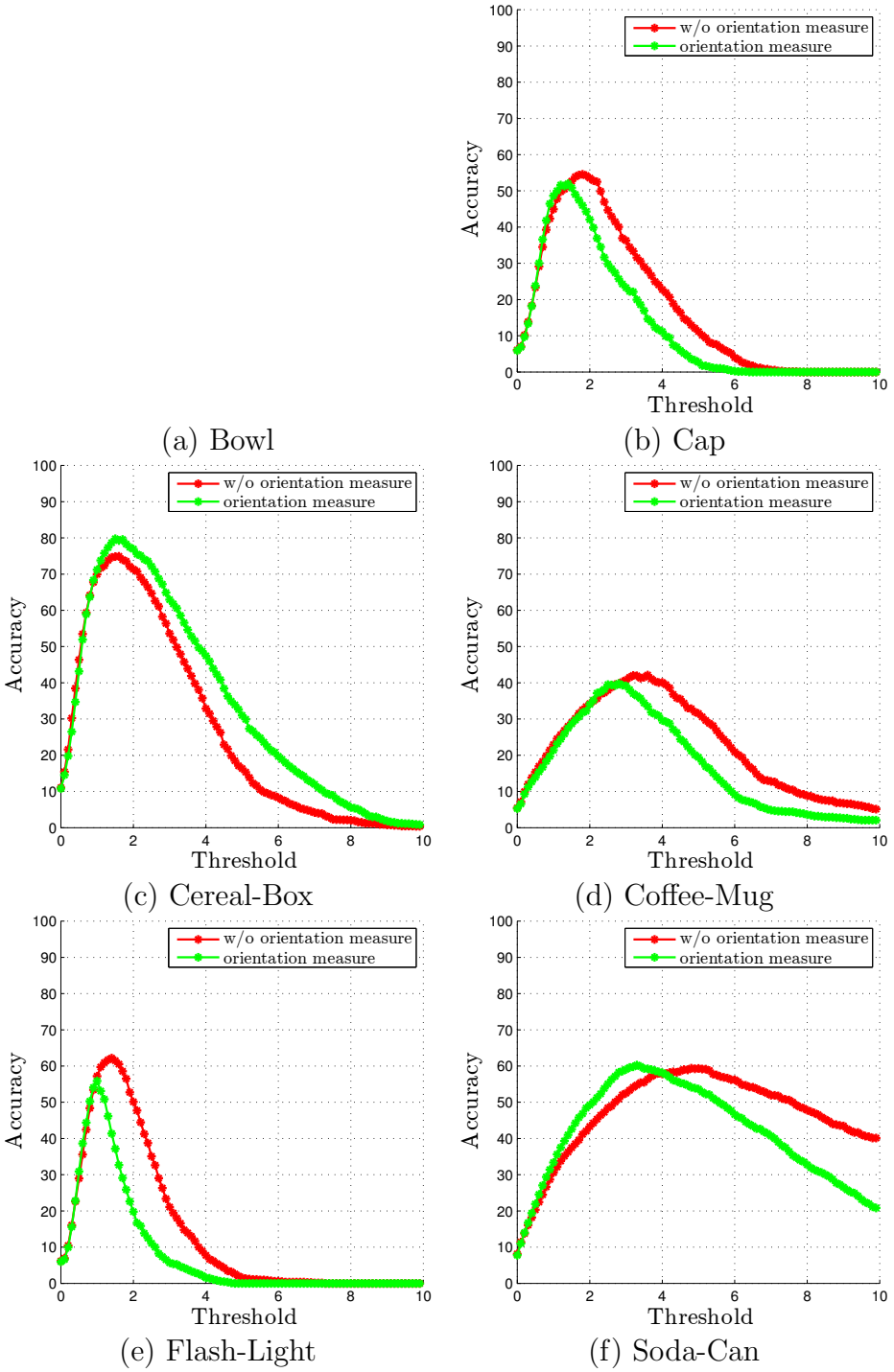


FIGURE 6.17: Detection performance comparison between classifiers trained with and without orientation uncertainty measure for all the objects: (a) Bowl, (b) Cap, (c) Cereal-Box, (d) Coffee-Mug, (e) Flash-Light and (f) Soda-Can,computed for all the eight *RGB-D Scene Datasets* [38]. All plots are computed for color + depth + surfel channel combination.

