Multiple-Cue Object Recognition for Interactionable Objects

Sarah Aboutalib

CMU-CS-10-153

December 8, 2010

School of Computer Science
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Manuela Veloso, Chair
Martial Hebert
Paul Rybski
Fernando De la Torre
Irfan Essa, Georgia Institute of Technology

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: Computer Vision, Multi-modal, Human Interaction, Object Recognition

Abstract

Category-level object recognition is a fundamental capability for the potential use of robots in the assistance of humans in useful tasks. There have been numerous vision-based object recognition systems yielding fast and accurate results in constrained environments. However, by depending on visual cues, these techniques are susceptible to object variations in size, lighting, rotation, and pose, all of which cannot be avoided in real video data. Thus, the task of object recognition still remains very challenging.

My thesis work builds upon the fact that robots can observe humans interacting with the objects in their environment. We refer to the set of objects, which can be involved in the interaction as 'interactionable' objects. The interaction of humans with the 'interactionable' objects provides numerous non-visual cues to the identity of objects.

In this thesis, I will introduce a flexible object recognition approach called *Multiple-Cue Object Recognition* (MCOR) that can use multiple cues of any predefined type, whether they are cues intrinsic to the object or provided by observation of a human.

In pursuit of this goal, the thesis will provide several contributions: A representation for the multiple cues including an object definition that allows for the flexible addition of these cues; Weights that reflect the various strength of association between a particular cue and a particular object using a probabilistic relational model, as well as object displacement values for localizing the information in an image; Tools for defining visual features, segmentation, tracking, and the values for the non-visual cues; Lastly, an object recognition algorithm for the incremental discrimination of potential object categories.

We evaluate these contributions through a number of methods including simulation to demonstrate the learning of weights and recognition based on an analytical model, an analytical model that demonstrates the robustness of the MCOR framework, and recognition results on real video data using a number of datasets including video taken from a humanoid robot (Sony QRIO), video captured from a meeting setting, scripted scenarios from outside universities, and unscripted TV cooking data.

Using the datasets, we demonstrate the basic features of the MCOR algorithm including its ability to use multiple cues of different types. We demonstrate the applicability of MCOR to an outside dataset. We show that MCOR has better recognition results over vision-only recognition systems, and show that performance only improves with the addition of more cue types.

Acknowledgments

First, and foremost, I would like to thank my advisor, Manuela Veloso, who has taught me so much and has given me invaluable amounts of encouragement and support. I hope some day to meet the standard she has set in research, demeanor, and life. I would like to thank my committee members, Martial Hebert, Irfan Essa, Paul Rybski, and Fernando De la Torre for their input, time and effort to help improve this thesis and my research work. Paul has helped me from the beginning starting with the CALO project and checking in throughout my time here. The suggestion given by Martial Hebert in our meetings have proven immensely valuable for giving direction in key points of my research. Fernando De La Torre (also a part of my early years) and Irfan Essa have also helped me signficantly. I would also like to thank my current (and former) lab mates, Douglas Vail, Sonia Chernova, Colin McMillen, Somchaya Liemhetcharat, Cetin Mericli, Joydeep Biswas, Stephani Rosenthal, and others in the CORAL lab, who were willing to star in my video datasets for this thesis and who have provided numerous helpful comments throughout the years.

I also want to thank my undergrad advisor, Prof. Jochen Triesch, who first introduced me into the world of research and computer vision, along with his previous grad student, Eric Murphy-Chutorian. I appreciate all the advice and knowledge they have given me in that initial first step.

I would like to acknowledge my friends and family who have provided support both in school and outside including Rozana Hussin (my first and best friend in Pittsburgh), Sajid Siddiqi, Khalid Harun, Nada Quraishi, Khalid El-Arini, Osmaan Akhtar, Abeer Saeed, Taysir ElBayly, Faheem Hussain, Mariam Salman, Romel Mostafa, Nurazlina Aziz and so many more. Thank you all for all that you have done. My father, Samih Aboutalib, mother, Salwa, and brothers (Samer, Saheil, and Sammy) have all given me untold amounts of support and encouragement. I love you. Later in the game, but so valuable, was the kindness and support given to me by my mother- and father-in-laws-to-be, Amal Kanbour-Shakir and Abdulrezak Shakir.

Lastly, I would like to thank my fiance, Nader Shakir, who has been an unimaginable source of support and help in every manner possible. Without him, I do not know where I would be.

And when I see you, I really see you upside down
But my brain knows better 🎶
Death Cab For Cutie

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Thesis Problem

The ability for an agent to accomplish a variety of tasks in an environment depends heavily on the reliability of visual recognition with regards to objects in that environment. Object recognition in the field of computer vision focuses on the task of identifying and locating a set of objects in an image or video sequence. The precise definition of object recognition can vary significantly depending upon the definition of "identify" and "locate". For our thesis work, we focus on category-level identification of the objects, in which the goal is to determine what category or class segmented regions belong to in an image. We define recognition as successful if the correct category label is attached to a point or region that lies somewhere within the true segmentation of that object in an image. Figure 1.1 demonstrates an example result.

Figure 1.1: Example of category-level object recognition

We make an additional distinction between object recognition with images and object

recognition with video. Object recognition has proven to be a significantly difficult challenge especially with the complexity of real world data, in which there is great variation in both the appearance of objects within a single object class (e.g., mugs may come in many shapes and colors - Figure 1.2), and in the appearance of the same object under various circumstances (e.g., the same object can appear different with changes in pose, size and lighting - Figure 1.3 top-left, bottom-left, and right respectively). The varying appearances and circumstances can extremely change the pixel values in an image for the same object. Static-image object recognition, as well as some video-based ones, have dealt with this complexity by focusing on learning robust visual features. Although great progress has been made along these lines, there is still much to be done in order to build an object recognition system that can be used under various real world scenarios. Our thesis attempts to address this problem by further utilizing the additional properties and information provided by the multimedia, temporal and sequential nature of video data.

Figure 1.2: Example of the many possible shapes and appearances of a single object

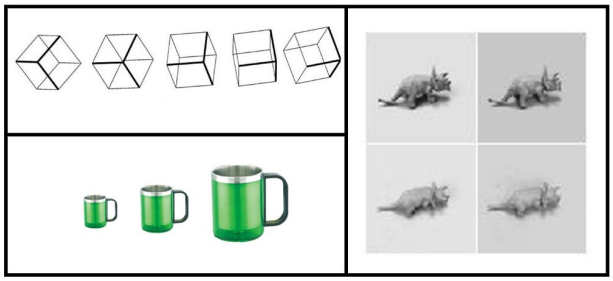


Figure 1.3: Example of numerous poses, sizes, and lighting effects for an object

When utilizing video information, we are particularly interested in the important observation that the environment and context around an object can provide numerous non-visual cues towards the identity of the objects, such as the interaction of humans with those objects. In this thesis, we build upon such interactions to improve object recognition. The agent can use activity recognition, speech recognition, and any other recognition system readily available to extract information from the interaction. For our thesis, we provide a framework that allows for the integration of these non-visual activity cues, speech cues, and other cues along with vision cues to produce more informed and robust object recognition, which we call **Multiple-Cue Object Recognition (MCOR)** .

Since Multiple-Cue Object Recognition can utilize information obtained through the observation of interactions with an object by a human or any other agent, we define the term *'interactionable'* objects for the set of objects that can obtain the greatest benefit from this algorithm for lack of an equivalent term currently existing. We define 'interactionable' objects as those which can be interacted with or can interact. Interaction in this definition means speaking about objects, having sounds coming from the objects, objects being acted upon, objects acting, and any other possible manipulation done to or by the objects.

Thus, MCOR is able to utilize the additional information and cues provided by videos of a human interacting with objects. Video data of predominantly static subject matter (such as data from a Mars Rover or of a landscape containing objects such as rocks, trees, or mountains) does not provide the additional interaction information utilized by MCOR. The determination of whether an object is interactionable or not is dataset-specific: Some objects may be interactionable in one dataset and not interactionable in another.

It is still possible to use the vision cue aspect of the MCOR algorithm to identify non-interactionable objects such as the tree example mentioned above. However, MCOR would only be as good as the vision cues it was given and thus would be on par with other visual recognition systems using those same cues. Thus, it would not be *best* utilized in that case. MCOR gains its advantage when used with a set of interactionable objects. 'Interactionable' may also be a useful term for other methods that depend on some type of interaction to aid in recognition such as in Functional Recognition, albeit not as extensively as MCOR.

This thesis work, thus, attempts to utilize the properties of video and interactionable objects in order to address the difficulties facing object recognition and the varying appearance of objects in real world data.

1.2 Thesis Approach

Our novel approach is to include multiple cues of any type for efficient and robust object recognition, whether they are vision cues intrinsic to the object or non-visual cues provided by the environment and interaction of humans. This approach includes:

Multiple Cues We create an object definition for each object category that allows for the inclusion of any number of cues of any type, such as activity, speech, vision, and gesture. Our object definition allows cues to be taken into account without special modification to the object recognition algorithm. In order to do this, we contribute a uniform cue representation, in which cues are organized with different types and

values. The representation of the cue requires a mapping to the image using object displacement values. Furthermore, multiple cues are weighted differently to capture the relative strength of the association from cue to object.

Probabilistic Weight Representation We recognize the interesting and challenging fact that some cues may be more indicative of an object category than others and thus the evidence given by that cue should have greater influence in the recognition of that object. In order to reflect this fact, we include weights whose values represent the strength of the association between a particular cue and object category for each possible cue and object class. We use *probabilistic relational models* (PRMs) to determine the strength of the association between a particular cue and an object category. The thesis work also utilizes a simulator for the generation of training and test data with cues based on real recognition systems. In addition, we use the simulator to test the accuracy of the probabilistic relational model for weights.

Utilization of Video We take advantage of the use of video rather than static single images to recognize objects. Video provides far more comprehensive information for the cues and allows for the use of several streams of information without limiting us to visual features, enabling the use of the information provided by the interaction of humans with the objects. In addition, video allows for the *incremental discrimination* of potential object categories, providing cue-driven equivalence classes where objects in the same equivalence classes, given a cue, can be distinguished with the addition of new cue information provided at each time step in the video.

Evaluation We use several methods and datasets in order to show the advantage of using multiple cues for object recognition including analytical, and experimental evaluations.

Several video datasets are used in these evaluations:

Meeting Data - There are two sets of meeting data: (1) video data taken from an omnidirectional camera consisting of a single person interacting with objects in office scenarios and (2) data taken from a Panasonic digital video camera consisting of multiple people interacting with objects in office scenarios. This data is used to demonstrates key advantages of using multiple cues.

QRIO(Humanoid Robot) - Video data was taken from a single camera eye of a QRIO Robot. This data is used to demonstrate the incremental discrimination of object labels based on cues.

Gupta & Davis [23] - Video data was taken from another research dataset [23], where a single person manipulates objects in a laboratory setting. This data is used to demonstrate MCOR on an outside data source.

Cooking Data - Video data was taken from DVDs of actual cooking shows aired on TV, as well as from scripted kitchen scenarios from the LACE dataset from the University of Rochester [47]. This data allows for learning based on real training data and shows recognition on a large dataset.

1.3 Thesis Contributions

This thesis makes several main contributions to object recognition with interactionable objects. These contributions consist of:

- An effective representation of each object class, cues, and the association between the cues and object categories.
- A probabilistic mechanism for the calculation of weights reflecting the strength of association between a cue and an object category.
- A recognition algorithm, MCOR (Multiple-Cue Object Recognition) that utilizes the representation of cues and weights to determine the evidence for the presence of an object of a particular category and generalizes new cues found in the recently recognized object to all objects of the same class, i.e., the object definition is changed to include the new information.
- The utilization of vision and image processing tools including tracking, segmentation, visual features, and scene gist information in order to take advantage of video for better object recognition.
- An analytical evaluation as a mathematical model representing the behavior of the multiple cue framework as the accuracy and configuration of the cue information varies. We show that, at worst, the introduction of inaccurate multiple cues does no additional harm to the recognition performance, while the inclusion of accurate multiple cues provides a significant improvement.
- Key scenarios where multiple cues provide a distinctive advantage over vision-only recognition in a number of empirical demonstrations, providing a validation of the multiple cue framework and the inclusion of vision, activity, gesture, and speech cues.
- Experimental results of the MCOR system showing a significant improvement in performance when comparing vision-only and multiple cue performance on a large video dataset.

1.4 Thesis Outline

The thesis is organized as follows:

Chapter 2 - Background and Related Work We give an overview of related work pertaining to this thesis.

Chapter 3 - MCOR Representation We provide a framework for the representation of cues and object definitions to allow for the flexible integration of multiple cues, and specify the cue types and values used.

Chapter 4 - Learning Weights and Dictionary Properties We present the Probabilistic Relational model framework used to calculate the weight association between a particular cue and object, and object displacement values for the localization of object category evidence in an image.

Chapter 5 - Cues, Segmentation, and Tracking We explain the various vision and image processing tools used in the MCOR algorithm for tracking, segmentation and the creation of visual features. We define the cue values and method used to obtain those values for each of the cue types.

Chapter 6 - Incremental Discrimination and Recognition We present the recognition algorithm which makes up MCOR and integrates the multiple cue representation, weights, and visual information in a coherent system utilizing incremental discrimination to determine object labels.

Chapter 7 - Recognition Robustness, Analysis and Cue Accuracy We provide an analytical evaluation of the robustness of the multiple cues in MCOR by using a performance metric on varying levels of cue accuracy.

Chapter 8 - Experiments and Evaluation We demonstrate the recognition capabilities of MCOR utilizing real video datasets obtained from internal and external research scripted scenarios and unscripted video datasets.

Chapter 9 - Conclusion We summarize major contributions and discuss future work.

Chapter 2

MCOR Representation

In order to perform category-level object recognition, the object recognition framework must have a model for each of the object categories that need to be found. This model defines which cues and features are used when determining the presence of an object belonging to a particular category. In most work, the model representation restricts the type of features and cue information used to recognize an object. In the MCOR framework, the fundamental model is an object definition. Each object definition contains a set of cues, each with a specified set of properties. This structure, called the object dictionary, allows for the inclusion of any number of cues of any type (e.g., activity, speech, vision, and sound) in the object models without special modification to the object recognition algorithm. In this chapter, we first define the dictionary structure used by the object recognition algorithm to access information about the object categories, including object definitions and cue representations. We then outline the probabilistic relational framework used to fill in the object dictionary definitions from video data information.

2.1 MCOR Object Dictionary Structure

In this section, we define the terms that make up the object dictionary structure used by MCOR to reference model information about each object category. This object dictionary structure consists of a set of object definitions, which in turn consists of a set of definition cues. The complex back structure used to fill in these object dictionary values is discussed in the next section.

Formally, we define an ***object dictionary*** as the set of object definitions for each category, o_i , from the set of all object categories, O given. Each object definition represents the

model of each object category.

2.1.1 Object Definition

In previous object recognition methods, an object model consists of a rigid specification of the types of cues and features used to represent an object category, e.g., a graph of visual features [45] or, as in the FOCUS algorithm [70], a functional and visual description. For these methods, the value of the features and cues can be adjusted to represent different object categories, but the type of features and cues that compose the model cannot. This inability to flexibly adjust the composition of the object model has two disadvantages: (1) It prevents the application of the framework to domains which may not have available the type of feature and cue information required by the model and (2) it lacks the potential of utilizing the full set of information provided by cue types which may be available in a domain but not used in the object model. In this thesis, we address these disadvantages by providing a flexible object model with the ability to represent object categories using an adjustable set of multiple cue types. We refer to the adjustable set for a particular object category as an ***object definition***. This flexibility is provided by representing each object definition as a set of cues, C_i for the i^{th} object category, o_i that can be of any size and contain any type of cue information. We call these cues ***definition cues***, i.e., cue information taken from the object definition. The definition cues are distinguished from the ***extracted cues*** taken from the video data, which we discuss later. For the unified use of the various cue types, each cue must follow a specified representation.

2.1.2 Definition Cue Representation

In order to integrate the evidence from multiple sources i.e., from multiple cue types, we introduce a standard representation for all cue types in the object dictionary. More specifically, all definition cues must define a set of properties:

- **Cue Type and Value -** Defines the type of the information provided by the cue and the value of the cue depending on the type.
- **Object Displacement Values -** Provide the spatial and temporal location of the object segment within a video for which the cue information is ascribed.
- **Weight -** Provides the strength of the association between a cue and an object class.

In Figure 2.1, we present a diagram of the object dictionary structure, including the cue representation properties defined by each definition cue. For each object category, our framework defines a set of definition cues that comprise an object definition. The object dictionary then consists of the entire set of object definitions provided for each object category. The cue properties of the individual definition cues in the structure consist of: the cue type and value, the object displacement values (including spatial and temporal offset information) and the weight. The definition cue property values are filled in using the probabilistic relational framework (defined in Section 2.3) based on the objects and extracted cues from data. We describe each of the dictionary components in the following subsections.

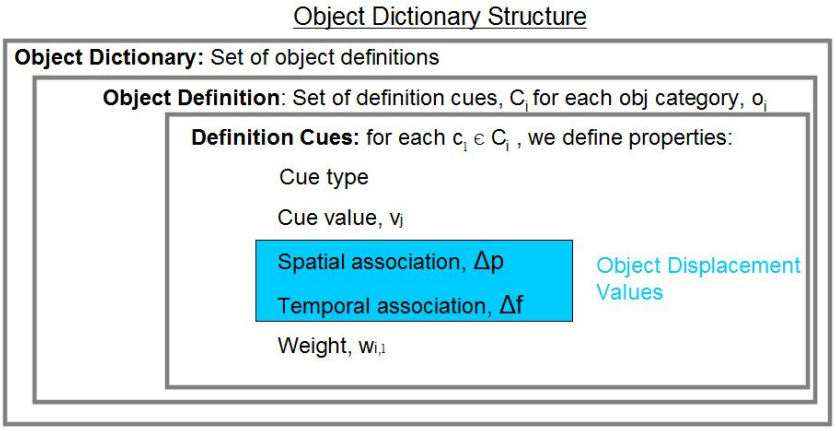


Figure 2.1: Object dictionary structure used by the MCOR algorithm

2.1.3 Cue Type and Value

The cue type and value properties of the definition cue representation identify the kind of information provided by the cue. The ***cue type*** is the recognition framework or features used to extract cue information. For instance, an activity cue type uses an activity recognition system to extract information. The ***cue value*** is then the content of the extracted information. For the activity cue case, the cue values consist of the activity recognition labels such as *Sit*, *Stand*, *Eat*, and *Write*. Each of the cue types defines the set of possible cue values.

In our work, we draw from the set of cue types, {activity, speech, color and shape, texture, sound}. The details of the set of values for each of these cue types is given in detail in Chapter 4. Object displacement values and weights are defined for each of these cue values.

2.1.4 Object Displacement Values

In order to utilize the object definition and cue information provided for object recognition in a video, the information needs to be attached to particular pixel locations in the video. We introduce object displacement values that localize in the video the actual pixel locations in space and time of the object region associated with a cue.

Spatial Association

In order to determine the location of objects in a video, we need to determine where cue information should be localized. We define the ***spatial association***, Δp , as the difference between the current cue's position (a cue-specific calculation)and the expected location of the object to which the cue is associated. Figure 2.2 gives an example: if the location of the speech cue value, *"laptop"*, was defined as the center of the face of the person speaking, then the spatial association would be the distance between the center point of the face and the expected position of the laptop. The arrow in the figure represents the spatial association between the speech cue and the object it should be associated with. This property will be used to determine which segmented region in the scene the cue belongs to.

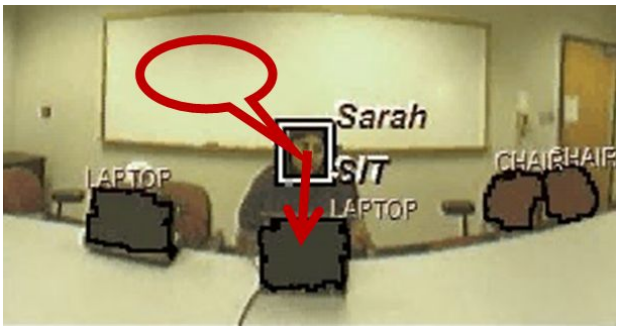


Figure 2.2: Demonstration of spatial association for a speech cue

In initial experiments, where the number of objects and scenarios were limited, we defined the spatial association for each cue and object by hand. In later experiments with larger datasets, we learned the spatial association from a training set, as we discuss in Chapter 3.

Temporal Association

We introduce a ***temporal association***, Δf , to represent the difference between the frame associated with the extracted cue and the frame where the object is clearly visible. This property is primarily important for cues that, in the process of being produced, might obscure the object they indicate. Figure 2.3 gives an example of the temporal association: the activity of sitting may obscure the chair for which it is providing evidence, i.e., when the person was sitting, the chair being interacted with was not visible. A clear view of the object is necessary in order for that object to be segmented and recognized. Thus, it is necessary to go about 4 frames previous to see the object. The 4 frames is the temporal association for the *Sit* cue value.

Figure 2.3: Demonstration of temporal association for an activity cue

As with the spatial association, in initial experiments where the number of objects and scenarios were limited, we defined the temporal association by hand for each cue and object. In later experiments where the datasets were larger, we learned the temporal association from a training set as we later discuss.

2.1.5 Weights

In order to reflect the fact that different cues may be more indicative of an object class than others, we introduce weights ascribed to each cue within each object definition. In Figure 2.4, we provide a visual representation of the varying strength of weights for a particular cue value (the color *Orange*) and object categories (chair and orange). Weights represent the strength of the association between that cue and the object. Objects that are more highly associated with a cue (such as the orange object and orange color cue) have higher

values, while those less strongly associated (such as the chair object and orange color cue) have smaller values. We determine the weight, $w_{i,j}$, by the probability of the object, o_i, being present given the cue value, cv_j, i.e., $\text{P}(o_i|\text{cv}_j)$.

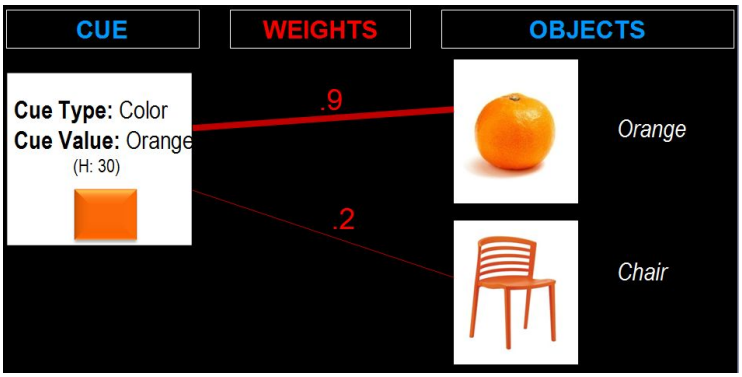


Figure 2.4: Example of varying weight values for a cue value

The calculation of the weight value is discussed later in Chapter 3. Given the information provided by the weight value, object displacement values, cue type and cue values, an object dictionary can be created for utilization by the object recognition algorithm.

2.1.6 Depictions of an Object Dictionary Instance and Dictionary Presence in Recognition

In this section, we provide a demonstration of an object dictionary with filled-in cue representation properties, and we show where the object dictionary lies in the process of recognition by the MCOR algorithm.

In order to depict all of the components of the object dictionary structure combined, we demonstrate in Figure 2.5 an instance of an object dictionary with filled in cue representation properties and object definitions. In Figure 2.5(a)., an object dictionary with definitions for two object categories, laptop and chair, are given with cue types, cue values, and weights labeled. Note how the *Sit* cue value for the laptop object category has a smaller weight value than the same cue value for the chair object category, since you would expect the activity of sitting to have a greater association with a chair than you would with a laptop. In Figure 2.5(b.), a more detailed look at the properties for a particular definition cue is given. In the example, the cue type and value, object displacement values and weight are defined for the *Sit* activity cue value in the chair object category definition. Spatial association is set to 0 since the activity of sitting occurs directly on top of the object, and the temporal association is set to 4 since it is necessary to go back 4 frames in order to see the object.

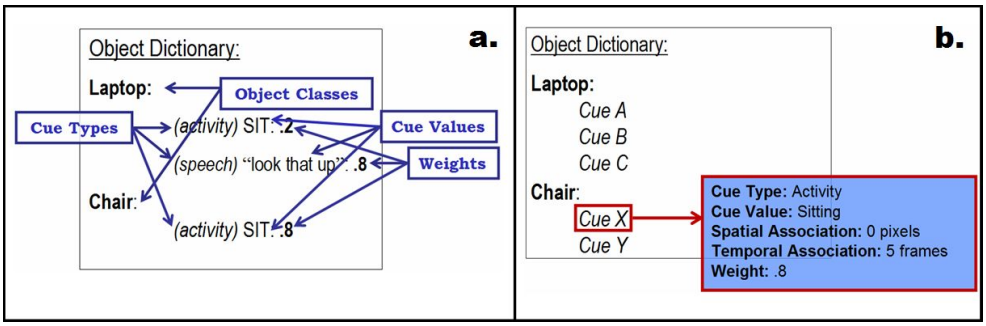


Figure 2.5: Example of object dictionary structure and detailed example of a definition cue with filled in properties

An object dictionary such as the one demonstrated in Figure 2.5 contributes to the overall object recognition algorithm as represented in the flow diagram in Figure 2.6. We provide the details of the MCOR algorithm in Chapter 5. As an overview, the MCOR algorithm (Algorithm 4) starts by extracting cues from the video (Figure 2.6(1.) & (2.)) based on recognition systems available, such as an activity recognizer, speech recognizer and sound recognition system. These extracted cues are then attached to segmented regions in the video based on the object displacement values described below (Figure 2.6 (3.)). Object recognition is then based on the collection of evidence for each extracted cue associated with each segmented region weighted by the values in the object dictionary (Figure 2.6 (4.) & (5.)). The object dictionary is then updated based on the recognition results to allow for generalization and updated weight values.

Using this framework and cue properties defined below, segmented regions in the image can then be recognized as objects by comparing the extracted cues from the scene with the definition cues in the object dictionary. This object dictionary structure allows for a recognition algorithm that is independent of the type and number of cues available and thus can utilize the wide variety and varying cues humans provide in their interaction with an object.

In the next section, we describe how the extracted cues and objects from video data are *represented* in a probabilistic relational framework in order to determine the values for the cue properties found in the object dictionary. In the next chapter, we go into more detail on how these values are calculated.

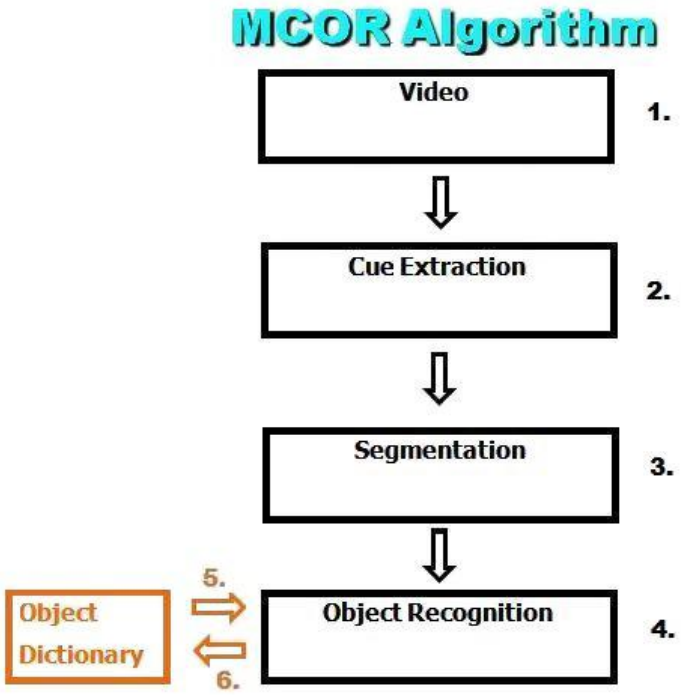


Figure 2.6: Flow of MCOR framework

2.2 Probabilistic Relational Models from Video Data

In this section, we describe the framework, **Probabilistic Relational Models (PRMs)**, used to determine the representation properties of a definition cue in the object dictionary structure, i.e., the temporal association, spatial association, and weight. We use PRMs because they can easily represent relational information such as the information provided by the different object categories and cues in a set of videos. In addition, the probabilistic relational model framework provides the advantage of representing multiple complex relationships that can be learned.

In order to determine the definition cue property values, we learn the values from data. This requires obtaining information from the objects, extracted cues, and humans in the video data. In this section, we define the probabilistic relational framework outlining the attributes taken from each of these sources and their relationships to each other.

2.3 Overview of Probabilistic Relational Models

The Probabilistic Relational Model [18, 19] can handle the inclusion of relation information of data as an extension of standard Bayesian networks. PRMs can learn associations between classes, attributes within a classes, and attributes related to another class rather than the flat, attribute-value data of Bayes nets. In this thesis, we use PRMs to represent and learn our proposed definition cue properties, which reflect the relationship of the various properties of the cues and objects. We first review the PRM framework:

Within a relation model, a ***schema*** consists of several components: A set of classes,

$$\mathbf{X} = X_1, ..., X_n$$

each of which has a set of ***attributes***,

$$A(X_i) = X_i.a_1, ..., X_i.a_2$$

The attributes can be ***fixed***, ***deterministic***, or ***probabilistic***. Fixed attributes have unchanging values and identify entities of a class. Deterministic attributes have values which are directly determined by the entity itself; once determined, their value does not change. The values of probabilistic attributes can vary based on the other attributes of the class or of related classes. In terms of notation, we use $E^I(X_i)$ to refer to all entities of the class, X_i in instance I. A single entity of a class can be referenced by the value of a fixed attribute belonging to that class, e.g., $\mathbf{xd_k}.a_j$ represents an entity with fixed attribute value $\mathbf{xd_k}$ and attribute a_j. Classes are given in upper case, X_i; attributes are given in lower case, a_j; fixed attribute values which identify entities of a class are given in lower case bold, $\mathbf{xd_k}$.

The second component is the set of ***relations***,

$$\mathbf{R} = R_1, ..., R_m$$

that defines the relationship between two classes. Relationships are significant in that the value of attributes in one class can depend not only on the other attributes of that class, but on the attributes of any related class.

The ability to depend on attributes from other classes is represented by the concept of ***slots***. If $R[X_1, ..., X_k]$ is any relation among classes $X_1, ..., X_k$, we can project R onto the i^{th} and j^{th} classes to obtain a binary relation, $R(\mathbf{f}, \mathbf{g})$, between entities $\mathbf{f}$ and $\mathbf{g}$. $\mathbf{f} \in E^I(X_i)$ and $\mathbf{g} \in E^I(X_j)$ and $R(\mathbf{f}, \mathbf{g}) = 1$ when entity $\mathbf{f}$ and $\mathbf{g}$ are related through relation R, otherwise, $R(\mathbf{f}, \mathbf{g}) = 0$. We refer to $\mathbf{f}.R$ as a slot of X_i. In order to refer to the n^{th} attribute, a_n of an entity of class X_j, we write $\mathbf{f}.R.\mathbf{g}.a_n$. In some cases, we wish to refer

to the set of all entities which relate to the entity $\mathbf{f}$, i.e., for any $\mathbf{f} \in E^I(X_i)$, we let $\mathbf{f}.R$ denote all entities $\mathbf{g} \in E^I(X_j)$ such that $R(f, g)$ holds. We then denote $\mathbf{f}.R.a_n$ as the n^{th} attribute of class X_j for all entities $\mathbf{g} \in E^I(X_j)$.

The notion of ***aggregation*** from database theory is used by PRMs to summarize the set of attribute values returned by a slot, $\mathbf{f}.R.a_n$. Aggregation of the set of values from a slot is represented by γ_H, where H is an aggregation function that defines how the set of values are summarized, $\gamma_H(\mathbf{f}.R.a_n)$. There are many natural and useful functions for aggregation such as mean and median. Our schema uses a number of aggregation values seen in Figure 3.1. The aggregated value can be used to determine the value of a related attribute.

Our object dictionary provides the structure of the network.

2.4 Object and Cue Representation

We can now describe the model used to represent the relationship between the various extracted cues and objects in order to determine the object definition property values. The object and cue classes defined here should not be confused with the object definitions and definition cues belonging to the object dictionary. Although they are linked to each other, the classes outlined here describe the information taken from objects and cues extracted from the data. This information is then processed in order to be used to fill in the property values in the object dictionary.

MCOR Schema To begin, we must first define the set of classes used by the MCOR schema and their attributes (see Figure 2.7). In terms of the classes, we define an OBJECT class, a class for each cue type <CUETYPE>, a PERSON class, and a SEGMENT class. We now describe the attributes for each of these classes.

We define an OBJECT class in order to describe the location, visibility and category for each object in the dataset. Thus, the OBJECT class has the following attributes:

object id, od (fixed) The identifier of an object entity.

object category, og (probabilistic) The object category for an object entity.

object location, ol (deterministic) The location of the object as given by a vector of 4 items describing the location and size of the bounding box on the image.

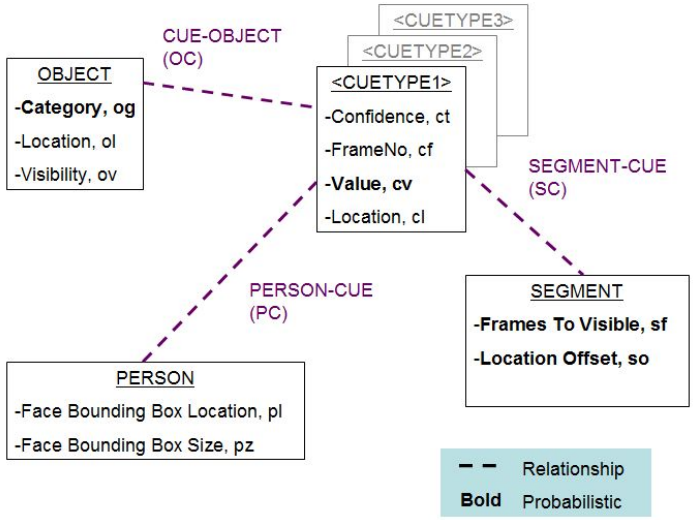


Figure 2.7: PRM schema of the extracted cues and objects used for determining definition cue properties

object visibility, ov (deterministic) The array of start and end frame number pairs for each consecutive set of frames the object is visible.

In addition, we have a class for each extracted cue type, <CUETYPE>. Each class has the same set of attributes:

cue id, cd (fixed) The identifier of an extracted cue entity.

cue value, cv (probabilistic) The value of the cue. The range of values depends on the cue type.

cue type confidence, cc (deterministic) The confidence in each cue type. Details are given in the next chapter.

cue frame number, cf (deterministic) The frame number the cue was extracted from.

cue location, cl (deterministic) The location of the extracted cue. Some cue types have location information given explicitly (for instance, vision cues provide bounding box and location information). All other cues are attached to the bounding box around the person's face whose interaction caused the cue.

The next class is the PERSON class. This gives information needed for determining location information. Thus, the attributes are:

person id, pd (fixed) The identifier of the specific person interacting.

face bounding box location, pl (deterministic) The bounding box location around the face of the person in the image.

face bounding box size, pz (deterministic) The bounding box size around the face of the person in the image.

The last class, SEGMENT, represents segment information, namely where the segment used for recognition should be located. Unlike the OBJECT, CUE, and PERSON classes, there is only one entity of this class, which is related to all the cues. It can be thought of as a global class. The attributes are as follows:

segment id, sd (fixed) The identifier of a segment entity. Since there is only one entity of this class, there is only one segment id.

frames to visible, sf (probabilistic) The number of frames till the object is visible given a cue value.

location offset, so (probabilistic) The location offset for an object category given a cue value.

These classes are related to each other through the relationships: OBJECT-CUE (OC), SEGMENT-CUE (SC), and PERSON-CUE (PC), as illustrated in Figure 3.1.

Through these relationships various dependencies define the calculation of the weights, spatial association, temporal association, and type confidence, as we introduce in the next chapter.

2.5 Conclusion

Using the object definitions and cue representations and properties, we utilize the resulting object dictionary for the core models for the object recognition algorithm. By defining an object as a flexible set of cues, we allow for the the use of multiple cues of varying types to provide as much evidence for determining object classes as possible. Requiring cues to define a cue type, values, object displacement values, and weights, we have the information needed to use cue information to localize and identify objects in a video.

Chapter 3

Learning Weights and Dictionary Properties

In this chapter, we define how the definition cue property values found in the object dictionary are learned using the PRM model of the data. We deal with the concept that not all cues are associated with a particular object class with the same strength. In terms of the cue representation and object dictionary this translates to the weight property. We define how this weight is calculated using the probabilistic relational models. We allow for the incremental update of these weights as well as the addition of new cue types during the recognition process.

In addition, since the probabilistic relational model framework provides the advantage of representing multiple complex relationships that can be learned simultaneously, we describe how the probabilistic models are used to learn temporal and spatial associations in addition to cue type confidence.

We demonstrate the learning of these property values using both simulated and real data.

3.1 Learning with PRMs

In this section, we describe how the probabilistic relational representation of the data is used to learn the property values for each definition cue. To begin, we define the probabilistic framework of PRMs:

PRMs describe a probability model over entities of a relational schema. An instance, I, of

the relational schema consists of the set of entities of each class,

$$E^{\sigma}(X_i) = e_1, ..., e_p$$

, where the attributes are defined and which relationships exist between them. A relational skeleton, σ, is when only the fixed attributes of the entities are defined. In our case, an instance would consist of all the objects in a scene, all the extracted cues in the scene, and the association between any of the extracted cues or objects. Some of the attributes however are not easily defined, such as the object category, and thus, it is necessary to determine the probability distribution of its values. A relational skeleton σ is then a partial instance where the probabilistic attributes are undefined. Some of these probabilities may depend on attributes from within the same class or from any other class related to it through one of the defined relationships. The set of dependencies between the attributes is referred to as the ***dependency structure***, S. The attributes that are depended upon are called the ***parent*** attributes. Thus, given a defined entity, e_k, and probabilistic attribute, a_j, the parent attributes are defined as $\text{pa}(e_k.a_j)$. The dependency structure and parents for the MCOR schema can be seen in Figure 3.1. Dependencies between attributes are shown with arrows, and aggregate functions for related slots are shown with boxes. Arrows marked with a circle represent the dependencies used to calculate the weight value; Arrows marked with a star represent the dependencies for the spatial association; Arrows marked with a triangle represent the dependencies for the temporal association; Arrows marked with a square represent the dependencies for the type confidence. As is the common case with Bayesian networks, each of the dependencies has parameters which represent the conditional probability of obtaining a particular attribute value given the value of the parents. The set of conditional probability distributions (CPDs) for each of the attributes makes up the parameters, δ_S, that are learned by the network.

PRMs then define the distribution of instantiations of attributes as:

$$P(I|\sigma, S, \delta_S) = \prod_{X_i \in \mathbf{X}} \prod_{a_j \in A(X_i)} \prod_{e_k \in E^{\sigma}(X_i)} P(I_{e_k.a_j}|I_{\text{pa}(e_k.a_j)}) \tag{3.1.1}$$

Given a training set, I, the parameters, δ_S, can be learned according to the following equation:

$$l(\delta_S|I, \sigma, S) = \log P(I|\sigma, S, \delta_S) \tag{3.1.2}$$

$$l(\delta_S|I, \sigma, S) = \sum_{X_i} \sum_{a_j \in A(X_i)} \left[\sum_{e_k \in E^{\sigma}(X_i)} \log P(I_{e_k.a_j}|I_{pa(e_k.a_j)}) \right] \tag{3.1.3}$$

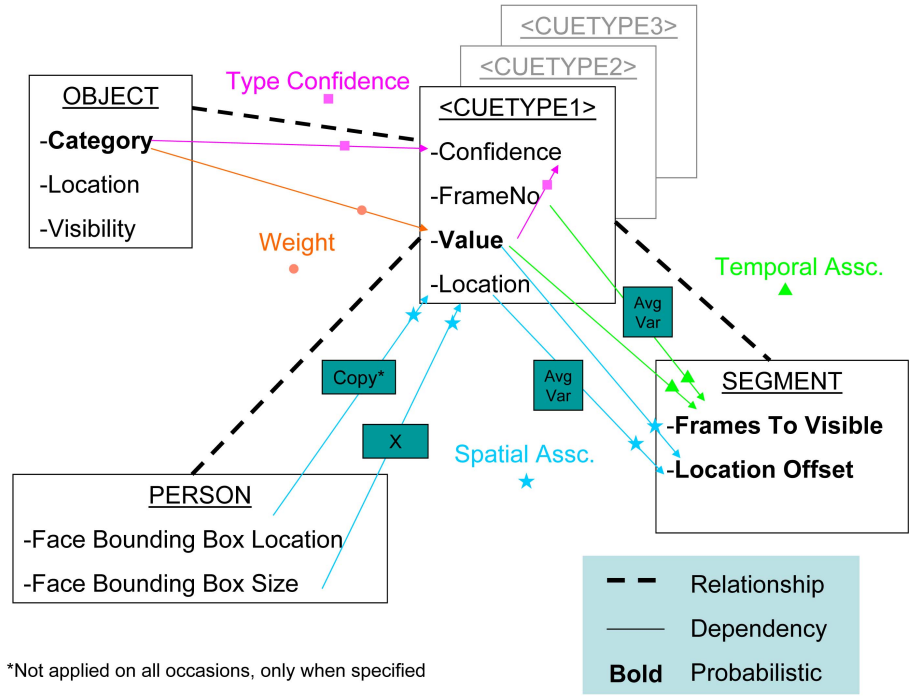


Figure 3.1: PRM schema of cues and objects showing dependencies and aggregate functions for related slots

Each of the terms in the square brackets can be maximized independently of the rest. Thus, maximum likelihood estimation reduces to an independent maximization problem, one for each CPD.

When dealing with multinomial CPDs, maximum likelihood estimation can be done through counts $C_{X_i.a_j}[v, \vec{u}]$ of the different values of $v, \vec{u}$, which the attribute $X_i.a_j$ and its parents can jointly take. The maximum likelihood parameter setting $\hat{\theta}_S$ is

$$P(X_i.a_j = v | Pa(X_i.a_j) = \vec{u}) = \frac{C_{X_i.a_j}[v, \vec{u}]}{\sum_{v'} C_{X_i.a_j}[v', \vec{u}]} \tag{3.1.4}$$

In order to learn the parameters online, we utilize a well established method of updating these conditional probabilities called the m-estimate of probability [40]. Using the learned parameters as priors,$\alpha_{X_i.a_j}[v, \vec{u}]$, we use a virtual training dataset size, m, called the *equivalent sample size* constant to determine how heavily to weight the prior relative to the observed data: The larger m, the more observed examples need to be seen in order to adjust the weights. In our experiments, we set m to the actual training dataset size since

we have that information.

The conditional probabilities are then updated with the introduction of the new data, I, as follows:

$$P(X_i.a_j = v|Pa(X_i.a_j) = \vec{u}|I) = \frac{C_{X_i.a_j}[v, \vec{u}] + \alpha_{X_i.a_j}[v, \vec{u}] * m}{\sum_{v'} C_{X_i.a_j}[v', \vec{u}] + m} \tag{3.1.5}$$

Weights As mentioned previously, we would like to calculate the weight, $w_{i,j}$, as the probability of an object category, o_i, being present given a cue value, cv_j, i.e., $P(o_i|\text{cv}_j)$.

In this schema, this relationship can be defined using the probabilities: P(OBJECT.og_i), P(CUETYPE.cv_j), and P(CUETYPE.cv_j|OBJECT.og_i) as given by the dependencies and attributes defined above. Note that there is a conditional table for each cue type across each of its values. Given a training set, the parameters δ_S are learned according to the equations defined above.

The other dependencies in the schema can then be used to calculate the spatial and temporal association and cue type confidence.

Spatial and Temporal Association As mentioned in Chapter 2, we need a method of determining the spatial and temporal association associated between each cue and potential object. We utilize the PRM schema to provide such values. We take advantage of the aggregation ability of PRMs to calculate the probability of getting different location offsets and number of frames until object is visible.

Given video data, we define the function, `SOFFSET`, which takes as a parameter an extracted cue entity, **cd**, from the data. It returns the spatial offset by taking the difference between a cue location, cl, of a related object (through the OBJECT-CUE relationship, OC). The cue location is either given by the cue or defined by the location, pl, of a person related to that cue (through the PERSON-CUE, PC, relationship), depending on the cue type. For instance, a COLOR&SHAPE cue type would return the location defined by the center of its boundaries, while a SPEECH cue type would define its location by the location of the person speaking. This value is scaled by the size of the bounding box around that persons head, pz:

$$\texttt{SOFFSET}(\mathbf{cd}.\text{cl}) = \frac{1}{\mathbf{cd}.\text{PC}.\mathbf{pd}.\text{pz}}(\mathbf{cd}.\text{PC}.\mathbf{pd}.\text{pl} - \mathbf{cd}.\text{OC}.\mathbf{od}.\text{ol})$$

The spatial offsets are potential spatial associations that can be used to locate the position

of a segment in an image and relate it to a specific cue.

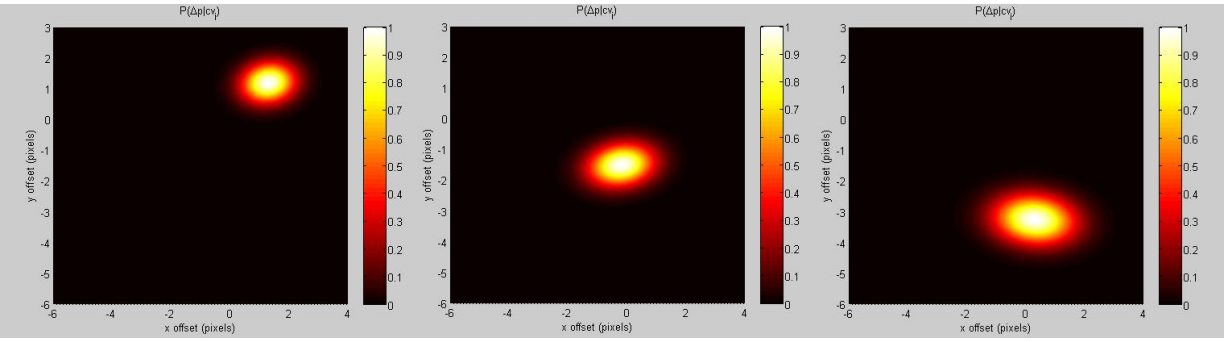


Figure 3.2: Spatial association probability ($\mathrm{P}(\Delta p|\mathrm{cv}_j)$ for *Close*, *Pour* and *Bring* activity cue value.

We then use the aggregation of the spatial offsets through the SEGMENT-CUE (SC) relationship to get an average (mu), standard deviation (std), and covariance (cov), of all the cue locations i.e., $\mathbf{sd}.\mathrm{so} = \gamma_{\mathrm{std,mu,cov}}(\texttt{SOFFSET}(\mathbf{sd}.\mathrm{CS.cl}))$. As a reminder, the aggregate value, $\gamma(\mathbf{x}.R.a_j)$ obtains all the values of attribute, a_j, for all entities related to entity $\mathbf{x}$ through relationship R. These values are summarized/aggregated, using a specified function. In our case, we take the average and standard deviation of the spatial offsets related to the segment entity. Since, the segment class has the unique characteristic of having only one entity and being related to all the cue entities, the aggregation takes the average of all the cue locations.

We then use the mean, $\mathbf{sd}.\mathrm{so}_{\mathrm{mu}}$, covariance, and standard deviation, $\mathrm{sd.so}_{\mathrm{std}}$, of the cue locations to determine the probability of these candidate spatial associations for a particular cue value. Because the spatial offset values of the spatial association are continuous, we use a continuous 2D Gaussian distribution to represent the probability of getting a particular offset given a cue value, $P(\Delta p|\mathrm{cv}_j)$, since we can assume each spatial association instance is an independent event given a cue value, cv_j. Figure 3.2 shows example distributions of the spatial association probability calculated from the Rochester dataset for the (a.)*Close*, (b.)*Pour*, and (c.)*Bring* activity cue values. These examples show activity cues whose probabilities vary from most variance to least. We then choose a spatial association, Δp^*, for each cue value as the offset with the greatest probability (Since the distribution is a Gaussian, this corresponds to mu):

$$\Delta p^* = \mathrm{argmax}_{\Delta p} P(\Delta p|\mathrm{cv}_j) \tag{3.1.6}$$

While we chose to define the spatial association used in the object dictionary as the offset with the greatest probability, an alternative strategy would include segmenting at all

locations touched by spatial association defined in the Gaussian and scaling the evidence by the probability of each. We decided to go with the former strategy as the majority of our datasets dealt with spatial association, which fell somewhere within the segmentation of the object.

Note that with our method, error rates (based on the percentage of times the maximum offset missed the object) grows with an increase in variance and with the size of the object. Large objects which tend to be around the same position given a particular cue value show a small variance and thus a large probability that the offset will fall around the learned position. Also, larger objects provide a larger margin of error as we only require the offset to hit somewhere on the object. On the opposite end, small objects, which vary a lot in terms of position given a cue value, will be harder to learn with the large variance and provide a smaller margin of error. The consequence of a mistaken offset could be corrected in MCOR through the introduction of more cues (see Chapter 6).

