TECHNIQUES FOR VISION-BASED HUMAN-COMPUTER INTERACTION

by

Jason J. Corso

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

August, 2005

Abstract

With the ubiquity of powerful, mobile computers and rapid advances in sensing and robot technologies, there exists a great potential for creating advanced, intelligent computing environments. We investigate techniques for integrating passive, vision-based sensing into such environments, which include both conventional interfaces and large-scale environments. We propose a new methodology for vision-based human-computer interaction called the Visual Interaction Cues (VICs) paradigm. VICs fundamentally relies on a shared perceptual space between the user and computer using monocular and stereoscopic video. In this space, we represent each interface component as a localized region in the image(s). By providing a clearly defined interaction locale, it is not necessary to visually track the user. Rather we model interaction as an expected stream of visual cues corresponding to a gesture. Example interaction cues are motion as when the finger moves to press a push-button, and 3D hand posture for a communicative gesture like a letter in sign language. We explore both procedurally defined parsers of the low-level visual cues and learning-based techniques from machine learning (e.g. neural networks) for the cue parsing.

Individual gestures are analogous to a language with only words and no grammar. We have constructed a high-level language model that integrates a set of low-level gestures into a single, coherent probabilistic framework. In the language model, every low-level gesture is called a gesture word. We build a probabilistic graphical model with each node being a gesture word, and use an unsupervised learning technique to train the gesture-language model. Then, a complete action is a sequence of these words through the graph and is called a gesture sentence.

We are especially interested in building mobile interactive systems in large-

scale, unknown environments. We study the associated *where am I* problem: the mobile system must be able to map the environment and localize itself in the environment using the video imagery. Under the VICs paradigm, we can solve the interaction problem using local geometry without requiring a complete metric map of the environment. Thus, we take an appearance-based approach to the image modeling, which suffices to localize the system. In our approach, *coherent regions* form the basis of the image description and are used for matching between pairs of images. A coherent region is a connected set of relatively homogeneous pixels in the image. For example, a red ball would project to a red circle in the image, or the stripes on a zebra's back would be coherent stripes. The philosophy is that coherent image regions provide a concise and stable basis for image representation: concise meaning that there is drastic reduction in the storage cost of an image, and stable meaning that the representation is robust to changes in the camera viewpoint.

We use a mixture-of-kernels modeling scheme in which each region is initialized using a scale-invariant detector (extrema of a coarsely sampled discrete Laplacian of Gaussian scale-space) and refined into a full (5-parameter) anisotropic region using a novel objective function minimized with standard continuous optimization techniques. The regions are represented using Gaussian weighting functions (kernels), which yields a concise, parametric description, permits spatially approximate matching, and permits the use of techniques from continuous optimization for matching and registration. We investigate such questions as the stability of region extraction, detection and description invariance, retrieval accuracy, and robustness to viewpoint change and occlusion.

Advisor: Professor Gregory D. Hager
Readers: Professor Gregory D. Hager (JHU) and
Professor René Vidal (JHU) and
Professor Trevor Darrell (MIT)

Acknowledgements

I would like to thank my advisor, Professor Gregory Hager, who over the past five years has played an integral role in my development as a graduate student. His guidance, enthusiasm in the field of computer vision and dedication to accuracy and concreteness have taught me innumerable lessons. I would also like to offer special thanks to Professor René Vidal and Professor Trevor Darrell for their roles on my thesis committee.

I would like to thank Professor Jon Cohen, who mentored me on a qualifying project about out-of-core rendering of large unstructured grids. I would also like to thank Dr. Subodh Kumar, who welcomed me into the graphics lab during my early days at Hopkins, taught me many things about geometry and graphics, and sat on my GBO. I am grateful to Professor Paul Smolensky, who introduced me to the interesting field of cognitive science and sat on my GBO. My sincerest gratitude goes to Professor Allison Okamura for guiding me on an interesting haptics project about rendering deformable surfaces and for sitting on my GBO. My thanks also goes to Professor Noah Cowan for sitting on my GBO.

My deepest gratitude goes to Dr. Jatin Chhugani who has shared his interest of graphics, haptics, and general computer science with me through many interesting chats, as both a friend and a mentor. Thanks to Budirijanto Purnomo and the other members of the Hopkins Graphics Lab where I spent a couple of years learning about the field of computer graphics.

I am very grateful to have joined the Computational Interaction and Robotics Lab. Here, I have had the pleasure of many interesting people. With my fellow VICs project members, Dr. Darius Burschka and Guangqi Ye, I was able to explore human-

computer interaction, which has interested me since I first touched a computer as a child. I thank Maneesh Dewan for his willingness to dive into deep discussion about vision and mathematics and for sharing a desk these past two years. I also thank Le Lu, the lab "librarian," for directing me to the *right* paper time and time again; his breadth of knowledge in vision has been a wonderful aid during my research. My thanks also goes to the other CIRL members: Xiangtian Dai, Henry Lin, Sharmi Seshamani, Nick Ramey, William Lau, and Stephen Lee. I am also grateful to have been brief member of the vision reading group with Professor Vidal's students.

I have had the pleasure of making many great friends through Johns Hopkins. I am especially grateful to the members of the Tolstoy Reading Group for our marvelous discussions and evenings. Among others, I thank Andy Lamora, for managing to get the department to purchase a foosball table, Ofri Sadowsky, the best poker player I know, and Nim Marayong, who shares my love of sweet candies.

My thanks to Chris Jengo and Dr. Jon Dykstra at Earth Satellite Corporation, and to Dr. Yakup Genc and Dr. Nassir Navab at Siemens Corporate Research for giving my the opportunity to experience a pair of rewarding summer internships in my undergraduate and graduate years. I am especially grateful to Sernam Lim, whom I met at Siemens Corporate Research, for sharing his great interest in vision and image analysis with me.