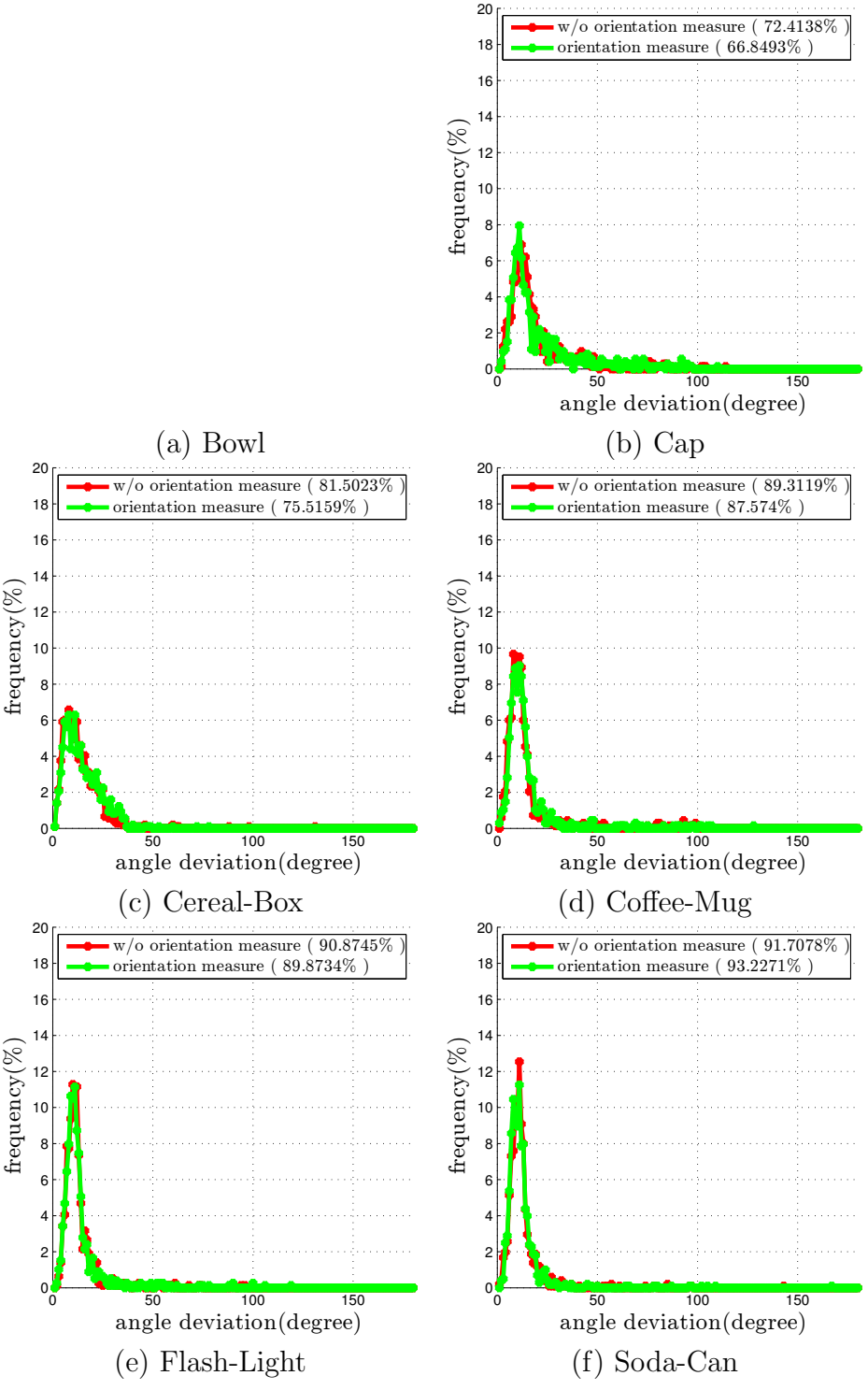


FIGURE 6.18: Comparison of orientation estimate between classifiers trained with and without orientation uncertainty measure for all objects (a) Bowl, (b) Cap, (c) Cereal-Box, (d) Coffee-Mug, (e) Flash-Light and (f) Soda-Can, computed for all the eight *RGB-D Scene Datasets* [38]. Percentage of observations fall within 20° is shown in bracket.

Chapter 7

Conclusion

We proposed a novel approach to object-class detection and a continuous canonical pose estimation from RGB-D images. We discriminatively train random decision forests to classify pixels and to vote for 3D object location and orientation.

During training we store class distribution, relative positions and relative orientations in leaf nodes. We have shown a memory efficient way to store the large amount of orientation votes. The features used in the binary node tests are made scale-invariant through depth-normalization. We furthermore use depth cues and surfel point-pair feature cues to make use of the geometry information contained in the RGB-D images for the detection.

We have illustrated two pass detection procedure by separating the location and orientation Hough space. This way first we achieved location hypothesis through dense voting in 3D position Hough space and then by taking support from each hypothesis. 3D orientation votes are cast from local reference frames that are created from local surface normals and 3D point configurations.

We proposed a simple but effective way of generating rendered images. We extracted object views from the acquired turn-table data and rendered new training scenes with varying background, clutter, lighting changes, and occlusions.

Experiments demonstrate that our approach yields good accuracy in detecting objects and recovering their canonical pose. It compares well with a state-of-the-art approach to object-class detection that only utilizes RGB information in the decision cascade.