We go through a similar process in order to get the probability distribution for the temporal association values. Replacing the subtraction of the spatial offset with a temporal offset, i.e. function, `TOFFSET`, returns the difference between the current cue frame and the frame where the object was last visible:

$$\texttt{TOFFSET}(\mathbf{cd}.\text{cf}) = \mathbf{cd}.\text{cf} - L(\mathbf{cd}.\text{OC.ov})$$

where the function L() returns the last frame where the object was visible before the cue frame number. Using the aggregation of these temporal offsets through the SEGMENT-CUE relationship, we have:

$$\mathbf{sd}.\text{sf} = \gamma_{\text{std,mu}}(\texttt{TOFFSET}(\mathbf{sd}.\text{CS.cf}))$$

We then use the mean, $\mathbf{sd}.\text{sf}_{\text{mu}}$ and standard deviation, $\mathbf{sd}.\text{sf}_{\text{std}}$, of the temporal offsets to determine the probability of these candidate temporal associations for a particular cue value. Because the temporal offset values of the temporal association are continuous and since we can assume each temporal association instance is an independent event given a cue value, cv_j, we use a continuous Gaussian distribution to represent the probability of getting a particular offset given a cue value, $P(\Delta f|\text{cv}_j)$. We then choose the temporal offset with the highest probability as the temporal association, Δf^* entered into the object dictionary.

$$\Delta f^* = \text{argmax}_{\Delta f} P(\Delta f|\text{cv}_j) \tag{3.1.7}$$

The temporal and spatial properties can then be used in the recognition algorithm to determine where to segment for possible objects.

Cue Type Confidence The last value that can be calculated from this schema is the cue type confidence term. This attribute defines the amount of confidence one can have in the entire cue type. This confidence is calculated by determining the mutual information provided by the cue values of a specified cue type and the object categories. We use mutual information since it tells us how much information one variable can give about another. We determine a good cue type to be one that gives a lot of information about an object.

We calculate this mutual information, MI(), for each cue type class, $[CT_1, ..., CT_m]$:

$$\text{MI(OBJECT.og;CT}_i\text{.cv)} = \sum_{\forall c \in \text{CT}_i\text{.cv}=c} \sum_{\forall o \in \text{OBJECT.og}=o} P(o,c) log \frac{P(o,c)}{P(o)P(c)} \tag{3.1.8}$$

In order to standardize this information and possibly use it as a confidence weight, we normalize the mutual information to a value of [0,1]. This **normalized mutual information (NMI)** provides the value for the cue type confidence CT_i.cc:

$$\text{CT}_i.cc = \text{NMI(OBJECT.og;CT}_i\text{)} = \frac{\text{MI(OBJECT.og;CT}_i\text{)}}{\min[\text{H(OBJECT.og), H(CT}_i\text{)}]} \tag{3.1.9}$$

where, H() is the entropy function:

$$H(X) = -\sum_{x \in V(X)} p(x) \log(p(x))$$

Although the schema provides the potential for using this term, it is not currently being used since the implicit strength of association is given through the $P(o_i|\text{cv}_j)$. This, however, could be a useful term for future work in context learning, i.e., determining which cue types are beneficial for different contexts and possibly initializing weights with the confidence values until additional data can be found to refine the benefit of individual cue values, as well as for possible uses in additional confidence weighting.

Using the schema, relationship, and dependencies defined above, we can learn and test the parameters and probabilities of this model. In the first experiment, we use simulated data to show the accuracy of the learned model given a known model. In the second experiment, we show the weights that are learned using real data.

3.2 Simulation

We begin with the simulated data. In order to determine, how weight learning would occur using different recognition systems, we developed a simulator that could represent the generation of cues based on the performance results of real existing systems. We define below the five cue types used:

ACTIVITY The activity class consists of the set of possible activities that can be recognized. The activity class provides important information as to the function of an object. Since in many cases, while visual attributes may vary, the functional characteristics often do not. The possible cue values are based off of two different recognition systems the activity system developed by Gupta et al. [23] with activities: *Eat*, *Drink*, *Bring*, *Return*, *Open*, *Close*, *Pour*, *Mix** and a recognition rate of 76.6 percent, and the system developed by Rybski and Veloso [60] which provides *Walk*, *Stand*, and *Sit* activity recognition. One can imagine adding additional values with input from additional activity recognition systems as is available.

SPEECH Speech consists of words or phrases that could be extracted by a speech recognizer. There are actually numerous speech recognition systems [10, 27] that can recognize a large portion of the English vocabulary, thus to limit the amount of variables, we focus just on the words pertaining to the objects being recognized. According to a survey of speech recognition error rates [73], error rates for word recognition varies from 8.2 percent to 12.3 percent for n-gram and HMM systems.

COLOR For Color, we use a discretized version of the HSV (Hue Saturation Value) color space where there will be 9 categories of colors: *Red*, *Yellow*, *Orange*, *Green*, *Cyan*, *Blue*, *Magenta*, *Black* and *White*. Each color, except *Black* and *White*, can be determined according to the hue color scale which ranges from 0 to 360. *White* and *Black* are determined by the value (or brightness). If the value is above 90 percent, the color will be labeled *White*. If it is below 10 percent, it is labeled *Black*.

SHAPE The shape cue value is determined by the shape aspect ratio of a tight bounding box around the object/segmented region i.e.,

$$\text{Aspect Ratio} = \text{Bounding\_Height}/\text{Bounding\_Length}$$

We can then discretize the range from 0 to Image_Height to get our categories. For now, we will assume an image of size 1085x260 pixels and where we will want 8 categories. We will start with an even division of the range, but future cut off points can be chosen by looking at a histogram of the objects and their aspect ratios, so more categories can be placed in higher density regions for better distinction.

SOUND For sound, a list of possible sounds that could be made by the object is used. These values are determined by the UPC AED/C (Acoustic Event Detection and Classification) System [67], which had the lowest error rate of any other system as tested by [68]. The system can detect 12 classes of sounds including *Knock*, *Door Slam*, *Steps*, *Keyboard Typing*, and *Phone Ringing*, which has a recognition rate of 88.6 percent. As well as the Speech recognition system developed by [74] with kitchen sounds: *Cut*, *Pour*, *Mix*, and *Stir*.

Additional classes can be added as needed or available.

The set of objects used in the simulation is represented by: Bowl, knife, chair, cereal, spoon, phone, laptop, and orange.

Now that we have a clear definition of possible cues and their values, we can build a simulator to generate scenarios the agent may encounter. Given the set of objects, the simulator generated cues based on a predetermined model that represents the probabilities of a cue being produced given the presence of an object. It is the weights of this model that we attempt to learn.

In order to mimic the imperfection of the real world, noise was added to the data generated by the simulator according to the error rate of the recognition systems that the cues were based on, whenever applicable.

3.3 Results From Simulation

Using the cues and relationships defined above, we created various model representations that could represent the real world model of the interaction and dependencies between all the entities.

In order to test the parameters learned by the simulation, the true model used to generate the cues were compared with the probabilities learned from the data generated by the simulator. The error rate (the normalized sum of the absolute difference between the learned and true parameters) was then calculated for dataset sizes ranging from 1 to 100. As can be seen from Figure 3.3, the error rate somewhat plateaus after a data set size of 50. Note how the plateau occurs at an error rate of .3. This is because the error rate for the recognition systems which max out at around a .3 percent error rate interferes with the learning.

The fact that the weights include the error rate when learning the values, however, is not

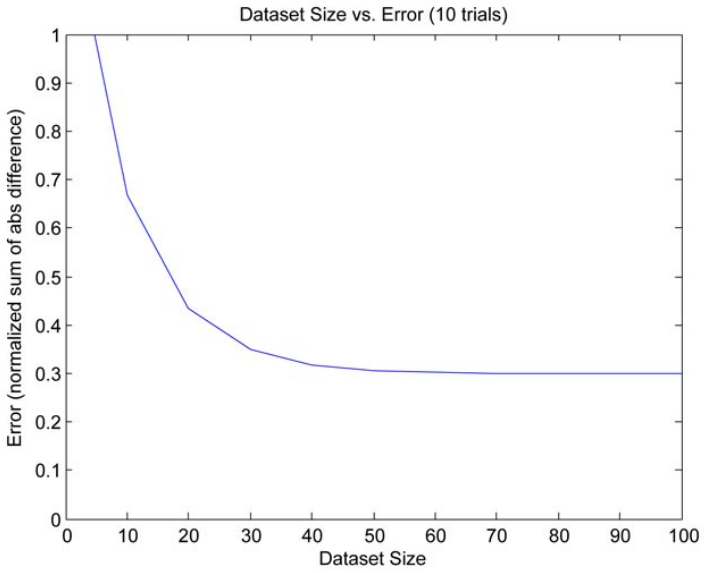


Figure 3.3: Error rate of learned parameters against dataset size using simulator

a bad thing. Actually, it allows the error rate to be a part of determining the amount of dependence that should be placed on that cue type. In Chapter 6, we see how a weight value that includes error can neutralize the effects of a bad cue type.

3.4 Learning from Real Data

In this section, we show training using real data. First, we show the labeling of data to be used in the training.

3.4.1 Labeling using ViPER

There are a number of video labeling products, but we chose to use **Video Performance Evaluation Resource (ViPER)** [33] by the Language And Media Processing (LAMP) at the University of Maryland because it allows you to easily define your own schema of labeling as well as allows for duplication and interpolation when labeling an object across a large number of frames.

Using ViPER, we developed an interactionable schema that can provide useful information about objects and their interactions. Thus, we labeled the following:

- Person
  - Face: Bounding box around frontal face
  - Body: Bounding box encompassing entire body seen
  - Hands: Bounding box around left and right hand
- Object
  - Label: Object class of that object
  - Location: Bounding box around object
  - Segment: Polygon around the true segmentation of the object.
  - Visibility: Vector that keeps track of frames where object visible.
- <Cue> - can be replaced with any cue type, ex: Activity.
  - Value: Value of the cue
  - Person: Person providing or related to the cue
  - Objects: Objects being affected or generating the cue

This schema allows for a large variety of learning opportunities. For example one could learn the average distance of an activity cue as related to the person doing the activity (for instance, the location of the face) from an object. This information can then be used to reduce the search area of the segment of the object being recognized.

We then can create using this schema and ViPER, XML files for easy processing which can then be used for easy training and performance evaluation. An example of such labeling can be seen in Figure 3.4.

3.4.2 Results

Training was done on two main sources:

- **Rachel Ray dataset**: Learning was done on 50 video clips from 1 to 30s, from 3 half-hour videos. Scenarios were unscripted.
- **LACE dataset**, University of Rochester [47]: Learning was done on 32 video clips from 1 to 3mn. Scenarios were scripted.

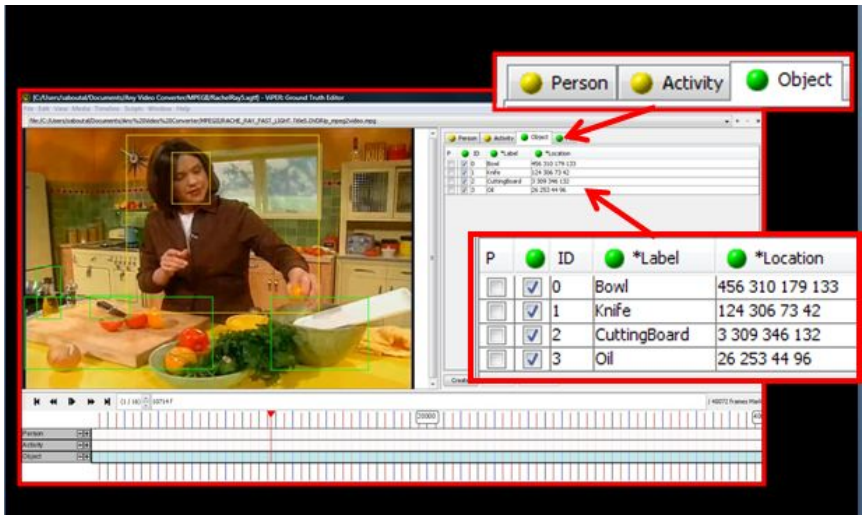


Figure 3.4: Example of labeling using ViPER

Figure 3.5: (Example of captured video frame from Rachel Ray cooking dataset and LACE Dataset

In Figure 3.5, an example video frame taken from each dataset is shown with the Rachel Ray example frame shown on the left, and the LACE dataset on the right.

In Figures 3.6 and 3.7, we provide tables showing the weights learned from the XML data generated by our ViPER schema. The weights represent probabilities, $P(o_i|\text{cv}_j)$, where o_i is the object category and cv_j is the cue value. For the tables shown, we focus on the activity cue values. For the Rachel Ray dataset, we labeled activity cue values: *Twist*, *Smash*, *Shake*, *Drop/Add*, *Cut*, *Mix*, *Pour*, *Squeeze*, *Pick up*, *Put down*, *Point*, *Scoop*. The object categories consisted of: Bowl, knife, oil jar, spice, wine bottle, cutting board, plate, grill, pan, and spoon. For the LACE dataset, we labeled the activity cue values: *Eat*, *Drink*, *Bring*, *Return*, *Open*, *Close*, *Pour*, and *Mix*. All the activity cue values except for *Mix* were taken from activity cue values labeled by the University of Rochester laboratory.

The object categories, also taken from the labels in the LACE dataset were: Bowl, knife, cup, cereal, spoon, cutting board, jug, fridge, place and cupboard.

In the results, one can see how certain activities are more strongly associated with a particular object such as *Pour* to bowl with a value of .8, while other activities such as *Put down* are more widely distributed across all the objects and thus less weighted when giving evidence for any particular object. Similar weights were learned for the speech, sound and vision cues.

In the LACE dataset, we see that some objects, which may be easily recognizable in the Rachel Ray dataset such as knife because of the strong association of *Cut* to knife may not be so in other datasets. Knife does not have a strong association with any activity value in the LACE dataset. This demonstrates how the weights for the object dictionary can be adapted to the given context based on the cue information available. On the other hand, since the interactions in the LACE dataset were scripted and the same set of interactions occurred with each object, there were a greater number of weights with value 1 than in the Rachel Ray dataset.

Object/ Activity	Twist	Smash	Shake	Drop/ Add	Cut	Mix	Pour (on/from)	Squeeze	PickUp (object/from)	PutDown (object/to)	Point	Scoop
Bowl	0	0	**.7**	**.9**	0	**1**	**.8**	**1**	**.4**	**.5**	**.3**	0
Knife	0	**1**	0	0	**1**	0	0	0	**.3**	**.3**	0	0
OilJar	0	0	0	0	0	0	**.2**	0	**.2**	**.1**	0	0
Spice	**1**	0	**.3**	**.6**	0	0	0	0	**.3**	**.3**	0	0
WineBottle	0	0	0	0	0	0	**.1**	0	**.1**	**.1**	0	0
CuttingBoard	0	0	0	0	**.8**	0	0	0	**.4**	**.5**	**.3**	0
Plate	0	0	0	0	0	0	**.2**	0	**.1**	**.1**	**.3**	0
Grill	0	0	0	0	0	0	0	0	0	0	0	0
Pan	0	0	0	**.1**	0	0	0	0	0	0	**.3**	0
Spoon	0	0	0	0	0	**.5**	0	0	0	0	0	**1**

Figure 3.6: Learned weights ($P(o_i|\text{cv}_j)$) for the Rachel Ray dataset and activity cue values

Object/ Activity	Eat	Drink	Bring	Return	Open	Close	Pour	Mix*
Bowl	**1**	0	**.1**	**.1**	0	0	**.5**	**1**
Knife	0	0	**.1**	**.1**	0	0	0	0
Cup	0	**1**	**.1**	**.1**	0	0	**.5**	0
Cereal	**1**	0	**.1**	**.1**	0	0	0	0
Spoon	**1**	0	**.1**	**.1**	0	0	**.4**	**1**
CuttingBoard	0	0	**.1**	**.1**	0	0	0	0
Jug	0	0	**.1**	**.1**	0	0	**.6**	0
Fridge	0	0	0	0	**.5**	**.5**	0	0
Plate	0	0	**.1**	**.1**	0	0	0	0
Cupboard	0	0	0	0	**.5**	**.5**	0	0

* mix was not part of the activities recognized by the Rochester group

Figure 3.7: Learned weights ($P(o_i|\text{cv}_j)$ for the LACE dataset and activity cue values

This training data then provided the weight values that were then used in the MCOR algorithm when recognizing objects (see Chapter 7).

3.5 Conclusion

In this chapter, we have provided a probabilistic framework for representing the interesting and challenging fact that some cues may be more indicative of an object category than others. We did so through the use of probabilistic relational models, which not only provided a way to learn these weights, but allowed for the learning of spatial and temporal associations and a confidence value based on the cue type.

We demonstrated the learning of these weights both on simulated data and data taken from real video datasets.

This framework allows for future flexibility in context learning as well as more complex association and relationships between entities such as object to object relationships yet to be explored.

Chapter 4

Segmentation, Tracking, and Cues

In this chapter, we focus on the tools used to segment and extract visual features and other cues for recognition and tracking. Since this thesis is focused on how the combination of multiple cue types can aid in the robust recognition of objects, we primarily focus on visual features and cue recognition systems which are simple to implement and can run in real time during recognition. Other works have focused their attention on the development of more advanced visual features and cue information (see related work), but the focus of this thesis is exactly on showing that simple visual features can be implemented with other cues. However, such advanced visual features and cues can be incorporated at any time into MCOR.

We begin by describing the segmentation technique, followed by the visual features we extract for use in recognition and tracking. Finally, we define the other cue types and systems used in MCOR.

4.1 Segmentation

In order to localize the cue information in an image for recognition, we need to be able to draw a bounding box around at least a portion of the object we wish to identify. Segmentation of objects is triggered by the extraction of a cue that is not already associated with a segmented region in the image. We use two techniques for segmentation: **Scale-Invariant Feature Transform (SIFT)** and a region growing algorithm. We use the SIFT method when possible, because of its robustness and high descriptive ability. If however, no match is found using SIFT, we use a color segmentation method.

4.1.1 Segmentation with SIFT

In this section, we describe how SIFT features are used in the segmentation. Section 4.2.2 provides the details on the calculation and training of the SIFT features. For this section, we can assume we are given an object dictionary with SIFT features already learned from a training set. The algorithm for segmentation with SIFT (Alg. 1) then goes as follows: First, we begin with the extraction of SIFT features (details in Section 4.2.2). These SIFT features are compared with those found in the object dictionary using the SIFT similarity measure defined in Section 4.2.2. If a match is found above a given threshold, the training image saved in the object dictionary used for the learned SIFT features is oriented and scaled to match the recognized region using an affine transformation specified by the Hough transform used in the SIFT object recognition (see Section 4.2.2). This boundary represents the location of the candidate object. Color region growing is then used to extract color and shape information. If however, no match is found, segmentation is based on the color region growing algorithm outlined below.

Algorithm 1: SIFT Segmentation

Given:
- object dictionary, OD, with learned SIFT features and object categories, O;
- set of extracted SIFT features, T;
- SIFT similarity threshold, θ;
- SIFT similarity measure, `SIFTSimilarity` (T, o_i, OD);
- frame f from video

for each $o_i \in O$ **do**
    s = `SIFTSimilarity` (T,o_i,OD);
    if $s > \theta$ **then**
        I ← getImage(OD,o_i,T);
        `// Get SIFT training image saved in dictionary which matches extracted features`
        It ← transform(I,Hough(T));
        `// Transform the image according to Hough transform parameters`
        segment ← apply(It,f);
        `// The segment is then the location of the transformed training image in the frame`
        colorshapeFeature ← extractColorShapeInfo(segment);
        save(colorshapeFeature,o_i,OD);
        return;
    end
end
Go to color region growing segmentation;

4.1.2 Segmentation with Color Region Growing

For the color and shape segmentation, we use a standard color region growing algorithm (Alg. 2), where the segmented region grows until it reaches neighboring pixels that are not of a similar color.

Algorithm 2: Color Region Growing

Given:
- pixel p at location x,y with RGB colors: p.r, p.g, p.b;
- empty region, S;
- threshold, θ;
- unchecked-8-Neighbors(p), returns set of pixel neighbors to pixel p that has not been checked for color similarity

```
while not empty do
    for n ∈ unchecked-8-Neighbors(p) do
        edist = sqrt((p.r − n.r)^2 + (p.g − n.g)^2 + (p.b − n.g)^2);
        if edist < θ then
            add n to S;
            RegionGrowing (n);
        end
    end
end
```

More specifically, the algorithm works by starting with a single defined pixel. This pixel is specified using the spatial association offset with the maximum probability value, Δp^*, given the cue value that initiated the segmentation. This probability value was given by the Gaussian distribution learned from the training dataset described in Chapter 3:

$$\Delta p^* = argmax_{\Delta p} P(\Delta p | \mathrm{cv}_j) \tag{4.1.1}$$

Then, for each neighbor in the 8-connected neighbor pixels that are unchecked, i.e. have not already been checked for color similarity, surrounding the target pixel, we determine whether they are within a Euclidean distance from the color of the chosen pixel. If it is, the pixel is added to the region. If not, the algorithm stops.

In Matlab, this algorithm is slightly modified to take advantage of Matlab's matrix optimizations. We do this by first determining the Euclidean distance of all the pixels in the image generating the distance matrix, E. We then compare this matrix to the threshold value returning a binary matrix, where values are 1 for pixels, whose Euclidean color dis-

Figure 4.1: Example of color segmentation results used in color and shape visual features for a knife, cereal box, and cup

tance is above the threshold, and 0 for all others. We then return all the connected regions (through the 8-neighbor method defined above) using a predefined Matlab function. The Matlab optimization not only runs quickly for segmenting the target region, it returns other potential object locations based on the color cue. This allows us to skip an additional step of finding all other segments with the learned vision cue at a separate time point in the MCOR recognition algorithm.

For segments which are not visible at the time of the cue occurrence, the algorithm goes back the number of frames specified by the temporal association described in Chapter 3. Visual features are then learned from that segment, the segment is assumed to be in that location until another cue interacts with it in which case it is assumed visible again and can be tracked.

Using this segmentation technique, we are able to localize candidate object locations. In addition, based on the color segment extracted, we can retrieve color and shape visual features. An example of color segmentation can be seen in Figure 4.1. Segmented regions in the figure are filled in with a uniform color (red). The figure shows example segmentation results for a knife (left), cereal box (middle) and cup (bottom). Note the better segmentation of the homogeneous colored knife and the less precise segmentation of the more textured cereal box. The color segmentation results provide color and shape information used in MCOR as a visual feature.

4.2 Visual Features

There are two types of visual features currently implemented in the MCOR framework: (1) Color and shape features and (2) Scale-Invariant Feature Transform (SIFT) features. The first provides a good reflection of objects with large homogeneous color regions. The second provides a representation of objects with texture.

4.2.1 Color and Shape

The color and shape feature we use is based on the visual features of the segmented region. It is a simple calculation, which allows faster processing during recognition.

Rather than simply use the color or shape features alone, we combine them into one in order to provide greater discrepancy, i.e. to reduce the set of object classes which may have the same color or the same shape to those which must have both. Since segmentation only occurs when cues are extracted, having the greater number of less-discriminative, separated color and shape features would result in a greater number of needless regions that have to be assessed for recognition. Combining the two features into one cue also cuts down the dimensionality of the potential object candidates that needs to be recognized.

While this vision cue provides a good description of objects with large homogeneous color regions, problems occur when dealing with objects that have a lot of texture and color variation. To address this weakness, we introduce the use of SIFT features.

4.2.2 Scale-Invariant Feature Transform in MCOR

Previously, when a visual description of the object being identified was grabbed, the visual description consisted of color information and the aspect ratio of the bounding-box around the segment produced by a region growing color segmentation algorithm [1]. In this section we describe another visual feature, in addition to the color and shape information, where we grab SIFT (Scale-Invariant Feature Transform) features.

SIFT Features

SIFT features [39] are well-known descriptors used widely through computer vision and object recognition tasks. These features are useful because they provide highly descriptive

texture-based features which are robust to most changes in scale and rotation.

Because it would be too computationally intensive and unnecessarily repetitive to calculate a feature for every pixel location, we calculate SIFT features only at interest points in the image such as areas with rapid intensity change called "corners" [24]. There are numerous corner detectors that can be used. We focused our approach on the common Harris corner detector [24], defined by the equation below:

$$A = \sum_u \sum_v G(\mathbf{u}, \sigma) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} < I_x^2 > & < I_x I_y > \\ < I_x I_y > & < I_y^2 > \end{bmatrix}$$

A is a Harris matrix whose eigenvectors and values can be used to determine areas of interest, i.e. if the first and second eigenvalues have a zero response, there are no features of interest. If one of the eigenvalues has a large response, an edge is present. If both eigenvalues have a large response, a corner is found. Brackets, <>, indicate averaging over the summation of all pixel locations, $\mathbf{u}$, and $G(u, v)$ corresponds to the Gaussian function used.

In SIFT, key locations are further deciphered by using the maxima and minima of the results of the difference of Gaussians, which are applied in scale space to a series of smoothed and sampled images in an image pyramid [39].

Dominant orientations are assigned to each interest point using a 128 dimensional vector formed from a histogram of image gradients in the neighborhood of the interest point [39]. This produces key points that are more stable and robust to changes in orientation and scale.

Object Recognition with SIFT Features

Object recognition then occurs by first computing the SIFT features as outlined above. Each key point specifies 4 parameters: 2D location, scale and orientation. To increase recognition robustness, a Hough transform is used to identify the cluster of matches that vote for a particular object pose. Each keypoint then votes for the set of object poses that are consistent with the keypoint's location, scale, and orientation. Locations in the Hough accumulator that accumulates at least 3 votes (3 was selected by Lowe as the optimum minimum amount of features needed to robustly recognize an object) are selected as candidate object matches. Each cluster identified by the Hough transform undergoes a geometric verification step in which a least-squares solution is performed for the best affine projection parameters relating the training image to the new image. This also allows us to

segment the object around the now matched and oriented training image. Features that do not fall within half the error range based on the least-squares solution are rejected as outliers.

The final decision to accept or reject a model hypothesis is then based on a probabilistic model defined by Lowe [38]. Appendix D provides the details of this probabilistic model. As an overview, given the projected size of the model (larger models expect to have more false positives), the number of features within a region, and the accuracy of the fit, the model computes the expected number of false matches to the model pose. A Bayesian analysis then gives the probability an object model (as taken from the object dictionary) is present based on the actual number of matching features found.

We use this probability as the *SIFT similarity measure* when determining its vote, both for recognition and segmentation. For segmentation, we use the .98 threshold used by Lowe [39].

In MCOR, we start with a set of initial training images for each object from which SIFT features are taken and stored in a database in the object definition, i.e. a 128 dimensional vector for each of the features extracted from the training images are saved into the object dictionary. Figure 4.2 shows examples of SIFT models extracted from training images for the object categories, knife (top-left), cereal box (top-right), and wine bottle (bottom). Each point in the figure corresponds to the location of an extracted SIFT feature. An enlarged version of the cereal box model is given in Figure 4.3. The original training image is shown on the left and the model with the SIFT features of the model (marked with circles) is on the right. Note how the objects with a more interesting texture pattern (such as the cereal box) provides more interest points for the SIFT model.

In the MCOR framework, additional model images are added to the database if a segment is recognized as a particular object category by the MCOR algorithm. In this way, new visual features can be learned for each object definition as recognition occurs.

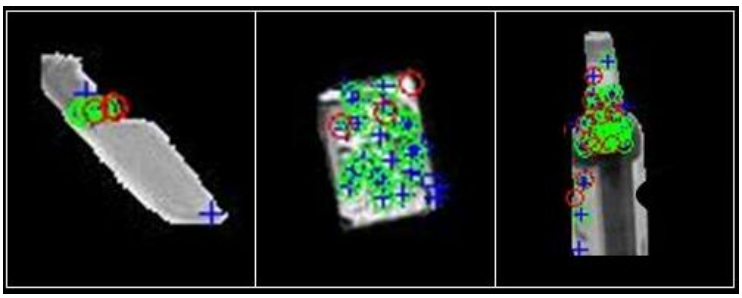


Figure 4.2: Examples of SIFT models extracted from a training example of a knife, cereal box, and a wine bottle

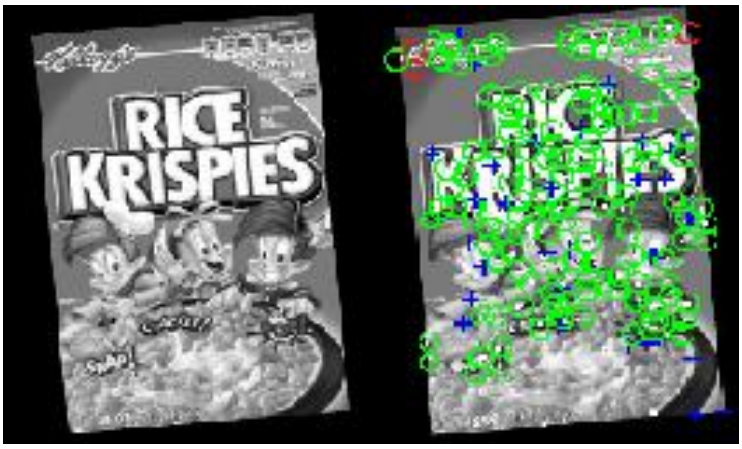


Figure 4.3: Enlarged example of cereal box SIFT model extracted from a training example

If a model of an object is already stored in the object dictionary, then we look for that object in each of the frames. Figure 4.4 shows the extracted SIFT features taken from a frame during recognition. These key points are then compared with the key points saved in each object dictionary using the Hough transform and SIFT similarity measure to determine the strength of the match. The similarity is used in the evidence equation for recognition (Equation 5.1.1). We assume that a SIFT model of an object is strongly associated with that object, i.e., we assume the value of the weight is set to 1. Similarity then determines the extent of the evidence provided by the SIFT response.

The inclusion of the SIFT features provides an added benefit to the MCOR algorithm by providing the ability to recognize textually complex objects which could not be effectively represented with color region segmentation. Figure 4.4 shows examples of extracted SIFT features, represented by + signs, from frames taken during recognition. The example frames contain a knife (top-left), cereal box (top-right), and wine bottle (bottom). Note the large number of matched SIFT features for the cereal box because of its large amounts of texture (an enlarged example is given in Figure 4.5), and the small number for the less textured knife. Complex textured items such as the cereal box are recognized well using a SIFT model, since it provides numerous distinctive texture features that can be used for recognition. When color region segmentation is used on textually complex objects such as the cereal box (see Figure 4.1 for example), small and irregular regions result, making recognition for these objects based solely on color segmentation information unreliable. With the inclusion of SIFT features, these objects are better recognized in the MCOR framework.

It is important to note, however, that using the SIFT features alone also does not provide enough information to reliably recognize all objects. In some cases, features taken from color region segmentation provide a more reliable description of some objects than a SIFT

Figure 4.4: Example of extracted SIFT features for frames containing a knife, cereal box, and wine bottle

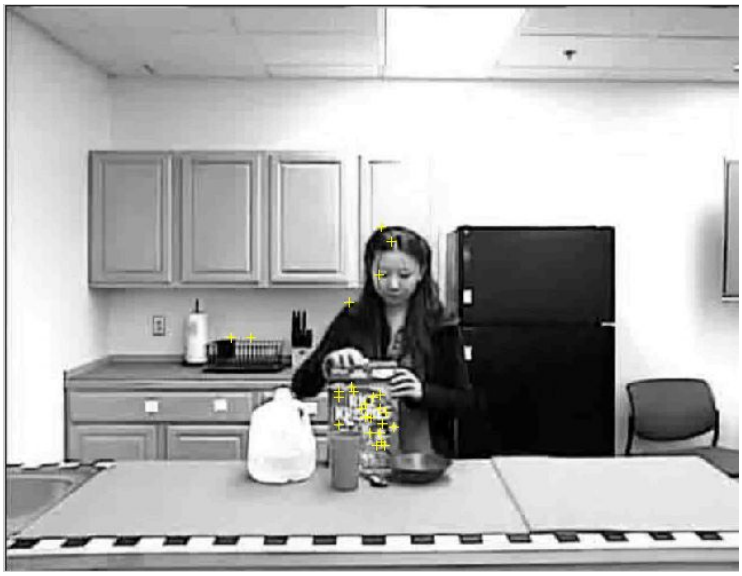


Figure 4.5: Enlarged example of extracted SIFT features from a frame taken during recognition

model. Color segmentation provides better features for objects that are made of a large homogeneous color region, for instance, the knife in Figure 4.4. Because of the smooth texture of the knife, the SIFT features are not reliable enough to find the knife in the frame. Color segmentation through region growing, however, is able to provide a more comprehensive representation of the knife (represented in red in Figure 4.1).

Thus, MCOR is able to utilize various visual descriptors, in addition to non-visual cues, to more robustly recognize objects of differing appearances and properties. SIFT features provide strong visual evidence for textually complex objects, while region growing color segmentation information provides good visual descriptors of objects that contain a large homogeneous color region.

4.3 Tracking

The problem still remains of tracking these segments and visual features across the different video frames. To deal with this issue we use a dual tracking system that changes from SIFT tracking to contiguous color region tracking depending on the reliability of the features available.

Based on which visual features were used to segment the region as described in the Segmentation section, either SIFT features or the color region is used for tracking.

4.3.1 SIFT Tracking

If SIFT features were used to segment the object, then tracking is done by looking for the matched SIFT features in the next image. Matching is done using the the SIFT similarity measure described above. Tracking is then done by determining the corresponding region in the next frame. This region is determined by the similarity measure used in the proximity

algorithm defined in algorithm 3.

Algorithm 3: Proximity Algorithm

Given:
- region, r, to track;
- object dictionary with SIFT features,OD;
- clusters, C, given by Hough Transform;

for $c \in C$ **do**
  sc = `SIFTSimilarity` (c);
  MaxMatchedRegions = all $argmax_{c \in C} S(c)$;
  for $cc \in$ MaxMatchedRegions **do**
    OD(cc) = $\sqrt{(cc.centroid.x - r.centroid.x)^2 + (cc.centroid.y - r.centroid.y)^2}$;
  end
  r = $argmax_i \in maxMatchedRegions$ D(i) ;
end

4.3.2 Contiguous Region Tracker

In the case when segmentation was done using the region growing algorithm, we use the contiguous region tracking code developed by Felix von Hundelshausen [72], which tracks contiguous regions of similar color by growing or shrinking the region accordingly. The segments are then tracked from image to image so a re-segmentation is not needed.

This algorithm however only works when the region being tracked has an area of overlap between its current position and its position in the next frame. If a gap occurs, than a research of the space is required to find the segment again.

When this occurs, we search the space using the proximity algorithm 3 outlined in the SIFT section, except rather than using the SIFT matching criteria, we use the Euclidean color distance between the two regions.

The proximity algorithm for both SIFT and color region growing fails when there is an object with an equivalent appearance (either in terms of SIFT features or color and shape) is closer to the region being tracked than the object being moved. This can be solved using a prediction model such as the Kalman or particle filtering. Since this scenario, did not come up frequently, we leave this to future work.

4.3.3 Tracking Avoidance

In addition, if the camera viewing angle of the video data is known to be stationary, the MCOR algorithm takes advantage of the fact that the objects being recognized will not move in the scene unless interacted with by a human. Not all cue types have the capability to move an object (for instance, speech cues never cause an object to be moved), however other cues (such as activity cues) can; we call these types of cues, ***mover cues***. Since during recognition we know when a particular cue begins and ends, we can determine when this interaction occurs. We then turn off tracking at any point in time a mover cue is not currently in progress. This method helps reduce the computational costs of tracking during that time. Table 4.1 shows the reduction in tracking time averaged across all objects for the meeting dataset, cooking dataset and Gupta dataset.

Dataset	Meeting	Rochester	TV Cooking	Gupta
% Reduction of Tracking	70	72	1	9

Table 4.1: Percent Reduction of Tracking Time By Taking Advantage of Mover Cues

Note how the meeting and Rochester datasets have a larger reduction in tracking time. The greater reduction occurs because these datasets contain more objects in the dictionary, which are interacted with, but only for short periods of time. The rest of the time the objects stand static. In the Gupta dataset, there is only one object per video found in the object dictionary and this object is always the one being moved. Thus, tracking occurs much of the time. For the TV Cooking dataset, there is a constant slight camera movement requiring tracking to be consistently on, thus there is such a small percentage (1 percent) of reduction.

4.4 Non-Visual Cue Types and Values

This section provides a description of the cue types and values that we used for my thesis work. For each cue source, we describe the exact output including what cue information is given and used by the MCOR algorithm.

4.4.1 Speech and Sound

There was some cue information which needed to be inputed manually. This includes speech and sound information. Note, at any point in time, it is possible to replace any of

the hand encoded information by an automated system.

Speech Information This gives cue information in imitation of speech recognition systems. At base levels, the set of words that can be returned will correspond to the object labels of the objects to be recognized. It is possible for particular datasets to include such phrases as *"look that up"* in order to identify a laptop. These speech cues will be clearly indicated when results are given. For each frame, a phrase or word is returned along with a pixel location representing the source of the speech (such as the mouth of a person as located by hand).

Sound Information This gives cue information in imitation of sound recognition systems. More specifically, we primarily used cues that could be recognized by the UPC AED/C (Acoustic Event Detection and Classification) System [67]. Thus, labels representing 12 possible classes of sounds including *Knock*, *Door Slam*, *Steps*, *Keyboard Typing*, and *Phone Ringing* will be returned. For each frame, a label indicating one of the 12 sound classes will be returned along with a pixel location representing the source of the sound (such as the center point of the phone that is ringing, which will be located by hand).

4.4.2 Activity

In order to get the MCOR algorithm to work on real robot data, it is necessary to have cues that can provide useful information. The most difficult of these to obtain is the activity recognition. In order to make the algorithm practical for real-time data, we created a simple activity recognizer. From these systems, the MCOR algorithm can extract activity cues, i.e., for every frame, we get an activity label associated with each person in the scene and a pixel value representing the location of the label.

Activity Recognition Using Face Displacement

One of the systems we employ to get activity information is a system designed by Rybski et al. [60]. This is an activity recognition which can label the activities of multiple people in a scene, as long as their faces are pointed towards the camera. At each frame, the system outputs one of six activity labels (*Sit*, *Stand*, *Fidget Left*, *Fidget Right*, *Walk Left*, *Walk Right*) for each person at a given pixel value (determined by the location of the person's face)[60].

Activity Recognition using Colored Hand Band

The standard approach to activity recognition is an HMM based on observations such as the change in the x and y position of a particular tracking point. There are many advanced activity recognition systems out there, but since our focus is primarily on the use of the cue, rather then developing an incredibly intricate cue recognition system, we have developed an activity recognition algorithm that can produce quick and simple results for some key activities.

We based the algorithm on HMMs and Viterbi algorithm, but doing so through a hierarchical structure. Below we first describe this model, then show how it is used for recognition.

Hierarchal Model Many of the activities we are interested in tend to be larger activities. Initially, we attempted to recognize using a single tier recognition system where large activities movements were attempted to be recognized by classifying the change in the x and y movement of the centroid of the region being tracked. In our case, the region is a bright pink wristband worn by the subject.

This approach, however, presented a problem, since for such activities as eating, you would get a very large distribution in the change of activities. In order to solve this problem, we divided the activities into smaller movements, which would later be used as observations for the larger activities.

Small Movements The smaller movements consisted of *Pick Up*, *Put Down*, *Move Left*, *Move Right*, and *Stay*. These movement distributions could then be easily separated and, thus, more easily recognized.

The HMM used to describe the small movement states is shown in Figure 4.7. Dx, dy represent the change in x and y coordinates. In order to do the recognition, we used the Baum-Welch algorithm to learn the parameters of the HMM (see Figure 4.8), including both the emission and transition probabilities. We then used the Viterbi algorithm (see Figure 4.9) to recognize the small movements.

In order to make the recognition online, we used a very small window for the Viterbi algorithm. Figure 4.9 shows an example of the recognition of a small movement. The Viterbi algorithm generated results with a .96 accuracy.

Large Activities The larger activities, *Eating*, *Writing*, *Talking*, and *Other* states (other states include all small movements), could then be recognized using the smaller movements

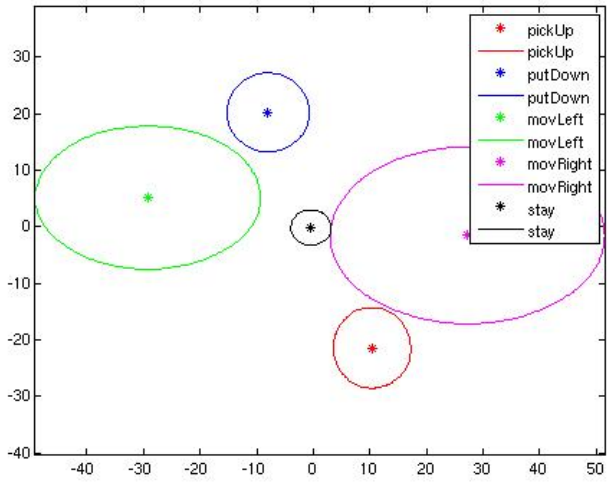


Figure 4.6: Gaussian Models of the observations representing the small movements

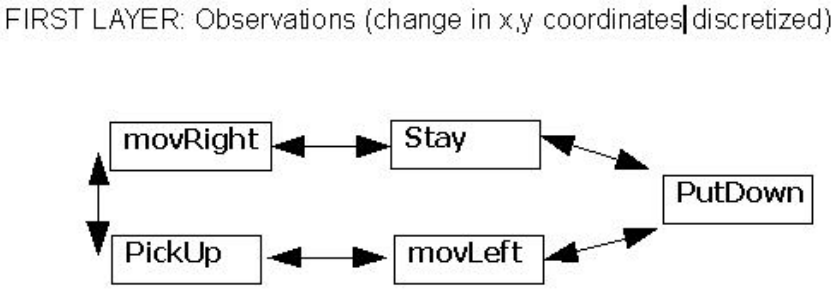


Figure 4.7: HMM representing the small movement states and their observations

as observations. The technique was the same as that used for the smaller movements, i.e., use the Baum-Welch algorithm to calculate the parameters of the HMM (see Figures 4.10 and 4.11), including both the emission and transition probabilities. We then used the Viterbi algorithm for recognition.

Activity Recognition By Hand

We label any additional activity information that falls out of the scope of the two activity recognition systems (Rybski and Color-band defined above) by hand. This includes activities difficult for current fast activity recognition algorithms, such as *Put Down*. This is also useful for simulated data in which the typical video data would not be available for

Emission Prob:

	Up	Down	Left	Right	Stay
dy > 10 & dy>dx	.001	**.911**	.020	.017	.007
dy < -10 & dx<dy	**.901**	.002	.004	.017	.006
dx > 10 & dx>dy	.025	.020	.001	**.933**	.007
dx < -10 & dx>dy	.033	.029	**.952**	.017	.008
-10<dx&dy<10	.040	.035	.011	.017	**.970**

Transition Prob:

	Up	Down	Left	Right	Stay
Up	.04	.83	.07	.25	.01
Down	.80	.05	.08	.15	.20
Left	.01	.08	.70	.21	.02
Right	.01	.04	.06	.21	.02
Stay	.10	.01	.09	.17	.75

Figure 4.8: Parameters learned for the HMM model of the small movements using the Baum-Welch algorithm

the activity recognition systems. The set of possible activities are: *Sit*, *Stand*, *Erasing*, *Pointing*, *Put Down*, *Talking*, *Writing*, *Pick Up*, *Drinking*, and *Eating*.

In some cases, we simulate activity recognition results of real systems, when the systems themselves are not available. This is the case with the Gupta Dataset [23].

4.5 Conclusion

In this chapter, we have outlined the vision and cue tools used to segment the image, extract visual features, and track segments across image sequences, as well as recognize the various cue values. We have shown the usefulness of using both color segmentation and color-shape visual information for identifying objects with large homogeneous colored regions and using SIFT features for textually complex features. These tools can then be utilized in the MCOR recognition algorithm for localizing information in the image and finding new object candidates based on learned visual descriptions.

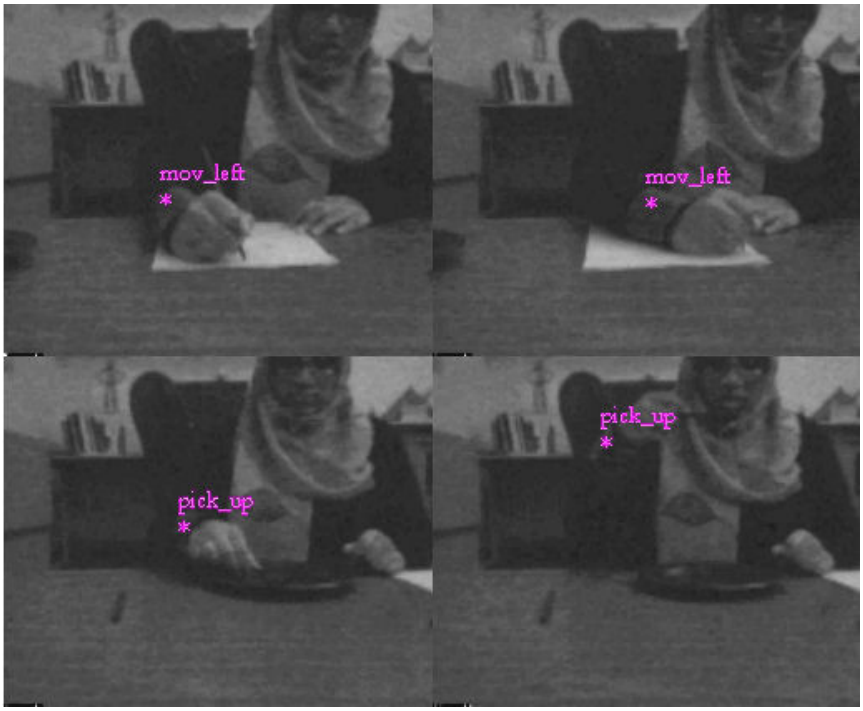


Figure 4.9: An example of the activity recognition of a small movement on data from the Sony QRIO observing a human

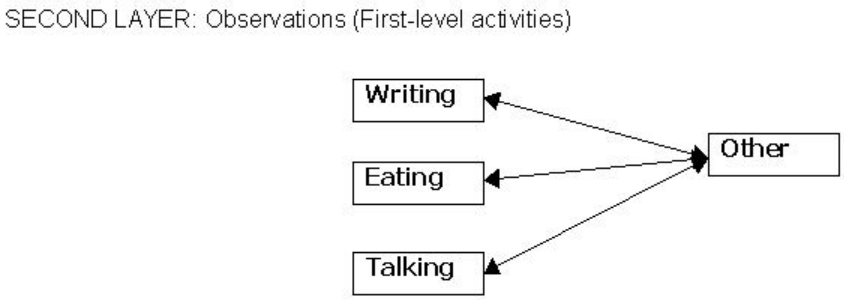


Figure 4.10: HMM representing the large activity states and their observations

Emission Prob:

	Eating	Writing	Speaking
Up	**.45**	.10	.00
Down	**.45**	.05	.00
Left	.05	**.60**	.00
Right	.00	**.20**	**.10**
Stay	.05	.05	**.90**

Transition Prob:

	Eating	Writing	Speaking	Other
Eating	.80	.00	.90	.30
Writing	.00	.85	.00	.40
Speaking	.00	.00	.00	.20
Other	.20	.15	.1	.10

Figure 4.11: Parameters learned for the HMM model of the larger activities using the Baum-Welch algorithm

Chapter 5

Incremental Discrimination and Recognition

In this chapter, we present the recognition algorithm which makes up MCOR. This chapter describes how all the previous pieces including the multiple cue representation, weights, and visual information are integrated in a coherent framework. The chapter then describes how the algorithm's use of the sequential introduction of cues can be used to disambiguate object categories in an incremental discrimination framework. Visual data captured by the Sony humanoid QRIO robot is used to demonstrate this discrimination.

5.1 MCOR Algorithm

The ability to recognize objects using multiple sources of information is rooted in the MCOR algorithm. In this section, we describe all the primary steps of the recognition algorithm including: region extraction, calculation of evidence, object recognition, and generalization. The overall algorithm for object recognition with multiple cues, i.e., Multiple-Cue Object Recognition (MCOR), can be found in Algorithm 4.