I am very lucky to have attended a small liberal arts college. The diverse classes I took did indeed pave the way for my graduate schooling. I am especially grateful to my undergraduate advisor and Hauber summer science research fellowship advisor, Professor Roger Eastman; during my research with him, I learned about the interesting problem of image registration. I would also like to thank Professor Arthur Delcher, Professor Keith Gallagher, and Professor David Binkley, who made the small computer science department at Loyola College a rewarding and interesting community.

My warmest thanks goes to Lynn Johnson: through her art classes at Chaminade High School, Thursday afternoon "OOF" sessions, and our many interesting conversations, she unintentionally laid the foundation for my interest in computer vision by opening my eyes to the world of visual things.

Finally, I would like to thank my wife, Aileen; my family, Jill and Jesse, Lois, and Joe; and my many friends for their omnipresent love, support, encouragement, and company during these years. I am indeed a lucky man.

Jason Corso
Baltimore,
18 August 2005.

Contents

List of Figures

List of Tables

To my grandparents,
whose lives in a new world
laid the foundation for my studies.
Florence Zitelli and Joseph M. Corso, Sr.
Stephanis Hagemaier and Vernon DeMeo

Chapter 1

Introduction

The disparity between the digital and the physical is shrinking rapidly. We hold more computing power in our pockets today than on our desks a decade ago. However, Moore's Law is not dictating the development of computing's every aspect. The practice of interacting with a computing environment has changed little since the inception of the graphical user interface (GUI) in the early 1980s [81]. The dominant interaction model[1] governing today's interfaces is Shneiderman's *direct manipulation* model [157]. In this context, direct manipulation describes the user's ability to effect immediate changes in the computer-state by directly interacting with the application objects through the keyboard and mouse. This is in contrast to earlier generations of interfaces that required the user to pre-program the whole session or learn a complex procedural command-language. Through this model, the language of interaction evolved from such complex command languages to rapid, sequential, reversible, direct, and intuitive actions. The direct manipulation model comprises four principles:

1. Continuous representation of the objects of interest.
2. Physical actions (movement and selection by mouse, joystick, touch screen, etc.) or labeled button presses instead of complex syntax.
3. Rapid, incremental, reversible operations whose impact on the object of interest is immediately visible.

[1] An interaction model [11] is a set of principles, rules, and properties that guide the design of an interface.

4. Layered or spiral approach to learning that permits usage with minimal knowledge. Novices can learn a model and useful set of commands, which they can exercise till they become an "expert" at level 1 of the system. After obtaining reinforcing feedback from successful operation, users can gracefully expand their knowledge of features and gain fluency.

The *direct* interaction model brought proficiency with the user interface to a broad spectrum of users. The model gave rise to the current generation of computer interfaces: the "Windows, Icons, Menus and Pointers" (WIMP) generation [171]. It is this style of interface to which we have become accustomed. Given WIMP's standardization [114] and its ubiquity, the so-called *desktop* metaphor clearly is the cornerstone of contemporary human-computer interaction (HCI).

With increasing computing power and many new technologies at the disposal of interface engineers, we are beginning to investigate the next generation of interfaces. Van Dam [171] writes, "A post-WIMP interface is one containing at least one interaction technique not dependent on classical 2D widgets such as menus and icons. Ultimately it will involve all senses in parallel, natural language communication and multiple users." We add two points to this statement: first, we expect the interaction to evolve into a duplex learning process on a per-user basis, for interaction is, essentially, a means of communication between the human and the computer. Second, humans are highly adaptable. They bring a vast amount of domain knowledge from everyday real-world activities. In an ideal situation, they would be able to directly apply such domain knowledge to interacting with the computer system.

A key enabling technology for the post-WIMP interface is computer vision: giving the computer the ability to see its surroundings and to interpret them. Computer vision holds great promise for effecting a natural and intuitive communication between human and machine. Being a passive input mechanism, vision can seamlessly integrate with many traditional environments ranging from the conference room to the factory floor. And inexpensive desktop cameras with sufficient resolution and speed are nearly already commonplace. Rich video information would allow a powerful, yet adaptable and intuitive language of interaction to be developed. Humans would be able to interact in a natural manner without the encumbrance of interaction devices.

However, humans are difficult to *understand*: even in human-human interaction miscommunications often occur. Such miscommunications can arise from three types of problems:

1. Physical Problem. Either party in the communication or the medium has a physical problem in the communication: for example, the speaker may have a speech impediment, the listener may have a hearing disability, or there may be noise over the communication channel (in the case of a cellular phone conversation, for instance).

2. Language Problem. It is possible that the two parties do not speak the same language or they speak different dialects of the same language.

3. Comprehension Problem. Even when the two communicating parties speak the same language, they may experience a comprehension problem during the communication. One potential cause is the parties have entered the conversation from different contexts and, as a result, are talking about completely different things. Another potential cause is a difference in the relative education level between the two parties; one may be speaking metaphorically with the other party interpreting the speech literally.

From the perspective of computer vision, we find the same *miscommunications* arising. There may be physical communication problems for the computer vision algorithms to correctly interpret the input video because humans require articulated modeling and exhibit highly complex spatio-temporal dynamics [1, 57, 135]. Second, learning the interaction language of an interface has proven to be a complex task. Even with current interfaces where the burden rests almost exclusively with the human, there is a steep learning curve for the average user. As stated earlier, a goal of post-WIMP interface development is a duplex learning process between the human and the computer; such a goal introduces complex requirements on the computer algorithms. Third, as we find comprehension problems between expert human-human communicators, similar problems are inevitable in HCI. In the dissertation, we study these problems in the context of vision-based human computer interaction.

1.1 Motivation

We are interested in the general problem of human-computer interaction and how computer vision techniques can be applied. Concretely, we state the vision-based human computer interaction (VBI) problem as follows:

> How does one efficiently model and *parse* a high dimensional video stream to maximize activity recognition reliability, interaction vocabulary and system usability?