In future work, we will evaluate our approach for scalable multi-class detection that detects classes in a taxonomy. For scalable training on large datasets or online interactive learning of the trees, we will pursue a GPU implementation for the method.

List of Figures

1.1 6DoF object class detection 1

2.1 Random Decision Tree 8
2.2 Implicit shape model 9

3.1 Training Data: Image Patche Samples 11
3.2 Image Appearance Channels 13
3.3 Probability map 17
3.4 Hough space 3D scale-space frustum 18
3.5 Bounding Box Construction Using Back Projection 19

4.1 Memory efficient storage of orientation votes 23
4.2 Full 6-DoF Object Detection 24
4.3 Depth normalized pixel-pair features 27
4.4 Surfel Pair Feature 28

5.1 Training images captured on turn-table 30
5.2 Image rendering pipeline 31
5.3 Rendered positive training images 32
5.4 Rendered negative training images 33
5.5 Object segmentation and 3D model 34
5.6 Smereaning of votes in Hough space 35
5.7 Depth Filling 35

6.1 Precision-Recall curves for appearance channel combination : color + depth 38
6.2 Precision-Recall curves for appearance channel combination : color + depth + surfel 39
6.3 Precision-Recall curves for appearance channel combination : color + depth + surfel + HoG 40
6.4 Precision-Recall curves for appearance channel combination : color + HoG 41
6.5 Detection performance comparison among appearance channel combinations 42
6.6 Comparison of orientation estimate for channel combinations 43
6.7 Performance comparison for tree depth for object *soda-can* 45

6.8 Precision recall curves for the object *soda-can* against change in tree depths . . . 46
6.9 Performance comparison for sample density for object *bowl* . . . 47
6.10 Precision recall curves for the object *bowl* for different pixel density per image: (a) samples = 50, (b) samples = 250, (c)samples = 450, (d) samples = 650, (e) samples = 850 and (f) samples = 1000, computed for all the eight *RGB-D Scene Datasets* [38]. All plots are computed for color + depth + surfel channel combination . . . 48
6.11 Performance comparison for tree density for object *flashlight* . . . 49
6.12 Precision recall curves for the object *flashlight* for different pixel density per image: (a) number of tree = 1, (b) number of tree = 3, (c) number of tree = 5 (d) number of tree = 8, (e) number of tree = 11 and (f) number of tree = 15, computed for all the eight *RGB-D Scene Datasets* [38]. All plots are computed for color + depth + surfel channel combination . . . 50
6.13 Precision-Recall curves for all the objects trained by turn-table-training dataset . . . 52
6.14 Detection performance comparison between classifiers trained by rendered and original turn-table training images . . . 53
6.15 Comparison of orientation estimate between classifiers trained by rendered and original turn-table training images . . . 54
6.16 Precision-Recall curves for all the objects trained with orientation uncertainty measure . . . 55
6.17 Detection performance comparison between classifiers trained with and without orientation uncertainty measure . . . 56
6.18 Comparison of orientation estimate between classifiers trained with and without orientation uncertainty measure . . . 57

B.1 Detection of object class: Bowl. . . . 68
B.2 Detection of object class: Cap. . . . 69
B.3 Detection of object class: Cereal-Box. . . . 70
B.4 Detection of object class: Coffee-Mug. . . . 71
B.5 Detection of object class: Flashlight. . . . 72
B.6 Detection of object class: Soda-Can. . . . 73

List of Tables

6.1 Average accuracy at EER for different channel combinations. See text for details. 44
6.2 Mean and standard deviation of error in orientation estimate. See text for details. 45

B.1 RGB-D Scene dataset.. For each video sequence, number of images and categories present are tabulated. 67

Abbreviations

RF	**R**andom **F**orest
GHT	**G**eneralized **H**ough **Transform**
ISM	**I**mplicit **S**hape **M**odel

Appendix A

Derivations

A.1 Parzen-Window Density Estimation

In case of parametric modeling of PDFs, from the empirical analysis of the observed data, parameters associated with the PDF is estimated (e.g. Gaussian noise). Generally parametric models are uni-model, whereas practical situations exhibit multi-model PDFs. Parzen window-density estimation [40] is a non-parametric density estimation technique. It is widely used in the spectrum of area such as pattern recognition, object detection, tracking etc. It is basically a data-interpolation technique. For a random sample x, Parzen-windowing estimates the PDF $P(x)$ from which the sample was derived. To estimate the value of the PDF at point x, a window function is placed at x which gathers the observations fall within the window. In other words it accumulates the contribution of each observation x_i to this window. The PDF value $P(x)$ is then the weighted sum of observations to this window. The Parzen-window estimate is defined as:

$$P(x) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{w^d}K\left(\frac{x - x_i}{w^d}\right). \tag{A.1}$$

Where w^d is the width of the window in d dimensional space and

$$\int_{\mathcal{R}^d} K(x)\,dx \;= 1 \tag{A.2}$$

The popular kernel function for Parzen window estimation is Gaussian PDF. Hence the density estimate with Gaussian function is written as:

$$P(x) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{(2\pi\sigma^2)^{d/2}}\exp\left(-\frac{1}{2}\left(\frac{x-x_i}{\sigma}\right)^2\right). \tag{A.3}$$

Where $\sigma^2_{(d\times d)}$ is a d dimensional covariance of a Gaussian window.