Algorithm 4: Multiple-Cue Object Recognition

Given: an object dictionary (Chapter 2), with:
- a definition for each object, o_i, in the set of object categories, O;
- a set of definition cues for each object, C_i;
- a cue value, cv_j for each definition cue, $c_j \in C_i$;
- a spatial association, Δp_j;
- a temporal association, Δf_j;
- a weight, $w_{i,j}$;
- a similarity measure, $s_{j,l}$ between definition cue c_j and extracted cue c_l;

for each frame of video, F_t **do**
  Extract all cues that belong to $\bigcup_i C_i$;
  for each new cue extracted, c_l, with cue value, cv_l and matching definition cue, c_{j^*} **do**
    Get current position, P_l;
    if $P_l - \Delta p_{j^*}$ at $f_{t-\Delta f_{j^*}}$ is within any region $r_k \in R$, where R is the set of segmented regions **then**
      Store c_l in C_k, the set of cues attached to that region;
    else
      Extract a new region, r_k, at position $P_j - \Delta p_{j^*}$ and frame $F_{t-\Delta f_{j^*}}$ and store it in R;
      Store c_l in the currently empty C_k;
    end
  end
  for each region, $r_k \in K$ **do**
    for each object, $o_i \in O$ **do**
      `// Calculate the evidence,` $e_{k,i}$`, that region` r_k `is object` o_i`:`
      $e_{k,i} = \sum_{c_j \in C_i} \sum_{c_l \in C_k} w_{i,j} s_{j,l}$ if the cue type of c_j is not the same as c_l, then $s_{j,l} = 0$;
    end
    `// Region` r_k `is then recognized as the object with the greatest evidence, if it is above a threshold,` θ`:`
    $label_k \leftarrow \text{argmax}_i\, e_{k,i}$, if $\max e_{k,i} > \theta$;
    Add all cues, $c_l \in C_k$, to the set of cues in the object definition, C_{label_k}, if $\vee c_j \in C_{label_k}, s_{j,l} \neq 1$;
    if current label, i.e., $label_k$ at f_t is different from $label_k$ at f_{t-1} and $label_k$ at f_{t-1} exists **then**
      Remove all $c_l \in C_k$ added before f_t from $C_{o_{old}}$, where $o_{old} = label_k$ at f_{t-1} ;
    end
  end
end

Given an object dictionary (as defined in Chapter 2) with a set of definition cues , C_i, associated with each object category, o_i, in the set of object categories specified, O, and a video, we can begin object recognition:

For each frame of the video, F_t, the first step is to extract all cues which belong to the union of the set of all cues in all of the object definitions, i.e., $\bigcup_i C_i$. In addition, the type of cues are extracted is dependent upon which tools are available. For instance, if an activity recognizer is implemented, it could be used to extract activity cues. If a speech recognizer is implemented, speech cues can be extracted, and so on. That is the major benefit of this algorithm: it does not require any specific type of extraction method, but rather utilizes whatever is available. It is this characteristic which allows the algorithm to be run by any robot or computer under any circumstances. In addition, it is the various recognizers that will track and extract cues from the interaction of humans with the object.

5.1.1 Region Extraction

Processing then continues with each new cue extracted, where a new cue is defined as a cue that was not present in the previous frame, F_{t-1}, with the same cue value cv_l and at a position P_l less than a cue-specific distance away. Then, for each new cue, c_l, the current position, P_l will be retrieved. This calculation is also cue-specific. Figure 5.1 gives an example of cue extraction on a frame taken during recognition. In the example, an activity cue is extracted with a cue value of *Sit*. The object dictionary with the object categories, cue values, and weights used during recognition at that time step is shown in the top-left corner.

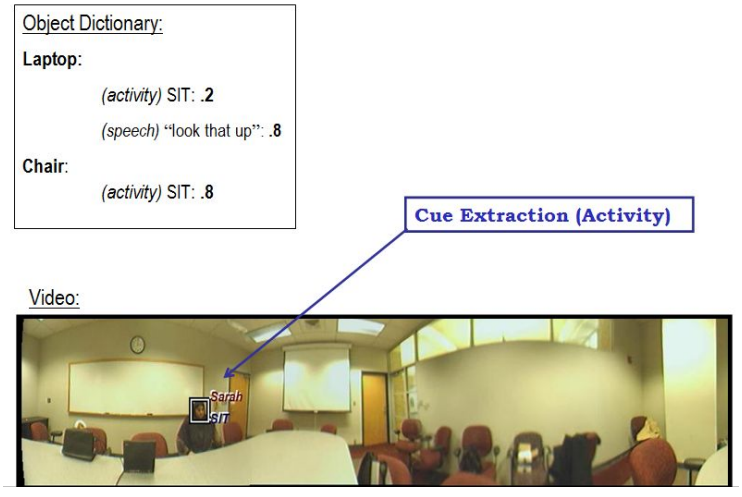


Figure 5.1: Example of cue extraction by the MCOR algorithm

In order to determine the spatial and temporal association to find the region where the extracted cue information should be attached, we retrieve the matching definition cue, c_{j^*} in the object dictionary, which matches the extracted cue value cv_l. We know that a match must be found, because only extracted cue information, whose value is present in the object dictionary, is used as outlined in the previous section. If the predicted location of the object belonging to that extracted cue specified by the spatial association,Δp_{j^*}, i.e., $P_l - \Delta p_{j^*}$ is within a region, r_k, from the set of segmented regions to be recognized as objects, K, that cue, c_l, is stored in a set of cues that belong to that region, r_k, i.e., C_k. If, however, $P_l - \Delta p_{j^*}$ does not fall within an already segmented region, a new region will be segmented at that location, but from frame $F_{t-\Delta f_{j^*}}$, where the object is clearly visible. This new region is then stored in K. As described in Chapter 4, there are two possible segmentation method SIFT and color. The segmentation method used depends on the availability of the SIFT features. An example of region extraction through color segmentation is shown in Figure 5.2.

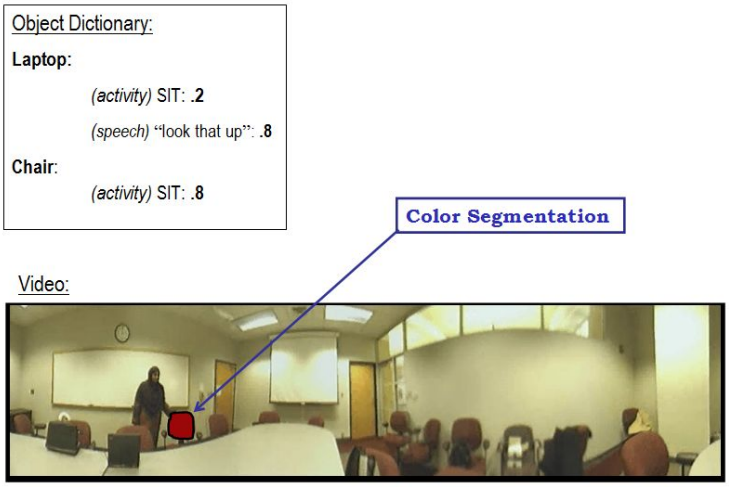


Figure 5.2: Example of segmentation for region extraction by the MCOR algorithm

With the regions extracted, we determine the evidence for each object category using the cues extracted.

5.1.2 Calculation of Evidence

In order to determine the presence of an object, we use an accumulator array as defined by the Hough transform approach (see Related Work Chapter 8)

In a typical use of the Hough transform, each feature/cue votes for a set of parameters. For object recognition, features often vote for a set of object properties such as pose position, scale, rotation, category and x,y location of the object center.

In a typical framework, a 3-dimensional Hough transform is used consisting of the object category, x location, and y location of the object center. The Hough transform is represented by an accumulator array, where the 3-d space is divided into bins of a given size (Size produces a trade off between evidence strength and location accuracy). Each feature extracted from an image then adds a weighted vote to a bin specified by the learned offset between a feature and an object center from training data. An object is then given the label to the bin that has the highest activation value, which is above a given threshold.

In MCOR, we use a modified version of this Hough transform space, where instead of dividing the entire image into corresponding bins in the accumulator array, bins represent each region currently segmented by the MCOR algorithm. Thus, we have a 2-d Hough transform consisting of the segmented region (position) parameter and object category parameter. Position parameter information for each cue is given by the spatial association. Currently, the spatial association represent the most likely offset between a cue value location and object location association as defined in Chapter 3. In future work, this can be extended be modified to vote for all regions within a given probability range and scaled according to the probability of the offset.

Using this Hough transform, the calculation of evidence is done by taking each region, r_k, in K, and each object category, o_i, in O to calculate the evidence, $e_{k,i}$, that r_k should be labeled o_i according to the following equation:

$$e_{k,i} = \sum_{j \in C_i} \sum_{l \in C_k} w_{i,j} s_{j,l} \tag{5.1.1}$$

where $w_{i,j}$, as described earlier in Chapter 3, is the predefined weight representing the strength of the association between the cue, j, in the object definition, C_i, and the object category, o_i. $s_{j,l}$ is the similarity between the cue, c_j, and the cue, c_l, belonging to C_k. If the cues are not of the same type, then $s_{j,l} = 0$. This value is multiplied by the probability of the spatial association.

5.1.3 Object Recognition

We use the calculated evidence to determine which object category to assign each region, i.e. region, r_k, is then recognized as the object with the greatest evidence, if it is above a given threshold, θ, i.e.,

$$label_k \leftarrow \text{argmax}_i \, e_{k,i}, \text{ if } \max e_{k,i} > \theta$$

The threshold is learned based on training data. See Chapter 6, for details on how this value is calculated. Once each of the regions above the threshold value is assigned a category, generalization of new cues is possible. Figure 5.3 shows an example of recognition and generalization. In the figure, a segment (the segment is behind person in this frame) is recognized as a chair by the MCOR algorithm based on the calculation of evidence. The evidence calculation is found in the top right corner. Based on this calculation, the chair object category has the greatest amount of evidence because of the extracted *Sit* cue value. Generalization then occurs, where color and shape information are grabbed from the recognized region and added to the chair definition in the object dictionary. This allows other chairs in the scene to be recognized.

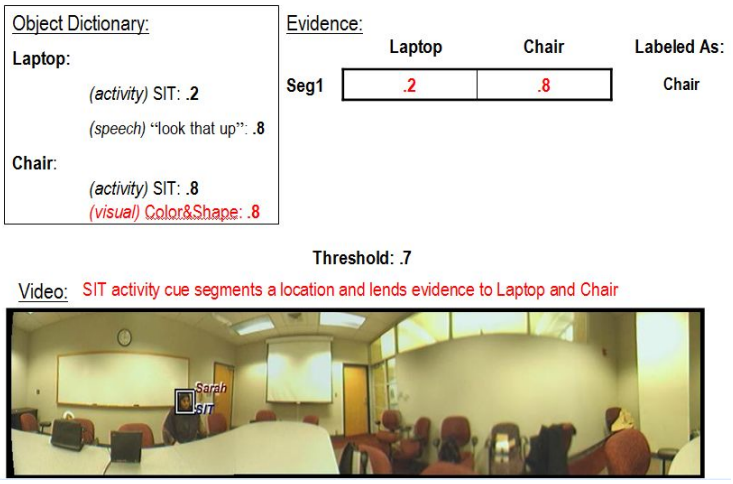


Figure 5.3: Example of recognition and generalization by the MCOR algorithm

5.1.4 Generalization

An important part of the algorithm is its ability to take cues, which may not have been previously associated with an object definition, and add them to object categories. This provides two important properties: (1) the ability to ascribe cue values to new object categories for additional evidence and (2) the ability to utilize cue types which were not found in the training dataset. This is done by the following method:

All the extracted cues, c_l, in the set, C_k, of cues belonging to the region, r_k, will now be added to the object definition of o_{label_k}, i.e., C_{label_k}, if there is no definition cue in the object dictionary with the same cv_l, since duplicate cues are not allowed, i.e., $\vee c_j \in C_{label_k}, s_{j,l} \neq 1$. If the same cue value is found, we can use this value to strengthen the weight for that value based on the weight update method found in Chapter 3. This step generalizes the cues learned from this particular object to all objects of the same class, so now new objects based on the newly added attributes can be found. For cue values belonging to cue types that are not found in the training dataset, and thus have no corresponding weights, we use a predefined default starting weight, which can then be updated as new examples are seen. Depending on the fact that objects of the same category within the same context often have similar features, we use a high (above threshold) value for the new cue types. Thus, with this method, we are able to not only generalize known cue values to new object categories, but we are also able to utilize new cue types which are not found in the training set.

If the algorithm were to stop here, however, there would be a problem if the region, r_k, had been previously labeled as another object, i.e., $label_k$ at $F_{t-1} \neq label_k$ at F_t (For simplicity, we will call the old object label, $label_k$ at F_{t-1}, o_{old}). The problem occurs because all the cues belonging to that region, except those that were newly extracted at the current frame, would have been added to o_{old}'s definition with the previous iteration. We now can assume, since the object label was changed with the addition of more evidence, that it was the wrong object label. Thus, the cues of that region should not belong to o_{old}'s definition. It is then necessary to remove any cues that were added to r_k before the current time step from the definition of o_{old}, i.e., remove all cues, $c_l \in C_k$ added before F_t from $C_{o_{old}}$.

With this algorithm, a sequential disambiguation of the possible object categories that can be ascribed to a segmented region is possible. This ability can be cast in the framework of incremental discrimination. We describe this further in the next section.

5.2 Incremental Discrimination

We now present a framework, which shows how any set of objects similar to each other given any cue can be incrementally divided through the addition of various cue types, until each set contains only one object, and thus can be recognized apart from the others. The recognition system provides cue-driven equivalence class discrimination, where objects in the same equivalence classes, given a cue, can be distinguished. We first describe the cue-based equivalence classes, followed by the incremental discrimination of these classes, and then finally show object recognition results with the QRIO humanoid robot video dataset demonstrating this concept.

5.2.1 Cue-Based Equivalence Classes

As mentioned earlier, we present a framework for describing the incremental discrimination of a set of objects similar to each other given a cue. This set of objects are what determines an equivalence class. We then start by formally defining *equivalence*.

Equivalence

An equivalence class is defined as the set of all objects that contain the same property given a specific cue. In typical object recognition, the cues are usually based on visual cues, such as color or texture. These, however, often produce classes that will encompass more than one type of object, and thus errors in recognition will occur. The goal then is to continually separate the objects into classes of smaller and smaller size until the set remaining has cardinality of one.

In terms of the algorithm, we define an equivalence class, eq_k, as the set of object categories that are applicable to the k^{th} region, which we would like to recognize.

An object category is determined to be applicable, if it satisfies the condition that the evidence provided by an extracted cue from the data is very close to the evidence given by the object category with the greatest amount of evidence, i.e., only object categories close to the highest evidence value will be placed in that equivalence class. Hence, ***equivalence*** is determined. The following gives the equation:

$$eq_k \leftarrow \left\{ x | e_{k,x} - \max_i e_{k,i} < \epsilon \right\}$$

i.e., an equivalence class for the k^{th} region, eq_k is the set of categories x whose evidence is "close", i.e., a small ϵ value away from the largest evidence value, $\text{argmax}_i e_{k,i}$.

The evidence mentioned above is calculated by the following equation:

$$e_{k,i} = w_{i,l} s_{j,l}$$

i.e., the evidence $e_{k,i}$ that region k is labeled as the i^{th} object category. Evidence is determined by the product of $w_{i,j}$, the weight between object i and cue j, and $s_{j,l}$, the similarity between the extracted cue l and the corresponding cue j in the dictionary. Calculation of the weight and similarity is outlined in previous sections and chapters.

Thus, the equivalence class for a particular region, eq_k, will consist of all the object categories which are applicable to that region given the cues seen thus far by the robot.

5.2.2 Incremental Discrimination

We introduce the idea of equivalence class separation through the addition of more cues. In this process, the algorithm begins with the equivalence classes outlined above where every region begins with all possible object categories. These categories are then divided with the introduction of a cue. Objects with the same property are then pooled together. The process continues until each object is in its own equivalence class or until all cues are utilized. Figures 5.4 and 5.5 provide diagrams of the incremental discrimination process and separation of equivalence classes with the introduction of various cues. Figure 5.4 gives a model of equivalence class separation starting with the application of the activity cue. In the first level, we have a circle containing the set of all potential object categories for a given segmented region. When no cues are scene, the set includes all object categories. The dotted line represents the application of a cue to the set; in the first step, an activity cue is applied. The original set is then divided into three smaller sets containing objects that are similar given that cue. For instance, pen and paper are grouped together because they are both associated with the *Writing* activity. The application of another cue, shape, represented by the second dotted line, further divides the sets based on those with similar shapes. Since this cue separates all the objects into their own individual sets, it allows them to be distinguished completely from one another for identification.

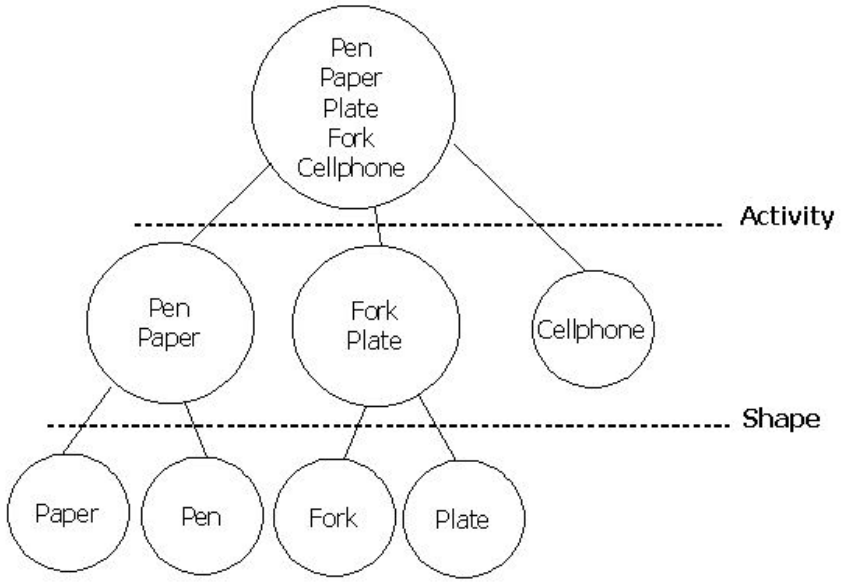


Figure 5.4: Equivalence class separation starting with the application of an activity cue

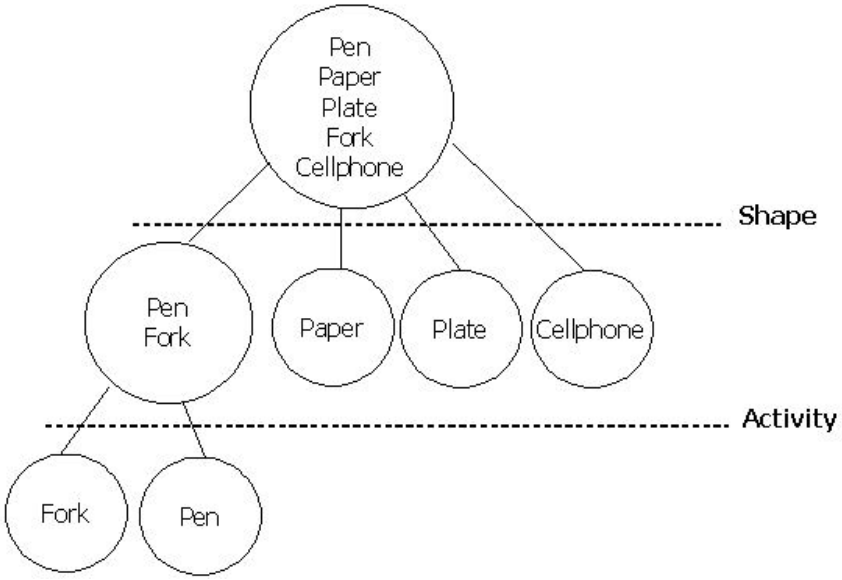


Figure 5.5: Equivalence class separation starting with the application of a shape cue

5.2.3 Results from QRIO

In order to demonstrate the concept of incremental discrimination on real robot data, we ran the MCOR algorithm on video from the Sony QRIO, a humanoid robot. Figure 5.6

shows a picture of the Sony QRIO. For each frame of the video retrieved from the QRIO, the equivalence class algorithm was applied with the weights calculated according the MCOR algorithm, using any of the cues outlined above, recognized by the QRIO at each time frame. The addition of each cue allowed the equivalence classes to be narrowed down until objects could be recognized. The equivalence classes and incremental discrimination of the MCOR algorithm allow information from multiple cues to be utilized incrementally as the data from the QRIO robot is processed.

Figure 5.6: Sony QRIO humanoid robot

Figure **??** shows the object dictionary weight values used in the QRIO experiments. Weights represent the probability of an object category given a cue value as defined in Chapter 3.

MCOR was run on approximately four minutes of video. Results are given in the next subsections. We illustrate three scenarios that demonstrate the concept of the equivalence classes described and the robustness of the algorithm in dealing with ambiguous cues. In the first instance, we illustrate its ability to recognize objects with similar shape. In the second, its ability to recognize objects with ambiguous activities, and in the third, its ability to recognize objects that can be recognized without ambiguity.

	Pen	Fork	Plate	Paper	Cellphone
ACTIVITY					
Eating	.01	.07	**.90**	.01	.06
Speaking	.08	**.92**	.02	.09	**.92**
Writing	**.91**	.01	.08	**.90**	.02
COLOR					
Blue	**.89**	.06	.04	.01	.01
Black	.10	**.90**	**.91**	.01	.01
White	.01	.02	.03	**.94**	.08
Yellow	.01	.02	.02	.05	**.90**
SHAPE (ratio)					
~1 (square)	.01	.02	**.91**	**.96**	.11
~.6 (rectangle)	.04	.05	.04	.03	**.85**
~.3 (long/thin)	**.95**	**.93**	.05	.01	.04

Figure 5.7: Object dictionary used by QRIO experiments

Distinguishing Ambiguous Shapes

In this first scenario, we demonstrate how objects with ambiguous shapes are distinguished by the algorithm. There were two possible sets of objects that could be confused based on shape: the fork and pen (which are both long and thin) and the plate and paper. Because the shape measure is based on the aspect ratio of the bounding box, the shape of the plate and paper come out to about the same shape. If more complicated shape recognition was used, it is possible to distinguish them without having to add additional cues. These results demonstrate how the algorithm can compensate for the weak cue method with the introduction and use of cue information provided by additional cue types. The activity of *Eating* distinguishes the segmented region of the fork from the pen object category. The segmented region of the fork is distinguished based on the introduction of color information, where the fork and pen are different colors. Figure 5.8 gives recognition results of the fork and pen scenario. Results of the object recognition algorithm are given as labels embedded on to each frame of the video. Video is taken from the left-eye camera of a QRIO robot. The plate and paper yielded similar results.

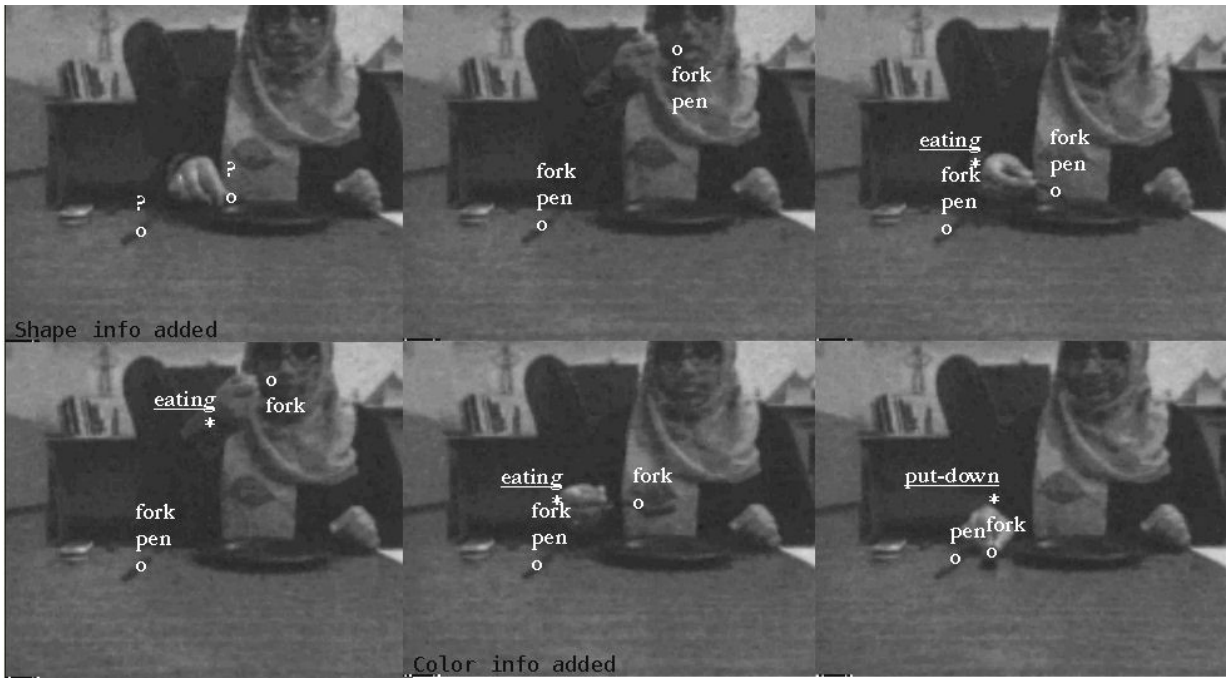


Figure 5.8: Recognition results of objects with ambiguous shapes: fork and pen

Distinguishing Ambiguous Activity

In the second scenario of ambiguous activities (see Figure 5.9 for an example of the results), we have two sets of objects: Fork and plate, and pen and paper. In each of these instances, the same activity (*Eating* in the former case, *Writing* in the latter) is associated strongly with both objects. Since our algorithm takes into account both color and shape, it can easily resolve these differences. With the equivalence classes, either cue type, i.e., color or shape, would be enough to put the objects in their separate classes. In Figure 5.9, results of the object recognition algorithm are given as labels embedded on to each frame of the video. Video is taken from the left-eye camera of a QRIO robot.

No Ambiguity

In the third task, we wanted to illustrate that the MCOR algorithm can recognize objects on the first shot, without having to disambiguate between other objects.

In Figure 5.10, the results of the algorithm are illustrated, where a cellphone was successfully recognized without ambiguity with another object, taking advantage of the fact that it has a very distinctive property not shared by the other objects, i.e., its lone association

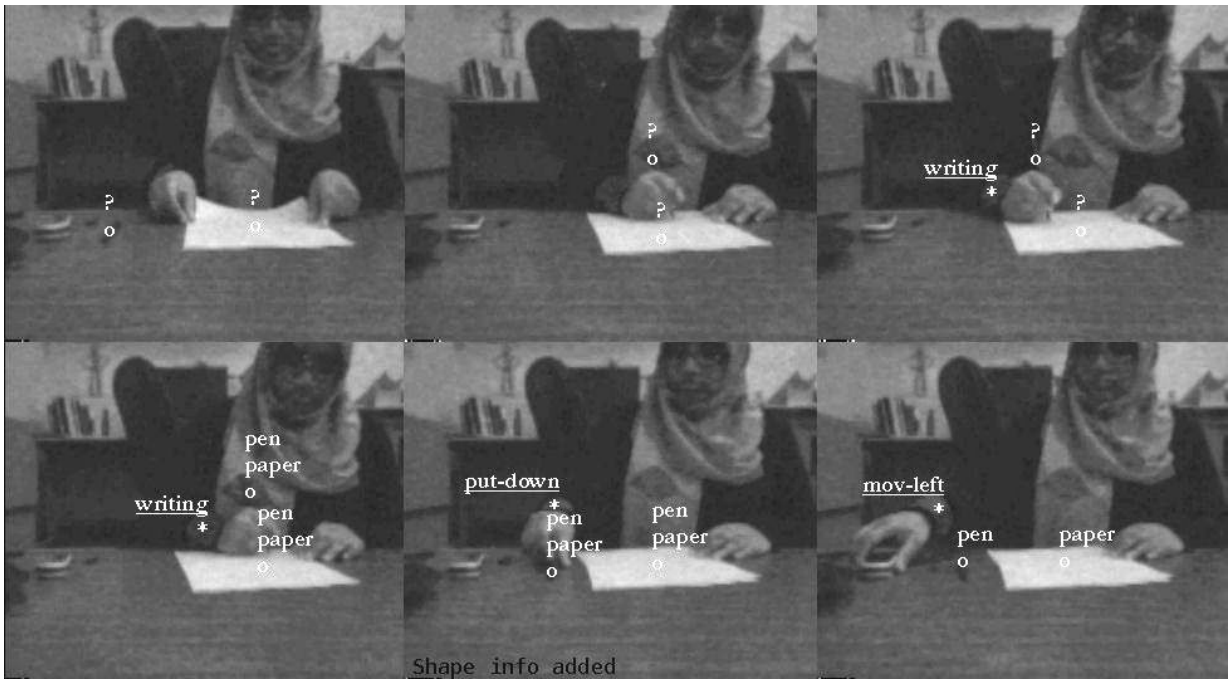


Figure 5.9: Recognition of objects with ambiguous activity: pen and paper

with the *Talking* activity. Results of the object recognition algorithm are given as labels embedded on to each frame of the video. Video is taken from the left-eye camera of a QRIO robot.

Thus, the algorithm was able to demonstrate its use in real robot data. Overall, every object was eventually correctly identified.

Figure 5.10: Recognition results for object with unambiguous cue information: cellphone.

5.3 Conclusion

In this chapter, we presented the MCOR algorithm which integrated the object and cue representations, weights, and visual features to produce a framework that can incrementally integrate information from multiple sources to disambiguate potential object categories. In addition, the MCOR algorithm has the ability to generalize learned cue values to new object categories for additional evidence, as well as the ability of using and learning the association of cue types not found in the original training dataset.

In the next chapters, we demonstrate the robustness of this MCOR algorithm using an analytical model (Chapter 6) and real recognition results (Chapter 7).

Chapter 6

Recognition Robustness, Analysis and Cue Accuracy

In this chapter, we present an analytical model demonstrating the robustness of the MCOR algorithm. This analytical model is important since in many of the real data experiments we hand-labeled the data, and so the benefit of multiple cues in cases when the cue accuracy is not perfect may not be clear from those results. Thus, we developed an analytical model, which shows mathematically the benefit of using the MCOR algorithm even in cases when cue uncertainty is increased. The goal of this analysis is to show that the use of multiple cues improves recognition performance given varying levels of cue noise. We do so by showing (1) a representation of the activation used for recognition, (2) how the distribution of this activation can be used to develop metrics for the performance of the system as represented by the rate of false positives and false negatives, (3) analytical and experimental evaluations using this metric to determine the advantage of using multiple cues over a single cue, and (4) a discussion and conclusion with future work based on these findings.

6.1 Object Recognition Activation Value

In order to determine how the recognition performance is affected by different levels of cue noise and the use of multiple cues, we need a mathematical model of the recognition process. The key component of this recognition is the activation value. Thus, in this section, we define the activation value and corresponding model. The activation value represents the evidence that an unknown segment in an image is a particular object category. The higher

the activation value, the more likely that object category belongs to that segment.

6.1.1 Activation Value

In Chapter 5, we introduced the equation of the evidence for a particular object category (Section 5.1.1). We can use the evidence to show how the activation value, a_i, for the i^{th} object category is calculated. While the evidence calculation represents the incremental accumulation of the evidence at each point in the video, this activation formula demonstrates the calculation of evidence at the end of the entire video:

$$e_{k,i} = \sum_{j \in C_i} \sum_{l \in C_k} w_{i,j} s_{j,l} \tag{6.1.1}$$

As a reminder, here is the evidence equation, where for the k^{th} region, r_k, and for i^{th} object, the algorithm calculates the evidence, $e_{k,i}$, that region r_k should be labeled with object category o_i.

where $w_{i,j}$, is the weight representing the strength of the association between the cue, j, in the set of cues in the object definition, C_i, and the object, i. $s_{j,l}$ is the similarity between the cue j and the cue, l, belonging to the set of extracted cues, C_k. If the cues are not of the same type, then $s_{c_j,cue_l} = 0$.

For the activation equation, we assume for simplicity that each of the extracted cues is found in the object dictionary with similarity of 1, as represented by C. We then modify the evidence equation, so that we are summing the evidence across the entire video, as represented by n_z. n_z holds the number of times that the z^{th} cue value occurred throughout the entire video. This is multiplied by the weight, $w_{i,z}$ is the weight or association strength between the i^{th} object category and the z^{th} cue value in the set of extracted cues found in the object dictionary, C:

$$a_i = \sum_{z \in C} w_{i,z} n_z \tag{6.1.2}$$

We focus on the maximum activation value a_{i^*} where i^* is the object category with the maximum activation value, since the maximum activation value determines which object category is given and thus effects recognition performance.

$$
\begin{aligned}
i^* &= \arg\max_j \sum_{z \in C} w_{j,z} n_z \\
a_{i^*} &= w_{i^*,z} n_z
\end{aligned} \tag{6.1.3}
$$

6.1.2 Threshold

Since recognition only occurs if the activation value is above a threshold, we define the calculation representing this threshold in this subsection. Given the maximum activation value, an object is recognized (i.e., the segment is given an object category) when the value is above a given threshold. The threshold can be determined by using a Maximum A Posteriori (MAP) estimation [45].

Let a_{i^*} be a continuous random variable corresponding to the activation value, and let O_{i^*} be a Bernoulli random variable describing the presence of the object with the i^{*th} category that has the maximum activation value. The MAP estimator, $\hat{O_{i^*}}$ is then:

$$
\hat{O_{i^*}} = \arg\max_{O_{i^*}} P(a_{i^*}|O_{i^*})P(O_{i^*}) \tag{6.1.4}
$$

$$
= \begin{cases} 0 & : P(a_{i^*}|0)P(0) \geq P(a_{i^*}|1)P(1) \\ 1 & : P(a_{i^*}|0)P(0) < P(a_{i^*}|1)P(1) \end{cases} \tag{6.1.5}
$$

Since each activation value event of whether an object is present or not is independent of any other activation event, we assume based on the central limit theorem that $P(a_{i^*}|O_{i^*} = \boldsymbol{o}_{i^*})$ is a Gaussian distribution. Figure 6.1 give a depiction of the Gaussian probability densities for $P(a_{i^*}|O_{i^*} = \boldsymbol{o}_{i^*})P(O_{i^*} = \boldsymbol{o}_{i^*})$, where $\boldsymbol{o}_{i^*} \in 1, 0$ and the MAP optimal threshold, θ. Error regions representing the false positives (light gray), and false negatives (dark gray) are also highlighted. We define the optimal threshold as the point, where it is more probable that the object is present in the scene for $a_{i^*} > \theta$ and it is more probable that the object is absent for $a_{i^*} < \theta$, i.e. the a_{i^*} value where:

$$
P(a_{i^*}|O_{i^*} = 0)P(O_{i^*} = 0) = P(a_{i^*}|O_{i^*} = 1)P(O_{i^*} = 1) \tag{6.1.6}
$$

We can determine the curves by knowing the first and second order moments, μ, σ when $O_{i^*} = 1$ and μ', σ' when $O_{i^*} = 0$. The threshold value can then be calculated by solving the quadratic equation [45]:

$$\frac{1-q}{\sqrt{2\pi\sigma'^2}}\exp^{-\frac{(\theta-\mu')^2}{2\sigma'^2}} = \frac{q}{\sqrt{2\pi\sigma^2}}\exp^{-\frac{(\theta-\mu)^2}{2\sigma^2}} \tag{6.1.7}$$

where $q = P(O_{i*} = 1)$. Assuming that when the object is present there is a higher activation distribution than when it is not, i.e. $\mu > \mu'$, the solution would then be:

$$\theta = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{6.1.8}$$

where

$$\begin{aligned} a &= \sigma'^2 - \sigma^2 \\ b &= 2(\mu'\sigma^2 - \mu\sigma'^2) \\ c &= \mu^2\sigma'^2 - \mu'^2\sigma^2 + 2\sigma'^2\sigma^2 \ln\left(\frac{(1-q)\sqrt{2\pi\sigma^2}}{q\sqrt{2\pi\sigma'^2}}\right) \end{aligned}$$

We then choose the root that lies between μ' and μ.

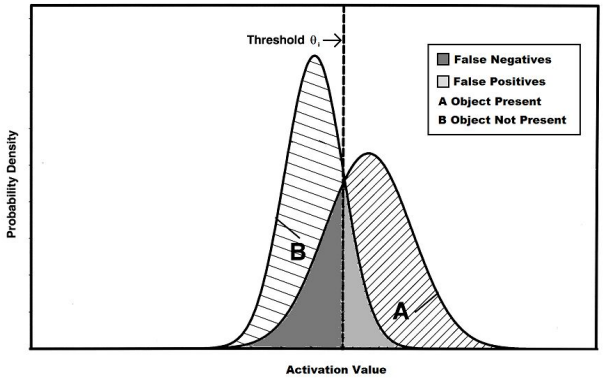


Figure 6.1: Gaussian probability densities for activation value when object present and not present, optimal threshold and error regions (false positive and negative rates)

With the threshold and activation distributions, a performance metric using the false positives and false negatives can be developed to measure recognition performance.

6.2 Performance Metric

In order to show the robustness of the MCOR system to cue noise with the use of multiple cues, we need a metric which can reflect the recognition performance.

6.2.1 Error Calculation

With the activation distributions and best threshold, we can determine the error value. Given the threshold, the false positive rate (fp) can be determined by the area under the $N(\mu', \sigma')$ curve where $a_{i^*} > \theta$ and false negative rate (fn) by the area under the $N(\mu, \sigma)$ curve where $a_{i^*} < \theta$, i.e.

$$fp = \int_{\theta}^{\infty} N(\mu', \sigma') \tag{6.2.1}$$

$$fn = \int_{-\infty}^{\theta} N(\mu, \sigma) \tag{6.2.2}$$

In Figure 6.1, the false positive and false negative areas are filled in with light and dark gray.

6.2.2 Metric

The main goal then is to show that these two regions are smaller or, in worst case, equal in the case when multiple cues are used, and larger when only a single cue is used. One metric we use for the comparison is the error rate (ER). The error rate is calculated as follows:

$$ER = \frac{fp + fn}{2} \tag{6.2.3}$$

where the false positive and false negative rate are averaged together. We then determine that this value is smaller in the multiple cue case, then in the single cue case.

Another metric we utilize is the disparity metric (D), which takes advantage of the observation that these two regions can be minimized by reducing the amount of overlap between these two curves, i.e. increasing the disparity. This is represented by the following calculation:

$$D = (\mu - \sigma) - (\mu' + \sigma') \tag{6.2.4}$$

Figure 6.2 shows a visual depiction of this metric. The advantage of this metric is that when the curves do not intersect, where the error rate simply shows zeros for all cases, this metric shows you how large the discrepancy is between the two curves. Thus, showing how likely a zero or one error rate is.

For the results, we show a graph of both metrics. Our goal is then to show that the metrics for the multiple cue case, D_m and ER_m have better or equal scores than the single cue case D_s and ER_s, For the error rate, smaller values are better. For the disparity metric, larger values shows better performance:

$$D_s \leq D_m \tag{6.2.5}$$
$$ER_s \geq ER_m \tag{6.2.6}$$

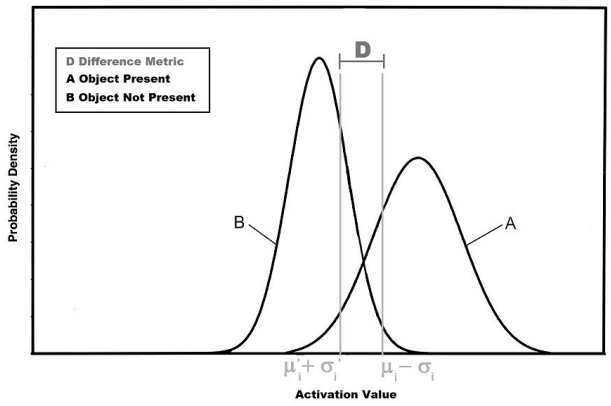


Figure 6.2: Graphical depiction of the metric, D, used to evaluate performance

6.2.3 Activation Distribution Model

We now need to look in more detail into the mean and standard deviation which define the shape of the activation distributions, i.e. $P(a_{i^*}|O_{i^*})$. As a reminder, if we take a look

at the activation calculation for one object category i^*, we have:

$$a_{i^*} = \sum_{z \in C} w_{i^*,z} n_z \tag{6.2.7}$$

We then need to calculate the mean and standard deviation for this activation. To get the mean and standard deviation, we then need a way to model the n_z term. We do this by using N_z to capture the number of times the z^{th} cue value is found for a particular object label. Let N_z be a binomial random variable describing the total unweighted activation value provided by cue value z for a given segment given that an object with category i^* is present.

As a reminder, for a Binomial distribution consider an experiment with two possible outcomes (success and failure), where success happens with a probability p, and failure with a probability 1-p. If the experiment is repeated n times, a binomial random variable, which we call X, counts the number of successes. The probability of having x successes is defined as

$$P(X = x) = \binom{n}{x} * (p)x * (1-p)n - x$$

the expected value is given by

$$E(X) = np$$

and the variance by

$$Var(X) = np(1-p)$$

The probability of success is defined as the probability that the unweighted activation value is increased, i.e. frame f contains a cue value z that provides evidence that the segment belongs to object category i^*. The probability of success, $P(X_z^{(f)} = 1)$, is given by p_z and the probability of failure, $P(X_z^{(f)})$, by $1 - p_z$, where $X_z^{(f)}$ is a random variable representing success or failure, i.e.:

$$X_z^{(f)} = \begin{cases} 1 & \text{if frame } f \text{ contains cue } z \text{ for object category } i^* \\ 0 & \text{otherwise} \end{cases} \tag{6.2.8}$$

Each activation value increase is assumed statistically independent. Similar variables can be defined for the case when $O_{i^*} = 0$ for frame f: N'_z, $X'^{(f)}_z$ and p'_z.

Let F be the set of frames for any given video [1]. Let F_{i^*} then be the set of frames where

[1]For simplicity, we assume all videos are of the same length. It is possible to include variable length videos into the equations, but it should not affect the main conclusions of this analysis.

$O_{i^*} = 1$, i.e. an object with category i^* is present and F'_{i^*} the set of frames where $O_{i^*} = 0$. We assume we are given all prior distributions for each object category $P(O_{i^*} = 1)$ and $P(O_{i^*} = 0)$. For shorthand, we use $P(\boldsymbol{o}_{i^*})$ for $P(O_{i^*} = 1)$ and $P(\boldsymbol{o}'_{i^*})$ for $P(O_{i^*} = 0)$.

To determine the probability of success, $P(X_z^{(f)})$, we use an *association* term and *cue accuracy* term. The association term defines the probability that a cue value is generated given the presence of an object, $P(c_z|o_i)$. The cue accuracy term is how we vary cue uncertainty in our experiments and is defined by the probability a cue value, c_z is observed as itself $\hat{c}_z$ plus all the times a different cue value, c_j, was seen as c_z. This can be put together in the following manner:

$$P(X_z^{(f)} = 1) = p_z = P(c_z|\boldsymbol{o}_{i^*})P(\hat{c}_z|c_z) + \sum_{j \neq z} P(\hat{c}_z|c_j)P(c_j|\boldsymbol{o}_{i^*}) \tag{6.2.9}$$

We do a similar calculation for the case when the object is not present, using the same cue accuracy terms:

$$P(X'^{(f)}_z = 1) = p'_z = P(c_z|o'_{i^*})P(\hat{c}_z|c_z) + \sum_{j \neq z} P(\hat{c}_z|c_j)P(c_j|o'_{i^*})$$

Using the expectation of the Binomial term, N_z, we can calculate the mean and standard deviation terms that define the activation distributions when only a single cue type is used and when multiple cue types are used. The derivation of these terms are defined in the next two subsections.

6.2.4 Single Cue Type Activation Distributions

Based on the Binomial random variable for the cue count, N_z, cue association and accuracy, we can calulate the mean and standard deviation for the activation distributions for a single cue type.

The mean and standard deviations for the single cue is then:

$$
\begin{aligned}
\mu_s &= E[a_{i^*}] = E\left[\sum_{z\in C_s}\sum_{f\in F_{i^*}} w_{i^*,z} X_z^{(f)}\right] \\
&= E[a_{i^*}] = \sum_{z\in C_s} w_{i^*,z} E\left[\sum_{f\in F_{i^*}} X_z^{(f)}\right] \\
&= E[a_{i^*}] = \sum_{z\in C_s} w_{i^*,z} E\left[N_z\right] \\
&= E[a_{i^*}] = \sum_{z\in C_s} w_{i^*,z} (|F_{i^*}| p_z) \\
\sigma_s &= \sqrt{Var[a_{i^*}]} = \sqrt{Var\left[\sum_{z\in C_s}\sum_{f\in F_{i^*}} w_{i^*,z} X_z^{(f)}\right]} \qquad (6.2.10) \\
&= \sqrt{Var[a_{i^*}]} = \sqrt{\sum_{z\in C_s} w_{i^*,z}^2 Var\left[\sum_{f\in F_{i^*}} X_z^{(f)}\right]} \\
&= \sqrt{Var[a_{i^*}]} = \sqrt{\sum_{z\in C_s} w_{i^*,z}^2 Var\left[N_z\right]} \\
&= \sqrt{Var[a_{i^*}]} = \sqrt{\sum_{z\in C_s} w_{i^*,z}^2 (|F_{i^*}| p_z(1-p_z))^2} \qquad (6.2.11) \\
\mu'_s &= E[a_{i^*}]' = \sum_{z\in C_s} w_{i^*,z} (|F'_{i^*}| p'_z) \\
\sigma'_s &= \sqrt{Var[a_{i^*}]'} = \sqrt{\sum_{z\in C_s} w_{i^*,z}^2 (|F'_{i^*}| p'_z(1-p'_z))^2} \qquad (6.2.12) \\
& \qquad (6.2.13)
\end{aligned}
$$

$$
\begin{aligned}
P(X_z^{(f)} = 1) &= p_z = P(c_z|\boldsymbol{o}_{i^*})P(\hat{c}_z|c_k) + \sum_{j \neq z} P(\hat{c}_z|c_j)P(c_j|\boldsymbol{o}_{i^*}) \\
&= \sum_{l \in C_s} P(\hat{c}_z|c_l)P(c_l|\boldsymbol{o}_{i^*}) \\
P(X'^{(f)}_z = 1) &= p'_z = P(c_z|o'_{i^*})P(\hat{c}_z|c_z) + \sum_{j \neq z} P(\hat{c}_z|c_j)P(c_j|o'_{i^*}) \\
&= \sum_{l \in C_s} P(\hat{c}_z|c_l)P(c_l|o'_{i^*})
\end{aligned}
\tag{6.2.14}
$$

V is the set of videos used for current recognition session. We assume that we are given the probability that a particular cue c_z is produced given the presence of an object $\boldsymbol{o}_{i^*}$, i.e. $P(c_z|\boldsymbol{o}_{i^*})$ and the prior probabilities for each cue value, i.e. $P(c_z)$.

In addition, the weight between an object and a cue, $w_{i^*,z}$, are learned. The weights represent the probability of an object o_{i^*} being present given a particular cue c_z, i.e. $P(\boldsymbol{o}_{i^*}|c_z)$. In order to reflect the strength of the association between a cue and object.

6.2.5 Multiple Cue Type Activation Distributions

The mean and standard deviations for the multiple cue case is calculated in a similar manner, using the expectation of N_z, cue association and accuracy:

$$
\begin{aligned}
\mu_m &= E[a_{i^*}] = E\left[\sum_{z\in C_m}\sum_{f\in F_{i^*}} w_{i^*,z} X_z^{(f)}\right] \\
&= E[a_{i^*}] = \sum_{z\in C_m} w_{i^*,z}(|F_{i^*}|p_z) \\
\sigma_m &= \sqrt{Var[a_{i^*}]} = \sqrt{Var\left[\sum_{z\in C_m}\sum_{f\in F_{i^*}} w_{i^*,z} X_z^{(f)}\right]} \qquad (6.2.15) \\
&= \sqrt{Var[a_{i^*}]} = \sqrt{\sum_{z\in C_m} w_{i^*,z}^2(|F_{i^*}|p_z(1-p_z))^2} \qquad (6.2.16) \\
\mu'_m &= E[a_{i^*}]' = \sum_{z\in C_m} w_{i^*,z}(|F'_{i^*}|p'_z) \\
\sigma'_m &= \sqrt{Var[a_{i^*}]'} = \sqrt{\sum_{z\in C_m} w_{i^*,z}^2(|F'_{i^*}|p'_z(1-p'_z))^2} \qquad (6.2.17) \\
& \qquad (6.2.18)
\end{aligned}
$$

$$
\begin{aligned}
P(X_z^{(f)}=1) &= p_z = P(c_z|\boldsymbol{o}_{i^*})P(\hat{c}_z|c_z) + \sum_{j\neq z} P(\hat{c}_z|c_j)P(c_j|\boldsymbol{o}_{i^*}) \\
&= \sum_{l\in C_m} P(\hat{c}_l|c_z)P(c_l|\boldsymbol{o}_{i^*}) \\
P(X'^{(f)}_z=1) &= p'_z = P(c_z|o'_{i^*})P(\hat{c}_z|c_z) + \sum_{j\neq z} P(\hat{c}_z|c_j)P(c_j|o'_{i^*}) \\
&= \sum_{l\in C_m} P(\hat{c}_z|c_l)P(c_l|o'_{i^*}) \\
& \qquad (6.2.19)
\end{aligned}
$$

6.3 Evaluation and Experimentation

Using the theoretical foundation outlined above, we now provide experimental evaluations demonstrating the advantages of the multiple cue framework. More precisely, there are three conclusions we demonstrate:

1. With increasing cue accuracy and number of cue types, performance is increased.
2. The addition of multiple cue types with better accuracy than a single cue type improves the performance.
3. The addition of multiple cue types with worse accuracy than a single cue type does not hurt performance.