In the dissertation, we are motivated by two specific aspects of this broad problem. First, we would like to remove the restrictive mediation through interaction devices like the mouse and replace it with a more natural communication between the human and machine. Second, we are interested in large-scale, multi-user, shared workspaces. We explain these motivations in more detail in the remainder of this section.

1.1.1 Interaction as Communication

The majority of computer users spend all of their computing time with standard 2D interfaces. The interaction with the computer is mediated through the mouse and keyboard, and, as such, is typically restricted to one user. Likewise, the interaction is one-way in the sense that the user must learn how to interact with the system; the system is not *adapting* to the user.

Incorporating computer vision into the system allows for new avenues of interaction techniques. For example, computer vision permits unencumbered interaction with freehand motion. Consider a jigsaw puzzle application. Using standard interaction techniques, a cumbersome language of interaction would be present: the user would have to click on one puzzle piece at a time, drag it to the proper placement, switch the *control* mode to rotation, rotate the piece, and then release the piece. The mode switching between translation and rotation is due to the restrictive click-and-drag nature of WIMP interfaces. Perhaps the application would also let the user select multiple pieces at one time. However, if we consider a vision enhanced scenario, the user would be able to use more natural and efficient actions. For example, the system could recognize both the translational and rotational motion at the same time, or the user could use both hands in tandem to move multiple separate sets of pieces at the same time.

1.1.2 Large-Scale Interaction

The possibility for constructing large-scale, smart environments for multiple users to share, explore and manipulate is rich. We are interested in dynamic man-machine interfaces in the context of mobile augmented computing. In mobile augmented computing, the user is carrying a mobile augmented computing system (MACS) (e.g. laptop, display glasses, cameras, etc.) that makes it possible to composite virtual objects into his or her visual pathway. For example, if a user is in a library setting with a MACS and they are searching for the shelves containing volumes from English Literature, then, the MACS could paint an arrow pointing them in the proper direction. When the shelf comes into view, the arrow might change into an information panel highlighting the contents or allowing a further search for a particular volume.

Typical state-of-the-art applications [46] in mobile augmented computing are *fixed* in the sense that the interface is constructed beforehand and is unchanging. There are two major implications in such fixed settings. First, an electronic map of the environment must be acquired prior to the interface development and usage. Second, the user has no ability to dynamically modify the environment by adding new virtual objects, manipulating current objects, or removing current objects.

In contrast, we are interested in allowing the user to dynamically modify the augmented environment to suit his or her needs. Returning to the augmented library example, we offer a case where dynamic manipulation can aid the user: at the English Literature shelf, the user needs to reference his or her (digital) notebook. Thus, he or she dynamically attaches a *virtual display* of the notebook to a vacant shelf nearby and can now interact with it.

Another application is an augmented, multi-user, shared workspace. In this setting, multiple users are capable of manipulating the virtual (and real) objects in the shared space. Recalling the single user example from the previous section, assembling the puzzle in collaboration is a candidate application. Another possible usage of the shared space is the organization of a set of virtual index cards scattered on a table. Each user has a view of the cards from his or her viewpoint and yet, can manipulate them thus affecting the shared interface. Alternatively, a shared workspace focused on education may enable teachers and students to explore new pedagogical techniques permitting better learning and retention rates.

There are a set of problems the MACS must be able to solve in order to function. The most important one is the *where am I* problem: i.e. the computing system must be able to localize the position of the user in the environment. Associated with the *where am I* is a rendering problem. Given a relative position in the environment, there are a set of visible virtual objects that must be rendered into the user's view. To that end, for each of the virtual objects, the system must be able to determine its location relative to the user and if it is visible from the user's viewpoint. The third problem is an information management and system integration problem arising in the case of multiple users collaborating in the shared space.

From this set, we have focused on the *where am I* problem. We leave the remaining problems for future work. We assume no prior knowledge of the scene structure. This is an important assumption that greatly increases the complexity of the problem, for a *fixed* MACS could not exist without any prior environment information. We also restrict our study to *passive* sensing. A passive input mechanism is any such input device that does not actively disturb the environment. For instance, placing infrared beacons or using a magnetic tracker are impermissible. This constraint is plausible considering that for some potential mobile augmented computing applications (an art gallery, for instance) actively disturbing the environment is not readily permitted.

Following Marr's paradigm [100], the immediate solution one attempts is to build a complete reconstruction of the scene, for it will enable later queries and localization. Indeed, many researchers have attempted such techniques (Section 1.4.5) with varying degrees of success. However, we propose an alternative strategy that makes no attempt to perform a full scene reconstruction. Instead, we argue that maintaining the relative coordination of a small set of *special* surfaces (or volumes) in the scene is sufficient for solving the localization problem. In the fifth chapter, we will discuss a novel approach at detecting and characterizing such surfaces.

1.2 Thesis Statement

Natural and intuitive human-computer interaction with robust recognition is possible in large-scale, unknown environments through the application of computer vision techniques without globally tracking and modeling the user.

1.3 Overview

In this section, we present an overview of the dissertation by introducing the three contributions. In the subsequent chapter conclusions, we concretely state the contributions in light of the current state-of-the-art.

1.3.1 Contribution 1: Novel Methodology for Applying Computer Vision to Human-Computer Interaction

We take a general approach to incorporating vision into the human-computer interaction problem that is applicable for both 2D and 3D interfaces. A brief survey of the literature (Section 1.4.4) reveals that most reported work on VBI relies heavily on visual tracking and visual template recognition algorithms as its core technology. While tracking and recognition are, in some sense, the most popular direction for developing advanced vision-based interfaces, one might ask if they are either necessary or sufficient. Take, for example, the real-world situation where a person dials a telephone number. When he or she presses the keys on the telephone, it (or the *world*), maintains no notion of the user. Instead, the telephone only recognizes the result of a key on the keypad being pressed. In contrast, typical methods for VBI would attempt to construct a model of the user's finger, track it through space, and perform some action recognition as the user pressed the keys on the telephone.