A.2 Quternion Interpolation

Because the unit-quaternion space is a closed Riemannian manifold, the difference between any two values on the manifold (in the tangent-space of the first value) can be defined as:

where the logarithm is the hypercomplex logarithm. This difference can be applied to the value in which it is a tangent-space member as

$$q_0 exp(d_{0,1}) = q_1 \tag{A.4}$$

where the hyper-complex exponential is used. Using the above definitions, the quaternion interpolation of values q with weights w can be defined (nearly identically to the unconstrained mean) as

$$\sum_i w_i \log(m^{-1}q_i) = 0 \tag{A.5}$$

which gives the weighted sum of all differences to m (in m's tangent-space) is zero.

Recursive Formulation[1]. The quaternion mean value defined above can be found in a recursive algorithm with some initial estimate (one of the points, for example) that will halt when the net-error is below some threshold or the algorithm has iterated beyond some time limit. Each iteration of the algorithm is as follows,

[1] Source:Wikipedia

with an initial mean estimate of m_0

$$e_{k-1} = \sum_i w_i \log(m_{k-1}^{-1} q_i) = 0 m_k = m_{k-1} exp(e_{k-1}) \tag{A.6}$$

as iteration index k increases, the value m_k will approach the true weighted-mean of the points.

Appendix B

Results

This section shows the test results obtained for each RGBD-Scene dataset [38]. This dataset contains 8 video sequences of home and office environments. Following table tabulates number of images and object categories present in each of the video sequence.

TABLE B.1: RGB-D Scene dataset.. For each video sequence, number of images and categories present are tabulated.

Video sequences	No. of images	Object category					
		bowl	cap	cereal-box	coffee-mug	flashlight	soda-can
desk-1	98		✓		✓		✓
desk-2	190	✓				✓	✓
desk-3	228	✓		✓		✓	
kitchen-small-1	180	✓	✓	✓	✓	✓	✓
meeting-room-small-1	180	✓	✓	✓	✓	✓	✓
table-1	125	✓	✓	✓	✓	✓	✓
table-small-1	199	✓		✓	✓		✓
table-small-2	234		✓	✓			✓

FIGURE B.1: Detection of object class: Bowl.