We demonstrate these conclusions under three different scenarios in each of the subsections below:

- Single cue value association: A single cue value is associated with the given object within each cue type.
- Multiple cue value associations: Every cue value is associated with the given object, with different distributions for each cue type.
- Ambiguous/Misleading cue values: Cue values for a single cue type are ambiguous or misleading.

For each of these scenarios, we graph the performance results as given by the disparity and error rate metrics. In addition to the graphical representation of the analytical functions as defined by the parameters below. We provide simulation and accuracy variation to affirm the accuracy of the mathematical results and graphs.

To begin, we need to define the constants, parameters, and functions used to determine the value of the terms outlined in the equations above in order to calculate the experimental results. We summarize here how each of these terms are determined:

CONSTANTS

$|\mathbf{F}|$ **:** Number of frames, $|\mathrm{F}| = 10000$

$P(\boldsymbol{o}_{i^*})$ **:** Prior probability for the presence of object, $P(\boldsymbol{o}_{i^*}) = .2$, except in Section 6.3.3, where $P(\boldsymbol{o}_{i^*}) = .5$

V : Number of cue values in each type. For simplicity, all cue types have the same number of cue values, V = 4

PARAMETERS

M : Number of cue types.

$P(\hat{c_l}|c_j)$ **:** Matrix which determines cue accuracy, where all elements on the diagonal, i.e. $l, j \in C$ where $l = j$, have $P(\hat{c_l}|c_j) = \alpha$. α is the accuracy value which varies from 0 to 1. All $l, j \in C$, where $l \neq j$, then have $P(\hat{c_l}|c_j) = (1 - a)/V$. Cue values from different cue types cannot be confused. (see Figure 6.3 for an example)

$P(c_z|\boldsymbol{o}_{i^*})$ **:** Probability of getting a cue value, c_z, given object presence, $\boldsymbol{o}_{i^*}$.

FUNCTIONS

$P(c_z)$ **:** Prior probability for each cue value, where $P(c_z|\boldsymbol{o}_{i^*})P(\boldsymbol{o}_{i^*}) \leq P(c_z) \leq P(c_z|\boldsymbol{o}_{i^*})$, since we would expect there to be less cues when the object is not present than when it is, but more than just the cues generated by the object due to noise. Thus we calculate $P(c_z) = P(c_z|\boldsymbol{o}_{i^*})P(\boldsymbol{o}_{i^*}) + \delta$. For all cases, except for Section 6.3.3, $\delta = .01$.

$P(c_z|\boldsymbol{o}'_{i^*})$ **:** Probability that a cue is seen given $\boldsymbol{o}_{i^*}$ is not present. This can be calculated using the terms above, $P(c_z|\boldsymbol{o}'_{i^*}) = \frac{P(c_z) - P(c_z|\boldsymbol{o}_{i^*})P(\boldsymbol{o}_{i^*})}{1 - P(\boldsymbol{o}_{i^*})}$

$w_{i,z}$ **:** Bayesian weights with noise included in the learning. Thus, $w_{i,z} = P(\boldsymbol{o}_{i^*}|c_z) = \frac{P(c_z|\boldsymbol{o}_{i^*})P(\hat{c_z}|c_z)P(\boldsymbol{o}_{i^*})}{P(c_z)}$

Figure 6.3 gives an example of various matrices representing $P(\hat{c_z}|c_j)$. Cue accuracy, α increases from left to right. First row contains graphs for one cue type, M=1. Second row contains graphs for five cue types, M=5. Note how in the five cue type, there are diagonal square patches with values; this is because cue values from one cue type can not be confused with cue values from another cue type. We now utilize these terms in the following experimentation and evaluations.

6.3.1 Single Cue Value Association

In this first experiment, we start with the simplest scenario, i.e. a single cue value for each object type associated with the object (as in the Gupta and Davis experiment). This

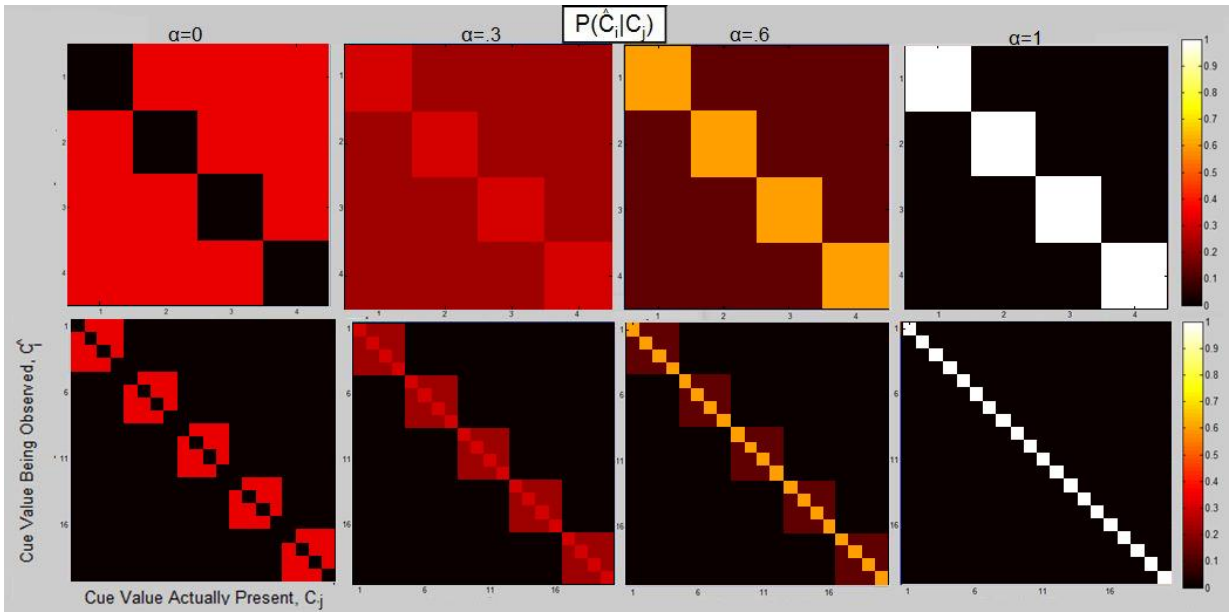


Figure 6.3: Example of various matrices representing cue accuracy, i.e., $P(\hat{c_z}|c_j)$

means, given no noise, we should be able to determine the presence of an object based on the presence of a single cue value. However, in order to demonstrate the noise one would normal find in a real world scenario, we use varying values of cue accuracy, α as defined above, with values ranging from 0 to 1. In Figure 6.3, examples of the accuracy used in the following experiments is given.

In addition, in Figure 6.4, examples of the parameters $P(c_z|\boldsymbol{o}_i)$, $w_{i,z}$ for $M = 1$ and $M = 5$ and varying levels of accuracy is given. Note how for each cue type a single cue value is given .9 association value. Also, note that the value of the weight depends both on $P(c_z|\boldsymbol{o}_i)$ and the level of cue accuracy. For instance, when $\alpha = 0$, all weights are 0, despite the $P(c_1|\boldsymbol{o}_i) = .9$.

If the multiple cue framework is beneficial, we would then expect to see an improvement in performance according to the three conclusions outlined above.

Varying Cue Accuracy

We now demonstrate that performance improves with increasing cue accuracy as well as with an increasing number of cue types. In Figure 6.5, we see that as accuracy is increased, the performance metric for all number of cue types also increases. This is a confirmation of what one would expect with an improvement in accuracy. Two metrics are shown:

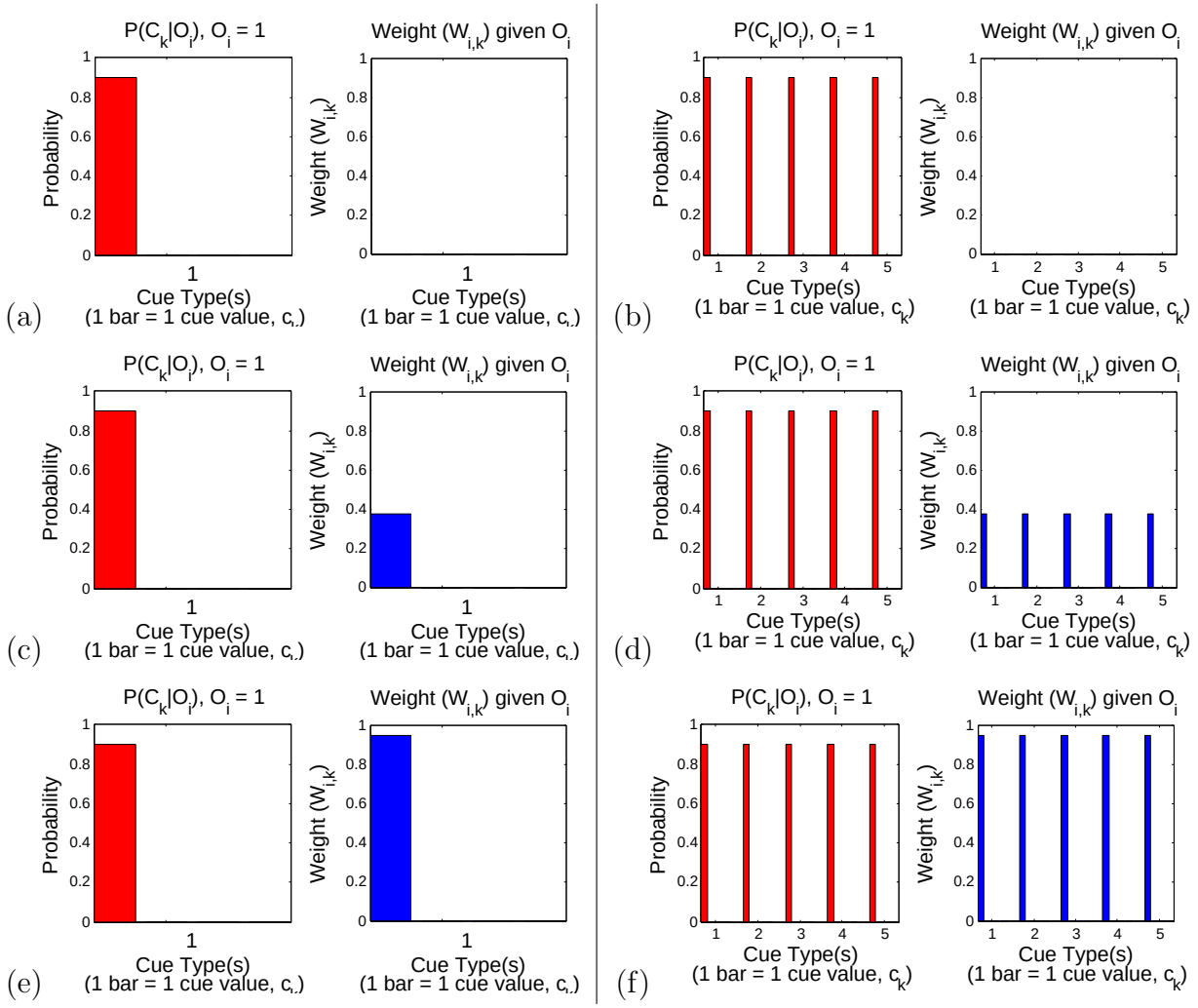


Figure 6.4: Example of association strength, $P(c_z|\boldsymbol{o}_{i^*})$, and weight, $w_{i,z}$, for single cue value association. First column corresponds to a single cue type (M=1), second column corresponds to five cue types (M=5). Rows correspond to accuracy: (a)&(b) $\alpha = 0$, (c)&(d) $\alpha = .4$, (e)&(f) $\alpha = 1$.

disparity and error rate.

The interesting point to note is that as the number of cue types increases, there is an overall increase in performance, i.e., a decrease in error rate. Thus, while it took .9 accuracy for a single cue to get a disparity of 1000 and an error rate of 0, 20 cue types only needed a .2 accuracy and 10 cue types, a .3. Thus, you can get the same level of performance with less accuracy by using multiple cue types.

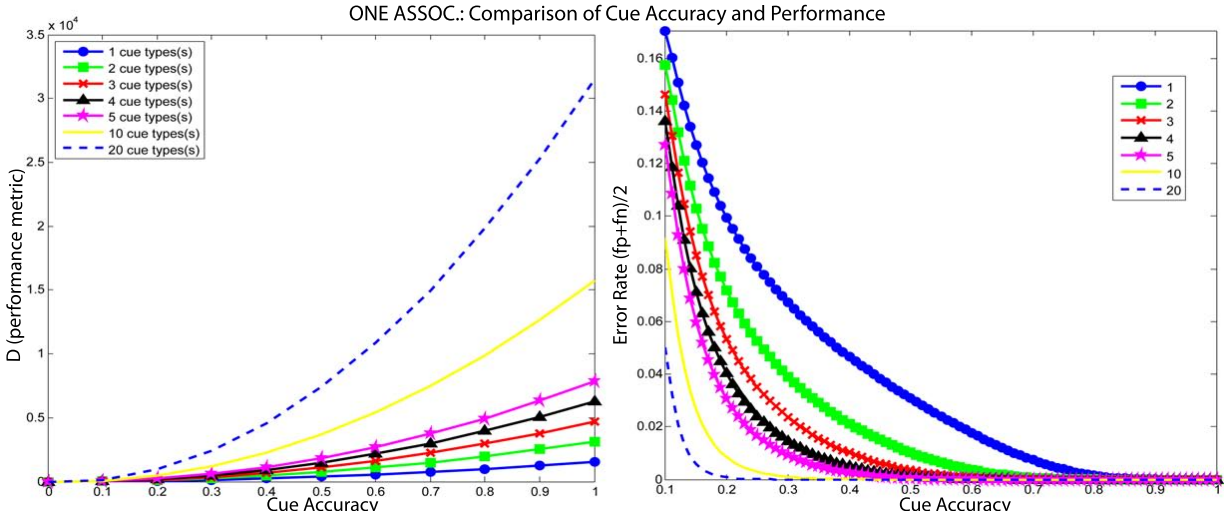


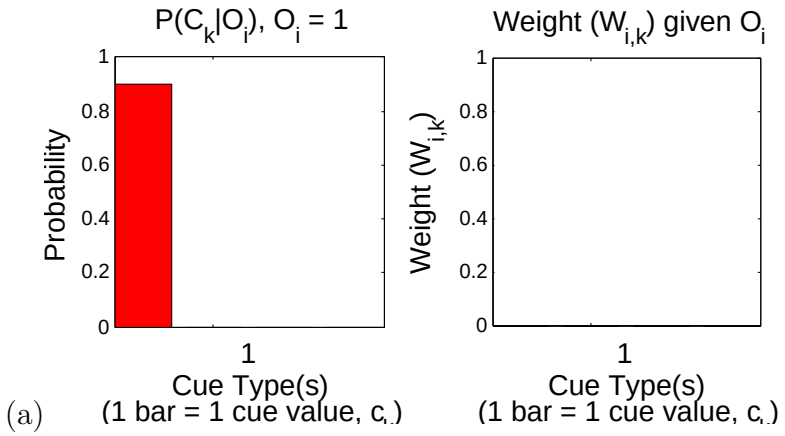
Figure 6.5: Comparison of cue accuracy and performance results for single cue value association in analytical model

Additional Cue Types have Better Accuracy than Single

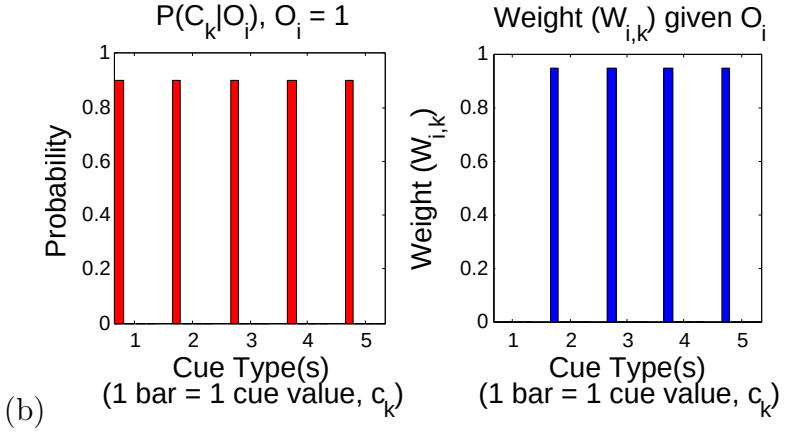
We now show that with the addition of multiple cue types that have higher cue value accuracy than a single cue type, we gain in performance results. To demonstrate this, we set the accuracy for the single cue type to 0, and all additional cues to 1 (see Figure 6.6). The corresponding weights and association strengths are show in the same figure. Note that the weights for the additional cues are much higher than the single cue, due to the much higher accuracy.

In Figure 6.7, we see the results, where as the number of cue types increase, the performance metric D increases and error rate decreases. This shows that the use of additional cues helps the overall performance for recognition. Two metrics are shown: disparity and error rate.

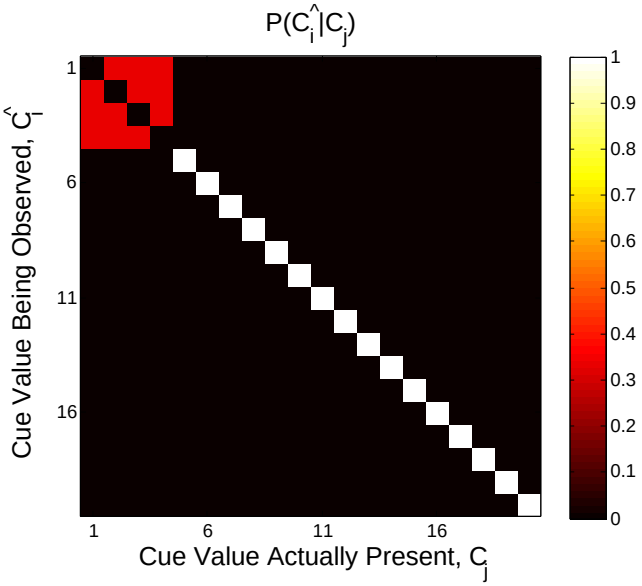
In Figure 6.8, we show the error rate for the single cue value association case, where a moderate increase ($\alpha = 0.3$) is used, rather than the high accuracy ($\alpha = 1$) used in the



(a)



(b)



(c)

Figure 6.6: Parameters used for single cue value association with one inaccurate ($\alpha = 0$) single cue and accurate ($\alpha = 1$) additional cues. Accuracy shown in (a). Association strength, $P(c_z|\boldsymbol{o}_{i^*})$, and weight, $w_{i,z}$, shown for (b) a single cue type (M=1) and (c) five cue types (M=5).

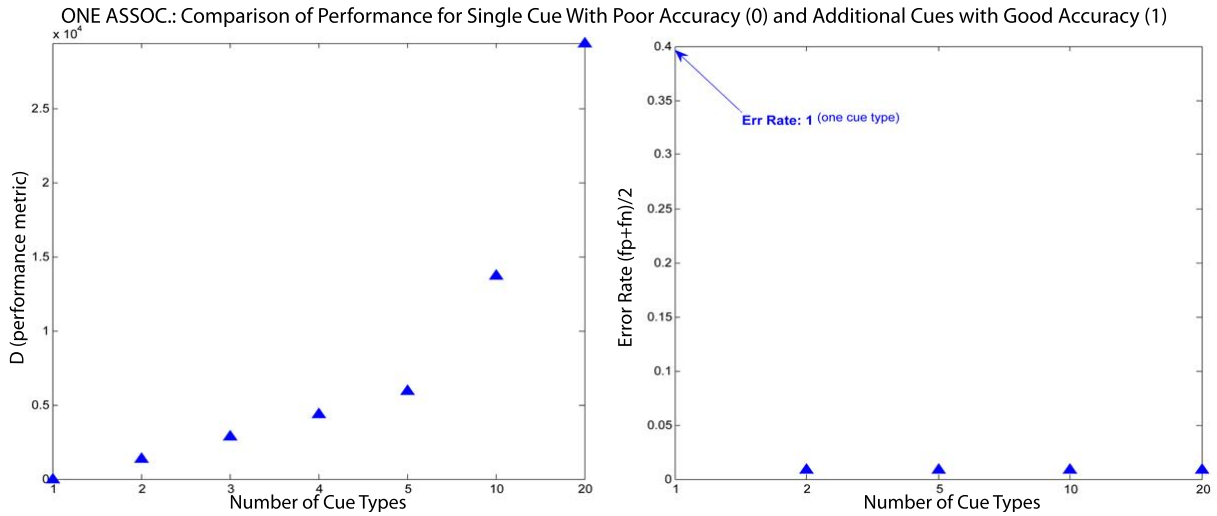


Figure 6.7: Comparison of performance for single cue type with poor accuracy ($\alpha = 0$) and additional cue types with good accuracy ($\alpha = 1$) for single cue value association

previous experiment. This moderate increase shows how the reduction in the error rate significantly decreases with increases in accuracy. The moderate increase does not show the near 0 error rate of the high accuracy, but it makes clear that as the number of cue types increase, the error rate can approach 0.

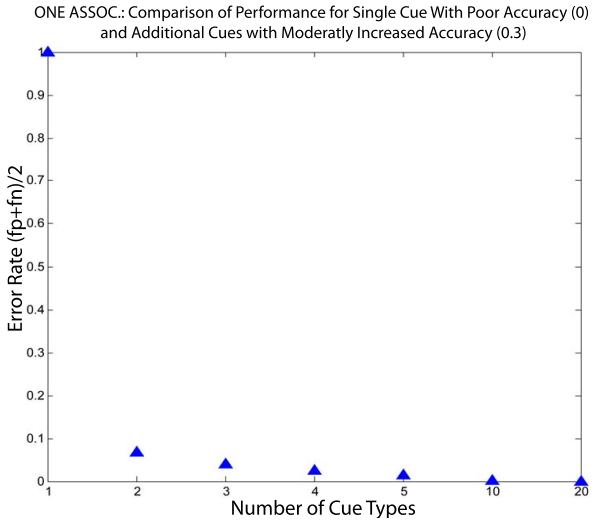


Figure 6.8: Comparison of performance for single cue type with poor accuracy ($\alpha = 0$) and additional cue types with moderately increased accuracy ($\alpha = 0.3$)

Single Cue Type has Better Accuracy than Additional

We now demonstrate that if the additional cues being added have a worse accuracy (in this case, the worst case scenario of $\alpha = 0$), than there is no harm done to the performance results of the single cue alone. This is important as it demonstrates that the addition of multiple cues can only benefit the overall performance.

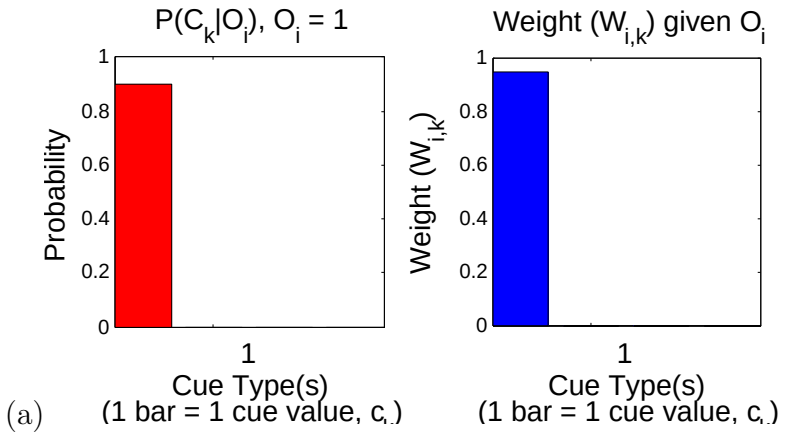
In Figure 6.9, the parameters used to define this scenario are given. Note that the single cue has better accuracy ($\alpha = 1$) than all the additional cues. The weights match this accuracy as well as the single cue value association of this scenario.

Figure 6.10 demonstrates that as the inaccurate additional cue types are added the overall performance stays at the level of the accurate single cue type. This means that given the worst case scenario, the algorithm will only ever do as bad as its best cue type. Additional inaccurate cue types will not bring the performance down. This property is due to the weights used in the framework. Since the weights are learned from training samples taken from the same scenario as the testing circumstance (reflected by the inclusion of inaccuracy in the weight term), the weights can counteract the effect of inaccurate cue types (see Figure 6.9 for weight values).

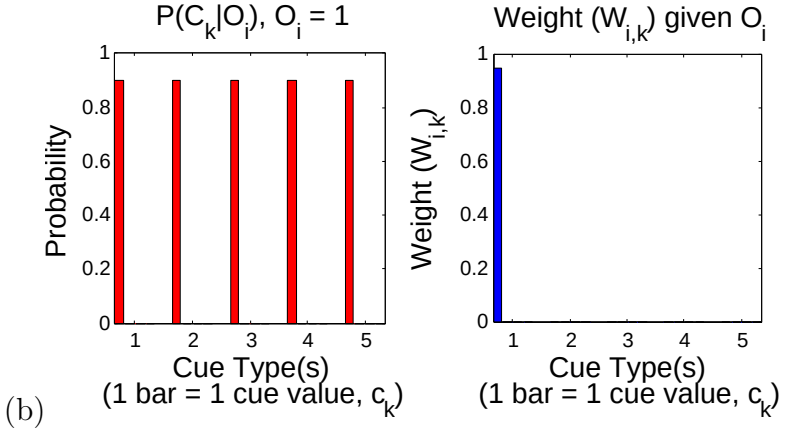
6.3.2 Multiple Cue Value Association

In this next scenario, we remove the single cue value restriction and allow multiple cue values within the same cue type to be associated with the given object. We do this by assigning a random value from 0 to 1 to each cue value. Cue values under the same cue type must add up to 1, since the cue values for one cue type are mutually exclusive. Figure 6.11 provides examples of the parameters $P(c_z|\boldsymbol{o}_{i^*})$ and $w_{i,z}$, reflecting the randomly generated values of multiple cue value association in each cue type for $M = 1$ and $M = 5$. This scenario reflects a more real world scenario where different cue values with the same cue type may contribute evidence to the presence of an object. For instance, an eraser board may get evidence from both the activity of erasing and the activity of writing. As in the scenario before, we vary the cue accuracy from 0 to 1 as show in Figure 6.3.

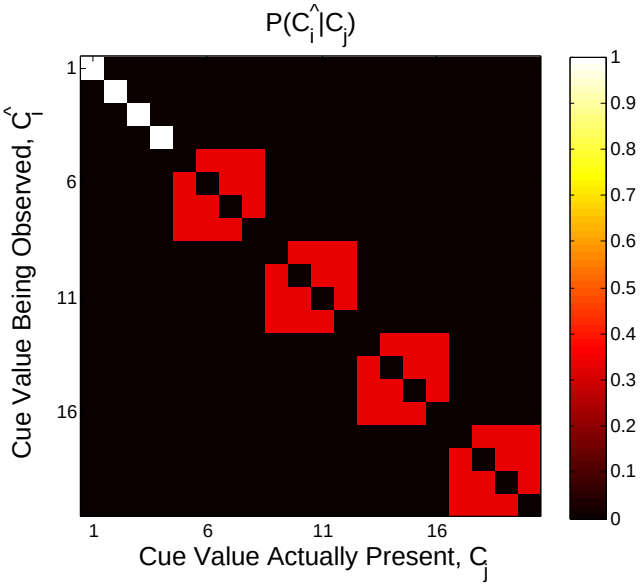
Despite the removal of the one cue value association, we would still expect to see an improvement in performance according to the three conclusions outlined above in order to show the usefulness of the MCOR framework.



(a)



(b)



(c)

Figure 6.9: Parameters used for single cue value association with one accurate ($\alpha = 1$) single cue and inaccurate ($\alpha = 0$) additional cues. Accuracy shown in (a). Association strength, $P(c_z|\boldsymbol{o}_{i*})$, and weight, $w_{i,z}$, shown for (b) a single cue type (M=1) and (c) five cue types (M=5).

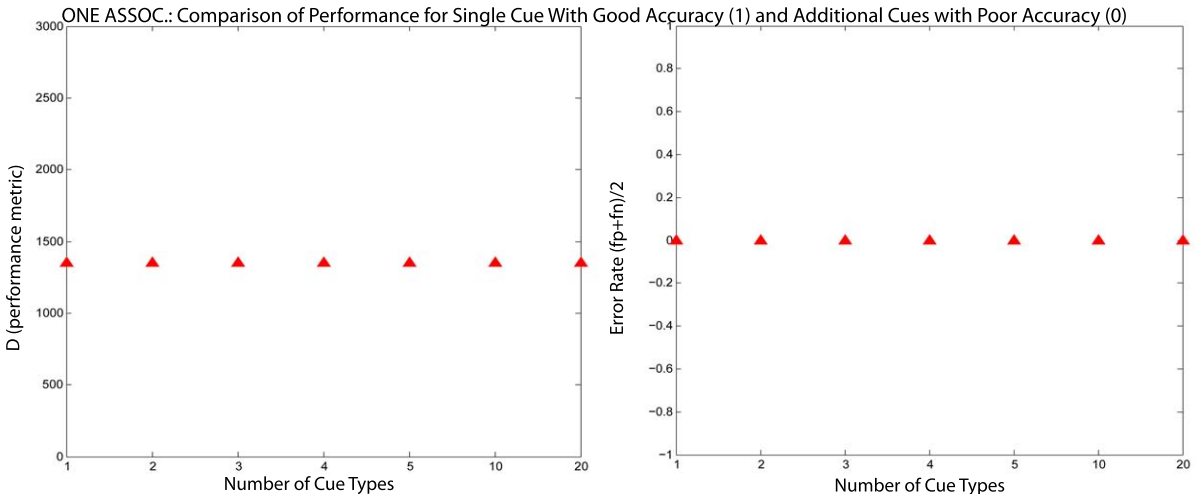


Figure 6.10: Comparison of performance of a single cue type with good accuracy ($\alpha = 1$) and additional cue types with poor accuracy ($\alpha = 0$)

Varying Cue Accuracy

As expected, the performance values still go up with the increase in accuracy (Figure 6.12. One difference, however, is that while the performance increased exponentially in the one cue value association case (Figure 6.5), the performance now increases much slower. This makes sense, however, as with the addition of multiple cue association, there is an increase in the standard deviation of the activation curve. There is more variation in the activation values since there are more combinations of cue values to deal with. An increase in standard deviation, according to the disparity performance metric, means a reduction in performance as overlap increases.

However, the more important point to note is the still overall increase in performance with the addition of more cue types. Thus, you can still get to the same level of performance with less accuracy by using multiple cue types.

Additional Cue Types have Better Accuracy than Single

We now show that with the addition of multiple cue types that have higher cue value accuracy than a single cue type, we gain in performance results. To demonstrate this, we set the accuracy for the single cue type to 0, and all additional cues to 1 (see Figure 6.13). The corresponding weights and association strengths are shown in the same figure. Note that the weights for the additional cues are much higher than the single cue, due to the

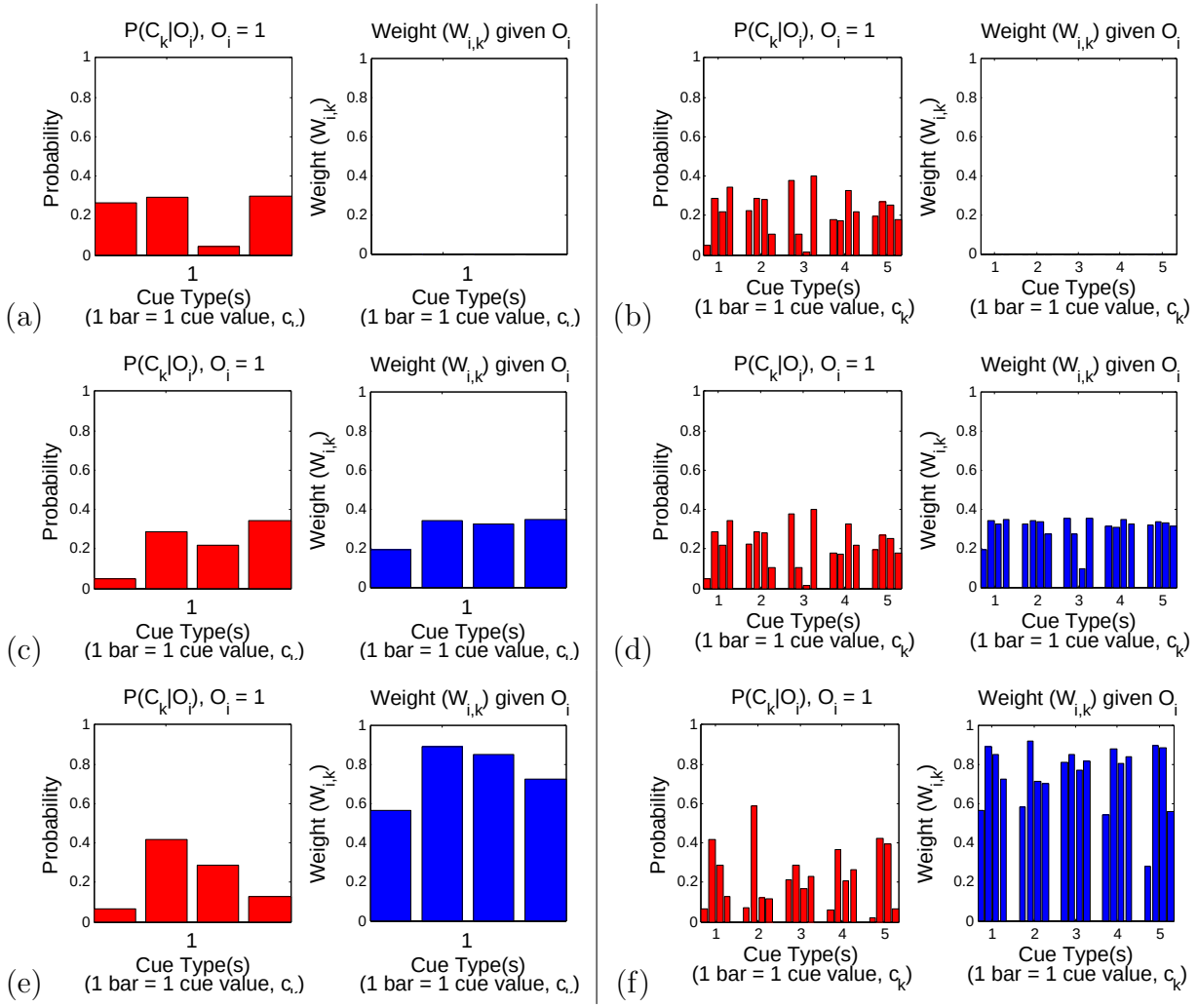


Figure 6.11: Example of association strength, $P(c_z|\boldsymbol{o}_{i^*})$, and weight, $w_{i,z}$, for multiple cue value association. First column corresponds to a single cue type (M=1), second column corresponds to five cue types (M=5). Rows correspond to accuracy: (a)&(b) $\alpha = 0$, (c)&(d) $\alpha = .4$, (e)&(f) $\alpha = 1$.

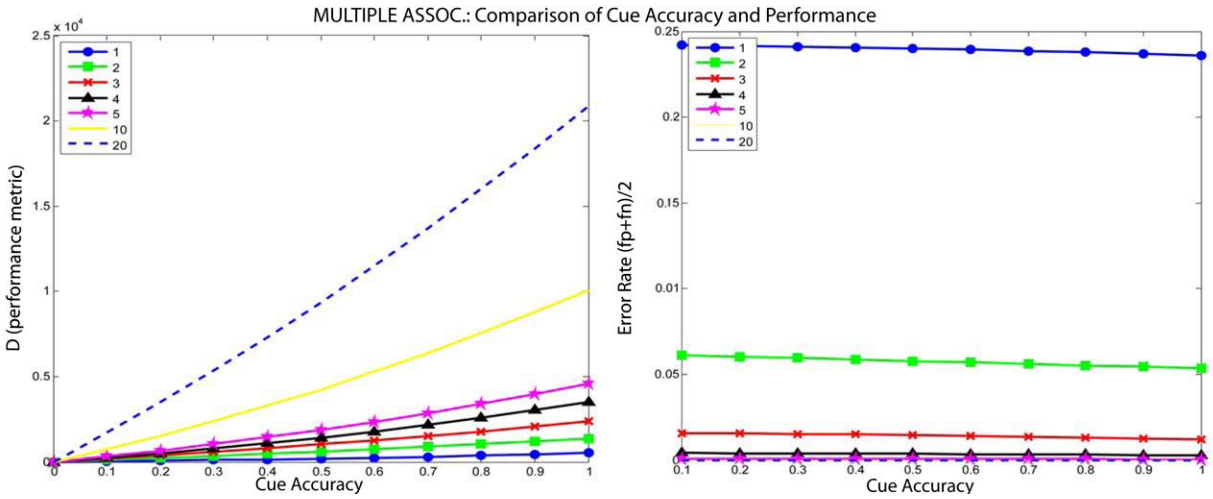


Figure 6.12: Comparison of cue accuracy and performance for multiple cue value association

much higher accuracy.

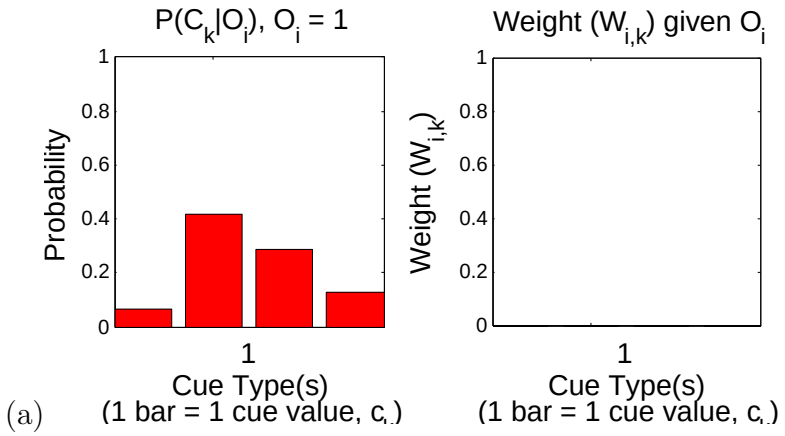
In Figure 6.14, we see the results, where as the number of cue types increase, the performance metric D increases. This shows that the use of additional cues helps the overall performance for recognition even with multiple cue value associations.

Single Cue Type has Better Accuracy than Additional

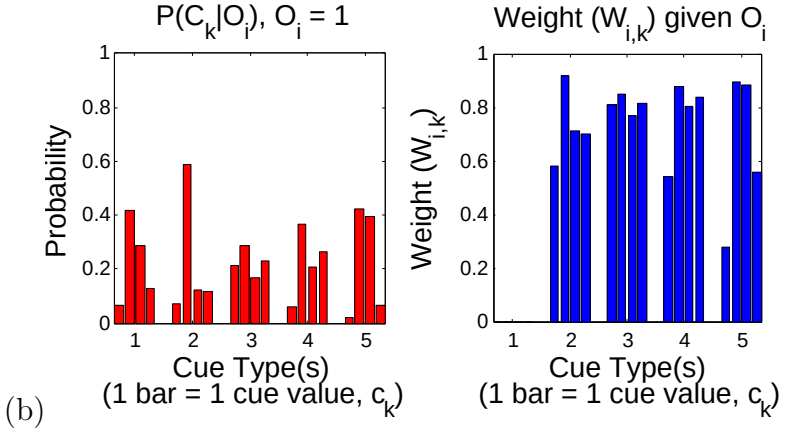
We now demonstrate that the addition of inaccurate cue types ($\alpha = 0$) does not worsen the performance results of the single cue alone in the multiple cue value association scenario. This is important as it demonstrates that the addition of multiple cue types can only benefit the overall performance.

In Figure 6.15, the parameters used to define this scenario are given. Note that the single cue has better accuracy ($\alpha = 1$) than all the additional cues. The weights match this accuracy as well as the multiple cue value association of this scenario.

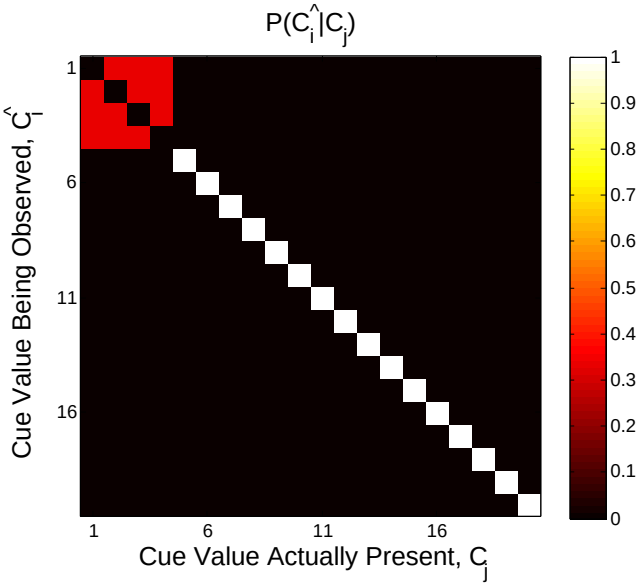
Figure 6.16 demonstrates that as the inaccurate additional cues are added the overall performance stays at the level of the accurate single cue. This means that given the worst case scenario, the algorithm will only ever do as bad as its best cue even with multiple cue values associated for each type.



(a)



(b)



(c)

Figure 6.13: Parameters used for multiple cue value association with one inaccurate ($\alpha = 0$) single cue and accurate ($\alpha = 1$) additional cues. Accuracy shown in (a). Association strength, $P(c_z|\boldsymbol{o}_{i^*})$, and weight, $w_{i,z}$, shown for (b) a single cue type (M=1) and (c) five cue types (M=5).

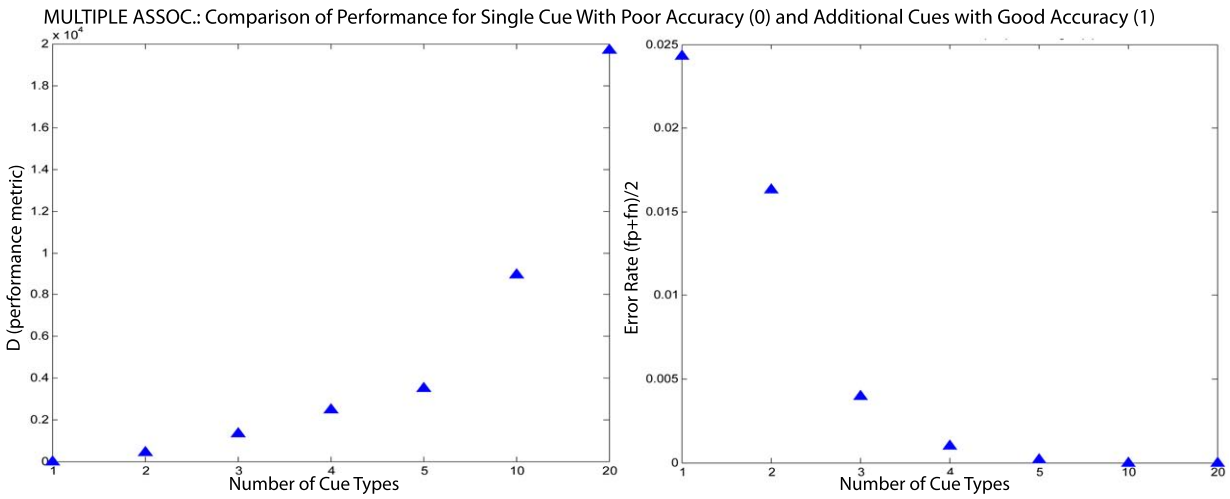
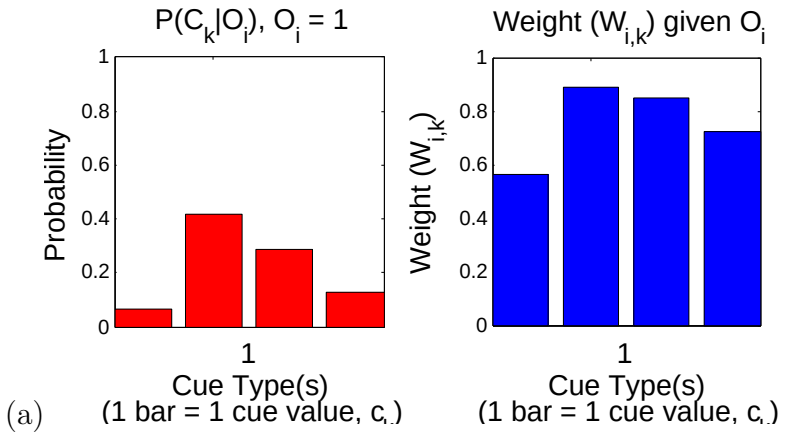


Figure 6.14: Comparison of performance for a single cue type with poor accuracy ($\alpha = 0$) and additional cues with good accuracy ($\alpha = 1$)

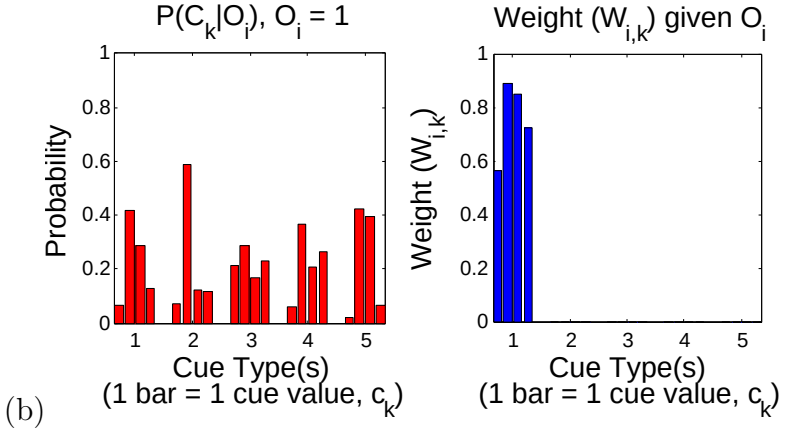
6.3.3 Ambiguous/Misleading cue values

In this section, we model two typical scenarios seen in our real data experiments. The first is the case when we have the same cue value associations for two different objects given the same cue type (For example, both a projector screen and eraser board having the same shape cue aspect ratios). In this case, it is not possible to recognize the objects based on the single shape cue alone, additional cues are needed to disambiguate the two. The second is the case of a misleading cue, i.e. someone incorrectly uses an object leading to the belief that it may be the incorrect object class (For example, if someone were repeatedly sitting on a table). It is then necessary to have additional cue types to correct for this mistake.

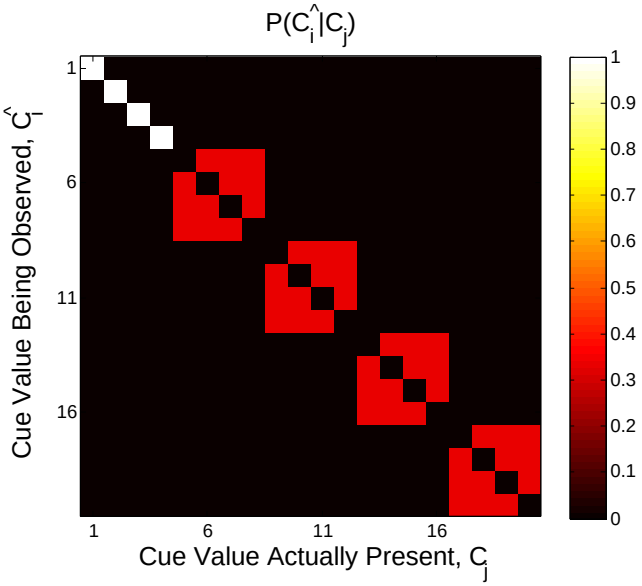
In both these instances, we can reflect the addition of extra cue representations besides those generated by the true object class or noise, by increasing the value of $P(c_v|\boldsymbol{o}'_{i^*})$ for the given ambiguous or misleading cue value, c_v. More specifically, we set $P(c_v) = P(c_v|\boldsymbol{o}_{i^*})$ so that $P(c_v|\boldsymbol{o}'_{i^*}) = P(c_v|\boldsymbol{o}_{i^*})$. We use single cue value associations for each type in order to focus on the effect of the ambiguity caused by the increase in $P(c_v|\boldsymbol{o}'_{i^*})$. Figure 6.17 shows examples of the parameters $P(c_v|\boldsymbol{o}_{i^*})$ and $w_{i,v}$, reflecting the ambiguous/misleading cue value association in each cue type for $M = 1$ and $M = 5$.



(a)



(b)



(c)

Figure 6.15: Parameters used for multiple cue value association with one accurate ($\alpha = 1$) single cue and inaccurate ($\alpha = 0$) additional cues. Accuracy shown in (a). Association strength, $P(c_z|\boldsymbol{o}_{i^*})$, and weight, $w_{i,z}$, shown for (b) a single cue type (M=1) and (c) five cue types (M=5).