In Chapter 2, we present a new and effective methodology for VBI which is based on the claim that global user tracking and modeling is generally unnecessary. We term our approach the Visual Interaction Cues Paradigm or VICs. VICs fundamentally relies on a shared perceptual space between the user and computer using monocular and stereoscopic video. In this space, we represent each interface component as a localized region in the image(s). By providing a clearly defined interaction locale, it is not necessary to visually track the user. Rather we model interaction as an expected stream of visual cues corresponding to a gesture. Example interaction cues are motion as when the finger moves to press a push-button and 3D hand posture for a communicative gesture like a letter in sign language. We explore both procedurally defined parsers of the low-level visual cues and learning-based techniques from machine learning (e.g. neural networks) for the cue parsing.

In the VICs project, we have implemented our methods in real-time interactive systems. In the 4D Touchpad project, we demonstrate an alternative desktop interface

based on the VICs interface model. On this platform the user can use natural gestures to perform common tasks on the interface like pressing buttons and scrolling windows.

In summary, the main contribution is the novel approach we take to modeling user interaction that does not use global tracking methods. Instead, we model the spatio-temporal signature of the gesture in local image regions to perform recognition. An in-depth analysis of this approach and comparison to the state-of-the-art in VBI is discussed in Section 2.7.

1.3.2 Contribution 2: Unified Gesture Language Model Including Different Gesture Types

As motivated earlier, we consider human-computer interaction as communication. In this communication, gestures[2] are part of the low-level vocabulary. They can be classified into three types:

1. Static postures [3, 98, 116, 129, 168, 185] model the gesture as a single key frame, thus discarding any dynamic characteristics. For example, in recent research on American Sign Language (ASL) [160, 197], static hand configuration is the only cue used to recognize a subset of the ASL consisting of alphabetical letters and numerical digits. The advantage of this approach is the efficiency of recognizing those gestures that display explicit static spatial configuration.

2. Dynamic gestures contain both spatial and temporal characteristics, thus providing more challenges for modeling. Many models have been proposed to characterize the temporal structure of dynamic gestures: including temporal template matching [17, 104, 121, 156], rule-based and state-based approaches [18, 129], hidden Markov models (HMM) [130, 160, 187, 189] and its variations [20, 118, 181], and Bayesian networks [155]. These models combine spatial and temporal cues to infer gestures that span a stochastic trajectory in a high-dimensional spatio-temporal space.

3. Parametric, dynamic gestures carry quantitative information like angle of a pointing finger or speed of waving arm. Most current systems model dynamic gestures qualitatively. That is, they represent the identity of the gesture, but they do not incorporate

[2] In our development, we restrict our focus to hand gestures.

any quantitative, parametric information about the geometry or dynamics of the motion involved. However, to cover all possible manipulative actions in the interaction language, we include parametric gestures. One example of this type of gesture modeling is the parametric HMM (PHMM) [181]. The PHMM includes a global parameter that carries an extra quantitative representation of each gesture.

Individual gestures are analogous to a language with only words and no grammar. To enable a natural and intuitive communication between the human and the computer, we have constructed a high-level language model that integrates the different low-level gestures into a single, coherent probabilistic framework. In the language model, every low-level gesture is called a *gesture word*. We build a forward graphical model with each node being a gesture word, and use an unsupervised learning technique to train the gesture language model. Then, a complete action is a sequence of these words through the graph and is called a *gesture sentence*. To the best of our knowledge, this is the first model to include the three different low-level gesture types into a unified model.

1.3.3 Contribution 3: Coherent Region-Based Image Modeling Scheme

Images are ambiguous. They are the result of complex physical and stochastic processes and have a very high dimensionality. The main task in computer vision is to use the images to infer properties of these underlying processes. The complex statistics, high image dimensionality, and large solution space make the inference problem difficult. In Chapter 5, we approach this inference problem in a maximum *a posteriori* (MAP) framework that uses *coherent* regions to summarize image content. A coherent region is a connected set of relatively homogeneous pixels in the image. For example, a red ball would project to a red circle in the image, or the stripes on a zebra's back would be coherent vertical stripes. The philosophy behind this work is that coherent image regions provide a concise and stable basis for image representation: concise meaning that there is drastic reduction in the storage required by the representation when compared to the original image and other modeling methods, and stable meaning that the representation is robust to changes in the camera viewpoint.

There are two parts in our approach. First, we propose to project the image into a new basis that emphasizes various coherency characteristics. Conceptually, each projection defines a feature space where a particular image character (e.g. red-ness, stripy-

ness) will project to a homogeneous region. An example of such a projection is a linear combination of pixel-color intensities to measure the red-ness of a pixel. Another example is the neighborhood variance, which is a coarse measure of texture. In the current work, these projections are defined using heuristics and have unrestricted form (linear, non-linear, etc). Since the projections form the basis of the image description, their invariance properties will be inherited by such a description.

Second, we propose an algorithm that gives a local, approximate solution to the MAP image modeling in the scalar projections (and thus, the original image given the new basis). We use a mixture-of-kernels modeling scheme in which each region is initialized using a scale-invariant detector (extrema of a coarsely sampled discrete Laplacian of Gaussian scale-space) and refined into a full (5-parameter) anisotropic region using a novel objective function minimized with standard continuous optimization techniques. The regions are represented using Gaussian weighting functions (kernels) yielding a concise parametric description and permitting spatially approximate matching.

To be concrete, the main contribution we make in this part of the dissertation is an interest region operator that extracts large regions of homogeneous character and represents them with full five-parameter Gaussian kernels in an anisotropic scale-space. We investigate such issues as the stability of region extraction, invariance properties of detection and description, and robustness to viewpoint change and occlusion. This approach to image summarization and matching lays the foundation for a solution to the mapping and localization problems discussed earlier.