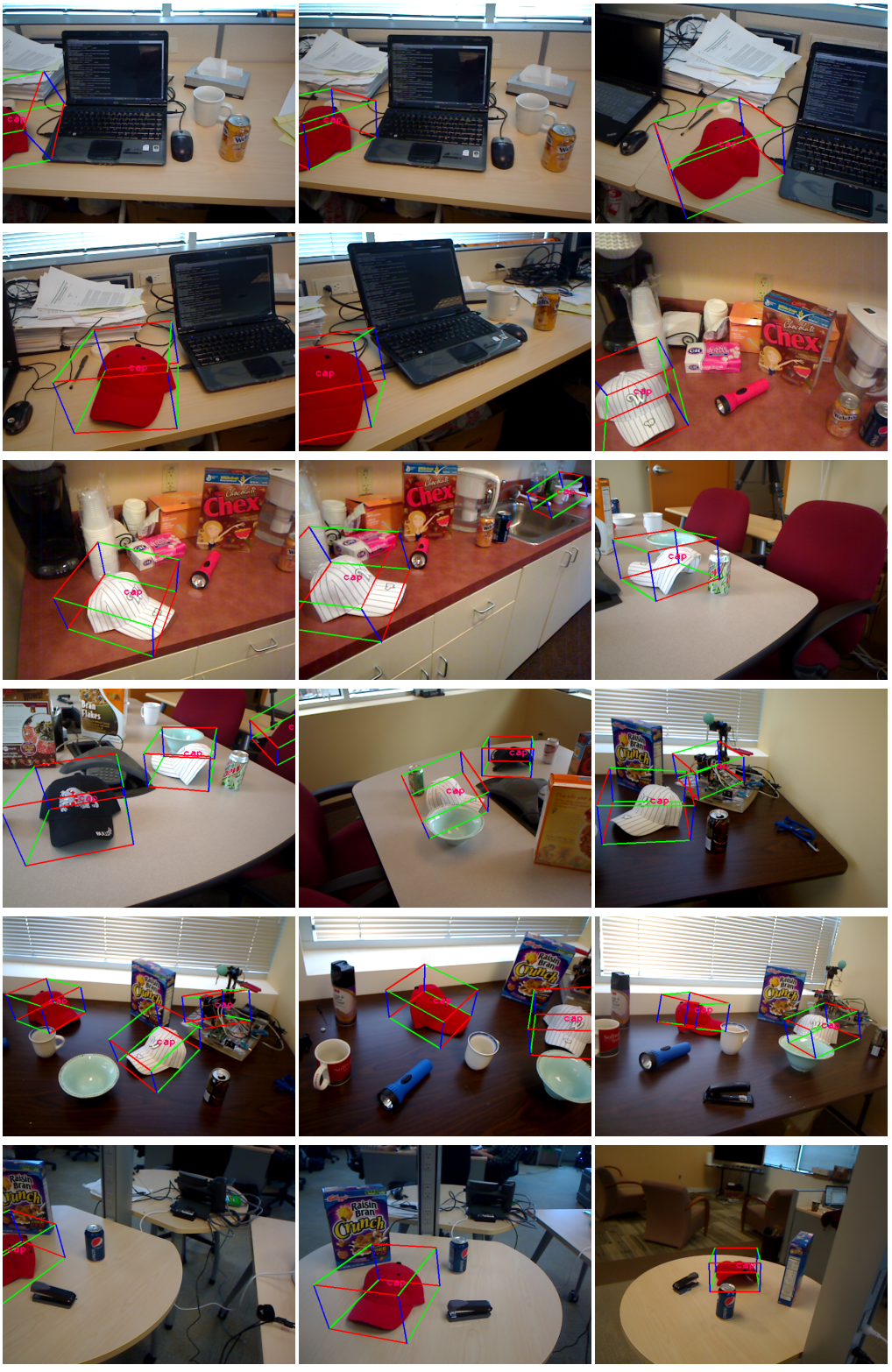


FIGURE B.2: Detection of object class: Cap.

FIGURE B.3: Detection of object class: Cereal-Box.