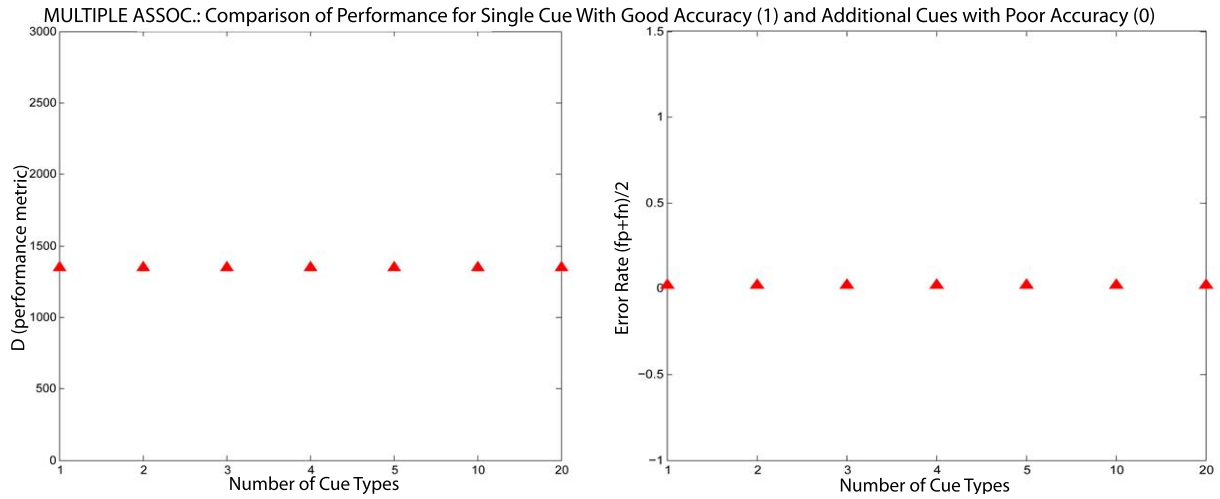


Figure 6.16: Comparison of performance for a single cue type with good accuracy ($\alpha = 1$) and additional cue types with poor accuracy ($\alpha = 0$)

Varying Cue Accuracy

Figure 6.18 demonstrates the performance results with varying cue accuracy for the ambiguous cue value association case. Two metrics are shown: disparity and error rate. The results show an increase in performance as additional cue types are added to help disambiguate the originally confusing cue data. As expected, with the addition of more and more information, i.e. a larger set of cue types, we have better performance results, which only increases as the accuracy increases.

An interesting results to note, is that in the one cue type case, you actually have a reduction in performance with an increase in accuracy. This makes sense since the cue is found both when an object is there and when it is not, more accuracy only pronounces this ambiguity.

Additional Cue Types have Better Accuracy than Single

As in the other two scenarios, we see how performance is affected with the addition of multiple cue types with high accuracy ($\alpha = 1$) (see Figure 6.19 for parameters). The corresponding weights and association strengths are shown in the same figure. Note that the weights for the additional cues are much higher than the single cue, due to the much higher accuracy.

In Figure 6.20, we see the results, where as the number of cue types increase, the performance metric D increases. This shows that the use of additional cues helps the overall

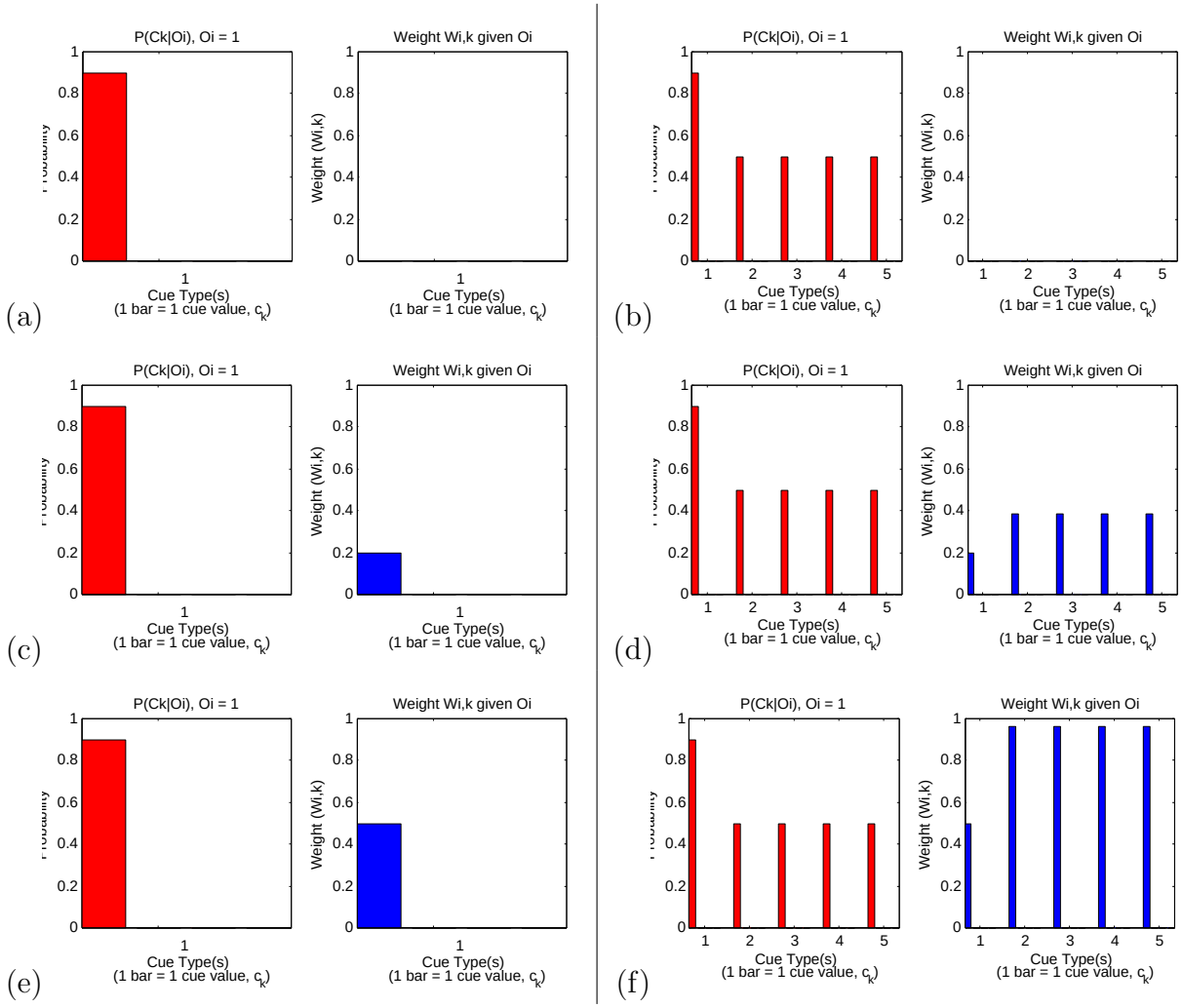


Figure 6.17: Example of association strength, $P(c_z|\boldsymbol{o}_{i^*})$, and weight, $w_{i,z}$, for ambiguous/misleading cue value association. First column corresponds to a single cue type (M=1), second column corresponds to five cue types (M=5). Rows correspond to accuracy: (a)&(b) $\alpha = 0$, (c)&(d) $\alpha = .4$, (e)&(f) $\alpha = 1$.

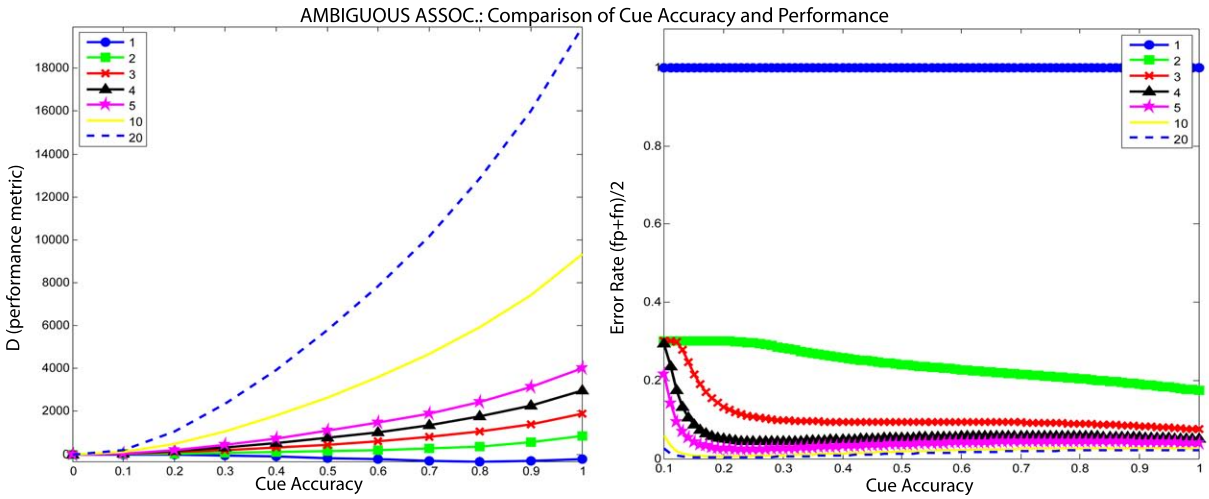


Figure 6.18: Comparison of cue accuracy and performance for ambiguous cue value association

performance for recognition when ambiguous or misleading cues exist.

Single Cue Type has Better Accuracy than Additional

We now demonstrate that the addition of inaccurate cue types ($\alpha = 0$) does not worsen the performance results of the single cue alone even with ambiguous cue association values.

In Figure 6.21, the parameters used to define this scenario are given. Note that the single cue has better accuracy ($\alpha = 1$) than all the additional cues. The weights match this accuracy as well as the multiple cue value association of this scenario.

Figure 6.22 demonstrates that as the inaccurate additional cues are added the overall performance stays at the level of the accurate single cue. This means that given the worst case scenario, the algorithm will only ever do as bad as its best cue even with multiple cue values associated for each type.

Thus, the multiple cue framework provides the ability to disambiguate the recognition of objects where a single cue is too ambiguous to distinguish between them.

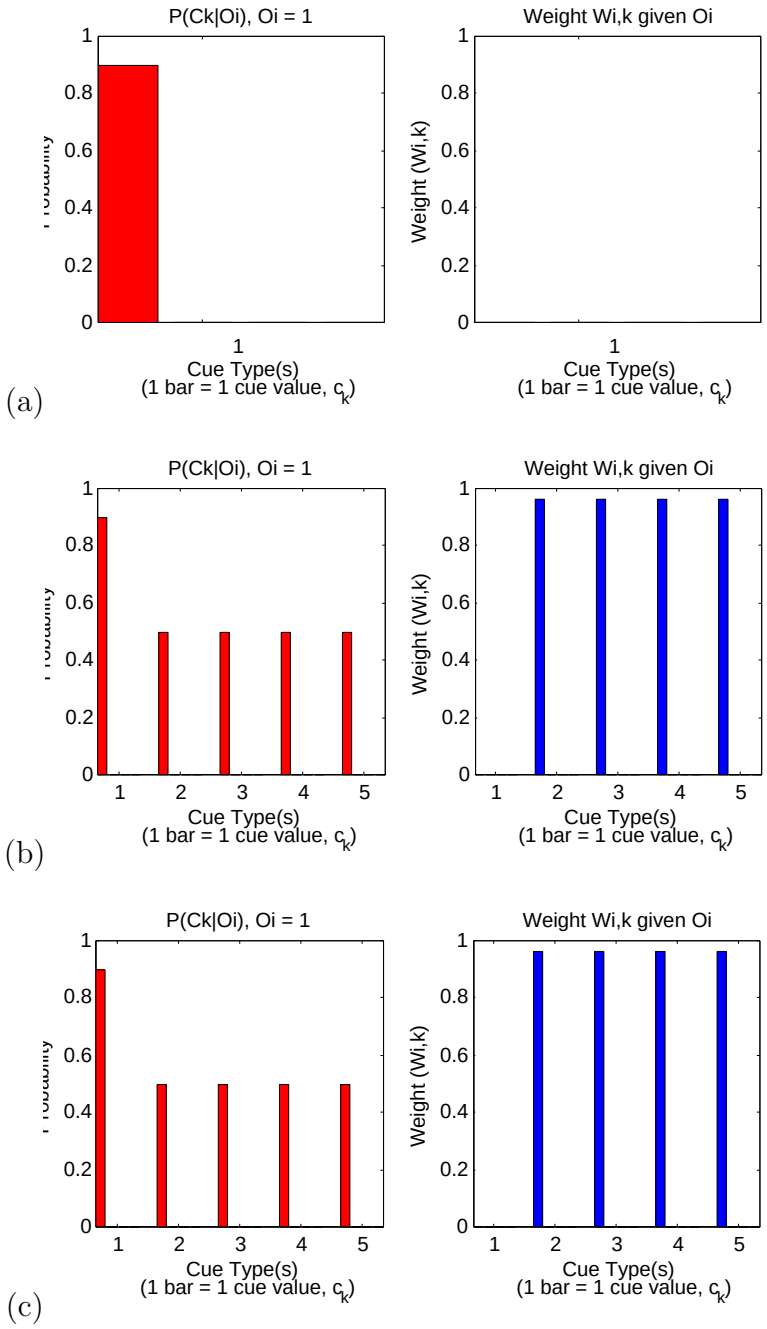


Figure 6.19: Parameters used for ambiguous/misleading cue value association with one inaccurate ($\alpha = 0$) single cue and accurate ($\alpha = 1$) additional cues. Accuracy shown in (a). Association strength, $P(c_z|\boldsymbol{o}_{i^*})$, and weight, $w_{i,z}$, shown for (b) a single cue type (M=1) and (c) five cue types (M=5).

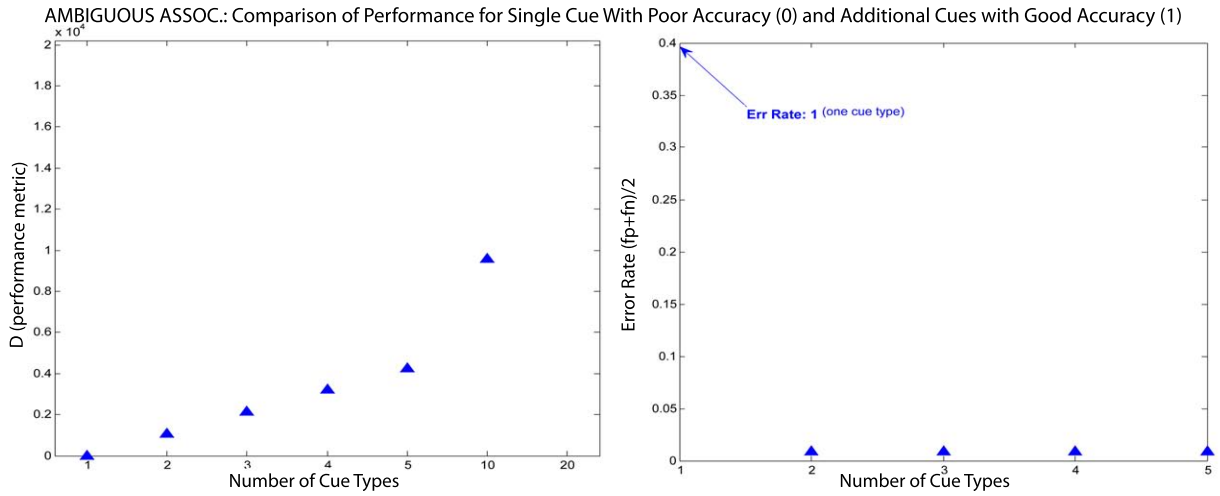


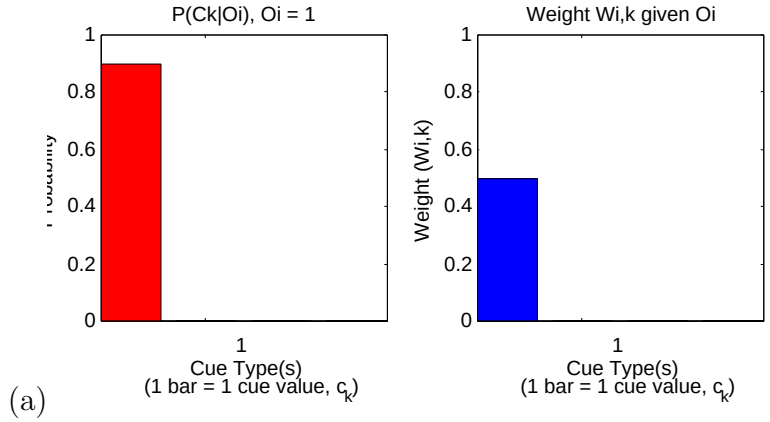
Figure 6.20: Comparison of performance for single cue type with poor accuracy ($\alpha = 0$) and additional cue types with good accuracy ($\alpha = 1$) for ambiguous cue value association

6.4 Simulated Results

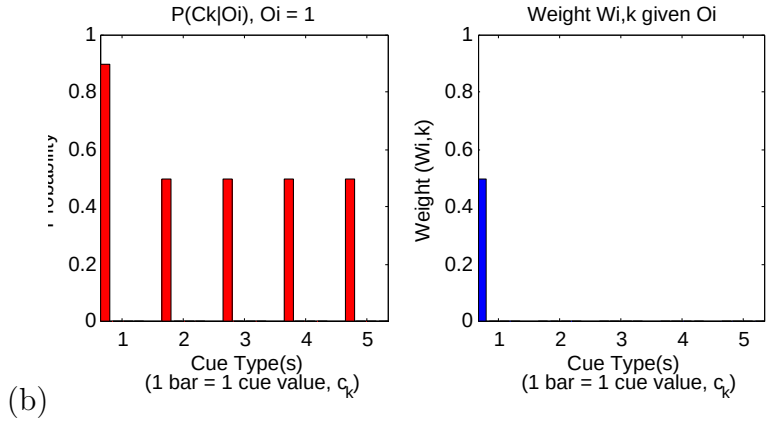
In order to illustrate the mathematical formulas and analytical model of the mean and standard deviation with simulated results, we ran each of the scenarios in a simulator based on the same probabilities and parameters outlined above. The results of this simulation should correspond with the performance results predicted by the analytical model.

Figure A.1 show simulation accuracy results for the single cue value association, multiple cue value association, and ambiguous cue value scenarios for both the disparity and error rate metrics. These graphs confirm the accuracy of the previous analytical graphs. In Figure A.1, we see that when we varied the accuracy, the disparity and error rate results of the simulation showed the same pattern described by the analytical model of the graphs: For the single cue value association, we see that there is slightly exponential improvement in performance (i.e., increased disparity, decreased error rate) as accuracy increases and overall better performance with the addition of more cue types. Similarly for the multiple cue value and ambiguous cue scenarios, except with a more linear improvement as accuracy increases for the multiple cue value scenario.

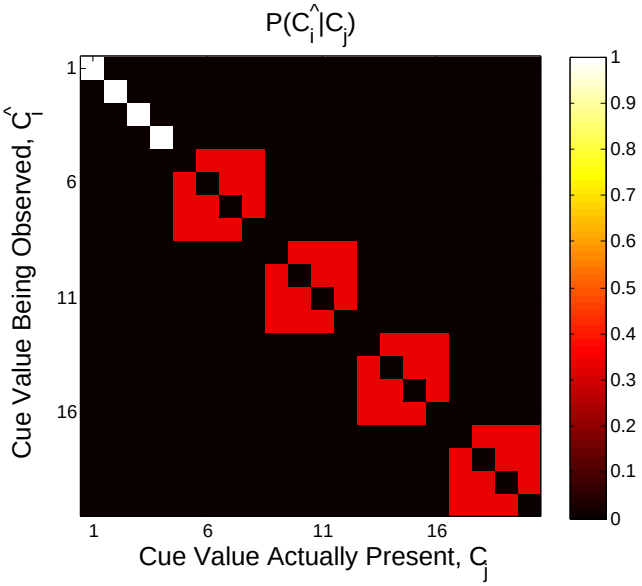
The simulation also illustrates an increase in performance with additional cue types when accuracy for those cue types are larger than a single cue type. FigureA.2 shows simulation results where additional cue types have greater accuracy than the single cue type for the single cue value association, multiple cue value association, and ambiguous cue value scenarios for both the disparity and error rate metrics.



(a)



(b)



(c)

Figure 6.21: Parameters used for ambiguous/misleading cue value association with one accurate ($\alpha = 1$) single cue and inaccurate ($\alpha = 0$) additional cues. Accuracy shown in (a). Association strength, $P(c_z|\boldsymbol{o}_{i^*})$, and weight, $w_{i,z}$, shown for (b) a single cue type (M=1) and (c) five cue types (M=5).

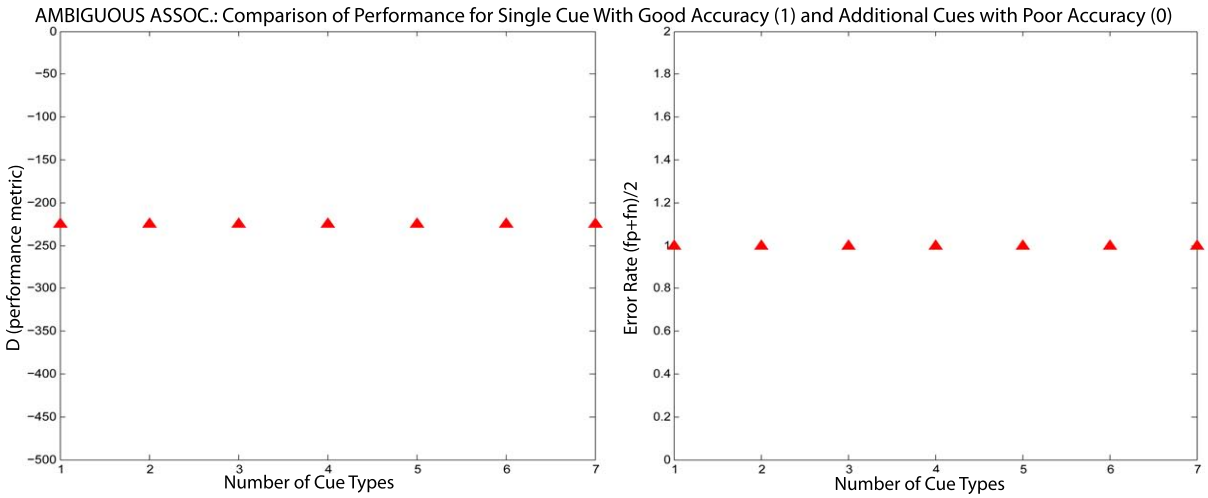


Figure 6.22: Comparison of performance for a single cue type with good accuracy ($\alpha = 1$ and additional cue types with poor accuracy ($\alpha = 0$) for ambiguous cue value association

In addition, no degradation in performance is shown when the accuracy of the additional cues are less than the single cue type. Figure A.3 show the simulation results where additional cue types have worse accuracy than a single cue type for the single cue value association, multiple cue value association, and ambiguous cue value scenarios for both the disparity and error rate metrics.

6.5 Non-Uniform Accuracy Distribution

Another aspect we illustrate is how the analytical results are affected when cue accuracy is not uniformly spread across all non-diagonal accuracy associations, i.e., $P(\hat{c}_j|c_i)$ where $i \neq j$.

In the previous sections, accuracy was calculated by setting the diagonal to an accuracy value and then dividing the rest of the value across the non-diagonal cue values. In these experiments, we choose a non-uniform distribution of the remaining accuracy, favoring some non-diagonal cue values over others. Intuitively, this represent the case when one cue value is more likely to be mistaken for another. For instance, if it was more likely to mistake the speech cue *"mug"* for *"jug"* than for *"whiteboard"*.

We calculate the submatrix for each cue type for this non-uniform accuracy probability, $P(\hat{c}_j|c_i)$ as follows:

index diff	$\|i-j\|=0$	$\|i-j\|=1$	$\|i-j\|=2$	...	$\|i-j\|=g-1$
probability	α	$a_1\beta$	$a_2\beta$	...	$a_g\beta$

where there are g cue values for each cue type and:

$$\sum_m a_m = 1 \tag{6.5.1}$$

$$\beta = 1-\alpha \tag{6.5.2}$$

Figure 6.23 shows an example of the accuracy values used where $\alpha = 0$, $g = 4$, and $[a_1...a_g] = [.6.3.1]$.

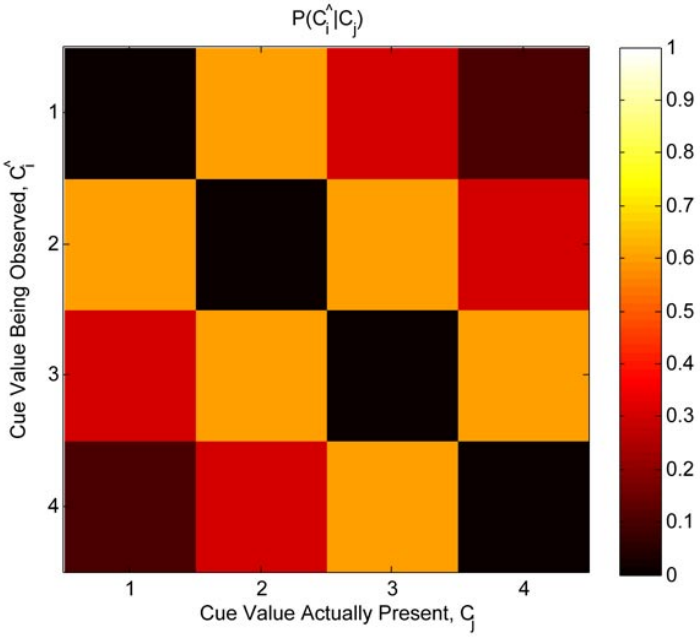


Figure 6.23: Example of non-uniform accuracy matrix $P(\hat{c}_j|c_i)$, where $\alpha = 0$, $g = 4$, and $[a_1...a_g] = [.6.3.1]$.

We ran experiments with the single cue value, multiple cue value association, and ambiguous value scenarios. We found the same pattern of results as with the uniform case (see Figures A.4, A.5, and A.6).

6.6 Relationship to Real Data Results

In this section, we compare the calculated analytical model error rate (AER) with real recognition error rates (RER) found in Chapter 7. We use the recognition results of the cooking video datasets, more specifically, we look at the error rate for individual cues vs. all cues with 100% spatial accuracy for .1(low), system, and 1(high) cue accuracy (Figure 7.15). We focus on 100% accuracy because the anaytical model does not include error from an incorrect spatial association.

The system cue accuracy, i.e., the accuracy based on real cue recognition systems, are based on the values shown in Table 7.2: Activity accuracy is .77, visual accuracy is .98, speech accuracy is .88, and sound accuracy is .89.

To implement the real recognition results in the analytical model, we take the weight values and prior probabilities learned from the probabilistic relational models learned in Chapter 3 and plug them into the analytical model. The number of cue values for each cue type are defined by the dataset cue values defined in Section 7.4.

In Table 6.1, we show a comparison of the error rate for the real recognition and the analytical model for experiment: Error rate for individual cues vs. all cues with 100% spatial accuracy for .1(low), system, and 1(high) cue accuracy. From the results, we see that the analytical error rate is a lot smaller than the real recognition error results with an average of .310 difference between the error rates. This is expected as the analytical model does not have to deal with as much noise as that found in real data. The important thing to note, however, is that the error rate for each case follow roughly the same pattern, where the speech cue typically has the greatest error rate, followed by the sound cue, then activity, and lastly, vision. The only exception is the low accuracy case, where in the real experiment activity has a slighlty higher error rate than sound. This is not unexpected as sound and activity have a fairly similar error rate, and thus any fluctuation in noise with the real recognition data could cause one to be slightly higher than the other. The analytical model, not subject to the same noise, remains steady across accuracy in regards to this pattern. Also, note that the highest analytical error rate for the individual cues with low accuracy is .24. This is consistent with the analytical results for the multiple cue value association figure (Figure 6.11), in which the error rate for a single cue is around .25 for the same cue accuracy. When all cues were used in recognition, the analytical rate dropped to .0. This is also consistent with Figure 6.11. This low error rate also corresponds to the signficantly lower all cue error rate for the real recognition results.

From these results, we see that the analytical model can be used to describe the overall pattern of error rate for the varying cue accuracy and number of cue types for real recognition results after some adjustment to noise.

Accuracy	.1(low)		system		1(high)	
Cue Type	RER	AER	RER	AER	RER	ARE
Activity	.545	.17	.335	.14	.255	.14
Speech	.525	.15	.210	.08	.140	.08
Vision	.690	.24	.410	.23	.400	.23
Sound	.530	.18	.335	.17	.295	.17
All	.460	.0	.125	.0	.095	.0

Table 6.1: Comparison of Analytical Error Rate and Real Error Rate for Individual Cue Types vs. All Cue Types

6.7 Conclusion

In this chapter, we have demonstrated the key concept that multiple cue information can only improve performance regardless of circumstance, where, at worse case, scenario when additional cue types have worse accuracy than the original signal cue type, performance does as well as the single cue. We have also shown that scenarios where a single cue type may be ambiguous or misleading for particular objects, multiple cues can help in disambiguating and correcting the error leading to better performance.

We demonstrated these finding using an analytical model, illustrated by simulation results.

Chapter 7

Experiments and Evaluation

In order to assess our framework, we provide four main methods of evaluation using different datasets: First, we provide empirical results which demonstrate the basic features and use of the MCOR algorithm using a data from a meeting setting (*Meeting dataset*, Section 7.2). Second, we demonstrate the use of the MCOR algorithm on a dataset used by another object recognition system which uses activity recognition information [23](*Gupta dataset*, Section 7.3), as well as comparing the MCOR algorithm to vision-only recognition methods. Third, we provide a more extensive exploration of MCOR's ability to recognize under varying conditions including the effect of varying cue and spatial association accuracy (*Cooking dataset*, Section 7.4). Lastly, we show results demonstrating the advantages of the MCOR framework with multiple people (*Meeting Extended dataset*, Section 7.5).

7.1 Experiment Process and Training

Before we go over the results of each experiment, we give a brief overview of the experiment process and training.

As mentioned in earlier chapters, we need an initial object dictionary before recognition can begin. The object dictionary tells us how strongly associated various cue values are with each object category, represented by the weight values. The set of cue values and object categories varies with each experiment. We summarize the set of cue values and object categories for each experiment. In some of these experiments, these object dictionaries were hand-defined (Meeting, Gupta, and Meeting Extended datasets), while, in others (Cooking dataset), the dictionaries were learned from data. For all of the experiments, during recognition, an update of the object dictionary occurred according to the generalization

process.

We now go over the experimental process. We then describe how training occurred for each experiment giving an example of an initial object dictionary and final object dictionary. To see the complete set of object dictionaries used for each dataset, see Appendix B.

7.1.1 Experimental Process

Although we describe in detail each experiment in the sections below, we give a brief outline of the experiment steps used in each of the experiments with an overview of the values.

- Collect video dataset.

  Meeting dataset Video taken from omnidirectional camera consisting of three 30-60s videos in a meeting setting with a single person in each video.

  Gupta dataset Video taken from University of Maryland dataset consisting of forty-six 5-10s videos in a laboratory setting with a single person in each video.

  Cooking dataset Video taken from three sources: Rachel Ray TV cooking show, LACE dataset (University of Rochester), and Gia De Laurentiis TV cooking show. Cooking datasets consist of fifty 1-30s videos, forty 1-60s videos, and twenty-five 1-20s videos in a kitchen setting with a single person in each video.

  Meeting Extended dataset Video taken from digital video camera consisting of five 10-60s videos in a meeting setting with multiple people.

- Define cue values and object categories.

  Meeting dataset Object categories and cue values for Meeting dataset consist of :

Object categories:	laptop, chair, projector screen, whiteboard, table
Activity cue values:	*sit, walk, stand*, point, erase, write, putdown (*Rybski recognizer*)
Speech cue values:	"look that up"
Color cue values:	hsv color
Shape cue values:	aspect ratio of bounding box

  (Note: cue values are hand-labeled unless otherwise specified)

  Gupta dataset Object categories and cue values for Gupta dataset consist of :

Object categories:	cup, flashlight, phone, spray can
Activity cue values:	drink, pour, light, answer, spray
Color cue values:	hsv color
Shape cue values:	aspect ratio of bounding box
SIFT cue values:	SIFT features

  (Note: cue values are hand-labeled unless otherwise specified)

Cooking dataset Video taken from three sources: Rachel Ray TV cooking show, LACE dataset (University of Rochester), and Gia De Laurentiis TV cooking show. Cooking datasets consist of fifty 1-30s videos, forty 1-60s videos, and twenty-five 1-20s videos in a kitchen setting with a single person in each video.

Rachel Ray and Gia De Laurentiis Datasets

Object categories:	bowl, knife, oil jar, wine bottle, cutting board, plate, grill, pan, spoon
Activity cue values:	twist, smash, shake, drop/add, cut, mix, pour, squeeze, pick up, put down, scoop, point
Speech cue values:	"bowl","knife", "oil jar", "wine bottle", "cutting board", "plate", "grill", pan", "spoon"
Sound cue values:	pour, sizzle, mix, chop
Color cue values:	hsv color
Shape cue values:	aspect ratio of bounding box
SIFT cue values:	SIFT features

LACE dataset

Object categories:	bowl, knife, cup, cereal, spoon, cutting board, jug, fridge, plate, cupboard
Activity cue values:	eat, drink, bring, return, open, close, pour, mix
Speech cue values:	"bowl","knife", "cup", "cereal", "spoon", "cutting board", "jug", "fridge", "plate", "cupboard"
Sound cue values:	pourLiquid, pourSolid, mix
Color cue values:	hsv color
Shape cue values:	aspect ratio of bounding box
SIFT cue values:	SIFT features

(Note: cue values are hand-labeled unless otherwise specified)

Meeting Extended dataset Video taken from digital video camera consisting of five 10-60s videos in a meeting setting with multiple people.

Object categories:	chair, mug/cup, pen
Activity cue values:	sit, drink, write
Speech cue values:	"chair", "pen", "cup", "mug"
Color cue values:	hsv color
Shape cue values:	aspect ratio of bounding box

- Define object dictionary. (see next subsection for training)

- Recognize specified object categories using defined object dictionary and cue values. (see Sections 7.2,7.3, 7.4, 7.5)

7.1.2 Training

In this subsection, we outline how training was done for each of the datasets (Meeting, Gupta, Cooking, and Meeting Extended) in order to determine the weight value in the object dictionary. The weight specifies the strength of the association between an object and a cue value (i.e., $p(o_i|cv_j)$. Training for each of the datasets occurred as follows:

Meeting dataset The initial object dictionary values were hand-defined. The initial object dictionary consisted of activity and speech information. Through generalization during recognition, visual cues (color and shape) were learned from objects recognized by the initial dictionary. New cues added to a definition were given a predefined value (.8). In Figure 7.1, we show an example of the initial object dictionary with weights hand-defined. In Figure 7.2, we see the final object dictionary where learned visual descriptors were added to the dictionary with the predefined weight. In order to make the color and shape information readable in the figure, we represented the HSV values with color names and indicated the different shapes through numbering.

OBJECT	Chair	Laptop	Projector Screen	Whiteboard	Table
Activity					
Sit	**.8**	**.2**	0	0	0
Stand	0	0	0	0	0
Walk	0	0	0	0	0
Erase	0	0	0	**.8**	0
Write	0	0	0	**.8**	0
Point	0	0	**.8**	**.2**	0
Putdown	0	0	0	0	**1**
Speech					
"look that up"	0	**.8**	0	0	0

Figure 7.1: Initial object dictionary for Meeting dataset

Gupta dataset The initial object dictionary values were hand-defined. The initial object dictionary consisted of association specified by the Gupta dataset where each activity was associated with only one object. Through generalization during recognition, visual cues (color and shape) were learned from objects recognized by the initial dictionary. New cues added to a definition were given a predefined value of .8. (see Appendix B for actual object dictionaries)

OBJECT	Chair	Laptop	Projector Screen	Whiteboard	Table
Activity					
Sit	**.8**	**.2**	0	0	0
Stand	0	0	0	0	0
Walk	0	0	0	0	0
Erase	0	0	0	**.8**	0
Write	0	0	0	**.8**	0
Point	0	0	**.8**	**.2**	0
Putdown	0	0	0	0	**1**
Speech					
"look that up"	0	**.8**	0	0	0
Visual					
White&Shape1	0	0	**.8**	**.8**	0
Red&Shape1	**.8**	0	0	0	0
White&Shape2	0	0	0	0	**.8**
Black&Shape1	0	**.8**	0	0	0

Figure 7.2: Final object dictionary for Meeting dataset

Cooking dataset The initial object dictionary was learned using a training dataset. The training dataset consisted of hand-labeled objects, activity information, speech information, and sound information using the ViPER software (described later) from a subset of the Cooking dataset videos. The object dictionary was updated based on recognized objects from the initial dictionary. How the object dictionary was affected by the learning during the recognition process depended on the experiment. In experiments where the initial object dictionary was learned from a training dataset where all cue types were included in the training dataset, only slight updates to the weight values occurred during recognition. Thus, the initial and final dictionaries did not differ much from each other. In Figure 7.3, a comparison of the activity weights in the object dictionary for the initial and final dictionaries of the Rachel Ray dataset are shown. For the full dictionaries including all cue types, see the appendix (Appendix B).

In other experiments where single or multiple cue types were left out of the training dataset. Final versions of the object dictionary included cue values not found in the initial dictionary. In Figure 7.4, we see an example of this updated dictionary where speech information was left out of the training dataset. The figure just focuses on the learning of the speech information at different points in the recognition process (i.e., percentage of video data

Initial Dictionary

OBJECT	Bowl	Knife	OilJar	Spice	WineBottle	Cuttingboard	Plate	Grill	Pan	Spoon
Activity										
Twist	0	0	0	**1**	0	0	0	0	0	0
Smash	0	**1**	0	0	0	0	0	0	0	0
Shake	**.2**	0	0	**.6**	0	0	0	0	0	0
Drop/Add	**1**	0	0	**.6**	0	0	0	0	0	0
Cut	0	**1**	0	0	0	**.9**	0	0	0	0
Mix	**.9**	0	0	0	0	0	0	0	**.1**	**.8**
Pour	**.8**	0	**.6**	0	**.3**	0	**.2**	0	0	0
Squeeze	**1**	0	0	0	0	0	0	0	0	0
PickUp	**.6**	**.2**	**.2**	**.3**	**.4**	**.4**	0	0	0	0
PutDown	**.6**	**.2**	**.1**	**.3**	**.2**	**.5**	**.1**	0	0	0
Scoop	0	0	0	0	0	0	0	0	0	**1**
Point	**.4**	0	0	0	0	**.3**	**.2**	0	**.3**	0

Final Dictionary

OBJECT	Bowl	Knife	OilJar	Spice	WineBottle	Cuttingboard	Plate	Grill	Pan	Spoon
Activity										
Twist	0	0	0	**1**	0	0	0	0	0	0
Smash	0	**1**	0	0	0	0	0	0	0	0
Shake	**.4**	0	0	**.6**	0	0	0	0	0	0
Drop/Add	**.9**	0	0	**.6**	0	0	0	0	**.1**	0
Cut	0	1	0	0	0	**.9**	0	0	0	0
Mix	**.9**	0	0	0	0	0	0	0	**.1**	**.8**
Pour	**.8**	0	**.6**	0	**.3**	0	**.2**	0	0	0
Squeeze	**1**	0	0	0	0	0	0	0	0	0
PickUp	**.4**	**.3**	**.2**	**.3**	**.1**	**.4**	**.1**	0	0	0
PutDown	**.5**	**.3**	**.1**	**.3**	**.1**	**.5**	**.1**	0	0	0
Scoop	0	0	0	0	0	0	0	0	0	**1**
Point	**.3**	0	0	0	0	**.3**	**.3**	0	**.3**	0

Figure 7.3: Comparison of initial and final activity object dictionary when training includes all cue types

seen): 0 percent (initial object dictionary without speech information), 50 percent, and 100 percent (final dictionary after all videos in the dataset was seen). As described in Chapter 3, weight updating occurs by updating the count estimates. In this case, where speech was left out, each time an object was recognized and a speech cue was associated with that object, the count for that object category was increased for that speech cue. From this figure, we can see how from recognition results based on multiple cues, new cues can be learned and their weights updated.

Meeting Extended dataset Initial object dictionary values were hand-defined. The initial object dictionary consisted of activity and speech information. Through generalization during recognition, visual cues (color and shape) were learned from objects recognized by the initial dictionary. New cues added to a definition were given a predefined value (.8). (see Appendix B for actual object dictionaries)

Using the experimental process and training we described in this section, we were able to produce recognition results demonstrating various aspects of the MCOR algorithm. We take about each of these aspects in the following sections.

0% - Initial Dictionary

OBJECT	Bowl	Knife	OilJar	Spice	WineBottle	Cuttingboard	Plate	Grill	Pan	Spoon
Speech										
"Bowl"	0	0	0	0	0	0	0	0	0	0
"Knife"	0	0	0	0	0	0	0	0	0	0
"OilJar"	0	0	0	0	0	0	0	0	0	0
"Spice"	0	0	0	0	0	0	0	0	0	0
"WineBottle"	0	0	0	0	0	0	0	0	0	0
"Cuttingboard"	0	0	0	0	0	0	0	0	0	0
"Plate"	0	0	0	0	0	0	0	0	0	0
"Grill"	0	0	0	0	0	0	0	0	0	0
"Pan"	0	0	0	0	0	0	0	0	0	0
"Spoon"	0	0	0	0	0	0	0	0	0	0

50%

OBJECT	Bowl	Knife	OilJar	Spice	WineBottle	Cuttingboard	Plate	Grill	Pan	Spoon
Speech										
"Bowl"	**.9**	0	0	0	0	0	0	0	0	0
"Knife"	0	0	0	0	0	0	0	0	0	0
"OilJar"	0	0	**.8**	0	0	0	0	0	0	0
"Spice"	0	0	0	0	0	0	0	0	0	0
"WineBottle"	0	0	0	0	0	0	0	0	0	0
"Cuttingboard"	0	0	0	0	0	**1**	0	0	0	0
"Plate"	0	0	0	0	0	0	**.8**	0	0	0
"Grill"	0	0	0	0	0	0	0	0	0	0
"Pan"	0	0	0	0	0	0	0	0	**.8**	0
"Spoon"	0	0	0	0	0	0	0	0	0	0

100% - Final Dictionary

OBJECT	Bowl	Knife	OilJar	Spice	WineBottle	Cuttingboard	Plate	Grill	Pan	Spoon
Speech										
"Bowl"	**1**	0	0	0	0	0	0	0	0	0
"Knife"	0	**.9**	0	0	0	0	0	0	0	0
"OilJar"	0	0	**.9**	0	0	0	0	0	0	0
"Spice"	0	0	0	**.7**	0	0	0	0	0	0
"WineBottle"	0	0	0	0	**.8**	0	0	0	0	0
"Cuttingboard"	0	0	0	0	0	**1**	0	0	0	0
"Plate"	0	0	0	0	0	0	**.9**	0	0	0
"Grill"	0	0	0	0	0	0	0	**.9**	0	0
"Pan"	0	0	0	0	0	0	0	0	**.8**	0
"Spoon"	0	0	0	0	0	0	0	0	0	**.5**

Figure 7.4: Comparison of initial, mid, and final speech object dictionaries when initial dictionary based on training data without speech

7.2 MCOR Basic Features

In this section, we demonstrate the feasibility of the MCOR algorithm on a meeting dataset taken from an omnidirectional camera (CAMEO) from the CALO project. We show results on three videos of $\tilde{3}$0-60 seconds long. Weights were hand-defined and learned through simulation as described in the next section. Activity, speech, and color and shape cue types were used. Activity information was extracted using an activity recognizer designed by [60], which included the activities cue values: *Sit*, *Stand*, *Walk Left*, *Walk Right*, *Fidget Left*, and *Fidget Right*. Additional cue values were added and labeled manually including: *Pointing*, *Erasing*, *Writing*, and *Put Down*. Speech cues consisted of one cue value: *"Look that up"*, which was manually labeled. Color and shape information was taken from the

color segmentation and region growing technique described in Chapter 4.

These results illustrate the feasibility of the MCOR framework. Three object recognition tasks were given to test various capabilities of the algorithm: the first demonstrates how the algorithm can deal with an unreliable cue type, the second demonstrates how the algorithm resolves cases where the same cue indicates the presence of different objects, and the third demonstrates how the algorithm deals with a misleading cue.

All the tasks started off with object definitions which contained cues that could be observed from the interaction of a human with the object and a weight value corresponding to the strength of the association between the cue and the object. The definitions are begun with cues that involve human interaction since those cues tend to be more obviously associated with the object since humans usually interact with the object in a manner implicit to its definition. For instance, most interactions tend to portray the function of the object which is usually an important part of its definition. Additional cue types are learned by the algorithm and added to the object definition through the video (see Figure 7.8(b,e) and Figure 7.9(b,d)) according to the generalization method described earlier.

For all recognition tasks, segmentation is based on a color-based region growing algorithm, described in detail in Chapter 4. The similarity measure for the activity and vision cues will be binary, where $s_{c_j,c_l} = 1$, if the cues are the same, 0 otherwise. 'Same' for the activity cue means that the activity label is the same for each cue and 'same' for the vision cue means the color distance between the cues and the shape ratio are within a threshold, t_c. Hence, the value of the evidence will simply depend on the sum of the weights of the cues found in the definition. For simplicity, we will not refer to the value or multiplication of s_{c_j,c_l} in the sum, although it is implicitly there.

7.2.1 Weight Learning

In order to illustrate how the weight values for each cue and object could be learned, synthetic data was generated by a simulator. Given a set of objects, the simulator generated cues based on predetermined model which represents the probabilities of a cue being produced given the presence of an object. It is this model which the weights attempt to learn using the PRM technique outlined below.

In order to demonstrate how the weights used in the real data scenarios described below could have been learned, the simulator was set up using the predefined weights shown in the next section as the model. Thus, three scenarios were produced by the simulator matching the three scenarios of the real data. The weights generated by the PRM learning are then compared with the true model values as shown in Figures 7.5, 7.6, 7.7. The simulation

generated 100 runs for each scenario in order to learn the weights. No cue error is assumed.

In the first scenario, there were two objects, a whiteboard and a projector screen, with activity cue values, i.e., *Pointing*, *Erasing*, and *Writing*. Their probabilities are represented in Figure 7.7, in addition to the learned weights. In the second scenario, there were two objects, laptop and chair. The extracted cues consisted of a speech and an activity cue, i.e., *"look that up"* and *Sit*. The third scenario consisted of a table and chair with the cues being the actions, *Put Down* and *Sit*.

With an average error of .003 between the true and the learned weights, one can see that the PRM learning technique was able to successfully learn the true model used by the simulator.

Object and Cues	True Weight Value	Learned Weight Value
Whiteboard		
POINTING	.2	.198
ERASING	.8	.799
WRITING	.8	.798
Projector Screen		
POINTING	.8	.801

Figure 7.5: Comparison of learned weights to true weights in first scenario of meeting dataset experiment

Object and Cues	True Weight Value	Learned Weight Value
Laptop		
SIT	.2	.199
"look that up"	.8	.799
Chair		
SIT	.9	.902

Figure 7.6: Comparison of learned weights to true weights in second scenario of meeting dataset experiment

Object and Cues	True Weight Value	Learned Weight Value
Chair		
SIT	.8	.798
Table		
PUT_DOWN	.3	.289

Figure 7.7: Comparison of learned weights to true weights in third scenario of meeting dataset experiment

Unreliable Cue Type

For the first object recognition task, the goal was to recognize two objects: a whiteboard and a projector screen, which tend to have very similar visual features, in this case, similar color and shape–a task which most vision-based object recognition systems would have extreme difficulty with. In both cases, however, MCOR was able to successfully label each object (see Figure 7.8(e)). Figure 7.8(a) illustrates how a new cue, a vision *Color&Shape* cue, is added to the definition of the projector screen. The figure shows each key time step as a row. In the row from right to left is a frame from the video, the status of the object definitions, and the current calculation of evidence. A key time step is when a new cue is extracted, since otherwise there is no change in the evidence and everything remains the same. In each frame, segmented regions are surrounded with a border and filled with a solid color, activity cues are labeled with all letters capitalized next to a box surrounding the face of the person doing the action, speech cues are white with quotes and object labels are white with all letters capitalized. In each object definition box, each object is in bold followed by the cues associated with it: the type of cue, the cue value, and the weight. The evidence column shows the current calculation of evidence that a particular region, r_n, is a particular object, o_i, for each region and each object. This step leads to an initial mislabeling of the whiteboard (see Figure 7.8(b)) as a projector screen. At this point, most vision-based recognition systems would have no recourse to correct the mislabel. MCOR, however, is able to overcome the obstacle by taking into account evidence from other types of cues (see Figure 7.8(c-d)), in this case, pointing, writing, and erasing activity cues.

Thus, more robust object recognition can be produced when the ability to include cues of various types is provided. This approach allows the system to be less dependent on the weaknesses of any single type.