1.4 Related Work

In this section, we survey the work related to the complete dissertation. In future chapters, we include any literature that is related exclusively to the content of the chapter.

1.4.1 2D Interfaces

We make the assumption that the reader is familiar with conventional WIMP-based 2D interfaces and do not elaborate on them in our discussion. We do note a slight extension of the desktop metaphor in the Rooms project [70]. They perform a statistical analysis of the window access which is used to separate various tasks into easily accessible *rooms* (workspaces).

There are also modern non-WIMP 2D interfaces. Most are augmented desk style interfaces. The most exemplary of these is Wellner's DigitalDesk [180] which attempts to fuse a real desktop environment with a computer workstation by projecting the digital display on top of the desk which may be covered with real papers, pens, etc. The Enhanced-Desk [89], influenced by Wellner's DigitalDesk, provides an infrastructure for applications developed in an augmented desktop environment. Another example of an unconventional 2D interface is the Pad by Perlin and Fox [126] which uses an infinite 2D plane as the interface and allows users to employ their learned spatial cognition abilities to navigate the information space.

1.4.2 3D Interfaces

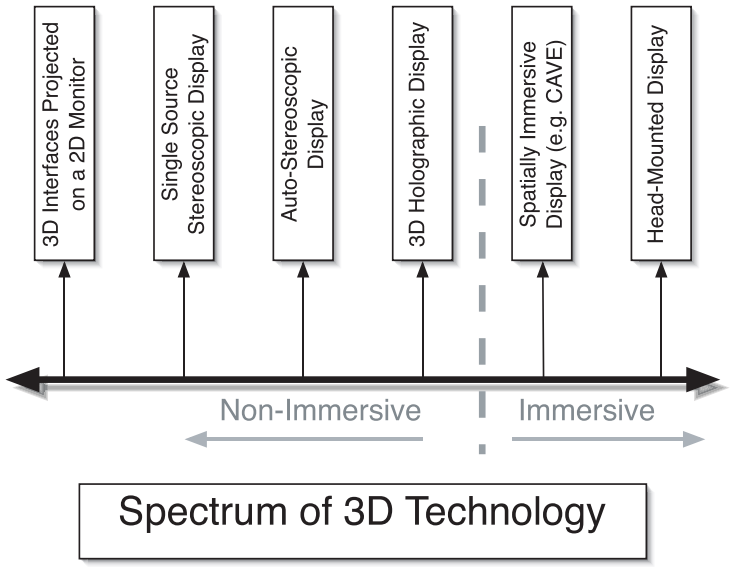


Figure 1.1: The spectrum of 3D interface technology ranging from 2D projections to fully immersive 3D.

The spectrum of current 3D interfaces ranges from those on a standard 2D monitor to fully immersive 3D interfaces. Figure 1.1 graphically depicts this spectrum. Starting with the left side of the spectrum, we see 3D interfaces that are projected onto a standard 2D screen [52]. This is the simplest 3D rendition and, among other uses, has been used extensively by the gaming industry and scientific visualization. In this case, the display is restrictive and may have a tendency toward ambiguity as the user has little sense of

presence. However, the technology required to generate such an interface is standard on modern PCs and laptops.

One step further is the single-source stereoscopic style display through a single monitor [147]. Most often, these come in the form of a pair of shutter glasses synchronized with the display which is rendering a sequence of left-right-left-etc. images. However, recently there has been an auto-stereoscopic display which exploits the natural operation of the human visual system to perform the depth-vergence [44]. The next type of 3D interface is the holographic style display in which the user is positioned in front of a display and a 3D visualization appears in front of their view [14, 27].

While the previous types of displays yield attractive visualization solutions, their output is quite different than those in the other half of the spectrum. The second half of the spectrum contains immersive displays. In such systems, it is typical that the user can actively navigate through the space and the virtual (or augmented) environment replaces (or enhances) the real world. The first class of these immersive displays is based on the same principle of stereoscopy as mentioned above. They are called *spatially immersive displays* (SID) as the user is placed inside a space of tiled projection screens. In the case of a full-cube, they are termed CAVEs and have found widespread use in a variety of immersive applications [37, 151].

The Office of the Future project uses a SID to create a distance-shared workspace [133]. They use techniques from computer vision and computer graphics to extract and track geometric and appearance properties of the environment to fuse real and virtual objects. While powerful, the use of these systems has been restricted to industrial grade applications because of the display cost and size. However, more recently, a cost effective solution has been provided which may increase the use of SIDs [142].

At the end of the spectrum on the right side is the Head-Mounted Display (HMD) [162]. Whether the task is augmented reality or virtual reality, HMDs offer the greatest sense of immersion to the user as the interface itself maintains a user-centric viewpoint with an ability to immediately localize oneself based on the environment. There are two types of basic HMDs: closed and half-silvered see-through [4, 5]. A closed HMD enables both virtual reality and video see-through augmented reality. In the video see-through augmented reality case there are cameras mounted on the closed HMD. The images rendered to the screens in the HMD are the composition of the real imagery being captured by the cameras and synthetic imagery generated by the computer. By contrast, in optical see-through augmented

reality, the user can see the real world through the half-silvered HMD while the computer is also rendering synthetic objects into view [7, 23, 47]; the half-silvered see-through HMD is used exclusively for augmented reality.

1.4.3 Ubiquitous Computing Environments

Ubiquitous computing is a well-studied topic [177]. The Office of the Future project presented a set of techniques to allow distant collaboration between members of a work-group [133]. Through complete control over the parameters of lighting and display, a spatially immersive display is used to allow shared telepresence and telecollaboration. The SmartOffice project focused on developing enhanced interaction techniques that can anticipate user intention [92]. Pinhanez et al. [128] developed a projector-mirror display device that permits the projection of a dynamic interface onto any arbitrary surface in the room.