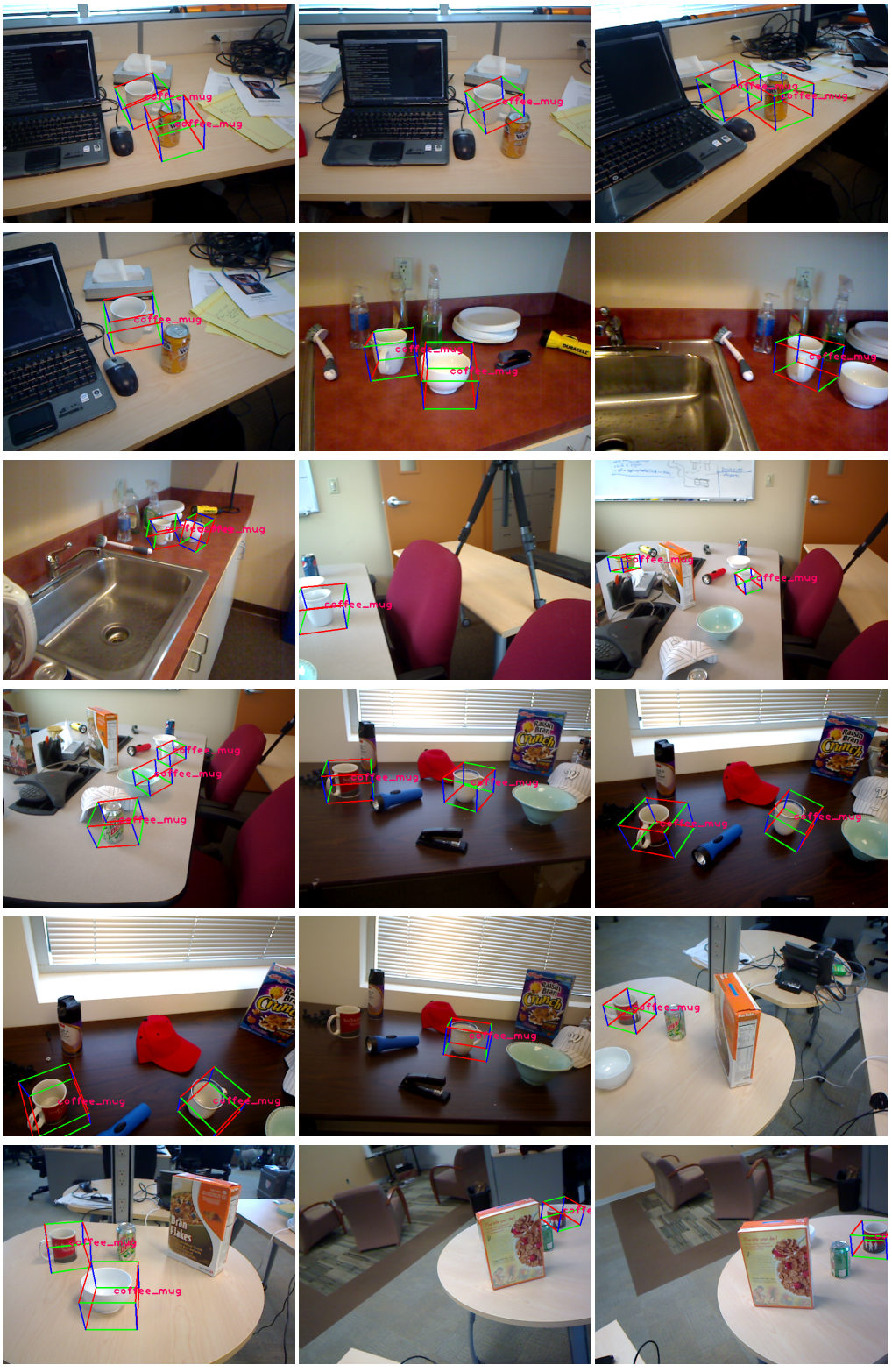


FIGURE B.4: Detection of object class: Coffee-Mug.

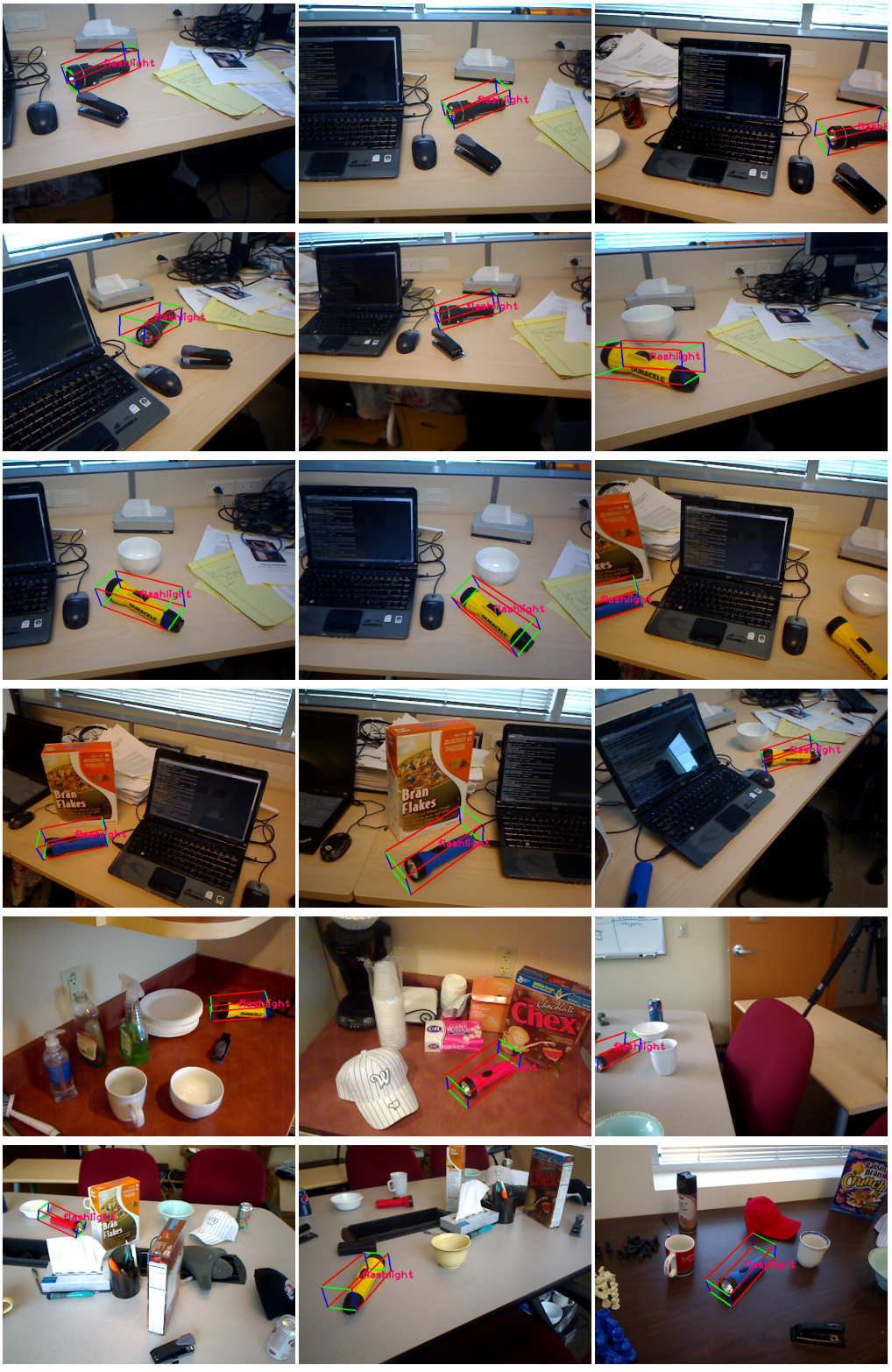


FIGURE B.5: Detection of object class: Flashlight.