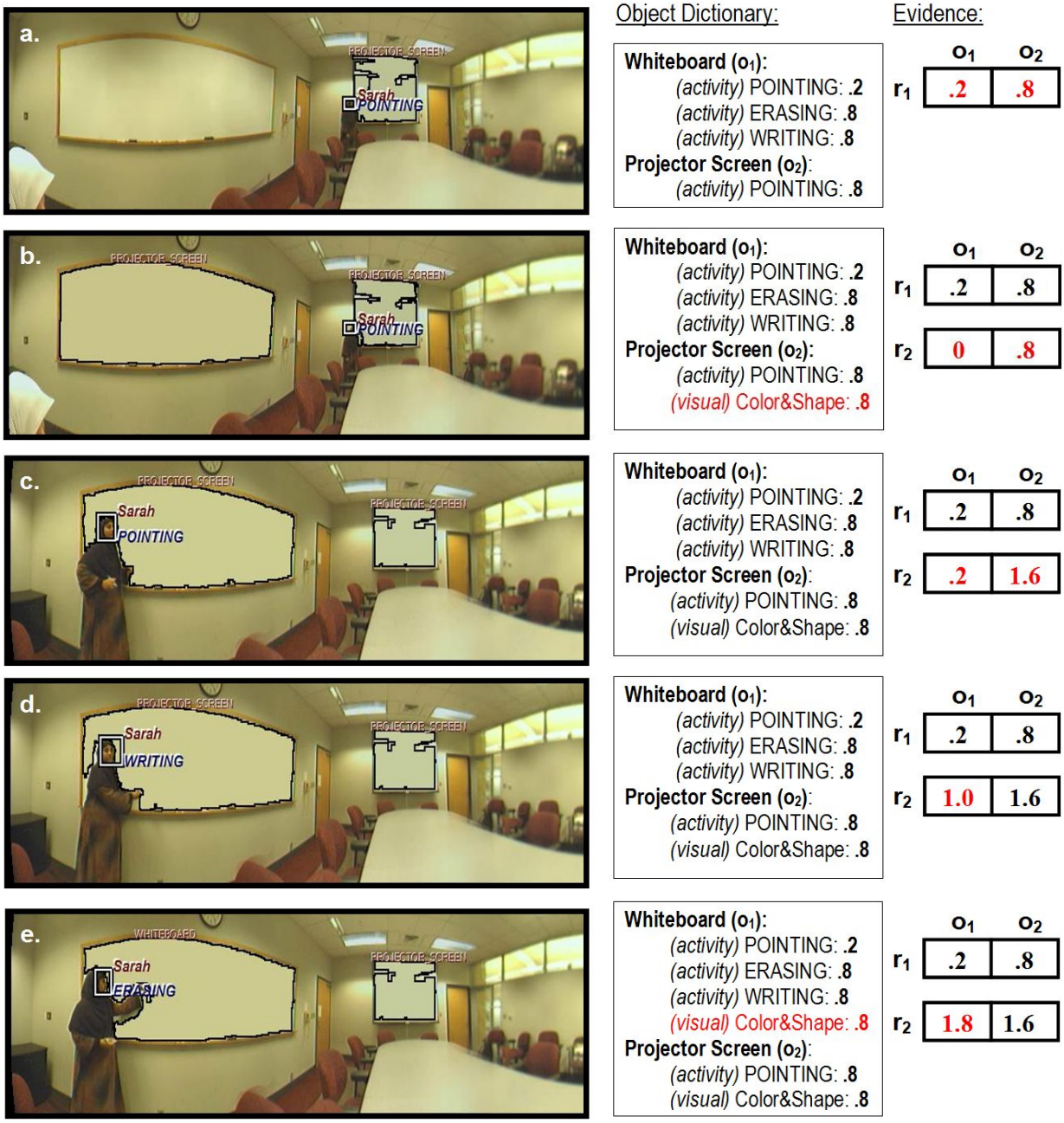


Figure 7.8: Meeting experiment recognition result for unreliable cue type scenario

Same Cue Associated with Different Objects

In the second task, two types of objects were to be recognized: laptops and chairs. An additional difficulty was added by having both the laptop and chair objects associated with

the same activity, i.e., *Sitting*. Figure 7.9(d) shows the results of the object recognition task, where despite the difficulty, the two laptops in the scene and a number of chairs were correctly labeled (Some chairs were missed because they were neither sat upon nor matched with the vision cue. Thus, additional cues would have to be learned or provided to recognize the rest).

Although initially both objects were labeled chair (see Figure 7.9(c)), due to the only cue provided thus far being associated with both objects, this ambiguity was able to be resolved by the algorithm's ability to take evidence from multiple cues, (in this case, additional evidence was given by a speech cue: see Figure 7.9(d)) which aided in distinguishing the laptops from the chairs.

Thus, having the same cue belong to multiple object definitions can lead to ambiguity in the recognition. This is, however, a realistic representation of the world, where it is very likely that the same cue may be correctly ascribed to more than one object, i.e., the set of cues belonging to each object will never be completely disjunct in real world scenarios, and so an algorithm must be able to handle such ambiguity. MCOR is able to do so, as demonstrated, by collecting evidence from multiple cues to decipher the difference.

Misleading Cue

In the third task, the table was to be recognized, although a chair definition was also provided. For added difficulty, a misleading cue was given. Although in most cases humans interact with objects according to their definition, on occasion, a human may misuse an object or interact act with it in an unconventional manner producing a misleading cue.

In this case, the misleading cue was produced by having the human sit on the table, an activity clearly attached to the function and definition of a chair, but somewhat adverse to that of a table. Initially, the algorithm is indeed confused and incorrectly labels the table, a chair (see Figure 7.10(a)). Although *Sitting* is a strong indication of a chair (as shown by the high value of its weight: Figure 7.10). The algorithm is able to take advantage of the fact that people tend to do the correct action far more frequently than the incorrect, as seen by the *Put Down* activities in Figure 7.10(b-d), to correct the mistake caused by the misleading cue (see Figure 7.10(d)).

In addition, Figure 7.10 shows how the color and shape of the table which was originally and incorrectly associated with the chair was placed in the proper definition after the correction of the label. A similar case can be noted in Figure 7.9.

Thus, the algorithm was able to demonstrate its robustness to misleading cues through the

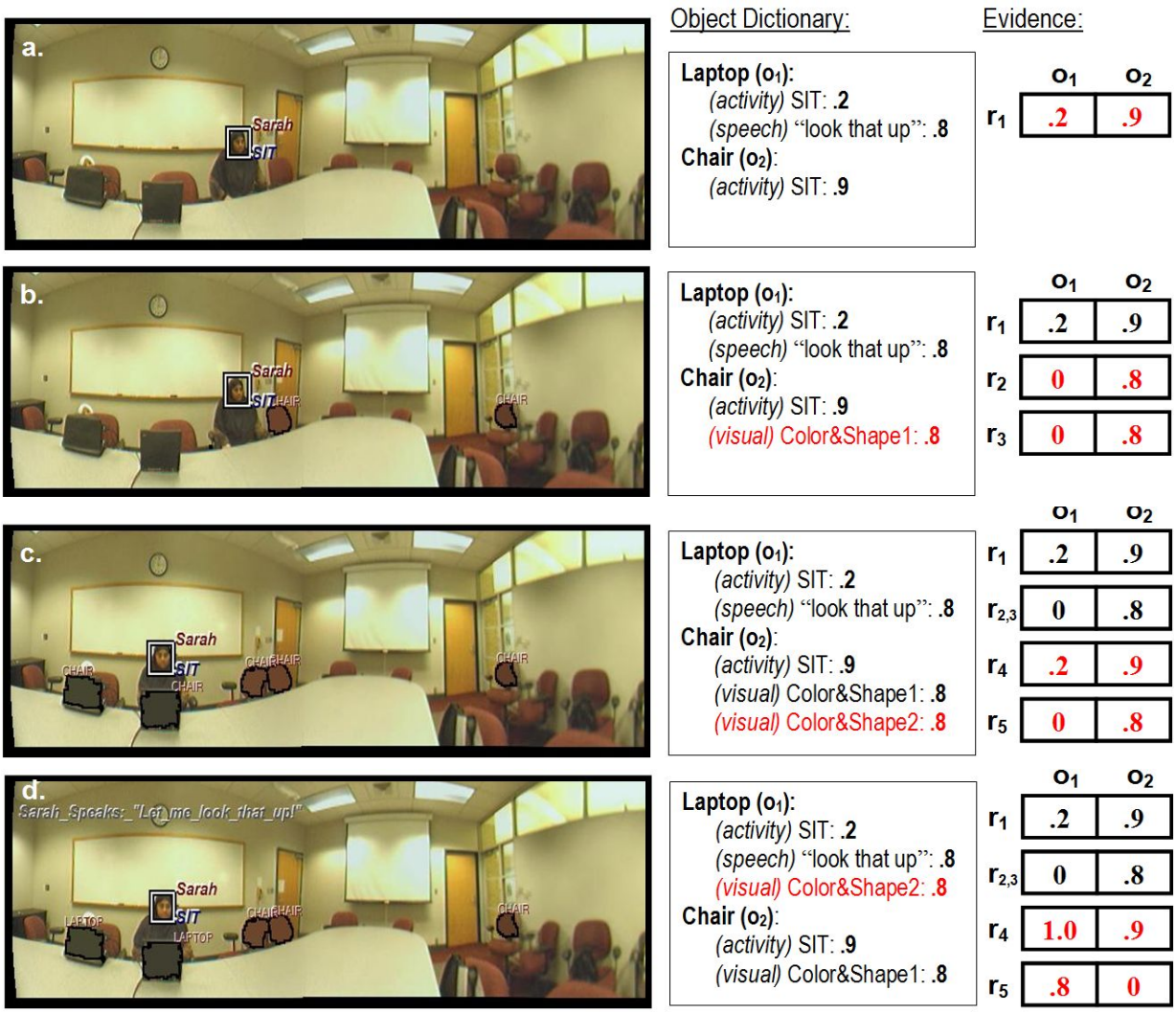


Figure 7.9: Meeting experiment recognition result for same cue Associated with different objects scenario

multiple cue framework.

7.3 Comparison to Other Methods

In this section, we demonstrate MCOR's ability to run on a datasets used by another object recognition system by Gupta et al. [23] which uses activity information. We compare the recognition rate and features of the MCOR algorithm as compared to the Gupta system

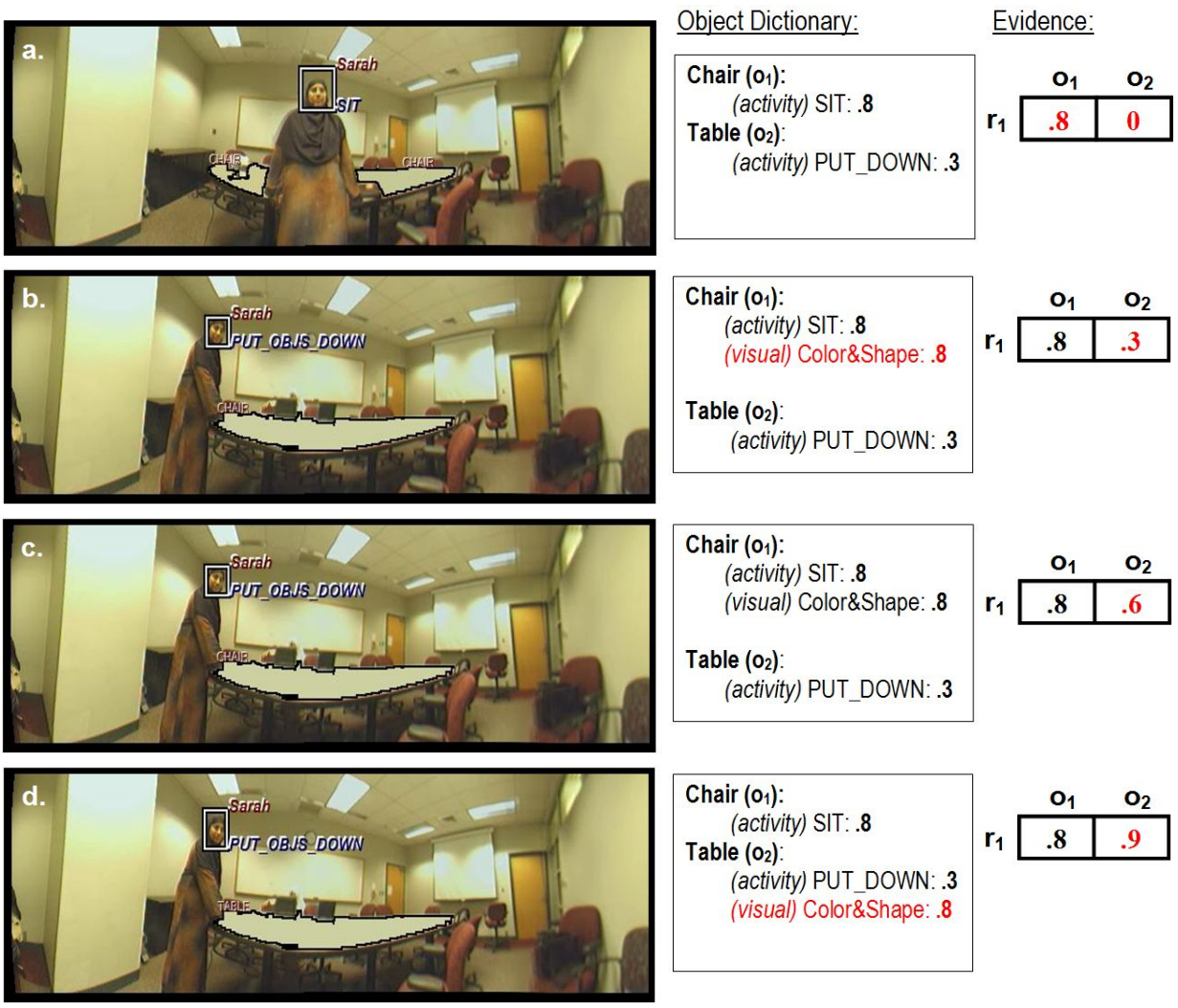


Figure 7.10: Meeting experiment recognition result for misleading cue scenario

as well as compared to the vision-only recognition system based on the popular SIFT [39] feature recognition.

We show that the recognition rate of the Gupta framework and MCOR to be comparably high. We show that MCOR provides an added feature, however, where visual features learned from one video can be generalized to recognize objects in another. We then show MCOR's improvement in recognition as compared to a vision-only system such as SIFT.

The dataset consist of forty-six videos that were five to ten seconds long. Frame size was 640x480 pixels.

The Gupta dataset is an optimal choice for frameworks interested in interactionable objects, i.e., objects that are interacted with and interact, such as the MCOR algorithm. The Davis's group used these videos to do activity recognition and then object recognition based on these activities. Here is an overall description of the dataset:

- Number of Videos: 46 videos each about 5-10 secs
- Objects Recognized: Spray can, Phone, Cup, Flashlight
- Activities Recognized: 5 activities: Spraying, Answering, Lighting, Pouring, Drinking

Gupta was able to recognize 98.67% of the objects using activity information based on a histogram of oriented gradients using an adaboost classifier [23], this was an improvement over the 78.33% recognition rate without activity information.

We did not have access to their activity recognition, so for our object recognition and for comparison with the results from Gupta, we manually created the activity recognition information for each of the videos according to the activities outlined by [23].

MCOR provides a color-based region segmentation and tracking algorithm that can segment objects in the images using color-based region growing. It then tracks that region according to proximity, shape and color.

We then ran the MCOR algorithm using the automated visual object segmentation and tracking and the manually annotated activity information.

Given an object dictionary with the activity information and the visual information, MCOR was able to achieve a 100% recognition rate, as the manual annotations on activity had no noise.

Both my 100% and the Gupta reported 96.67% object recognition excellent performance, are not surprising as this Gupta dataset has a one-to-one association of an activity to an object. Example of our results can be found in Figures 7.11 and 7.12.

Figure 7.11 shows captured frames taken from two different videos from the Gupta and Davis dataset. On the left, the 150th frame was captured at the point when the activity of *Pouring* was recognized. This allowed the cup to be recognized and the visual features to be stored in the cups dictionary. Thus, when the next video was processed with the updated definition. The cup was able to be recognized on the first frame without any other cue information.

Figure 7.12 shows captured frames taken from results video generated after processing various video clips from the Gupta and Davis Dataset. Each frame shows the point in the

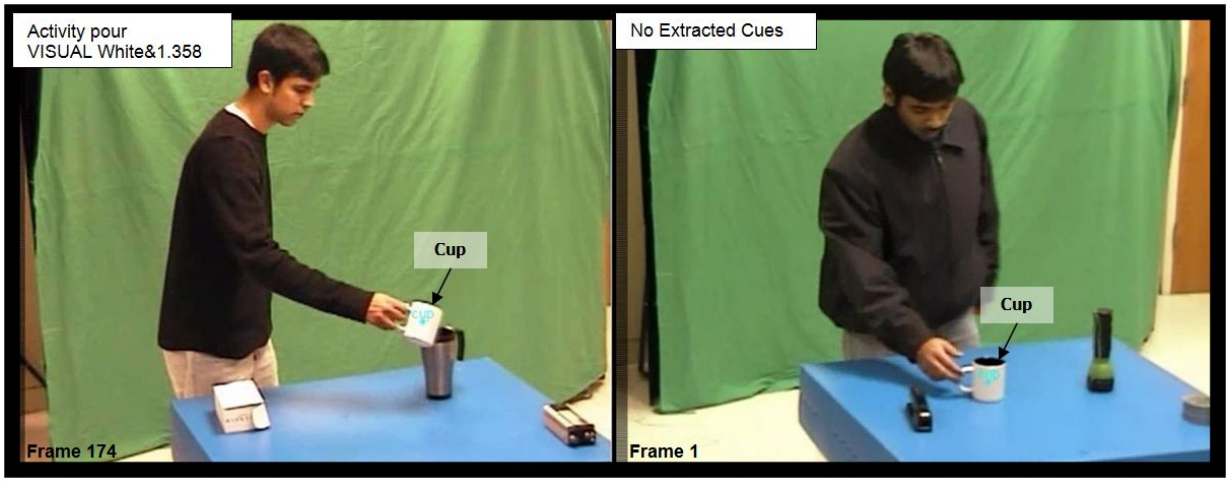


Figure 7.11: Gupta experiment results for recognition based on activity and generalized vision cues

video when an activity cue was recognized, which allowed for the object being used to be recognized and the color and shape information (top right of each frame) to be stored.

Based on the average recognition rate across all the objects in the Gupta dataset (Table 7.1), we found that there was a recognition rate of .47 for recognition based on color region features with a false positive rate of .1. The reason for the low recognition rate is because of the high propensity for the color region feature to recognize false positives, since color and shape information can be easily applied to a number of objects if the threshold is high enough. Thus, we chose to use a high threshold in order to keep the false positive rate under .15. Using the SIFT features, we found there to be a .63 recognition rate with no false positives due to the high matching requirement. Since, however, the objects which are easily recognized by one type are not easily recognized by the other. The combined recognition rate when using both features (taking preference over SIFT features when there is a good match) gives us 81%, where there is only a 29% of overlap between the two features. The false positive rate also gets reduced to .05, since 63% of the recognition rate was produced by the SIFT feature, not the color region feature.

Visual Feature	TP Rate	FP Rate
Color Region	.47	.13
SIFT	.63	0
Both	.81	.05

Table 7.1: Object Recognition Results for Visual Features

Thus, with by using the entire MCOR framework for object recognition, we can increase

Figure 7.12: Gupta experiment recognition results for various video clips from dataset

this recognition rate by 19%.

One of the strengths of the MCOR approach is that it allows a many-to-many weighted associations between objects and cues, which is not tested or demonstrated with the Gupta dataset. Furthermore, another main contribution of MCOR is the ability to learn an association and generalize from it to similar visual situations.

Interestingly, using the Gupta dataset, after MCOR recognizes an object through the association of activity cues, it is able to generalize and recognize objects solely from their updated visual description without the need for the use of the activity information and without offline training beforehand. For example, the white cup in one video (captured frame shown in left image in Figure 7.11) was actually recognized based on the color and shape information learned from the recognition results of another video (captured frame shown in right image in Figure 7.11).

In summary, with this dataset, we have shown that the MCOR algorithm can utilize

datasets outside those generated by our own work. MCOR got comparable results and showed its generalization capabilities.

7.4 Evaluation of MCOR Performance

In this section, we show the performance results for the MCOR algorithm using kitchen setting dataset with a complex background. These datasets were processed and labeled using a ViPER [33] as described in Chapter 3. With these datasets and labels, we are able to vary the cue accuracy, object dictionary, and spatial association accuracy to see how performance changed with the MCOR algorithm.

We use two datasets, all of which have a kitchen setting. We chose a kitchen setting because these videos are most likely to have a person interacting with objects in front of them. One dataset is taken from a kitchen simulated laboratory setting. The other dataset is taken from cooking TV show DVDs. Both these datasets have a complex environment setting (i.e. no color screen for background). Cooking shows were chosen because in addition to the activity cues that are provided by the constant interaction, the actors are constantly talking about the objects that they are holding, identifying them. As well as the sounds provided by cooking. Thus, they provide a rich source of multiple types of cue information that can be utilized. Here is a summary of the datasets used:

LACE Dataset [47] : 50 video clips around 1 to 60 seconds long of different people (one per video) performing similar scenarios of removing a box of cereal, retrieving a spoon, knife, bowl, jug and cutting board. Opening and closing cabinets.

- ACTIVITY CUE VALUES: Eat, Drink, Bring, Return, Open, Close, Pour, Mix*
  All activity cue values were labeled by the Rochester group except for Mix.
- SPEECH CUE VALUES: Word for each object category.
- VISION CUE VALUES: SIFT and Color region growing values
- SOUND CUE VALUES: Pour, Mix
- OBJECT CATEGORIES: Bowl, Knife, Cup, Cereal, Spoon, CuttingBoard, Jug, Fridge, Plate, Cupboard

Rachel Ray Dataset [55] : 50 video clips, around 1 to 30 seconds long of one person, Rachel Ray, performing and describing cooking recipes.

- ACTIVITY CUE VALUES: Twist, Smash, Shake, Drop/Add, Cue, Mix, Pour, PickUp, PutDown, Point, Scoop

- SPEECH CUE VALUES: Word for each object category.
- VISION CUE VALUES: SIFT and Color region values
- SOUND CUE VALUES: Sizzle, Chop, Mix, Pour
- OBJECT CATEGORIES: Bowl, Knife, OilJar, Spice, WineBottle, CuttingBoard, Plate, Grill, Pan, Spoon

A few videos were taken from another cooking TV show with Gia De Laurentiis [34] for a demonstration in some experiments.

Figure 7.13: Example frames from cooking datasets: (Left) Rachel Ray cooking dataset, (Middle) LACE Dataset, (Right) De Laurentiis Dataset

Using these datasets, we now present the recognition results in several experiments. To begin, we first pre-process the video data to group like viewing angles and to add labels. We then compare the performance results under four main varying conditions: a variation in cue accuracy and spatial association accuracy, a variation in the initial object dictionary and current number of videos seen, and finally a variation in the initial object dictionary when recognizing on a different dataset.

7.4.1 Pre-Processing

For the datasets, pre-processing of the videos occurred before hand in order to provide labels for training and evaluation. For the cooking dataset, addition processing had to be done in order to sort videos into different camera angles, so camera shifts would not interrupt the results. We then labeled all the datasets using ViPER.

Scene Sorting

We want training datasets which are primarily generated from real world datasets in order to show the advantages of the MCOR algorithm in the real world. The problem is most real world video datasets, except for those filmed in a restricted laboratory, consist of frequent shifts in camera angles (think of any real world TV show such as cooking shows). In order to compensate for these sudden shifts in camera angle, we outline a scene gist algorithm we developed to sort different camera angles or scenes into bins with similar scenes, so they can be processed using the same tracking or computer vision parameters that would need to be adjusted if the scene were constantly shifting.

In our scene sorting algorithm (Alg. 5), we use local-intensity histograms to separate viewing angles. This is done by dividing images into four 2x2 quadrants. A gray value histogram of each section is then taken. The image is divided into four regions in order to provide some spatial information, if a single histogram was used for the entire image all spatial information would be lost. See Figure 7.14 for an example of the gist descriptor.

Algorithm 5: Scene Sorting

```
Given:
- set of videos V to cut and sort;
- empty set, CV, of average image for each camera view;
- threshold to determine same camera view, θ;
for v ∈ V do
    pf ← 0;
    /* previous frame                                                    */
    diff ← 0;
    /* difference between histograms                                     */
    clip ← ∅;
    /* set of current frames in clip                                     */
    for f ∈ frames(v) do
        /* determine if this is the end of same camera view              */
        diff = euclideanHistDist (f,pf);
        if diff > θ then a new camera view has begun
            cv ← findMatch(CV);
            /* find matching camera view if available, 0 otherwise       */
            if cv == 0 then
                cv = cv ∪ pf;
                saveClip (clip,pf)
            else
                saveClip (clip,cv) clip ← f;
            end
        else
            clip ← f;
        end
    end
end
function diff = euclideanHistDist (I1,I2)
[I1.q1, I1.q2, I1.q3, I1.q4] = quadrants (I1); [I2.q1, I2.q2, I2.q3, I2.q4] = quadrants
(I2);
for i = 1 : 4 do
    h(i) = hist (rgb2gray(f.qi)) diff = diff + euclideanDist (h(i),ph(i))
end
return diff;
```

The histogram is then compared to a database of stored histogram values using the sum of square difference between each histogram bin. If the histogram is similar to one already found in the database it is labeled under the same scene category. If it is different then any other histogram according to a particular histogram, it is then added to the database as a new scene category.

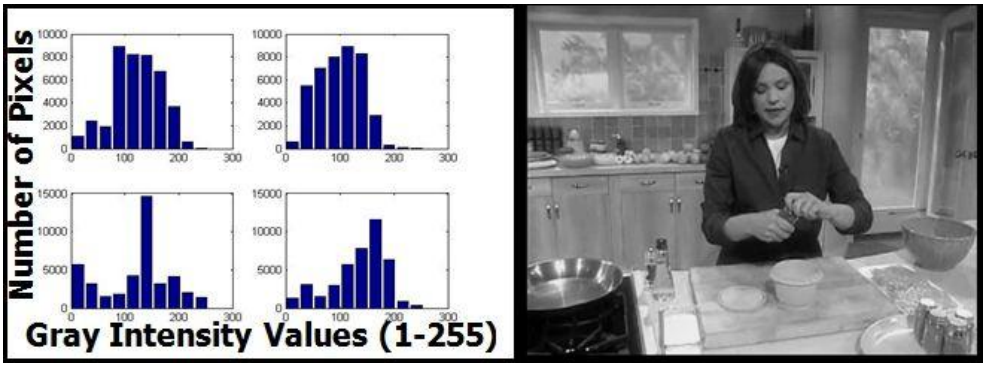


Figure 7.14: Gist information extracted and used in order to separate different scenes in cooking dataset.

Once the video clips are properly sorted into their corresponding scene/camera viewing angle categories, we then need a way to label each of the clips with all the information that would be useful for learning for an interactionable based algorithm.

Labeling using ViPER

In order to provide training and testing datasets for these experiments, videos were labeled with the Video Performance Evaluation Resource (ViPER). The details as well as the schema used for labeling are provided in Section 3.4.

As a reminder, all objects, cues, and people interacting in the scene were labeled along with there values and locations.

Using these processed datasets, we can now determine the performance results under varying conditions.

7.4.2 Error Rate with Varying Cue Accuracy and Spatial Association Accuracy

We start by comparing a baseline error rate of each of the individual cue types if object recognition was done with a single cue types as compared with recognition using all cue types.

Baseline Individual Cue Type Rates and All Cue Types Error Rate

We compare these error rates to that of the entire MCOR system using all cue types . These results are determined with a varying cue accuracy and learned spatial association (Figure 7.15).

In these figures, we see that sound and speech cue types have the best overall individual cue error rate, followed by activity, and then vision cues. Error rate was calculated by determining the number of objects that were missed over the total number of objects present (false negative rate) and the number of objects that were incorrectly labeled over the number of objects (false positive rate). We then add these two values up to get the total error rate and divide by two so error ranges from [0,1]. Speech and Sound most likely have a better error rate given a cue accuracy because these values are the most specific to a particular sort of object given the dataset and cues. In the graph, we show how the error rate changes when each cue had an accuracy rate of 100%, system estimated and 1%. The system estimated values follow the accuracy rate described by each of the cue systems based on an actual system. The system accuracy values are based on are described in Chapter 3. Table 7.2 summarizes the cue accuracy for all the cue systems.

Activity	76.7
Vision	98.3
Speech	87.7
Sound	88.6

Table 7.2: Accuracy Rate for Cue Recognition Systems

Figure 7.15 show the (a) error rate for each cue type and all cue types against cue accuracy with a 100% accurate spatial association and (b) the error rate for each cue type and all cue types against cue accuracy with learned spatial association. White portions indicate false negative rate. Colored portions indicate false positive rate. From Figure 7.15, we can see that the error rate for MCOR as a whole (i.e. error rate when all cues are used) is consistently lower than any of the individual features by themselves.

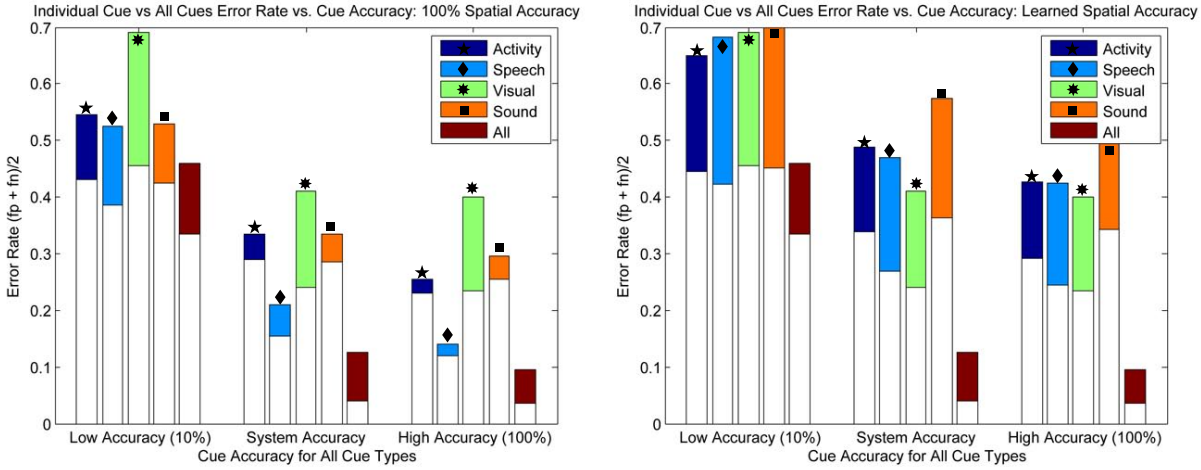


Figure 7.15: Cooking experiment error rate comparing individual cue types and all cue types with 100% spatial accuracy and learned spatial accuracy

This makes sense since MCOR is able to combine the set of object recognizable by each of the cue types to boost its recognition rate. For example, even though activities overall error rate is around 51% for 100% cue accuracy, if we look at the error rate for recognizing just the objects the activity cue interacts with (Figure 7.16), we see a much higher recognition rate. This is because most of the missed objects for the activity error rate were because the cue activity had no interaction with them. For the color cue, on the other hand, all these objects can be recognized but since the color cue is not a particularly good indicator for all objects in the datasets, it misses alot of the objects which the activity cue finds easy to recognize. Since MCOR uses both of these cues, we get the advantage of both. This unfortunately means a slight disadvantage as well as MCOR is affected by the false positive rates as well. This increase, however, is less than the combined false positive rate of the individual features due to the weeding out of misclassified segments through the evidence given by other cues. Thus, the overall performance is better than any individual cue.

Another aspect to note is the effect of spatial association accuracy on the overall error rates. We can see that with the learned spatial association accuracy some of the error rates are scaled up. This scaling up is based on the probability of the spatial association with the maximum accuracy not hitting an object based on the cue value. The scaling value for each of the cue type taken from data is shown as follows:

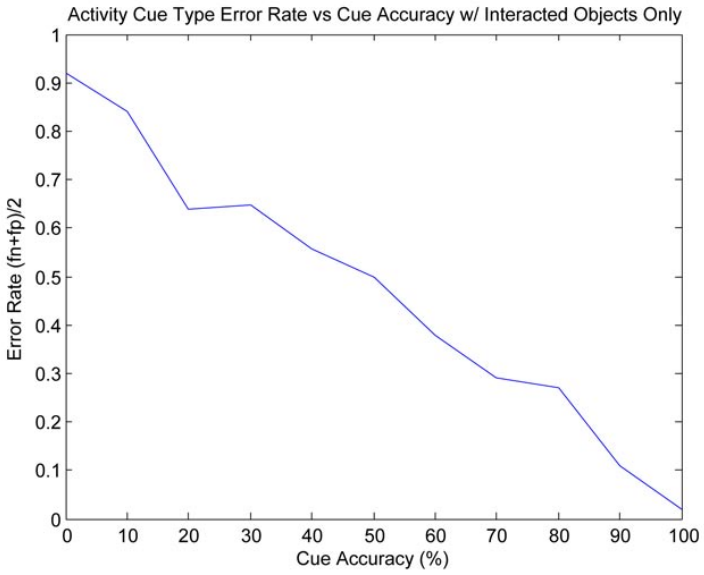


Figure 7.16: Cooking experiment error rate of activity cue for set of objects accuracy cue interacted with

Activity	.23
Vision	0
Speech	.33
Sound	.36

Since spatial association effects the recognition results by scaling the values up or down, improved recognition can occur with better methods of associating a cue with an object location. For cue types like the visual features, no improvement in spatial association is needed since the cues are derived directly for the location of the object. These results, however, do not detract from the main advantage of the MCOR algorithm, i.e. better error rates are still seen by using multiple cue values over any single cue type.

Leave-One Cue Type Out

In this next experiment, we demonstrate how the error rate for object recognition is affected by the addition of each cue. In Figure 7.17, we see how error rate is affected when each of the cue types are removed from recognition for 100% cue accuracy. Lower cue accuracy results are not shown since results are proportional to 100% accuracy.

From the results, we see that each of the cues provide some additional improvement to the performance results. In other words, the more cues that are added the better the overall

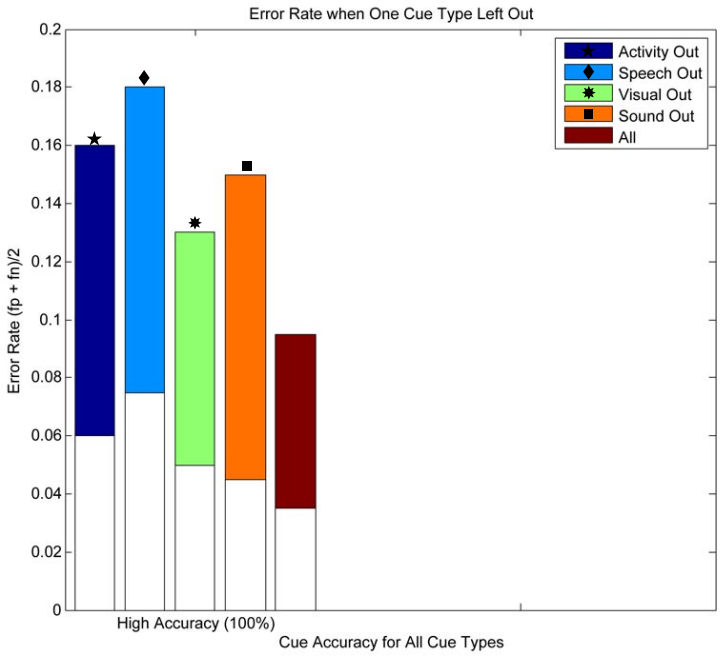


Figure 7.17: Cooking experiment error rate when one cue type is left out of object dictionary and recognition with 100 % cue accuracy

performance. This demonstrates the advantage of MCOR's ability to include all of these types, where most other systems would just include a select subset.

While including a number of beneficial cues is obviously an advantage as shown by the results above, Figure 7.18 shows that the inclusion of a cue with low accuracy will not harm the recognition rate. Lower cue accuracy results for single cue not shown since results are proportional to 100% accuracy.

In this experiment, we show the error rate when all but one of the cue types have a low accuracy of 0%. We see that the error rate is then defined by the best cue. The best cue is varied to each of the cue types.

We now take a look into how error rate varies as the number of videos seen during recognition is increased.

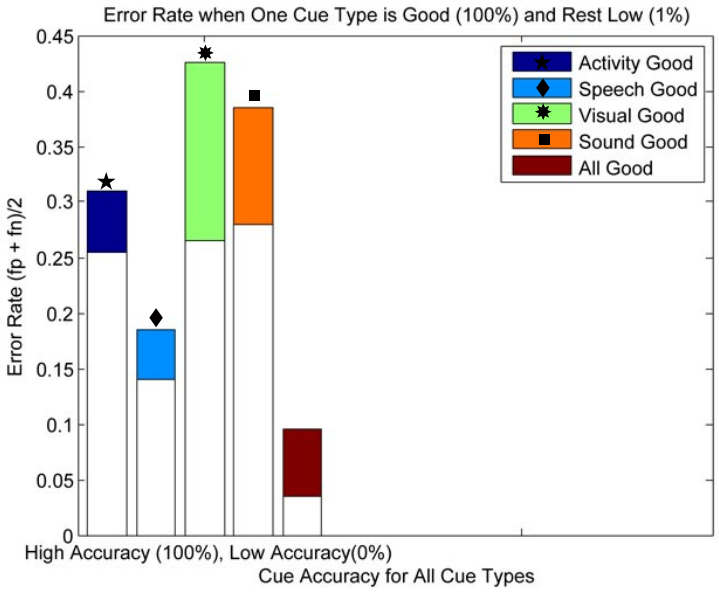


Figure 7.18: Cooking experiment error rate when all cue types have poor accuracy (0%) except for one cue type (100% cue accuracy)

7.4.3 Varying Initial Cues in Object Dictionary with Error Rate over Videos Seen

In this experiment, we show how the MCOR algorithm can improve the error rate of recognition as the number of video seen increases. Figure 7.19 show the results from this graph. Each of the results represent the error rate when recognition is begun where the object dictionary only contains the values from one cue type. As more videos are seen, new cues are learned and generalized by the MCOR algorithm.

Note how the recognition rate depends on the strength of the initial cue in the beginning. This strength corresponds to the average cue type confidence value weighted by the number of each object in the dataset (i.e. the mutual information provided by the cue type to an object. Objects are uniformly distributed in our case). The false positive rate of the initial cue type determines the rate for all the cue types. With the learning of more cues as more videos are seen, the number of missed objects decreases. Thus, the overall recognition rate with the number of video data seen goes down.

As these results show, MCOR is able to utilize the videos it has seen to find more objects in later videos. This demonstrates the ability of MCOR to begin with information that a person may know about an object and improve the recognition results from it. Thus, unlike many other frameworks, MCOR can take whatever knowledge is available and increase the

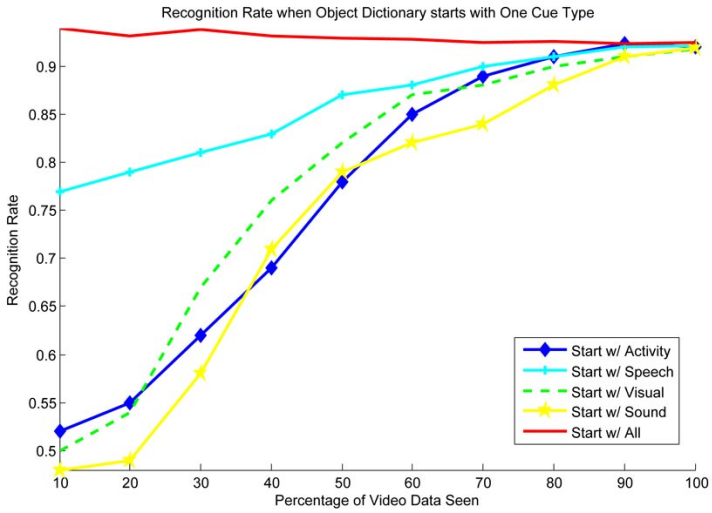


Figure 7.19: Cooking experiment recognition rate comparing initial object dictionary beginning with each cue type and all cue types against the percentage of video data seen at the time

recognition rate.

7.4.4 Using Object Dictionary of One Dataset on Another

In this last experiment, we show how object dictionaries built from previous databases can be used as an initial point for recognition in a new database. We do this by taking the object dictionaries built from the Rachel Ray dataset to recognize similar object categories in the Gia De Laurentiis dataset. Table 7.3 shows the results.

Initial Weights	Recognition Rate
Uniform	.27
Direct	.51
Scaled	.69

Table 7.3: Object Recognition Rate with Varying Initial Weight Values

In the figure, we show the recognition rate for three cases: (1) using an initial uniform weight distribution, (2) using the object dictionary values directly from the previous databases, (3) using the object dictionary values weighted by the cue type confidence.

We see from these results a slight increase in performance when using the object dictionary

values directly as compared to a uniform weight distribution. We see an even higher improvement for the case that includes the cue type confidence values. This makes sense considering that some of the cue values, such as color maybe more specific to the objects found in the Rachel Ray dataset, while the values for activity would be especially useful in the Laurentiis dataset.

These results begin the initial investigation in the use of using cue values learned from previous datasets for application in new contexts. This leads the way into an exploration of the learning of a general representation of weights that can be applied in any scenario while being update for specific contexts when available. The use of the cue type confidence term can be a useful term for this exploration.

7.4.5 Summary

Thus, across all the experiments, we have successfully exhibited the benefit of using as many cue types as are available. Since our MCOR framework provides the ability to this, we have show the great advantage of utilizing such a framework. More specifically, we have shown the improvement in error rate through the use of all cue types with varying amounts of cue accuracy and spatial association accuracy. In addition, we have shown that MCOR can take advantage of the videos which it sees during recognition time to improve its rate. We have also shown the possibility of utilizing object dictionaries from one datasets to help aid in the recognition of objects in another.

7.5 Advantage of Multiple Cues

In these experiments, we demonstrate the flexibility of MCOR due to its use of multiple cues. Videos were taken using a Panasonic Digital Camcorder with 700x500 resolution frames. Data was taken from different office settings located in Carnegie Mellon University. These videos demonstrate four key concepts: (1)the ability of MCOR to utilize multiple cues from multiple people, (2) the ability to recognize objects based on vision cues learned from a previous video, (3) the ability to recognize an object after it has returned within view after being off screen, and (4) the ability to recognize the different appearances of an object within the same class based on multiple cues of the same type as well as cues of different types.

7.5.1 Utilizing Cues from Multiple People

In this demonstration, we demonstrate the ability to recognize categories of objects despite the variance of appearance of particular instances of those objects through the interaction of multiple people. In this scenario, the object dictionary contains a pen object definition which contains one cue value, *Writing*, from the activity cue type. The video for evaluation contains two people, pa1 and pa2. Pa1 first interacts with a highlighter pen by writing. This allows the MCOR algorithm to grab the color and shape information from the pink highlighter. Using this information, MCOR then recognized the other pink highlighter on the table. Pa2 then writes with a highlighter of another color. The MCOR algorithm then generalizes this color to the pen object definition. The other two yellow highlighters were then recognized using this updated definition.

In Figure 7.20, a frame capture of the resulting video is given which demonstrates MCOR's use of the activity cues from two different people. Thus, the figure shows the ability of MCOR to utilize multiple cues (two activity cues) from multiple people to recognize pen objects of different appearances (pink and yellow).

Figure 7.20: Meeting experiment with multiple people recognition result using multiple activity cues for objects of different appearances

In the second demonstration (see Figure 7.21), a similar scenario is given where an object, cup, has multiple visual appearances (red and blue). The object dictionary contains the object category, cup, with the activity cue, *Drinking*, and the speech cue, *"cup"*. In this demonstration, there are two people, pb1 and pb2. Pb1 first generates a speech cue by using the word *"cup"* while talking about the cup. Pb2 simultaneously drinks from the cup generating an activity cue. Based on these two cues, MCOR segments around those regions and recognizing each of the different cups as a cup. The *Red* and *Blue* color and

shape visual descriptors are then added to the object definition of cup. This allows all the other cups in the scene to be recognized. Thus, MCOR is able to recognize objects utilizing not only the activity information of one person to recognize all the blue cups, but its able to recognize all the red cups because of the speech cue given by another person.

Figure 7.21: Meeting experiment with multiple people recognition result using different cue types to recognize different object appearances

With these demonstrations, we show how the MCOR algorithm was able to recognize not only the objects being interacted with, but additional objects in the scene based on visual descriptions learned from the initial interaction. This interaction was able to utilize both multiple people as well as multiple cue types.

7.5.2 Recognition Using Learned Vision Cues

In this demonstration, we show MCOR's ability to utilize the updated dictionary definition from the first demonstration to recognize objects in a new scene without the need of additional interactions.

Figure 7.22 show the ability to recognize objects based on vision cues learned from a previous video (Figure 7.20).

In the video for this scenario, we have a completely different context from the first demonstration with a different room, table, and people. MCOR uses the dictionary with the updated pen visual features learned from the earlier video. Using this dictionary, the al-

gorithm has the ability to recognize all the pen objects without the need of additional cues.

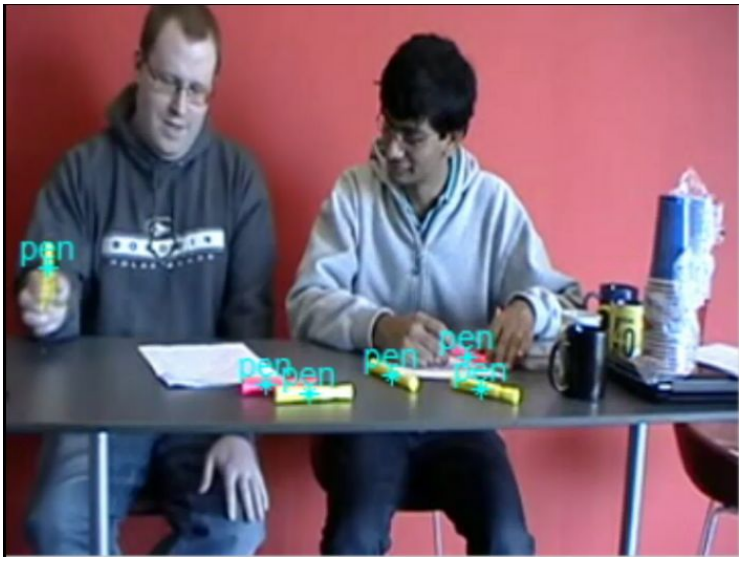


Figure 7.22: Meeting experiment with multiple people recognition result using vision cues learned from previous video

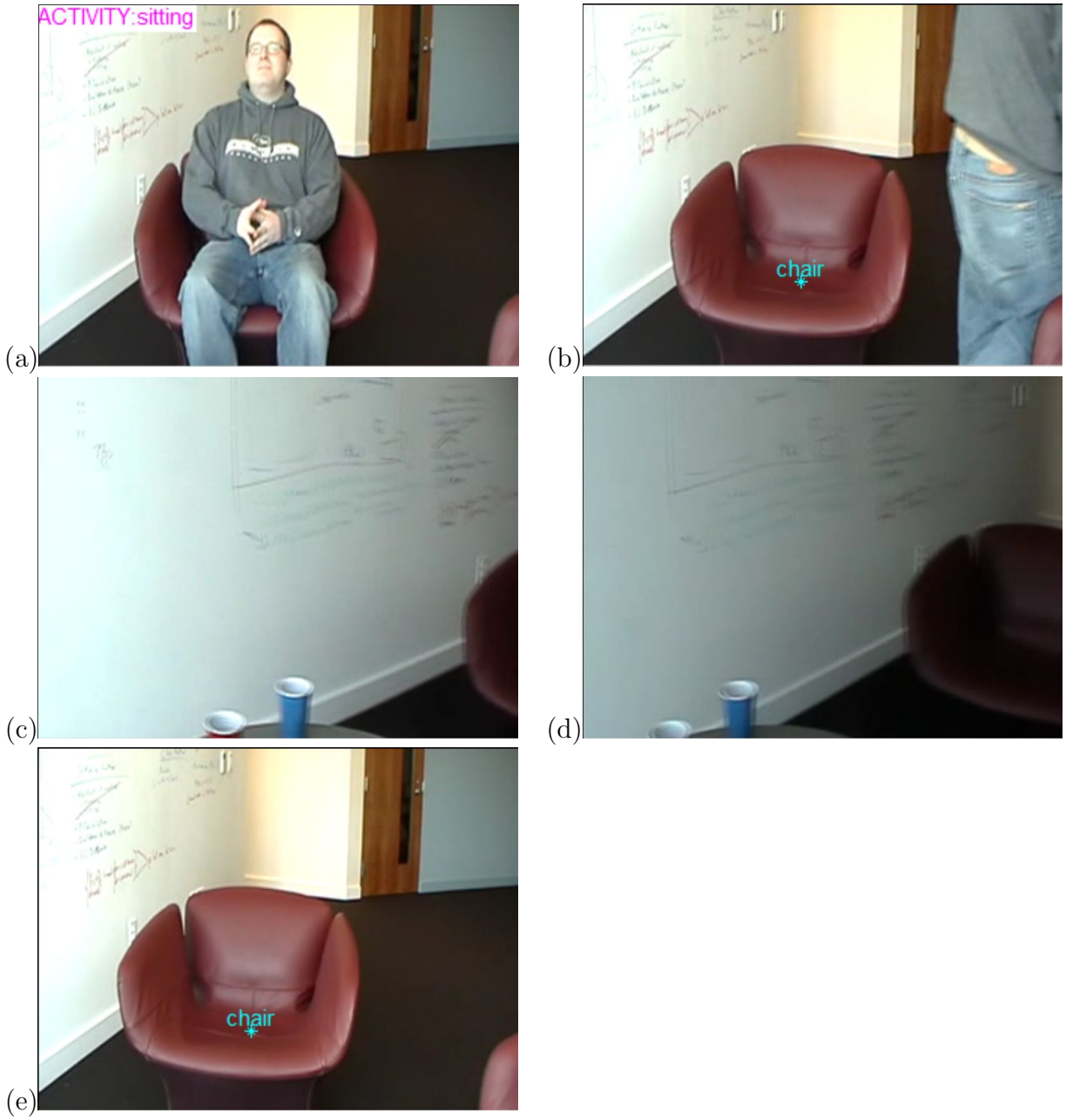
Thus, we are able to show MCOR's ability to recognize objects based on vision cues learned from a previous video.