The Interactive Workspaces Project at Stanford aims at creating a general framework for the design and development of a dedicated digital workspace in which there is an abundance of advanced display technology and the interface software allows users to seamlessly interact in the shared workspace [80]. The XWeb system is a communication system based on the WWW protocols that allows seamless integration of new input modalities into an interface [120]. iStuff is a user interface toolkit for the development of shared workspace style environments [8]. It includes a set of hardware devices and accompanying software that were designed to permit the exploration of novel interaction techniques in the post-desktop era of computing. The toolkit includes a dynamic mapping-intermediary to map the input devices to applications and can be updated in real-time.

Tangibility is a key factor in the naturalness of an interface be it 2D or 3D. Ishii introduced the *Tangible Bits* [76] theory to better bridge the gap between the real and the virtual in augmented environments. In his Tangible Bits theory, various real-world objects will double as avatars of digital information. The theory moves the focus away from a window into the virtual world to our everyday physical world. Many researchers have studied various methods of adding tangibility and graspability into the user interface [49, 67, 136, 146]. One system of particular interest is the Virtual Round Table wherein arbitrary real-world objects are used to proxy for virtual buildings in a landscape program [23]. The proxy object can be picked up by the users and they are tracked in real-time to allow for

quick manipulation of their virtual counterparts.

1.4.4 Vision for Human-Computer Interaction

Using computer vision in human-computer interaction systems has become a popular approach to enhance current interfaces. As discussed earlier, the majority of techniques that use vision rely on global user tracking and modeling. In this section, we provide example works from the field and cluster them into three parts: full-body motion, head and face motion, and hand and arm motion.

Full Body Motion

The Pfinder system [183] and related applications [97] is a commonly cited example of a vision-based interface. Pfinder uses a statistically-based segmentation technique to detect and track a human user as a set of connected "blobs." A variety of filtering and estimation algorithms use the information from these blobs to produce a running state estimate of body configuration and motion [184, 125]. Most applications make use of body motion estimates to animate a character or allow a user to interact with virtual objects. [21] presented a visual motion estimation technique to recover articulated human body configurations which is the product of exponential maps and twist motions. [58] use a skeleton-based model of the 3D human body pose with 17 degrees-of-freedom and a variation of dynamic-time warping [113] for the recognition of movement.

Head and Face Motion

Basu et al. [9] proposed an algorithm for robust, full 3D tracking of the head using model-regularized optical flow estimation. Bradski [19] developed an extension of the mean-shift algorithm that continuously adapts to the dynamically changing color probability distributions involved in face tracking. He applies the tracking algorithm in explorative tasks for computer interfaces. Gorodnichy et al. [61] developed an algorithm to track the face (the nose) and map its motion to the cursor. They have successfully applied their techniques in multiple HCI settings.

Black and Yacoob [16] presented a technique for recognizing facial expressions based on a coupling of global rigid motion information with local non-rigid features. The

local features are tracked with parametric motion models. The model gives a 90% recognition rate on a data-set of 40 subjects.

Hand and Arm Motion

Modeling the dynamic human hand is a very complex problem. It is highly articulated object that requires as many as 27 degrees-of-freedom for complete modeling [135]. Pavlovic et al. [123] review recognition of hand-gestures splitting the techniques into 3D model-based approaches and 2D image-based techniques. Goncalves et al. [60] take a model-based approach to tracking the human arm in 3D without any behavioral constraints or markers. Segen and Kumar [152, 153] also use a model-based approach in the "GestureVR" system to perform fast gesture recognition in 3D.

Cui and Weng propose a set of techniques for recognizing the hand posture in communicative gestures. They model hand gestures as three-stage processes [38]: (1) temporally normalized sequence acquisition, (2) segmentation [39], and (3) recognition. In [40], they use multiclass, multi-dimensional linear discriminant analysis and show it outperforms nearest neighbor classification in the eigen-subspace.

Kjeldsen and Kender [86] use hand-tracking to mirror the cursor input. They show that a non-linear motion model of the cursor is required to smooth the camera input to facilitate comfortable user-interaction with on-screen objects. Hardenberg and Berard [175] have developed a simple, real-time finger-finding, tracking, and hand posture recognition algorithm and incorporated it into perceptual user interface settings. Wilson and Oliver [182] use 3D hand and arm motion to control a standard WIMP system.

1.4.5 Euclidean Mapping and Reconstruction

In this section, we survey the related work in metric scene mapping in unknown environments. Considering the scene is unknown *a priori*, an immediate approach is to phrase the problem as one of constructing a Euclidean map on-line. It is equivalent to the general problem of scene acquisition. This approach is common in the fields of mobile robotics and computer vision because its solution facilitates efficient answers to queries of obstacle avoidance, localization, and navigation. However, we claim that such an approach attempts to provide more information than is needed for large-scale VBI, and the inherent difficulty in the global Euclidean mapping problem renders it implausible in our case.

Specifically, global Euclidean reconstruction lends itself well to situation-specific solutions based on active sensing in the environment; for example, placing coded, infrared beacons at calibrated locations in the environment greatly simplifies the pose problem and thus, provides an aid to the reconstruction problem.

The problem of scene reconstruction is well-studied in the computer vision literature. We briefly survey techniques based on a depth map representation and not structure from motion [10, 164]. Most techniques in the literature separate the imaging process from the reconstruction process; they assume as input a depth map. Slambaugh et al. [159] provide a comprehensive survey of volumetric techniques for the reconstruction of visual scenes. They divide the previous volumetric techniques into three categories: volumetric visual hulls which use geometric space carving, voxel color methods which use color consistency, and volumetric stereo vision techniques which fit a level set surface to the depth values in a voxel grid. We refer to [159] for a more detailed discussion of these techniques. Other volumetric methods using ICP [112, 140] and global graph-cut optimization [122] have been proposed more recently.