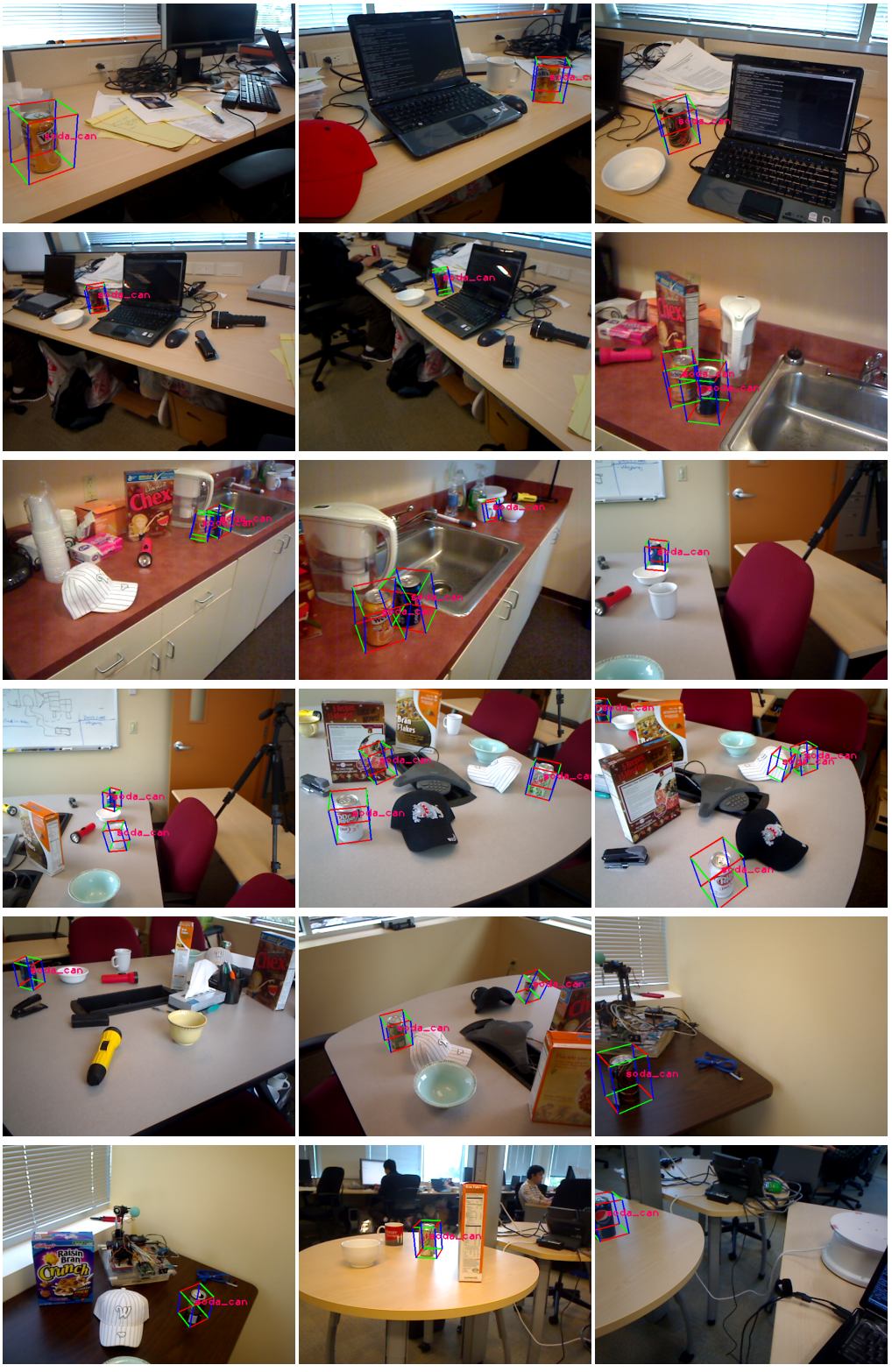


FIGURE B.6: Detection of object class: Soda-Can.

Bibliography

[1] N. Razavi, J. Gall, and L. J. V. Gool, "Scalable multi-class object detection," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1505–1512, 2011.

[2] K. Lai, L. Bo, X. Ren, and D. Fox, "A scalable tree-based approach for joint object and pose recognition," in *Twenty-Fifth Conference on Artificial Intelligence (AAAI)*, 2011.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[4] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 404–417, 2006.

[5] A. Collet, M. Martinez, and S. S. Srinivasa, "The MOPED framework: Object Recognition and Pose Estimation for Manipulation," 2011.

[6] A. Johnson, *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 1997.

[7] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *Proceedings of the 11th European Conference on Computer Vision*, pp. 356–369, 2010.