7.5.3 Handling Off Screen

With this demonstration, we show the use of MCOR to re-recognize objects that may go off screen with a panning camera.

Figure 7.23 show the ability to recognize an object after it has returned in view after being off screen.

In the video used for this evaluation, we have one chair object in the scene and one person. The object dictionary contains the a chair definition with the sitting activity cue value associated with it. The person then sits on the chair, allowing the chair to be recognized. The *Red* visual features and shape of the chair are then added to the definition. The camera then pans to the left such that the chair object moves off screen. When the camera pans back, so that the chair is back into full view, MCOR is able to re-recognize the object. The image captures from this video are shown in Figure 7.23.



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

Figure 7.23: Meeting experiment with multiple people demonstrating robustness to momentarily off screen objects

Using this example, we were able to show the advantage of using MCOR in re-recognizing objects which may have panned off screen for a period of time.

7.5.4 Handling Multiple Object Appearances

In the final demonstration of the advantages of the MCOR algorithm (Figure 7.24), we show the ability of the framework to recognize the very different appearances of an object within the same class based on multiple cues of the same type as well as cues of different types.

In the video, we have four people all with very different looking mugs. The object dictionary contains the mug object category which has the drinking activity cue value and the *"mug"* speech cue value associated with it. Because we have multiple people, all interacting with each of the mugs, MCOR is able to very quickly recognize all the mugs despite their vary different appearances. This was possible because of MCOR's ability to use the interaction of multiple people as cues for the identity of each of mugs.

7.6 Conclusion

In conclusion, we have demonstrated the ability of the MCOR framework to recognize multiple object categories using cues of various types. We demonstrated this by providing empirical results which showed the basic features of the MCOR algorithm. We showed how MCOR could be used to recognize objects in outside datasets successfully with the added ability of being able to generalize other cue information. We showed how the performance rate varies with a change in cue and spatial association accuracy. Finally, we gave video results demonstrating the advantages of the MCOR algorithm to handle object categories with varying appearances, objects that move off screen, and objects in different contexts.

Figure 7.24: Meeting experiment with multiple people using multiple cues of the same type and of different types

Chapter 8

Background and Related Work

In this chapter, we discuss related work to this thesis. We start by placing the concept of multiple-cue object recognition in the context of related object recognition work (Section 8.1). We focus on the areas of vision-based object recognition, multiple feature, functional recognition and multi-modal object recognition.

We then provide background and related work for the tools and methods used in our MCOR framework (Section 8.2). This includes the probabilistic relational model used for object and cue representation, the Hough transform used for the collection of evidence during recognition, discrimination used to define the recognition process and tracking and segmentation used to ground the information in video data.

8.1 Object Recognition and Relationship to MCOR

Object recognition is concerned with identifying and locating an object within an image or video frame.

The openness of the term 'identify' in this definition has led to a number of object recognition types. The MCOR framework focuses on one type of object recognition, *category-level* object recognition. Category-level recognition determines whether or not an image contains a specific category of object. This differs from instance-based recognition which attempts to focus only on recognizing particular object instances. There are many ways to represent a category of objects, such as shape analysis, bag of words models, or local descriptors.

Object recognition encompasses a vast field and set of approaches, the entirety of which

would be impossible to cover here. Since our object thesis focuses on the integration of multiple cues of various types, we organize this section with an increase in the type of cue information utilized for recognition: we start with vision-only recognition systems, followed by approaches based on visual and functional cue information, and finally, working our way up to multi-modal object recognition approaches.

Thus, we begin by focusing on conventional vision-based recognition approaches. We focus on key concepts, which have provided the foundation for substantial progress in the area of object recognition.

8.1.1 Vision-based Object Recognition

In this subsection, we summarize object recognition techniques which use visual information alone. There are a variety of approaches to vision-based object recognition. *Model-based* approaches represent objects as a collection of three-dimensional, geometrical primitives (e.g., spheres, cylinders, boxes, cones). Appearance-based models use extracted features from the image and shape of an object as templates for matching [12, 22, 39]. In our thesis, visual features are based on the latter model.

In shape and feature-based models, a model of an object is typically built from interest points found on the object. Interest points are often located by corner detectors [24]. From the interest points ***features*** consisting of visual descriptions are extracted. Features are often color and texture information generated by a number of methods including geometric blur [6], pyramidal histograms of oriented gradients [8] and histograms [39]. For category-level recognition, a model for each object appearance in an object category is attempted to be learned. These features are typically learned from training data. [17].

Most visual features, however, are very susceptible to one, if not all of the following changes: scale, noise, illumination, geometric distortion, clutter, and occlusion. A number of approaches attempt to avoid these problems by confining their techniques to training and test data where only a single object is found in the image on a solid background for easy segmentation [42, 71].

Robust Feature Building

There have been a number of works that have attempted to develop more robust visual features. One such well-known method is SIFT or Scale-Invariant Feature Transform [39]. This technique attempts to find highly distinctive features by matching individual features to a database of features from known objects using a fast nearest-neighbor algorithm, fol-

lowed by a Hough transform to identify clusters belonging to a single object, and finally performing verification through least-squares solution for consistent pose parameters. The features are invariant to image scale and rotation, and are shown to provide robust matching across a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. However, it can only partially handle changes in lighting and pose. In MCOR, we use SIFT as only one of the set of cues used to identify an object category.

There are several alternative methods such as RIFT, a rotation-invariant generalization of SIFT using circular normalized patches divided into concentric rings of equal width, where the orientation is measured at each point relative to the direction pointing outward from the center in order to maintain the rotation-invariance [35] and SURF or Speeded Up Robust Features, which relies on integral images for image convolutions to reduce the computation time of the other methods [5].

Others visual systems have tried reducing the amount of storage needed by the robust feature descriptions in order to gain an edge and compact as much information as possible. Such methods often do this compression through shared features or object parts [2, 45, 54].

These techniques however still limit the type of information that can be used to visual information and thus is still susceptible to the inherent weaknesses of that cue type.

Integration of Multiple Visual Cues

While the previous techniques attempted to build feature descriptors that were robust, a number of techniques have attempted to address the problem by building methods that could better integrate information from various visual cue types.

One method which attempts to do so using a generalized feature is G-RIF, or Generalized Robust Invariant Feature, which uses a general context descriptor which encodes edge orientation, edge density and hue information in a unified form combining perceptual information with spatial encoding. The object recognition scheme uses neighboring context-based voting to estimate object models [30].

Numerous works have described techniques for integrating multiple visual cues such as texture, shape, color, and brightness. [13, 46, 56].

Multiple techniques do so by accumulating the various types of information, i.e., they accumulate cues by summing probabilities or by joint regularization. [3, 53, 69]

These methods however tend to work with very simplistic data.

8.1.2 Functional Recognition

In order to compensate for the weakness of visual features, another form of recognition, ***functional*** recognition, attempts to use identify objects based on their function. A functional definition categorizes and defines objects according to their potential uses and their utility in performing a task [64]. In figure 8.1, the differences between a functional and a classic categorization of objects is shown. Note that a functional category may put together two objects that would normally be categorized separately as in the mug and the water-can and classic object categories may include several function categories. Thus, if one wanted to give the objects classic object recognition labels, one could not do so on the basis of a functional definition alone.

Figure 8.1: Classic object categories (left) vs. functional object categories (right) [64]

The functional definition of object categories depends on the notion of affordances introduced by [20, 21]. In Gibsons theory of affordances, certain aspects of an objects form can be used to determine their function. Affordances were 'action possibilities' defined in relation to the actor and thus, the actors capabilities. So, if we are talking about a human, a door knob affords the ability to be turned because of its form, i.e. its a turnable object for that human. (For a bird on the other hand, the door knob would not provide the same affordance since its form does not lend to the birds ability to turn it. However, we will stick with the perspective of an adult human as that makes most sense in terms of most object recognition problems). Thus, affordance properties are how an object can be interfaced with, defining the objects utility. For object recognition, this is a useful concept as it allows the approach to use affordance properties or cues to determine the function of an object. For example, in some works affordance properties are defined as the relative spatial locations of an object's components [7].

Functional recognition then uses these affordance properties to determine the function and finally the identity of the object. In most functional recognition approaches, the interaction of a human with an object is used in order to identify in the object either in place of or in conjunction with its visual attributes. In many cases, visual recognition techniques are used first to recognize parts of an object in order to determine its use (and thus functional object category) [59, 64, 76], with one of the first being [75], who used visual properties in order to determine geometric properties to inform the function of the objects, such as the bent (arch) nature of a handle on a cup or watering-can, which allows for grasping. In

some cases, the affordance properties are used first in order to identify objects from which visual properties are taken to find additional objects [70]. But under almost all cases, an object is labeled according to its functional definition, not the classical.

8.1.3 Multi-Modal Object Recognition

Other approaches to object recognition use alternate sources of non-visual information outside of or in addition to the functional definition. These approaches, along with our thesis approach, identify the fact that humans usually do not recognize objects in isolation. Objects are often being used, talked about, and touched – all of which provide additional cues to the label of the object. In fact, in laboratory studies where much of the context is removed and a person is only shown an image of the object to be recognized, if that object is in a non-canonical position, it becomes more difficult for the person to name it, as shown by an increase in response time [50]. Although there are numerous theories as to why that maybe [14, 52], most would agree it seems highly unlikely that a person would no longer recognize the teapot he/she was pouring tea from simply because they are just now viewing it from a non-canonical viewpoint. Other non-visual cues associated with the object allow the person to be confident that the object he/she is holding is a teapot, even though the visual cues of the object itself may be lacking when looked at in isolation.

Through the utilization of different modalities, such as speech, activity, and sound, object recognition systems can progress towards the same level of robust recognition human's have.

Other attempts have tried to compensate for the weaknesses by including a cue that does not depend on the visual properties of the object itself. One method uses scene context information [44] to provide additional evidence for the presence of an object. For instance, if one were in an office, one could expect to see a computer screen or keyboard. The scene context, the office, can thus provide evidence for those two objects in addition to their particular visual features. Graphical models are used to describe the relationship between the 'gist' of the scene, a feature vector extracted from the entire image, and the objects. This method, however, lends a way for incorrect classification or a missed recognition caused by an object being placed in a scene it normally is not found.

Others such as [43] attempt to add a completely non-visual cue, i.e., activities, in addition to the visual properties. However, it also depends on a training set of the visual appearance of the object, albeit less strongly than visual-based approaches alone. It also depends on learning the background before hand and removing it when doing the recognition. This approach limits its ability to be placed in any given room to recognize objects, and thus makes it harder to be applied to real world situations.

Although there are numerous other attempts which have included other non-visual cue information: activity [70, 77], speech [61, 63], touch [26], knowledge [9, 41]. Every type of cue in isolation has its own set of problems and weaknesses. For instance, with speech cues, although there are a number of instances where its linguistic content can be used to disambiguate objects in a picture [63], it is far from a perfected science and chances of false positives or failures in a real-world situation is not unexpected. If one were to use activity cues alone, there would also be numerous possibilities of error such as how to deal with an activity that could describe more than one object or false positives caused by an activity that misused an object. Other types of cues have similar weaknesses. Thus, in our framework, we attempt to include information from any and all types of cues, so that the weakness of one can be compensated by another. While category-level objection recognition is a well-studied problem, to the best of our knowledge, no other work has combined it with a flexible framework, which can include information taken from a large variety of modalities.

Multi-modal recognition systems can fall under two broad categories: early fusion and late fusion [61]. In early fusion, low-level features, which would be used in the recognizer of an individual cue type are used directly. For instance, motion features used by an activity recognizer are used to directly in the object definition for recognition, rather than the activity labels generated by the activity recognizer. In late fusion, the classification results of the recognition system (e.g. activity labels) for each modality/cue type are used in the object definition. In our thesis work, we utilize late fusion. An advantage of using late fusion is the greater compartmentalization of the different modalities. The compartmentalization allows each modalities/cue type to be weighted. This reduces the computational cost of weighting each individual low-level feature, while still retaining the advantage of having the different modalities weighted separately from one another. The trade off between computational efficiency of the late fusion and potential accuracy gain of using early fusion is left for future work (Chapter 9). For our current work, we preferred to err on the side of computational efficiency since we would like to be as close to real-time during recognition of the video.

In the next section, we describe the work related to the methods and tools used to learn the weight values for each cue, to represent each object model, and to recognize the object categories in a given video.

8.2 Methods and Tools

In this section, we provide the background and related work for the methods and tools used in our MCOR framework (Section 8.2). This includes the probabilistic relational model we use for object and cue representation, the Hough transform used for the collection of evidence during recognition, the discrimination approach used to define the recognition process and tracking and segmentation used to ground the information in video data.

8.2.1 Probabilistic Relational Models

An important aspect of the integration of the information is determine how much weight to give to each of the different type of cues, since it would be unrealistic to assume that each type of cue provides an equal indication of the presence of a particular object. There are multiple probabilistic techniques [40]which could be used to determine the weight of the association between a cue and an object such as voting algorithms [51]. One of the most popular techniques is the use of Bayesian networks [11, 29].

In all probabilistic work, the structure and representation of the probabilistic relationship need to be defined. The standard approach is to utilize a Bayesian probabilistic model. Bayesian networks allows us to provide a compact representation of a complex probability distribution over some fixed set of attributes or random variables. The representation exploits the locality of influence that is present in many domains.

Numerous methods have used Bayesian models [29] including many of the methods described above for integrating multi-modal cue information,e.g., [44]. However, most of these techniques operate under very strict assumptions [32]. One assumption in particular, that the data is not relational, does not apply well to real world data and the relationship between objects and cues in a real world environment.

Probabilistic relational models (PRMs) [18, 19] can handle the inclusion of relation information of data as an extension of standard Bayesian networks. PRMs can learn associations between classes, attributes within a classes, and attributes related to another class rather than the flat, attribute-value [18] data that Bayes nets must use. In addition, PRMs provide a framework for representing database-like structures, such as the object dictionary of the MCOR framework. PRMs provide the necessary tools and concepts for learning the multiple relationships between the various object, cue, human, and segment properties, allowing not only for the probabilistic learning of the weight values, but for the learning of the object displacement values (spatial and temporal association) needed to ground the evidence information to a location in the image and for the learning of cue type confidence

values, which can be used to scale the information coming from the various modalities.

As a reminder, within the relation model, a schema is defined containing several components: The set of classes,

$$\mathbf{X} = X_1, ..., X_n$$

each of which has a set of attributes,

$$A(X_i) = X_i.a_1, ..., X_i.a_2$$

. The attributes can be either *fixed*, *deterministic*, or *probabilistic*. *Fixed* attributes are there to identify instances of the class (referred to as *entities*) and thus their value does not change. The value of *probabilistic* attributes however can vary based on the other attributes of the entity or of related entities. It is this affect that we attempt to model and learn the parameters.

The second component is the set of relations,

$$\mathbf{R} = R_1, ..., R_m$$

which defines the relationship between two classes. Relationships are significant in that the value of attributes in one class can depend not only on the other attributes of that class, but on the attributes of any related class.

While PRMs learn the dependency structure, S, between the attributes of the classes using heuristic structure search and an adaptation of Bayesian model selection [18], the MCOR framework had a known dependency structure, which did not need to be learned.

PRMs then describe a probability model over instances of a relational schema. An instance, I, of the relational schema consists of the set of entities of each class,

$$O^{\sigma}(X_i) = e_1, ..., e_p$$

, where the attributes are defined and which relationships exist between them. A skeleton, σ, is when only the fixed attributes of the entities are defined. In our case, an instance would consist of all the objects in a scene, all the cues in the scene, and the association between any of the cues or objects. Some of the attributes however are not easily defined such as the object label and thus it is necessary to determine the probability distribution of its values. A relational skeleton σ is then a partial instance where the probabilistic attributes are undefined.

PRM then defines the distribution of instantiations of attributes as:

$$P(I|\sigma, S, \delta_S) = \prod_{X_i \in \mathbf{X}} \prod_{a_j \in A(X_i)} \prod_{e_k \in O^{\sigma}(X_i)} P(I_{e_k.a_j}|I_{pa(e_k.a_j)}) \tag{8.2.1}$$

Given a training set, the parameters δ_S can be learned according to the following equation:

$$l(\delta_S|I,\sigma,S) = logP(I|\sigma,S,\delta_S) \tag{8.2.2}$$

$$l(\delta_S|I,\sigma,S) = \sum_{X_i} \sum_{A \in A(X_i)} \left[\sum_{x \in O^\sigma(X_i)} logP(I_{x.a}|I_{pa(x.a)}) \right] \tag{8.2.3}$$

Standard maximum likelihood estimation can then be applied where δ is chosen in order to maximize l.

Although PRMs have been used in a number of other domains such as the web [66], movies [18], and genes [18], the application of PRMs for object recognition using multiple cues is a novel one.

With the probabilistic representation of the weight and object displacement values, a framework is needed to determine the evidence that an object category is present given the weight features and location information.

8.2.2 Hough Transform

There are a vast number of recognition schemas. In our framework, we need a recognition approach that can efficiently cluster evidence for object category hypothesis as well as possible segmentation locations. The *generalized Hough transform* [4, 25] is a well established framework that does just that. The Hough transform has been used in a number of applications varying from line detection to activity recognition [4, 25, 78]. A popular use of the Hough transform is in object recognition [37, 39, 45].

A major advantage of the Hough transform is its built in capability to localize evidence information in an image, while determining evidence for a particular object category. The Hough transform works by creating an accumulator array with dimensions corresponding to the different parameter information desired for recognition of an object. In a typical object recognition Hough transform, parameters can consist of pose, scale, rotation, position, and object category. Each feature/cue then votes for a set of parameters. The bin in the accumulator array with the greatest value determines the most likely object category and parameter values. Weighted votes are often used in the Hough transform recognition schemas to adjust the value placed on different features/cues. The method of weight calculation vary depending on the approach. In a shared-feature object recognition

approach[45], the log-likelihood ratio of the probability of a feature given the presence of an object and given the absence of the object is used to weight each vote. In another approach [36, 37], the votes are weighted by $P(o_i|I_j, l)$, the probability of an object category, o_i, given a codebook interpretation, I_j observed at location l. A codebook interpretation is a shape cue that is matched to a learned set of codebook entries that can be used to describe each object. A similar approach is used in our thesis, where votes are weighted by $P(o_i|\text{cv}_j)$, i.e. the weight value calculated using the probabilistic relational model.

As an example of the use of the Hough transform, in the shared-feature object recognition system introduced by [45], a 3-dimensional Hough transform is used consisting of the object label, x location, and y location of the object center in an image. The Hough transform is represented by an accumulator array, where the 3-d space is divided into bins of a given size (Size produces a trade off between evidence strength and location accuracy). The position dimensions in the accumulator array cover the entire image. Each feature extracted from an image then adds a weighted vote to a bin specified by the learned offset between a feature and an object center from training data. An object is then given the label to the bin that has the highest activation value, which is above a given threshold.

In MCOR, we use a modified version of this Hough transform space, where instead of dividing the entire image into corresponding bins in the accumulator array, bins represent each region currently segmented by the MCOR algorithm. Position parameter information for each cue is given by the spatial association. The threshold is determined by a maximum a posteriori estimate given a training dataset (see Chapter 6).

Using the Hough voting strategy to determine the evidence towards each object category, we use a recognition algorithm that incrementally discriminates between potential object categories based on the cues received across time from a given video. We know define the work related to this incremental discrimination process.

8.2.3 Discrimination

In the thesis work, the ability to discriminate between each of the objects is an important ability necessary for a fast and accurate recognition process. This section describes the basis for such discriminative techniques.

The foundation of most of these techniques is found in the EPAM model [15, 16, 57]. EPAM (Elementary Perceiver and Memorizer) is a psychological theory of learning and memory implemented as a computer program. Originally designed by Herbert Simon and Edward Feigenbaum to simulate phenomena in verbal learning, it has been later adapted to account for data on the psychology of expertise and concept formation. In EPAM, learning consists

in the growth of a discrimination net, where words are recalled through the comparison of an encoding of the word to each node of a discrimination net (tree) that contain feature tests until a match is found; if by the end of the tree, no match is found, the features of the word are added to the net. The primary concept to note is the incremental comparison of the word stimulus to features until recognition occurs. The addition of features to the net corresponds to the generalization technique of the MCOR algorithm.

Another approach [46] combines decision trees, support vector machines (SVM), and the probabilistic cue evidence accumulation schema defined by [3, 53, 69]. The method uses the discriminative classifier defined by the SVM for each of the cues to generate object classifications. Each of the cue classifier then vote for the presence of an object. A decision tree is used to reduce the dimensionality problem when there is a large set of object categories. The method creates a decision tree, which splits the object classes into two subsets at each step. The splitting is determined by the standard information theory method. The cue accumulation recognition algorithm is then preformed at each node of the tree. Thus, at each step only two classification labels are considered.

While these works attempt to discriminate on the basis of information theory, my thesis work will be focused more on taking advantage of the sequential nature of data in order to choose the order of cue introduction as the discriminating factor. This method eliminates the need for extra calculation in terms of information gain and possible tree configurations.

With a method for determining the recognition process, we now go over work related to the grounding of the recognition information to pixel locations in the image, i.e. the segmentation and tracking methods used to extract and localize regions for recognition.

8.2.4 Segmentation and Tracking

Since the exact methods for segmentation and tracking are described in detail in Chapter 4, we focus in this subsection on a general summary of the methods used and a brief overview of related work in each of the areas. As with the cue recognition systems, the focus of MCOR is not to use the most advanced methods in tracking and segmentation, but rather to demonstrate the ability of producing robust recognition results through the combination of these systems despite their weaknesses. We outline here some of the more advanced approaches in these areas, which could be implemented in future work.

In our thesis work, segmentation was done using a *color region growing* approach [72]. This method provided the color and shape information used for our visual cues, as well as providing the foundation for the region tracking algorithms used. Segmentation, however, does not have to be based on color. Other approaches use characteristics such as intensity and

texture. These approaches can be useful when dealing with objects with non-homogeneous color regions. We deal these objects by creating segments based on the affine transformation parameters identified by the SIFT features. In future work, it would be interesting to see if other texture based segmentation algorithm would prove more accurate. We used the SIFT approach, since SIFT features were already extracted for recognition evidence, and so segmentation did not add much computational cost. Other advanced segmentation techniques include graph partitioning methods, such *normalized graph cuts* [62]. This method has proven very effective for segmenting objects from their background. The algorithm, however, primarily works on figure-ground segmentation. Since there are often multiple objects in the scene, the current method would not be as useful in the MCOR framework. There are numerous other segmentation methods, such as *model-based segmentation*, segmentation based on motion cues [65], *histogram-based segmentation*, and *clustering* methods, which we cannot cover all here [48].

Tracking was done using a two-fold method: (1) *Region-* or *blob-based tracking* based on the tracking of color segmentation regions (contiguous region tracking code by Felix von Hundelshausen [72] was used along with Matlab code developed by ourselves - Chapter 4), and (2) *feature-based tracking* using SIFT features as the template for matching across scenes. Other tracking methods include *model-based tracking*, where object models are compared with each frame to determine the location of an object and *active contour-based tracking*, which represents objects as bounding contours and updating such contours dynamically in successive frames.

In both the color region and SIFT feature tracking methods, we used a standard approach of proximity matching [17] to determine the location of the tracked object in the next frame. More advanced technique in tracking predict the movement of the tracked region through model estimation techniques such as the Kalman [28] filter, which uses an optimal recursive Bayesian filter for linear functions subjected to Gaussian noise and the more sophisticated, particle filter [31], which samples the underlying stat-space distribution of non-linear, non-Gaussian processes. While these methods can produce fast and accurate results, they also tend to be more computationally complex [49].

8.3 Summary

Although there has been significant contributions to the area of category-level object recognition. The problem still remains exceedingly challenging.

In our work, we realize the world does not provide clean-cut definitions of what an object is. The world is made up of "messy concepts" [58]. Concepts that are gray at the borders,

that are changing, and contain exceptions, so attempting to recognize an object based on any one single aspect or cue can be a problem.

We understand the difficulty of using multi-modal information, where each cue recognition system, segmentation, and tracking are all complex problems in themselves. With this thesis work, we combine these different components with PRMs, Hough transforms, and incremental discrimination to produce a framework that can yield the best recognition performance results possible despite the inaccuracies of the subcomponents.

Chapter 9

Conclusion

Flexibility and robustness are the major goals of the multiple-cue object recognition approach and thesis work. Although there are a large number of works that provide accurate recognition results in very specific environments, none provide a cohesive way to combine all the various types of information available given any environment.

In my thesis, I attempt to take advantage of the vast amount of cues human interaction and video data provide in revealing the identity of an object. Although any particular cue may fail, the evidence provided by numerous other types can compensate. Thus, we have shown in this thesis an object recognition algorithm which can use multiple cues of different types and number.

9.1 Contributions

The main contributions of this thesis can then be summarized as:

- We provided an object definition and cue representation for the utilization of the resulting object dictionary for robust multi-cue object recognition. By defining an object as a flexible set of cues, we allow for the the use of multiple cues of varying types to provide as much evidence for determining object classes as possible. Requiring cues to define a cue type, values, object displacement values, and weights, we have the information needed to use cue information to localize and identify objects in a video.
- We provided a probabilistic framework for representing the important fact that some

cues may be more indicative of an object category than others. We did so through the use of probabilistic relational models (PRMs) which not only provided a way to learn these weights, but allowed for the learning of spatial and temporal associations and a confidence value based on the cue type. We demonstrated the learning of these weights both on simulated data and data taken from real video datasets.

- We outlined vision tools used to segment the image, extract visual features, and track segments across image sequences. We have shown the usefulness of using both color segmentation and color-shape visual information for identifying objects with large homogeneous colored regions and using SIFT features for textually complex features. These tools can then be utilized in the MCOR recognition algorithm for localizing information in the image and finding new object candidates based on learn visual descriptions.

- We presented MCOR algorithm which integrated the object and cue representations, weights, and visual features to produce a framework that can integrate information from multiple sources to incrementally discriminate potential object categories. In addition, the MCOR algorithm has the ability to generalize learned cue values to new object categories for additional evidence in addition to the ability of using and learning the association of cue types not found in the original training dataset.

- We demonstrated using a detailed analytical model, supported by simulation results, the key concept that multiple cue information improves performance regardless of a variation in cue accuracy. In the worse case scenario, where additional cue types have worse accuracy than the original single cue type, performance does as well as the single cue. Thus, performance results are never as bad as its best cue. We have also shown that scenarios where a single cue type may be ambiguous or misleading for particular objects, multiple cues can help in disambiguating and correcting the error leading to better performance.

- We demonstrated the ability of the MCOR framework to recognize multiple object categories using cues of various types. We demonstrated this by providing empirical results which showed the basic features of the MCOR algorithm. We showed how MCOR could be used to recognize objects in outside datasets successfully, with the added ability of being able to generalize other cue information. We showed how the performance rate varies with a change in cue and spatial association accuracy. Finally, we gave video results demonstrating the advantages of the MCOR algorithm to handle object categories with varying appearances, objects that move off screen, and objects in different contexts.

Representation Representations that can provide beneficial, complex information, while allowing flexibility, which includes an object definition and cue representation which allows

for the addition of multiple cue of any type, the learning of weights and relationships between the various cues and objects using Probabilistic Relational Models, and increasingly complex hierarchies.

Recognition Recognition algorithms which can utilize the information provided by the various representations in order to accurately identify objects, which includes the Multiple-Cue object recognition (MCOR) algorithm which takes advantage of the object definition and weights to determine the identity of an object, the integration of the more complex representation of cues and objects in that recognition, the incremental discrimination for faster recognition results, and the use of context-based information for better recognition given an environment.

9.2 Future Directions

There are a number of possible future directions for this work:

- **Improved Segmentation, Tracking, and Spatial Association Accuracy**

  There are a number of engineering issues which would could be used to improve overall recognition rate, one of these include an improved segmentation method. Better segmentation can improve recognition by providing a better platform for learning visual features. This would be especially helpful for texture based features such as SIFT, in this case a full segmentation of the object will increase the number of available features for building the object model and also for matching. In addition, improved segmentation would help extend the MCOR framework from focusing on labeling object categories with localization focused on finding a point or region somewhere on the object, to a framework which can return a more complete boundary of the object location. Because of the utilization of human interaction with the objects, we have a unique position of utilizing this information for finding seed points to begin segmentation. Currently, our seed points were generated by the spatial association term, which we calculated by finding the offset with the greatest likelihood given a cue value. It would be interesting to find a better method of determining the position of the object in relation to the human. This can be done by looking into other human parts which maybe more consistent in the location of the object, such as hands. Since our datasets were focused on scenarios where most objects being interacted with was in front of the human, this was not a problem, but one can imagine other

datasets where head position would not be a very good indicator of object placement. Further work could explore the possibilities of better features and calculation methods. Along these lines, the burden of re-recognizing objects after they have been lost by the tracking algorithm can be greatly reduced by a more advanced tracking technique. The more successful tracking is, the less need there is to repeatedly find the objects.

- **Cue Recognition Systems and Dynamic Features**

  Although the cue recognition systems used by MCOR are not the primary focus of our thesis work, future work could include finding more robust, accurate, and informative cue systems. One of the advantages of MCOR is its ability to use whatever recognition systems are available, but an improvement of the overall recognition rate through an improvement in the cue recognition systems plugged in would be interesting to determine how MCOR could preform on even more complex datasets using these more enhanced cues. Along these lines, it would be interesting to explore the use of the dynamic features used by activity recognition systems directly to see if it results in an improvement in performance, rather than the activity labels generated by the system. This engineering enhancement would be an interesting next step for a wider application of the system.

- **Probabilistic Representation and Weight**

  In regards to the weights and probabilistic relational models, future work can be focused on taking further advantage of the relational aspect provided by the PRM model and new areas of weighting methods. The PRM model could be used in future work to not only include cue information but also to use the known presence or likely presence of other objects in order to provide evidence for an object category. For instance, the evidence for a table could be increased given that a laptop is found in a region above. Distance and position of these objects and cues can be another source of additional information that the algorithm could use to identify possible object regions. The use of negative weights is another potential use of the probabilistic model, since its possible that the presence of some items may help eliminate otherwise potential object category labels. For instance, the presence of a car would decrease the likelihood of getting a car.

- **Extension of Application**
  As briefly mentioned earlier, another interesting area of future work would be the exploration of MCOR on more extended applications. This includes video datasets such as more general How-to videos (not just those focused on kitchen settings) as well as more datasets that include the interactions of multiple people simultaneously

from an outside data source. In addition, further progression can lead to the testing of the approach on a mobile platform such as a robot. The MCOR algorithm would then have to include

- **Automatic Labeling of Training Data**
  While some aspects of the training data used had automatic labels (such as some of the activity labels and face bounding boxes), it would be a greater advantage to have a more automatic process of labeling the training data, such as using the range of color in human skin to recognize not only face, but hand locations automatically. In cases when automatic labeling is not possible, having a large unrelated group of people labeling items would remove an element of bias from the data and also would allow learning of weights for more general datasets, such that the association between activities and cues could be determined by a vast input of worldly knowledge from the general public. Amazon's mechanical turk would be a good option for collecting such data.

- **Inclusion of Cue Type Confidence**

  As briefly mentioned in an earlier chapter, the cue type confidence calculated using the PRM model would be an interesting source of future work, as there are a number of possible uses for this information. Since the cue type confidence provides the normalized mutual information between each object label and cue type in general. It would be interesting to explore uses of the confidence value to start prior values for unlearned cue values from other datasets. For instance, we could determine that a color cue provides no evidence to the presence of a pen, so there is no need to extract, learn, or use the evidence provided by any particular color value when looking for a pen. Thus, the use of entire cue class can be eliminated, or highly valued. In addition, the cue type value could be a useful term when it comes to context learning.

- **Context**
  An area of exploration that would be intriguing for future work would be developing an algorithm, which can give an accurate reflection of the relationship between the general and context-specific information. More specifically however, further extension could include: (1) a more robust and flexible context-based algorithm, (2) adaptation to dynamic contexts, and (3) the evaluation across various contexts.

  Context-Based Algorithm An important aspect of the future thesis work is the development of a robust, flexible context-based algorithm. In preliminary work C, the algorithm learned the weights for each of the contexts by treating each individual context as a separate learning problem. Thus, it simply replaced the general dictionary values with its own.

In future work, I propose a greater interaction between the general and the context-specific object dictionaries, where there will be a weighted contribution between the general definition and the context. This approach will allow the general definition to be more slowly molded to fit each individual as more evidence is accumulated, rather than the blind and total dependence on the context information.

Dynamic Contexts Future work could also explore the idea of changing contexts, i.e., how the definitions for a context-specific dictionary can be adjusted to not just contexts which remain the same within themselves, but which take into account the fact that the context may change.

Appendix A

Additional Results from Analytical Model

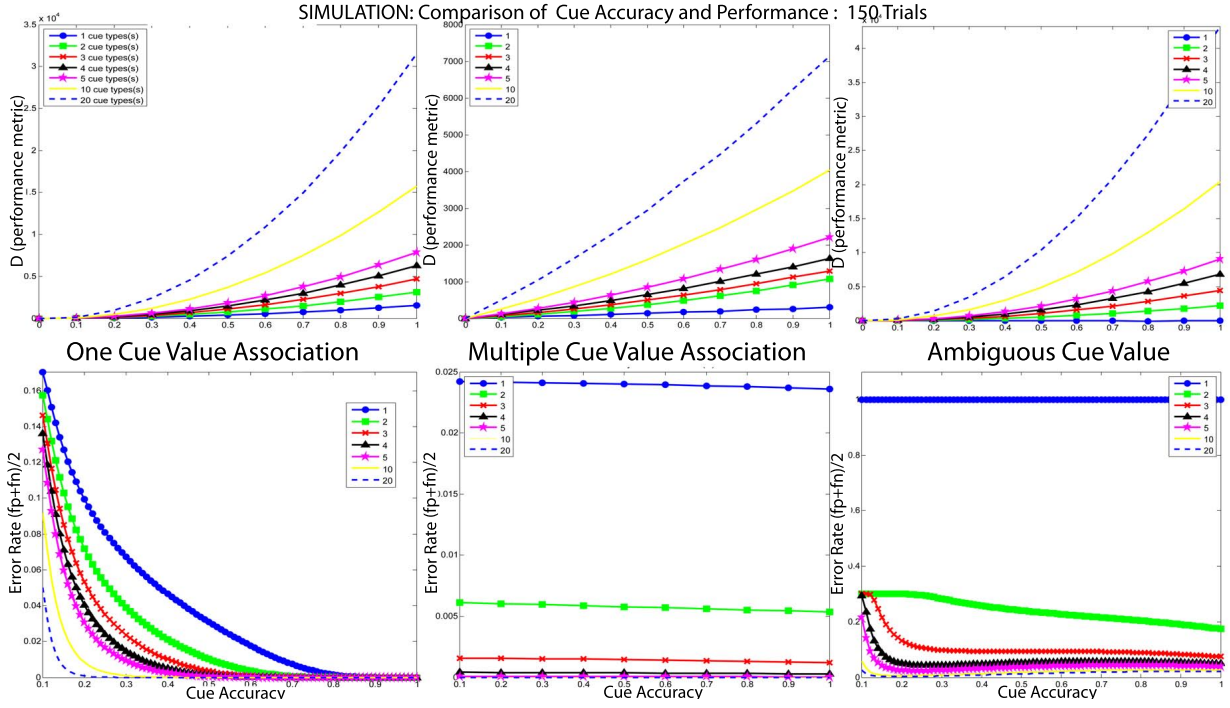


Figure A.1: Comparison of cue accuracy and performance for simulation results of single, multiple, and ambiguous cue value associations

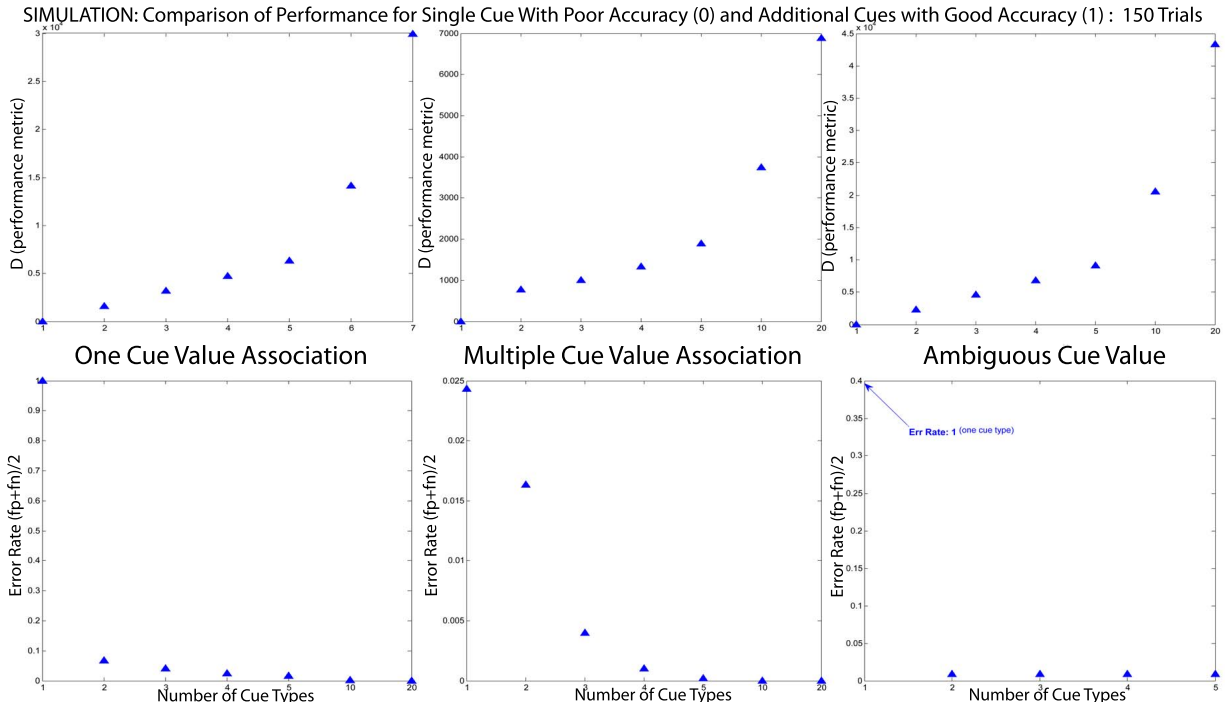


Figure A.2: Comparison of performance for a single cue type with poor accuracy ($\alpha = 0$) and additional cue types with good accuracy ($\alpha = 1$) from simulation of single, multiple, and ambiguous cue value associations

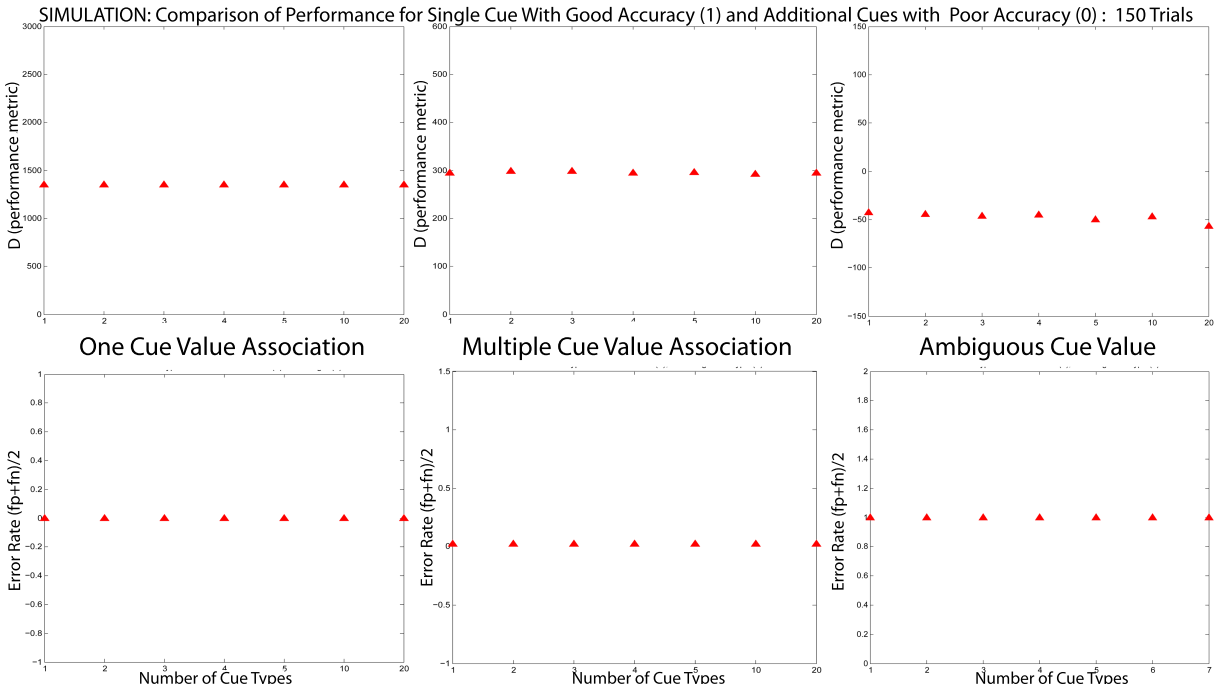


Figure A.3: Comparison of performance for a single cue type with good accuracy ($\alpha = 1$) and additional cue types with poor accuracy ($\alpha = 0$) for simulation of single, multiple, and ambiguous cue value association

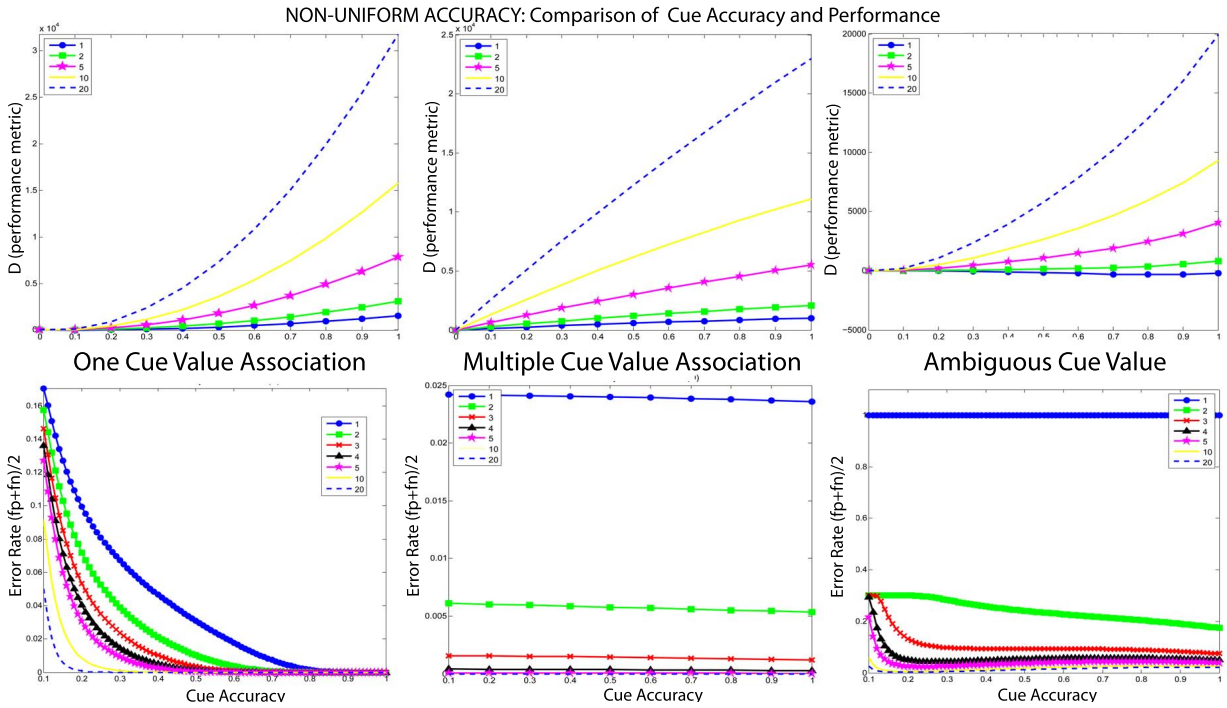


Figure A.4: Comparison of cue accuracy and performance for non-uniform accuracy for the single, multiple, and ambiguous cue value association

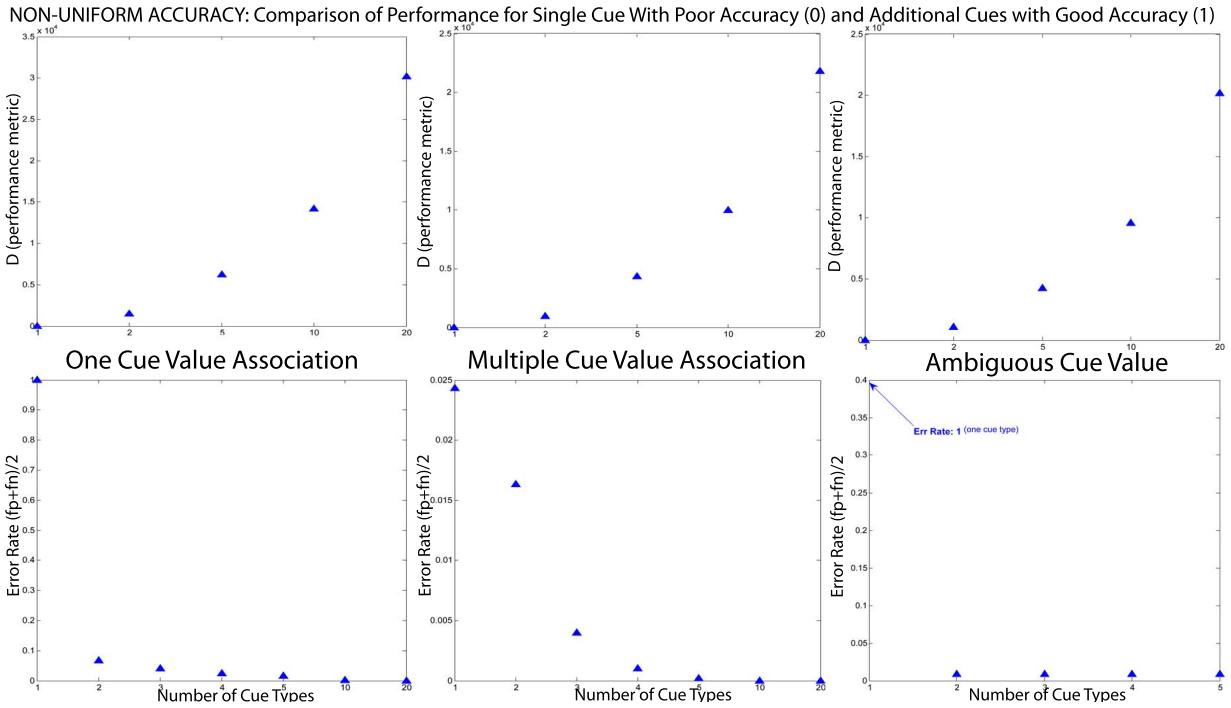


Figure A.5: Comparison of performance for a single cue type with poor accuracy ($\alpha = 0$) and additional cue types with good accuracy ($\alpha = 1$) for non-uniform accuracy of single, multiple, and ambiguous cue value association

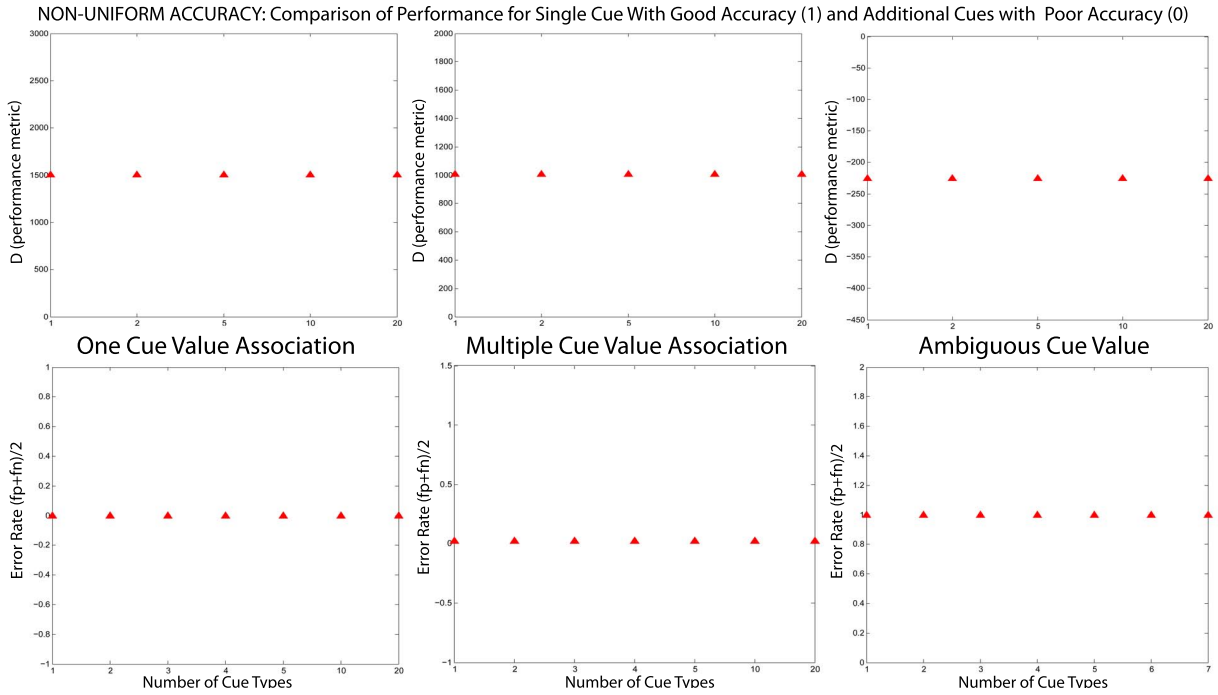


Figure A.6: Comparison of performance for a single cue type with good accuracy ($\alpha = 1$) and additional cue types with bad accuracy ($\alpha = 0$) for non-uniform accuracy of single, multiple, and ambiguous cue value association

Appendix B

Object Dictionaries

In this appendix, we provide the object dictionaries for each of the experiments. In cases where the object dictionary was updated during recognition, we first provide the initial object dictionary followed by the updated one. The object dictionaries given here come from the following datasets: Meeting dataset, QRIO dataset, Gupta dataset, Cooking dataset, and Meeting extended dataset.