Alternative to volumetric representations of the reconstructed scene, methods using surface descriptions have been well-studied [6, 72, 131]. A variant of the surface based techniques employs adaptive meshes to compute a surface description of the acquired object. Terzopoulos and Vaseliscu [167] developed a technique based on adaptive meshes, dynamic models which are assembled by interconnecting nodal masses with adjustable springs, that non-uniformly sample and reconstruct intensity and range data. The nodal springs automatically adjust their stiffness to distribute the degrees-of-freedom of the model based on the complexity of the input data. Chen and Medioni [25] developed a technique based on a dynamic balloon modeled as an adaptive mesh. The balloon model is driven by an inflationary force toward the object (from the inside). The balloon model inflates until each node of the mesh is anchored on the object surface (the inter-node spring tension causes the resulting surface to be smooth).

Work by Fua [51], motivated by [165], builds a set of oriented particles uniformly dispersed in reconstruction space. From this initial reconstruction, it refines the surface description by minimizing an objective function (on the surface smoothness and grayscale correlation in the projection). The output is a set of segmented, reconstructed 3D objects.

1.5 Notation

Denote the Euclidean space of dimension n by $\Re^n$ and the projective space of dimension n by $\Pp^n$. Let the image $\mathbf{I} \doteq \{\mathcal{I}, I, t\}$ be a finite set of pixel locations $\mathcal{I}$ (points in $\Re^2$) together with a map $I : \mathcal{I} \to \mathcal{X}$, where $\mathcal{X}$ is some arbitrary value space, and t is a time parameter. Thus, for our purposes the *image* is any scalar or vector field: a simple grayscale image, an YUV color image, a disparity map, a texture-filtered image, or any combination thereof. The image band j at pixel location i is denoted $I_j(i)$. We overload this notation in the case of image sequences: define $\mathcal{S} = \{\mathbf{I}_1 \dots \mathbf{I}_m\}$ to be a sequence of images with length $m \geq 1$. While the distinction should be clear from the context, we make it explicit whenever there is ambiguity.

1.6 Relevant Publications

This dissertation is based on the following publications:

1. G. Ye, J. Corso, D. Burschka, and G. Hager. VICs: A Modular Vision-Based HCI Framework. In *Proceedings of 3rd International Conference on Computer Vision Systems (ICVS)*, April 2003. Pages 257–267.

2. J. Corso, D. Burschka, and G. Hager. The 4D Touchpad: Unencumbered HCI With VICs. 1st IEEE Workshop on Computer Vision and Pattern Recognition for Human Computer Interaction, CVPRHCI. June 2003.

3. G. Ye, J. Corso and G. Hager. Gesture Recognition Using 3D Appearance and Motion Features. 2005. (Extended version of the paper by the same title in *Proceedings of Workshop on Real-time Vision for Human-Computer Interaction at CVPR 2004*).

4. J. Corso. Vision-Based Techniques for Dynamic, Collaborative Mixed-Realities. In *Research Papers of the Link Foundation Fellows.* Volume 4. Ed. Brian J. Thompson. University of Rochester Press, 2004. (Invited report).

5. G. Ye, J. Corso, D. Burschka, and G. Hager. VICs: A Modular HCI Framework Using Spatio-temporal Dynamics. *Machine Vision and Applications*, 16(1):13-20, 2004.

6. D. Burschka, G. Ye, J. Corso, and G. Hager. A Practical Approach for Integrating Vision-Based Methods into Interactive 2D/3D Applications. Technical Report:

Computational Interaction and Robotics Lab, Dept. of Computer Science, The Johns Hopkins University. CIRL-TR-05-01. 2005.

7. J. Corso and G. Hager. Coherent Regions for Concise and Stable Image Description. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

8. J. Corso, G. Ye, and G. Hager. Analysis of Multi-Modal Gestures with a Coherent Probabilistic Graphical Model. *Virtual Reality*, 2005.

Chapter 2

The Visual Interaction Cues Paradigm*

Vision-based human-computer interaction is a promising approach to building more natural and intuitive interfaces. As discussed in the introductory chapter, using vision techniques could allow large-scale, unencumbered motion from multiple concurrent users. The information-rich video signals contain far more information than current interaction devices. With the additional information and the naturalness of unencumbered motion, we expect the interaction between human and computer to be far more direct, robust, and efficient.

However, using video in human-computer interaction has proved to be a difficult task. The difficulty is evident simply in the absence of vision-based interaction systems in production. As noted in Section 1.4.4, most reported work on vision-based human-computer interaction (VBI) relies heavily on visual tracking and visual template recognition algorithms as its core technology. It is well understood that visual tracking of articulated objects (humans) exhibiting complex spatio-temporal dynamics is a difficult problem [1, 57, 135].

In contrast, we present an approach that does not attempt to globally track and model the user. Our methodology, the Visual Interaction Cues paradigm (VICs), uses a shared perceptual space between the user and the computer. In the shared space, the computer is monitoring the environment for sequences of expected user activity at the

*Parts of this chapter are joint work with Prof. Dr. D. Burschka and G. Ye.

locations corresponding to interface elements.

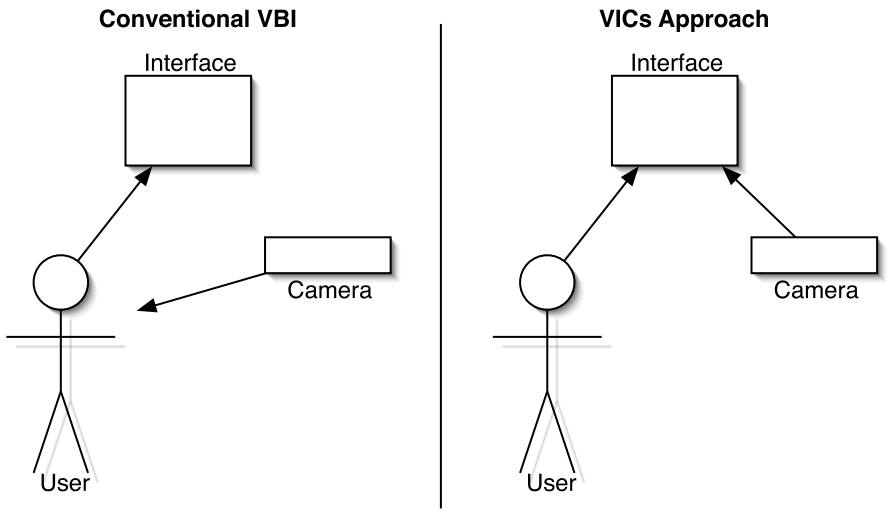


Figure 2.1: Schematic comparing conventional VBI with the VICs approach. Here, each arrow represents a direction of observation; i.e. on the left, the camera is observing the human while the human is observing the interface, and on the right, both the human and the camera are observing the interface.