[8] E. Wahl, U. Hillenbrand, and G. Hirzinger, "Surflet-pair-relation histograms: a statistical 3d-shape representation for rapid classification," in *3-D Digital*

Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on, pp. 474 –481, oct. 2003.

[9] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, ICRA'09, (Piscataway, NJ, USA), pp. 1848–1853, IEEE Press, 2009.

[10] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Taipei, Taiwan), 2010.

[11] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

[12] C. Papazov, S. Haddadin, S. Parusel, K. Krieger, and D. Burschka, "Rigid 3D geometry matching for grasping of known objects in cluttered scenes," *I. J. Robotic Res.*, vol. 31, no. 4, pp. 538–553, 2012.

[13] E. Kim and G. G. Medioni, "3D object recognition in range images using visibility context.," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 3800–3807, 2011.

[14] C. Choi, Y. Taguchi, O. Tuzel, M.-Y. Liu, and S. Ramalingam, "Voting-based pose estimation for robotic assembly using a 3d sensor," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1724–1731, 2012.

[15] C. Choi and H. I. Christensen, "3D pose estimation of daily objects using an RGB-D camera," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 3342–3349, 2012.

[16] B. Leibe, A. Leonardis, and B. Schiele, "Combined object categorization and segmentation with an implicit shape model," in *In ECCV workshop on statistical learning in computer vision*, pp. 17–32, 2004.

[17] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[18] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, (Washington, DC, USA), pp. 886–893, IEEE Computer Society, 2005.

[19] J. Gall and V. Lempitsky, "Class-Specific Hough Forests for Object Detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.

[20] H. Su, M. Sun, L. Fei-Fei, and S. Savarese, "Learning a dense multi-view representation for detection, viewpoint classification and synthesis of object categories.," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2009.

[21] R. J. Lpez-Sastre, T. Tuytelaars, and S. Savarese, "Deformable part models revisited: A performance evaluation for object category pose estimation.," in *Proceedings of the ICCV Workshops*, pp. 1052–1059, 2011.

[22] C. Gu and X. Ren, "Discriminative mixture-of-templates for viewpoint classification," in *Proceedings of the 11th European Conference on Computer Vision (ECCV)*, pp. 408–421, 2010.

[23] B. Pepik, P. Gehler, M. Stark, and B. Schiele, "3D2PM - 3D deformable part models," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012.

[24] M. Sun, G. Bradski, B.-X. Xu, and S. Savarese, "Depth-encoded hough voting for joint object detection and shape recovery," in *Proceedings of the 11th European Conference on Computer Vision*, pp. 658–671, 2010.

[25] D. Glasner, M. Galun, S. Alpert, R. Basri, and G. Shakhnarovich, "Viewpoint-aware object detection and continuous pose estimation," *Image and Vision Computing*, 2012.

[26] M. Zia, M. Stark, B. Schiele, and K. Schindler, "Revisiting 3d geometric models for accurate object shape and pose," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 569–576.

[27] J. Liebelt, C. Schmid, and K. Schertler, "Viewpoint-independent object class detection using 3d feature maps," in *Proceedings of the International Conference on Computer Vision & Pattern Recognition (CVPR)*, 2008.

[28] T. Wang, X. He, and N. Barnes, "Learning hough forest with depth-encoded context for object detection," in *Digital Image Computing Techniques and Applications (DICTA), 2012 International Conference on*, pp. 1–8.

[29] B. Leibe, A. Leonardis, and B. Schiele, "Combined object categorization and segmentation with an implicit shape model," in *In ECCV workshop on statistical learning in computer vision*, pp. 17–32, 2004.

[30] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, pp. 5–32, Oct. 2001.

[31] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural Comput.*, vol. 9, pp. 1545–1588, Oct. 1997.

[32] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, pp. 81–106, Mar. 1986.

[33] D. H. Ballard, "Readings in computer vision: issues, problems, principles, and paradigms," ch. Generalizing the hough transform to detect arbitrary shapes, pp. 714–725, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1987.

[34] E. Wahl, U. Hillenbrand, and G. Hirzinger, "Surflet-pair-relation histograms: a statistical 3d-shape representation for rapid classification," in *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on*, pp. 474 –481, oct. 2003.

[35] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, ICRA'09, (Piscataway, NJ, USA), pp. 1848–1853, IEEE Press, 2009.

[36] J. Stückler, N. Biresev, and S. Behnke, "Semantic mapping using object-class segmentation of rgb-d images," in *IROS*, pp. 3005–3010, 2012.

[37] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon, "Efficient regression of general-activity human poses from depth images," in *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, (Washington, DC, USA), pp. 415–422, IEEE Computer Society, 2011.

[38] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view RGB-D object dataset," in *ICRA*, pp. 1817–1824, IEEE, 2011.

[39] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama, "Digital photography with flash and no-flash image pairs," in *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, (New York, NY, USA), pp. 664–672, ACM, 2004.

[40] E. Parzen, "On estimation of a probability density function and mode," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.