B.1 Meeting dataset dictionaries

The initial object dictionary values were hand-defined. The initial object dictionary consisted of activity and speech information. Through generalization during recognition, visual cues (color and shape) were learned from objects recognized by the initial dictionary. New cues added to a definition were given a predefined value (.8).

B.2 QRIO object dictionary

The initial object dictionary values were hand-defined. The initial object dictionary consisted of all cues used during recognition: activity, shape, and color cues.

OBJECT	Chair	Laptop	Projector Screen	Whiteboard	Table
Activity					
Sit	**.8**	**.2**	0	0	0
Stand	0	0	0	0	0
Walk	0	0	0	0	0
Erase	0	0	0	**.8**	0
Write	0	0	0	**.8**	0
Point	0	0	**.8**	**.2**	0
Putdown	0	0	0	0	**1**
Speech					
"look that up"	0	**.8**	0	0	0

Figure B.1: Initial object dictionary for Meeting dataset

B.3 Gupta dataset dictionaries

The initial object dictionary values were hand-defined. The initial object dictionary consisted of association specified by the Gupta dataset where each activity was associated with only one object. Through generalization during recognition, visual cues (color and shape) were learned from objects recognized by the initial dictionary.

B.4 Cooking dataset dictionaries

The initial object dictionary was learned using a training dataset. The training dataset consisted of hand-labeled objects, activity information, speech information, and sound information using the ViPER software from a subset of the Cooking dataset videos. The object dictionary was updated based on recognized objects from the initial dictionary. Initial dictionary shown here was learned on a training dataset using all cues found during recognition. Thus, only slight updates to the weight values occurred during recognition, i.e., the initial and final dictionaries did not differ much from each other. This is especially apparent in the LACE dataset, which consisted of a laboratory setting where actions were scripted and objects consistent, thus cue values were the same for training and testing datasets.

OBJECT	Chair	Laptop	Projector Screen	Whiteboard	Table
Activity					
Sit	**.8**	**.2**	0	0	0
Stand	0	0	0	0	0
Walk	0	0	0	0	0
Erase	0	0	0	**.8**	0
Write	0	0	0	**.8**	0
Point	0	0	**.8**	**.2**	0
Putdown	0	0	0	0	**1**
Speech					
"look that up"	0	**.8**	0	0	0
Visual					
White&Shape1	0	0	**.8**	**.8**	0
Red&Shape1	**.8**	0	0	0	0
White&Shape2	0	0	0	0	**.8**
Black&Shape1	0	**.8**	0	0	0

Figure B.2: Final object dictionary for Meeting dataset

B.5 Extended Meeting dataset

Initial object dictionary values were hand-defined. The initial object dictionary consisted of activity and speech information. Through generalization during recognition, visual cues (color and shape) were learned from objects recognized by the initial dictionary. New cues added to a definition were given a predefined value (.8).

	Pen	Fork	Plate	Paper	Cellphone
ACTIVITY					
Eating	.01	.07	**.90**	.01	.06
Speaking	.08	**.92**	.02	.09	**.92**
Writing	**.91**	.01	.08	**.90**	.02
COLOR					
Blue	**.89**	.06	.04	.01	.01
Black	.10	**.90**	**.91**	.01	.01
White	.01	.02	.03	**.94**	.08
Yellow	.01	.02	.02	.05	**.90**
SHAPE (ratio)					
~1 (square)	.01	.02	**.91**	**.96**	.11
~.6 (rectangle)	.04	.05	.04	.03	**.85**
~.3 (long/thin)	**.95**	**.93**	.05	.01	.04

Figure B.3: Object dictionary used by QRIO experiments

OBJECT	Cup	Phone	SprayCan	Flashlight
Activity				
Spray	0	0	**1**	0
Answer	0	**1**	0	0
Light	0	0	0	**1**
Pour	**1**	0	0	0
Drink	**1**	0	0	0

Figure B.4: Initial object dictionary for Gupta dataset

OBJECT	Cup	Phone	SprayCan	Flashlight
Activity				
Spray	0	0	**1**	0
Answer	0	**1**	0	0
Light	0	0	0	**1**
Pour	**1**	0	0	0
Drink	**1**	0	0	0
Visual				
White&Shape1	**.8**	0	0	0
Black&Shape1	0	**.8**	0	0
Green&Shape2	0	0	0	**.8**
Black&Shape2	0	0	**.8**	0

Figure B.5: Final object dictionary for Gupta dataset

OBJECT	Bowl	Knife	OilJar	Spice	WineBottle	Cuttingboard	Plate	Grill	Pan	Spoon
Activity										
Twist	0	0	0	**1**	0	0	0	0	0	0
Smash	0	**1**	0	0	0	0	0	0	0	0
Shake	**.2**	0	0	**.6**	0	0	0	0	0	0
Drop/Add	**1**	0	0	**.6**	0	0	0	0	0	0
Cut	0	**1**	0	0	0	**.9**	0	0	0	0
Mix	**.9**	0	0	0	0	0	0	0	**.1**	**.8**
Pour	**.8**	0	**.6**	0	**.3**	0	**.2**	0	0	0
Squeeze	**1**	0	0	0	0	0	0	0	0	0
PickUp	**.6**	**.2**	**.2**	**.3**	**.4**	**.4**	0	0	0	0
PutDown	**.6**	**.2**	**.1**	**.3**	**.2**	**.5**	**.1**	0	0	0
Scoop	0	0	0	0	0	0	0	0	0	**1**
Point	**.4**	0	0	0	0	**.3**	**.2**	0	**.3**	0
Speech										
"Bowl"	**1**	0	0	0	0	0	0	0	0	0
"Knife"	0	**1**	0	0	0	0	0	0	0	0
"OilJar"	0	0	**1**	0	0	0	0	0	0	0
"Spice"	0	0	0	**1**	0	0	0	0	0	0
"WineBottle"	0	0	0	0	**1**	0	0	0	0	0
"Cuttingboard"	0	0	0	0	0	**1**	0	0	0	0
"Plate"	0	0	0	0	0	0	**1**	0	0	0
"Grill"	0	0	0	0	0	0	0	**1**	0	0
"Pan"	0	0	0	0	0	0	0	0	**1**	0
"Spoon"	0	0	0	0	0	0	0	0	0	**1**
Sound										
pour	**.8**	0	**.6**	0	**.3**	0	**.2**	0	0	0
sizzle	0	0	0	0	0	0	0	.4	.6	0
mix	**.9**	0	0	0	0	0	0	0	.1	.8
chop	**1**	0	0	0	0	**.9**	0	0	0	0
Color&Shape										
Yellow&Shape1	**.3**	0	**.1**	0	0	**.1**	0	0	0	0
Green&Shape1	**.3**	0	0	0	0	0	0	0	0	0
Orange&Shape 1	**.4**	0	0	0	0	0	0	**1**	0	0
White&Shape1	**.2**	0	0	0	0	0	0	0	0	0
Black&Shape1	0	**.1**	0	0	0	0	0	0	0	0
White&Shape2	0	**.1**	0	0	0	0	0	0	0	0
Black&Shape2	0	0	0	0	0	0	0	**.7**	0	0
Yellow&Shape2	0	0	0	0	0	**.9**	0	0	0	0

Figure B.6: Initial object dictionary for Cooking/Rachel Ray dataset

OBJECT	Bowl	Knife	OilJar	Spice	WineBottle	Cuttingboard	Plate	Grill	Pan	Spoon
Activity										
Twist	0	0	0	**1**	0	0	0	0	0	0
Smash	0	**1**	0	0	0	0	0	0	0	0
Shake	**.4**	0	0	**.6**	0	0	0	0	0	0
Drop/Add	**.9**	0	0	**.6**	0	0	0	0	**.1**	0
Cut	0	1	0	0	0	**.9**	0	0	0	0
Mix	**.9**	0	0	0	0	0	0	0	**.1**	**.8**
Pour	**.8**	0	**.6**	0	**.3**	0	**.2**	0	0	0
Squeeze	**1**	0	0	0	0	0	0	0	0	0
PickUp	**.4**	**.3**	**.2**	**.3**	**.1**	**.4**	**.1**	0	0	0
PutDown	**.5**	**.3**	**.1**	**.3**	**.1**	**.5**	**.1**	0	0	0
Scoop	0	0	0	0	0	0	0	0	0	**1**
Point	**.3**	0	0	0	0	**.3**	**.3**	0	**.3**	0
Speech										
"Bowl"	**1**	0	0	0	0	0	0	0	0	0
"Knife"	0	**.9**	0	0	0	**.1**	0	0	0	0
"OilJar"	0	0	**1**	0	0	0	0	0	0	0
"Spice"	0	0	0	**1**	0	0	0	0	0	0
"WineBottle"	0	0	0	0	**1**	0	0	0	0	0
"Cuttingboard"	0	0	0	0	0	**1**	0	0	0	0
"Plate"	0	0	0	0	0	0	**1**	0	0	0
"Grill"	0	0	0	0	0	0	0	**1**	0	0
"Pan"	0	0	0	0	0	0	0	0	**1**	0
"Spoon"	0	0	0	0	0	0	0	0	0	**1**
Sound										
pour	**.7**	0	**.6**	0	**.3**	0	**.3**	0	0	0
sizzle	0	0	0	0	0	0	0	**.4**	**.6**	0
mix	**.9**	0	0	0	0	0	0	0	**.1**	**.8**
chop	**1**	0	0	0	0	**1**	0	0	0	0
Color&Shape										
Yellow&Shape1	**.4**	0	**.1**	0	0	**.2**	0	0	0	0
Green&Shape1	**.2**	0	0	0	0	0	0	0	0	0
Orange&Shape 1	**.5**	0	0	0	0	0	0	0	0	0
White&Shape1	**.2**	0	0	0	0	0	0	0	0	0
Black&Shape1	0	**.1**	0	0	0	0	0	0	0	0
White&Shape2	0	**.1**	0	0	0	0	0	0	0	0
Black&Shape2	0	0	0	0	0	0	0	**.8**	0	0
Yellow&Shape2	0	0	0	0	0	**.9**	0	0	0	0
...										

Figure B.7: Final object dictionary for Cooking/Rachel Ray dataset

OBJECT	Bowl	Knife	Cup	Cereal	Spoon	Cuttingboard	Jug	Fridge	Plate	Cupboard
Activity										
Eat	**1**	0	0	**1**	**1**	0	0	0	0	0
Drink	0	0	**.1**	**.1**	0	0	0	0	0	0
Bring	**.1**	**.1**	**.1**	**.1**	**.1**	**.1**	**.1**	0	**.1**	0
Return	**.1**	**.1**	**.1**	**.1**	**.1**	**.1**	**.1**	0	**.1**	0
Open	0	0	0	0	0	0	0	**.5**	0	**.5**
Close	0	0	0	0	0	0	0	**.5**	0	**.5**
Pour	**.5**	0	**.5**	0	**.4**	0	**.6**	0	0	0
Mix	**1**	0	0	0	**1**	0	0	0	0	0
Speech										
"Bowl"	**1**	0	0	0	0	0	0	0	0	0
"Knife"	0	**1**	0	0	0	0	0	0	0	0
"Cup"	0	0	**1**	0	0	0	0	0	0	0
"Cereal"	0	0	0	**1**	0	0	0	0	0	0
"Spoon"	0	0	0	0	**1**	0	0	0	0	0
"Cuttingboard"	0	0	0	0	0	**1**	0	0	0	0
"Jug"	0	0	0	0	0	0	**1**	0	0	0
"Fridge"	0	0	0	0	0	0	0	**1**	0	0
"Plate"	0	0	0	0	0	0	0	0	**1**	0
"Cupboard"	0	0	0	0	0	0	0	0	0	**1**
Sound										
pourLiquid	**.5**	0	**.5**	0	0	0	**1**	0	0	0
pourSolid	1	0	0	**1**	0	0	0	0	0	0
mix	**1**	0	0	**1**	0	0	0	0	0	0
Color&Shape										
Blue&Shape1	**1**	0	0	0	0	0	0	0	0	0
Black&Shape1	0	**.4**	0	0	**.6**	0	0	0	0	0
White&Shape1	0	**.5**	0	0	**.5**	0	0	0	0	0
Green&Shape1	0	0	**1**	0	0	0	0	0	0	0
Blue&Shape2	0	0	0	**.3**	0	0	0	0	0	0
White&Shape2	0	0	0	**.3**	0	0	0	0	0	0
Black&Shape2	0	0	0	0	0	**1**	0	0	0	0
White&Shape3	0	0	0	0	0	0	**1**	0	0	0
Black&Shape3	0	0	0	0	0	0	0	**1**	0	0
Green&Shape2	0	0	0	0	0	0	0	0	**1**	0
Yellow&Shape1	0	0	0	0	0	0	0	0	0	**1**

Figure B.8: Initial object dictionary for Cooking/LACE dataset

OBJECT	Bowl	Knife	Cup	Cereal	Spoon	Cuttingboard	Jug	Fridge	Plate	Cupboard
Activity										
Eat	**1**	0	0	**1**	**1**	0	0	0	0	0
Drink	0	0	**.1**	**.1**	0	0	0	0	0	0
Bring	**.1**	**.1**	**.1**	**.1**	**.1**	**.1**	**.1**	0	**.1**	0
Return	**.1**	**.1**	**.1**	**.1**	**.1**	**.1**	**.1**	0	**.1**	0
Open	0	0	0	0	0	0	0	**.5**	0	**.5**
Close	0	0	0	0	0	0	0	**.5**	0	**.5**
Pour	**.5**	0	**.5**	0	**.4**	0	**.6**	0	0	0
Mix	**1**	0	0	0	**1**	0	0	0	0	0
Speech										
"Bowl"	**1**	0	0	0	0	0	0	0	0	0
"Knife"	0	**1**	0	0	0	0	0	0	0	0
"Cup"	0	0	**1**	0	0	0	0	0	0	0
"Cereal"	0	0	0	**1**	0	0	0	0	0	0
"Spoon"	0	0	0	0	**1**	0	0	0	0	0
"Cuttingboard"	0	0	0	0	0	**1**	0	0	0	0
"Jug"	0	0	0	0	0	0	**1**	0	0	0
"Fridge"	0	0	0	0	0	0	0	**1**	0	0
"Plate"	0	0	0	0	0	0	0	0	**1**	0
"Cupboard"	0	0	0	0	0	0	0	0	0	**1**
Sound										
pourLiquid	**.5**	0	**.5**	0	0	0	**1**	0	0	0
pourSolid	1	0	0	**1**	0	0	0	0	0	0
mix	**1**	0	0	**1**	0	0	0	0	0	0
Color&Shape										
Blue&Shape1	**1**	0	0	0	0	0	0	0	0	0
Black&Shape1	0	**.4**	0	0	**.6**	0	0	0	0	0
White&Shape1	0	**.5**	0	0	**.5**	0	0	0	0	0
Green&Shape1	0	0	**1**	0	0	0	0	0	0	0
Blue&Shape2	0	0	0	**.3**	0	0	0	0	0	0
White&Shape2	0	0	0	**.3**	0	0	0	0	0	0
Black&Shape2	0	0	0	0	0	**1**	0	0	0	0
White&Shape3	0	0	0	0	0	0	**1**	0	0	0
Black&Shape3	0	0	0	0	0	0	0	**1**	0	0
Green&Shape2	0	0	0	0	0	0	0	0	**1**	0
Yellow&Shape1	0	0	0	0	0	0	0	0	0	**1**

Figure B.9: Final object dictionary for Cooking/LACE dataset

OBJECT	Chair	Mug/Cup	Pen
Activity			
Sit	**1**	0	0
Drink	0	**1**	0
Write	0	0	**1**
Speech			
"Chair"	**1**	0	0
"Pen"	0	0	**1**
"Mug"	0	**1**	
"Cup"	0	**1**	0

Figure B.10: Initial object dictionary for Meeting Extended dataset

OBJECT	Chair	Mug/Cup	Pen
Activity			
Sit	**1**	0	0
Drink	0	**1**	0
Write	0	0	**1**
Speech			
"Chair"	**1**	0	0
"Pen"	0	0	**1**
"Mug"	0	**1**	
"Cup"	0	**1**	0
Visual			
Red&Shape1	**.8**	0	0
Pink&Shape1	0	0	**.8**
Yellow&Shape1	0	0	**.8**
Red&Shape2	0	**.8**	0
Blue&Shape1	0	**.8**	0
Yellow&Shape2	0	**.8**	0
Blue&Shape2	0	**.8**	0
White&Shape1	0	**.8**	0
Black&Shape1	0	**.8**	0

Figure B.11: Final object dictionary for Meeting Extended dataset

Appendix C

Context Learning

We demonstrate in the appendix some initial experiments into the idea of context learning for MCOR. Context learning focuses on two aspects: (1) adaptation of the weights learned to a specific context and (2) the building of a general object definition which can serve as a good starting point for all contexts based on the general properties.

C.1 Adaptation to Specific Contexts

Within each object dictionary, there is a list of each of the objects to be recognized. For each object, the cues associated with that object are listed along with the weight of the association between the object and cue (valued between 0 and 1). See Figure C.1 for an example. In most cases, if the weight is equal to 0, the cue is not shown in that object's definition.

In this initial model of context-specific dictionaries, the general object dictionary is updated for a particular context by recalculating the probability based on that context specific data alone, i.e., the weight, $w_{i,j,k}$ for the object class, o_i, the cue c_j, and the context x_k is calculated using the following equation:

$$w_{i,j,k} = \#(c_{i,j,k})/\#(o_{i,k})$$

where $\#(c_{i,j,k})$ is the number of cues counted from the context-specific data with the value j and associated with the ith object class and $\#(o_{i,k})$ is the number of objects in the context-specific data that falls under the object class, o_i.

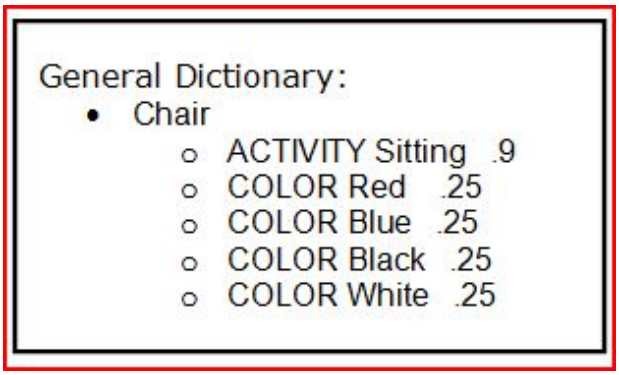


Figure C.1: Example of an object dictionary. Specifically, this is the general object dictionary that is later updated in Figure C.3.

C.2 Building a General Definition

The general definition is updated in a similar manner: To calculate the weights, a similar count is done as in the equation above for each cue and object association, except across all the contexts, i.e.,

$$w_{i,j} = \sum_{k} \#(c_{i,j,k})/\#(o_{i,k})$$

Below we give a small demonstration of the two techniques.

C.3 Experiment and Results

In the experiments below, video was taken from a Panasonic hand-held digital video camera.

C.3.1 Adaptation to Specific Contexts

In this first demonstration, we illustrate adaptation of the general object dictionary to a context-specific one. The original dictionary can be found in Figure C.1. Note that there is an equal value across all the color value weights for chair, since generally chairs come in all colors without preference.

However, in a specific context, as that shown in Figure C.2, chairs can often come in a single color, (in this case, black) making it a good indicator for the chair class. Because of the general dictionary (Figure C.1), the first chair can be recognized based on the activity cue of sitting alone (Figure C.2).

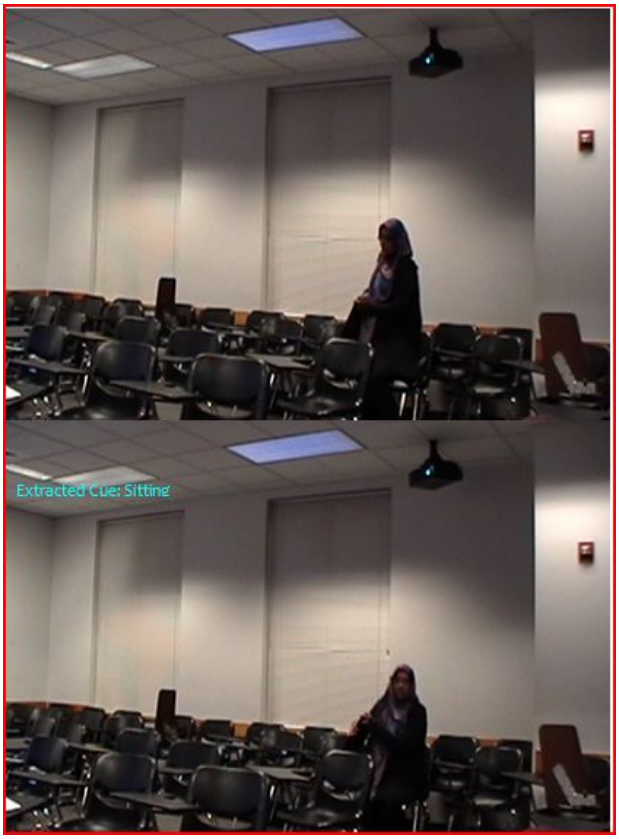


Figure C.2: Demonstration of sitting cue extracted from video recorded in the first context.

The other chairs can then be recognized based on the additional cue of color learned from the labeling (see Figure C.3). The object dictionary is then updated for this specific context accordingly (Figure C.3): The weight for black is changed to 1, since all the chairs in the context that are recognized are black, and all other colors are changed to 0.

We can now use this updated context-specific dictionary to recognize objects from different video sequences which were taken in the same context (Figure C.4). Thus, the chairs were able to be correctly recognized even without activity cue, which would have been necessary previously, thus taking advantage of the context-specific information learned previously.

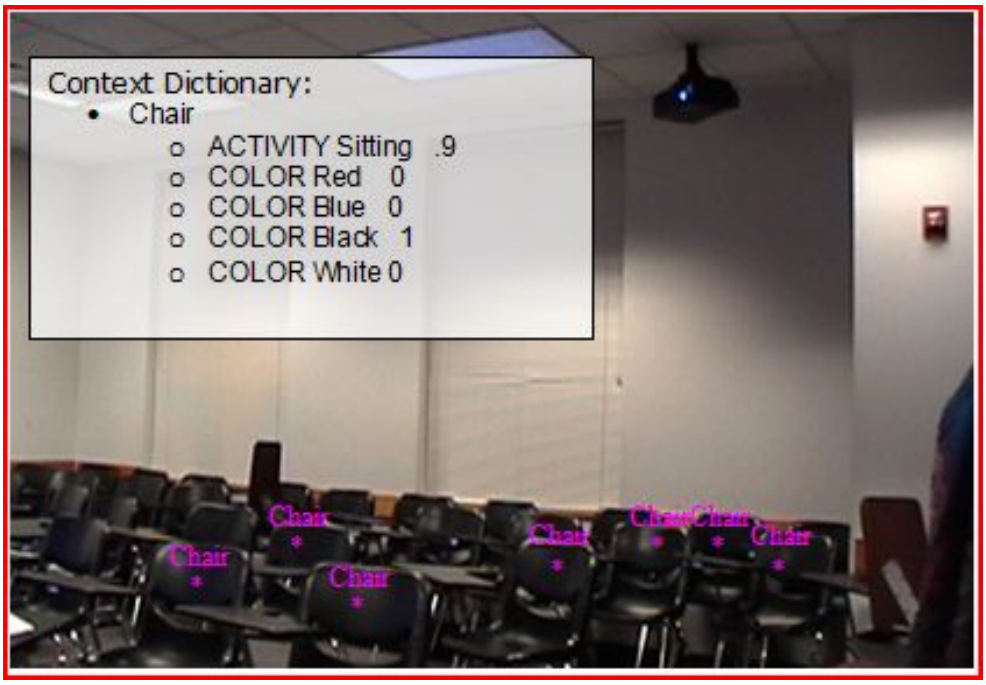


Figure C.3: Recognition results of the chair in the first context, and the updated context definition. Original dictionary can be found in figure C.1

C.3.2 Building a General Definition

In the results below, we show a simple demonstration of the learning of the general dictionary from multiple context data. In the original dictionary (Figure C.5) the weights were hand labeled.

An updated general dictionary was then learned using two different contexts, both containing a chalkboard, which could be recognized by the original dictionary information. Thus, additional cue information (color) was added to the general dictionary looking across contexts.

C.4 Conclusion

In the preliminary results above, previous work showed only two contexts. For future thesis work, The inclusion of various buildings (e.g.,Wean, Newel-Simon) and various room style (e.g.,classroom, conference room, atrium) allows for the investigation of the relationship of the objects and cues across each of the types of contexts. For instance, it may be that

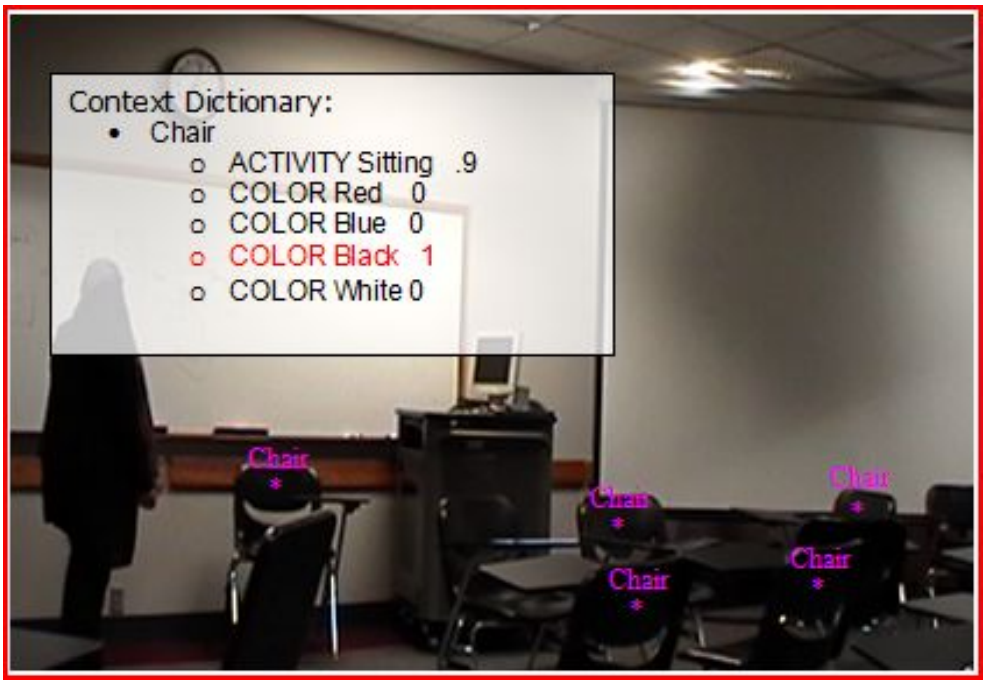


Figure C.4: Recognition results for 'chair' based on the context-specific dictionary learned from the previous video sequence in the same context (see Figure C.3)

color of chairs in Wean are all the same regardless of room style, while the shape of all tables in conference rooms are the same regardless of the building.

In addition, further progression could lead to the testing of the approach on a mobile platform such as a robot. This evaluation will test the ability of the algorithm to recognize objects as it moves from room to room.

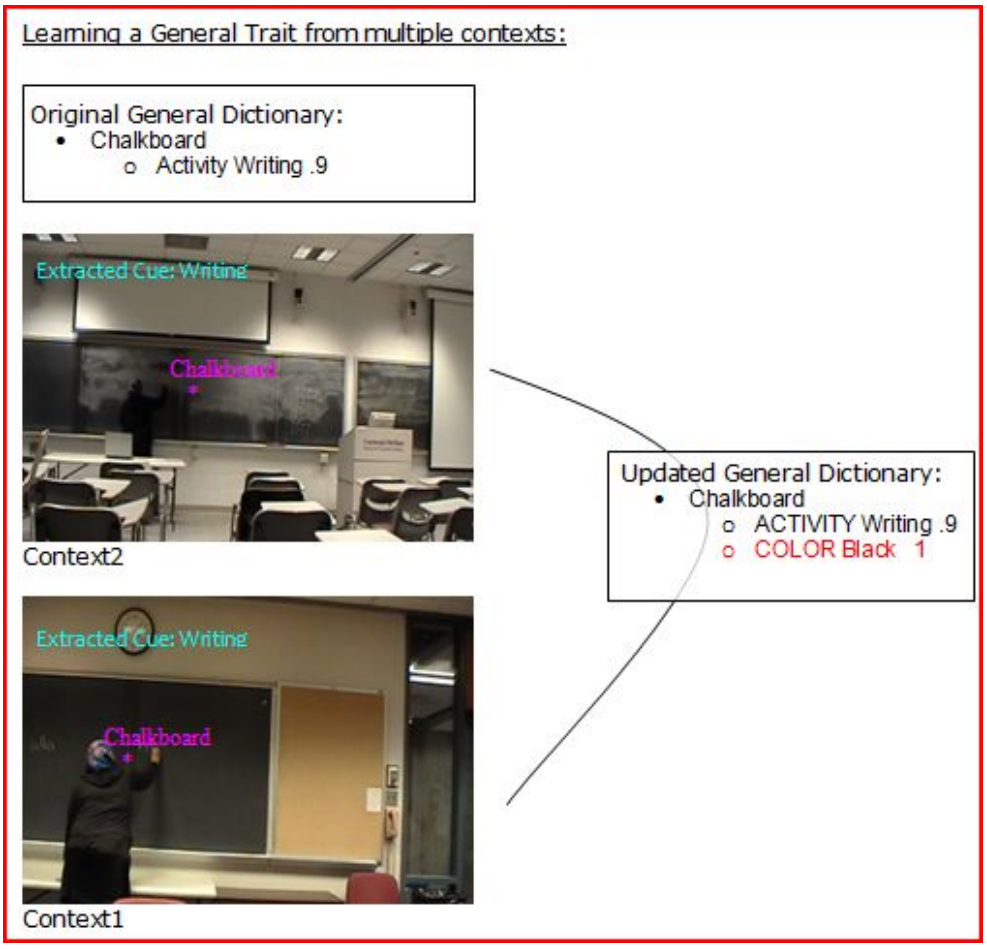


Figure C.5: Demonstration of the learning of a general trait from multiple contexts. The original general dictionary is found at the top of the figure, the updated version is found on the center-right. The new dictionary was calculated from the data provided by context 1 and context 2.

Appendix D

Probability Model for SIFT Similarity

In Chapter 4, we used SIFT features as one of the vision cues used in our multiple-cue object recognition (MCOR). In our MCOR framework, each cue must define how the similarity between an extracted cue taken from an image and a definition cue taken from the object dictionary are compared in order to determine a match. For the SIFT features, we use Lowe's probabilistic verification model [38], which determines the probability of the presence of an object model (taken from the object dictionary) based on the actual number of matching features found, to determine the similarity measure for a SIFT visual cue. This appendix, thus, outlines the probabilistic model given by Lowe and used for our similarity calculation for the SIFT visual cue.

In Lowe's work, object model verification is determined by the probability: $P(m|f)$, where m is the presence of the model at a given pose and f is the set of k features that have been matched between a model and an image (i.e. a definition cue and an extracted cue in the MCOR framework). It is this probability, which we use as the SIFT similarity measure.

$P(m|f)$ can be determined using the Bayes' theorem:

$$P(m|f) = \frac{P(f|m)P(m)}{P(f)} \tag{D.0.1}$$

$$= \frac{P(f|m)P(m)}{P(f|m)P(m) + P(f|\neg m)P(\neg m)} \tag{D.0.2}$$

According to Lowe, $P(f|m)$ and $P(\neg m)$ can be approximated as 1. For the former, this

is possible because we normally expect to see at least k features present when the model is present (for the small values of k for which this evaluation is relevant). For the latter, there is a very low prior probability of a model appearing at a particular pose. Thus, we can approximate $P(f|m)$ as:

$$P(m|f) \approx \frac{P(m)}{P(m) + P(f|\neg m)} \tag{D.0.3}$$

We let $P(f|\neg m)$ be the probability that the matched features, f, would arise by accident if the model, m, is not present. The model assumes that the k features matches arose from n possible features, each of which matches by accident with probability, p. The cumulative binomial distribution can then be used to calculate, $P(f|\neg m)$:

$$P(f|\neg m) \sum_{j=k}^{n} \binom{n}{j} p^j (1-p)^{n-j} \tag{D.0.4}$$

We let p be the probability of accidentally matching a single image feature to the current object model. p can be calculated as follows:

$$p = dlrs \tag{D.0.5}$$

where d is the probability of accidentally selecting a database match to the current object model. d is given by the fraction of the database (i.e. object models saved in the object dictionary) occupied by features from this model view. l, r, and s are the probabilities of satisfying the geometric constraints (location, orientation, and scale respectively). As defined by Lowe, we constrain the location, l, to 20 percent of the average model size in each direction, i.e. $l = 0.2^2 = 0.04$. Orientation is constrained to a range of 30 degrees: $r = 30/360 = 0.085$. Scale, s, is estimated as 0.5 since keypoint scale can only be constrained within a one-octave range due to inaccuracy in scale measurements and since most keypoints are detected at the finest scales.

$P(m)$ is calculated by taking the ratio of correctly matched features to all matched features in a typical image. This gives the probability that a single feature match is correct. This corresponds to the probability of an object model since we can assume that one matching features is used to determine the initial pose hypothesis (incorporated into m), and the remaining features are used for verification (reducing k by 1).

Using $P(m)$ and $P(f|\neg m)$ in the Bayesian equation, the resulting $P(f|m)$ can be used as the similarity measure for the SIFT vision cue.

Appendix E

List of Notation

O	Set of all object categories . . . 7
o_i	i^{th} object category . . . 7
C_i	Set of definition cues for i^{th} object category . . . 8
$w_{i,j}$	Weight strength between the i^{th} object category and j^{th} cue value . 12
Δp	Spatial association . . . 10
Δf	Temporal association . . . 11
Δp^*	Spatial association with greatest likelihood . . . 23
Δf^*	Temporal association with greatest likelihood . . . 24
cv_j	j^{th} cue value . . . 12
X	Set of classes for PRM . . . 15
I	Instance of a PRM schema with a set of entities and all attributes filled in . . . 15
σ	Partial specification of an instance for a PRM schema, only fixed attributes filled in . . . 20
$A(X_i)$	Attributes for i^{th} class in PRM . . . 15
$E^I(X_i)$	All entities of instance, I, for class, X_i, in PRM . . . 15
$E^\sigma(X_i)$	All entities of skeleton, σ, for class, X_i, in PRM . . . 20
$\mathbf{xd_k}.a_j$	Entity with fixed attribute value, $\mathbf{xd_k}$, and attribute, a_j, in PRM . 15
$\mathbf{xd_k}$	k^{th} fixed attribute with value $\mathbf{xd}$ in PRM . . . 15
$\mathbf{f}.R.a_n$	Slot for X_i, i.e., all entities $\in X_i$ whose attribute is $a_n \in A(X_i)$ related to entity, $\mathbf{f}$, through relation, R, in PRM . . . 16
$\gamma_H()$	Aggregation of a slot using a function, H, in PRM . . . 16
od	Object id for entity in OBJECT class in PRM . . . 16
og	Object category for entity in OBJECT class in PRM . . . 16
ol	Object location (bounding box) for entity in OBJECT class in PRM 16
ov	Object visibility for entity in OBJECT class in PRM . . . 17

σ'	Standard deviation for activation value given absence of object	69
fp	False positive rate	71
fn	False negative rate	71
ER	Error rate metric for recognition performance	71
D	Disparity metric for recognition performance	72
N_z	Binomial random variable representing total unweighted activation value provided by cue value, z given object present	73
$X_z^{(f)}$	Random variable representing success or failure of cue value, z, found on frame, f given object present	73
p_z	Probability of success, i.e. $P(X_z^{(f)}) = 1$ given object present	73
N'_z	Binomial random variable representing total unweighted activation value provided by cue value, z given object absent	73
$X'^{(f)}_z$	Random variable representing success or failure of cue value, z, found on frame, f given object absent	73
p'_z	Probability of success, i.e. $P(X_z^{(f)}) = 1$ given object absent	73
F	Set of frames from a given video	73
F_i	Set of frames where object of i^{th} category is present	73
F'_i	Set of frames where object of i^{th} category is absent	74
V	Number of cue values for each cue type in analytical model	79
M	Number of cue types in analytical model	79
α	Parameter for cue accuracy	79
$\hat{c}$	Cue value observed	79
c	Actual cue value	79
RER	Real error rate for recognition	101
AER	Analytical error rate for recognition	101

Bibliography

[1] Sarah Aboutalib and Manuela Veloso. Towards using multiple cues for robust object recognition. In *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8, New York, NY, USA, 2007. ACM. ISBN 978-81-904262-7-5. doi: http://doi.acm.org/10.1145/1329125.1329354.

[2] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(11):1475–1490, November 2004.

[3] J. Aloimonos and D. Shulman. Integration of visual modules: an extension of the marr paradigm. 1989.

[4] D.H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 1981.

[5] H. Bay, T. Tuytelaars, and L.V. Gool. "surf: Speeded up robust features". *Proceedings of the ninth European Conference on Computer Vision*, 2006.

[6] A. Berg and J. Malik. Geometric blur for template matching. *ICVPR*, 2001.

[7] A. M. Borghi. Object concepts and action: Extracting affordances from object parts. *Acta Psychologica*, (115):69–96, 2004.

[8] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. *Proceedings of IEEE International Conference of Computer Vision*, 2007.

[9] Jr. Claude Lester Fennema. *Interweaving reason, action and perception.* PhD thesis, Amherst, MA, USA, 1992.

[10] Ronald Cole, Joseph Mariani, and Hans Uszkoreit. Survey of the state of the art in human language technology. *Studies In Natural Language Processing, XIIXIII*, 1997.

[11] R. Cooper, P. Yule, and J. Fox. Cue selection and category learning: A systematic comparison of three theories. *Cognitive Science Quarterly*, 2003.

[12] N. Dalal and B. Triggs. Histogram of oriented gradients for fast human detection. *Proceedings IEEE Conference of Computer Vision and Pattern Recognition*, 2005.

[13] Zijian Deng, Bicheng Li, and Jun Zhuang. Image object recognition by svms and evidence theory. In *CIVR*, pages 560–567, 2005.

[14] S. Edelman and D. Weinshall. A self-organizing multiple-view representation of 3d objects. *Biological Cybernetics*, pages 209–219, 1991.

[15] E. A. Feigenbaum and H. A. Simon. A theory of the serial position effect. british journal of psychology. pages 307–320, 1962.

[16] E. A. Feigenbaum and H. A. Simon. Epam-like models of recognition and learning. *Cognitive Science*, pages 305–336, 1984.

[17] David A. Forsyth and Jean Ponce. Prentice Hall.

[18] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *IJCAI*, pages 1300–1309, 1999.

[19] Lise Getoor, Eran Segal, Ben Taskar, and Daphne Koller. Probabilistic models of text and link structure for hypertext classification. In *In IJCAI01 Workshop on Text Learning: Beyond Supervision*, 2001.

[20] James Gibson. *The Ecological Approach to Visual Perception*. Lawrence Erlbaum Associates, New Jersey,USA, 1979.

[21] J.J. Gibson. *The theory of affordance. In: Percieving, Acting, and Knowing*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1977.

[22] A. Gupta, J. Shi, and L. Davis. A shape aware model for semi-supervised learning of objects and its context. *Proceedings Conference of Neural Information Processing Systems*, 2008.

[23] Abhinav Gupta and Larry S. Davis. Objects in action:an approach for combining action understanding and object perception. *CVPR*, 2007.

[24] C. Harris and M. Stephens. A combined corner and edge detector. *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[25] P. Hough. Method and means for recognizing complex patterns. *U.S. Patent 3069654*, 1962.

[26] Martin Jttner, Erol Osman, and Ingo Rentschler. When touch forms vision- object recognition as a function of polysensory prior knowledge, 2001.

[27] J.-C. Junqua and J.-P. Haton. *Robustness in Automatic Speech Recognition: Fundamentals and Applications*. Kluwer Academic Publishers, 1995.

[28] R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 1960.

[29] D. Kersten and A. Yuille. Bayesian models of object perception. *Current Opinion in Neurobiology*, (13), 2003.

[30] Sungho Kim, Kuk-Jin Yoon, and In So Kweon. Object recognition using a generalized robust invariant feature and gestalts law of proximity and similarity. *Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, 2006.

[31] G. Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 1996.

[32] Josef Kittler, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas. On combining classifiers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(3):226–239, 1998. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/34.667881.

[33] Language and Media Processing Laboratory. The video performance evaluation resource. *http://viper-toolkit.sourceforge.net/*.

[34] Gia De Laurentiis. Everyday italian. *Cooking Show DVD*.

[35] S. Lazebnik, C. Schmid, and J. Ponce. Semi-local affine parts for object recognition. *BMVC*, 2004.

[36] B. Leibe and B. Schiele. Interleaved object categorization and segmentation. *BMVC*, 2003.

[37] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision Special Issue on Learning for Recognition and Recognition for Learning*, 77(1-3):259–289, 2008.

[38] David G. Lowe. Local feature view clustering for 3d object recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 682–688, 2001.

[39] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[40] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[41] Hoi Shun Miu, Kwong-Sak Leung, and Yee Leung. An evolutionary multi-agent system for object recognition in satellite images. In *IEEE Congress on Evolutionary Computation (1)*, pages 520–527, 2003.

[42] B. Moghaddam and A. Pentland. Probabilistic visual learning for object detection. In *International Conference on Computer Vision (ICCV'95)*, pages 786–793, Cambridge, USA, June 1995. URL `citeseer.ist.psu.edu/moghaddam95probabilistic.html`.

[43] Darnell J. Moore, Irfan A. Essa, and Monson H. Hayes. Exploiting human actions and object context for recognition tasks. In *ICCV (1)*, pages 80–86, 1999. URL `citeseer.ist.psu.edu/moore99exploiting.html`.

[44] K.P. Murphy, A. Torralba, and W.T. Freeman. Using the forest to see the trees: a graphical model realting features, objects, and scenes. *NIPS*, 16, 2003.

[45] E. Murphy-Chutorian and J. Triesch. Shared features for scalable appearance-based object recognition. *Proc. IEEE Workshop Applications of Computer Vision*, January 2005.

[46] M.E. Nilsback and B. Caputo. Cue integration through discriminative accumulation. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:578–585, 2004. ISSN 1063-6919. doi: http://doi.ieeecomputersociety.org/10.1109/CVPR.2004.67.

[47] University of Rochester. Lace: Indoor activity benchmark dataset. *http://www.cs.rochester.edu/ spark/muri/*.

[48] Wikipedia on Segmentation. http://en.wikipedia.org/wiki/segmentation_(image_processing).

[49] Wikipedia on Tracking. http://en.wikipedia.org/wiki/video_tracking.

[50] S.E. Palmer, E. Rosch, and P. Chase. Canonical perspective and the perception of objects. In Long J. and A. Baddeley, editors, *Attention and Performance*, pages 135–151. Erlbaum Hillsdale, N.J., 1981.

[51] B. Parhami. Voting algorithms. *Reliability, IEEE Transactions on*, 43(4):617–629, 1994.

[52] Gabriele Peters. Theories of Three-Dimensional Object Perception - A Survey. In *accepted for: Recent Research Developments in Pattern Recognition*. Transworld Research Network, 2000. URL `citeseer.ist.psu.edu/peters00theories.html`.

[53] T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *Nature*, 1985.

[54] Fatih Porikli. Achieving real-time object detection and tracking under extreme conditions, 2006.

[55] Rachel Ray. 30 minute meals. *Cooking Show DVD*.

[56] A. Reno and D. Booth. object recognition with multiple cues. *10th Mediterranean Electrotechnical Conference*, 2000.

[57] H. B. Richman, F. Gobet, J. J. Staszewski, and H. A. Simon. *Perceptual and memory processes in the acquisition of expert performance: The EPAM model*. 1996.

[58] Edwina L. Rissland. Ai and similarity. *IEEE Intelligent Systems*, 21(3):39–49, 2006.

[59] Ehud Rivlin, Sven J. Dickinson, and Azriel Rosenfeld. Recognition by functional parts. *Computer Vision and Image Understanding*, 62:164–176, 1995.

[60] Paul E. Rybski and Manuela M. Veloso. Robust real-time human activity recognition from tracked face displacements. *In Proceedings of EPIA'05, the 12th Portuguese Conference on Artificial Intelligence*, 2005.

[61] K. Saenko and T. Darrell. Object category recognition using probabilistic fusion of speech and image classifiers. *Proc. MLMI*, 2007.

[62] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

[63] Rohini K. Srihari. Computational models for integrating linguistic and visual information: A survey. *Artificial Intelligence Review*, 8(5-6):349–369, 1994. URL `citeseer.ist.psu.edu/srihari95computational.html`.

[64] M. Stark, P. Lies, M. Zillich, J. Wyatt, and B. Schiele. Functional object class detection based on learned affordance cues. *Computer Vision Systems: 6th International Conference (ICVS)*, pages 435–444, 2008.

[65] A. Stein, D. Hoiem, and M. Hebert. Learning to find object boundaries using motion cues. *IEEE International Conference on Computer Vision*, 2007.

[66] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. *In Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI02)*, 2002.

[67] A. Temko and C. Nadeu. Classification of meeting-room acoustic events with support vector machines and variable-feature-set clustering. *In IEEE International Conference on Acoustics, Speech, and Signal Processing(ICASSP)*, 5:505–508, 2005.

[68] Andrey Temko, Robert Malkin, Christian Zieger, Dusan Macho, Climent Nadeu, and Maurizio Omologo. Clear evaluation of acoustic event detection and classification systems. *CLEAR*, pages 311–322, 2006.

[69] D. Terzopoulos. Integrating visual information from multiple sources. *From pixels to predicates*, 1986.

[70] Manuela M. Veloso, Paul E. Rybski, and Felix von Hundelshausen. Focus: a generalized method for object discovery for robots that observe and interact with humans. In *Proceedings of the 2006 Conference on Human-Robot Interaction*, March 2006.

[71] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2001.

[72] Felix von Hundelshausen and Raul Rojas. Tracking regions and edges by shrinking and growing. *In Proceedings of the RoboCup 2003 International Symposium*, 2003.

[73] Ye-Yi Wang and Alex Acero. Is word error rate a good indicator for spoken language understanding accuracy. *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2003.

[74] Jamie A. Ward, Nagendra Bharatula, Gerhard Troster, and Paul Lukowicz. Continuous activity recognition in the kitchen using miniaturised sensor button. *Technical Notes*, 2002.

[75] P.H. Winston, B. Katz, T.O. Binford, and M.R. Lowry. Learning physical descriptions from functional definitions, examples, and precedents. *AAAI*, 1983.

[76] Kevin Woods, Diane Cook, Lawrence Hall, Kevin W. Bowyer, and Louise Stark. Learning membership functions in a function-based object recognition system. *Journal of Artificial Intelligence Research*, (3):187–222, 1995.

[77] D. Wyatt, M. Philipose, and T. Choudhury. Unsupervised activity recognition using automatically mined common sense. *Proceedings of AAAI-05*, pages 21–27, 2005.

[78] Angela Yao, Juergen Gall, and Luc Van Gool. A hough transform-based voting framework for action recognition. *CVPR*, 2010.