In Figure 2.1, we compare the conventional, tracking-based VBI approaches with the VICs method. On the left, we find the camera monitoring the user while the user is interacting with the computer. On the right, we show the VICs approach: the camera is monitoring the interface. Approaching the VBI problem in this manner removes the need to globally track and model the user. Instead, the interaction problem is solved by modeling the stream of localized visual cues that correspond to the user interacting with various interface elements. We claim that this additional structure renders a more efficient and reliable solution to the VBI problem. In this chapter, we discuss the Visual Interaction Cues approach to the VBI problem. To the best of our knowledge, the only similar approach in the literature is the Everywhere Displays projector [128] and related software algorithms [87], which also models the interaction as a sequence of image processing primitives defined in a local image region. In their work, a special projector can render interface components at arbitrary planar locations in the environment. Each interface component has an associated tree of image processing functions that operate on a local image region in a video camera that is calibrated to the projector. The exact image processing routines used by each

interface component for gesture recognition is function specific.

2.1 The VICs Interaction Model

An interaction model [11] is a set of principles, rules, and properties that guide the design of an interface. It describes how to combine interaction techniques in a meaningful and consistent way and defines the "look and feel" of the interaction from the user's perspective.

2.1.1 Current Interaction Models

The current interface technology based on "Windows, Icons, Menus and Pointers" (WIMP) [171] is a realization of the direct manipulation interaction model.[1] In the WIMP model,[2] the user is mapped to the interface by means of a pointing device, which is a mouse in most cases. While such a simple mapping has helped novice users gain mastery of the interface, it has notable drawbacks. First, the mapping limits the number of active users to one at any given time. Second, the mapping restricts the actions a user can perform on an interface component to a relatively small set: click and drag. Figure 2.2 depicts the life-cycle of an interface component under the WIMP model. Last, because of this limited set of actions, the user is often forced to (learn and) perform a complex sequence of actions to issue some interface commands. Thus, the restrictive mapping often results in the user manipulating the interface itself instead of the application objects [11].

A number of researchers have noticed the drawbacks inherent in the WIMP model and suggested improvements [173, 172] while others have proposed alternative models [11, 71, 154]. In fact, numerous so-called post-WIMP interface systems have been presented in the literature for spoken language [79, 176], haptics [75, 141, 191, 192, 193] and vision [1, 123, 186].

One way to quantify the added benefit of using computer vision (and other information-rich modalities like speech, for example) for the interaction problem is to compare the components of the two interfaces directly. We have already presented the state machine for a standard WIMP interface component (Figure 2.2) and explained that such a simplistic

[1] The principles of the direct manipulation model [157] are listed in Chapter 1.

[2] For brevity, we will write "WIMP model" to mean the "WIMP realization of the direction interaction model."

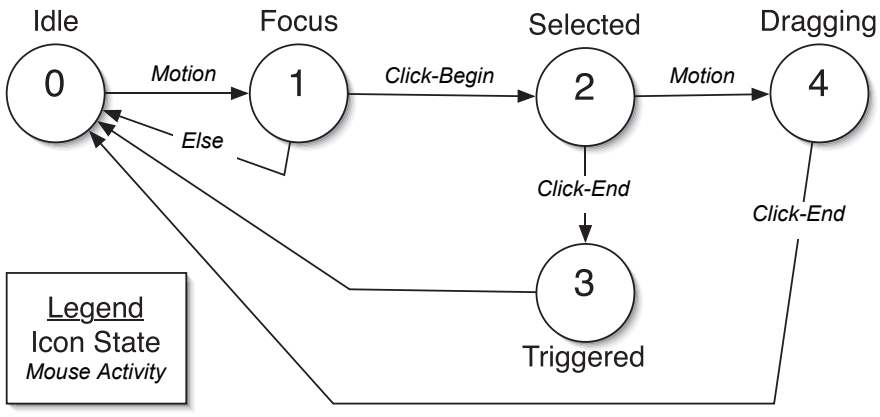


Figure 2.2: The icon state model for a WIMP interface.

scheme has led to the development of complex interaction languages. The number of actions associated with each interface component can be increased with proper use of the higher-dimensional input stream. For standard WIMP interfaces the size of this set is 1: point-and-click. We call a *super*-WIMP interface one that includes multi-button input or mouse-gesture input. One such example is the SKETCH framework [194] in which mouse gestures are interpreted as drawing primitives. For the *super*-WIMP interfaces the size of this set is larger, but still relatively small; it is limited by the coarse nature of mouse input. In general, for vision-based extensions, the number of possible user inputs can increase greatly by using the increased spatial input dimensionality. A candidate state-machine for a post-WIMP interface component is presented in Figure 2.3.

2.1.2 Principles

As noted earlier, mediating the user interaction with a pointing device greatly restricts the naturalness and intuitiveness of the interface. By using the video signals[3] as input, the need for such mediation is removed. With video input, the user is unencumbered and free to interact with the computer much in the same way they interact with objects in the real-world. The user would bring their prior real-world experience, and they could immediately apply it in the HCI setting.

We have developed a new interaction model which extends the direct interaction

[3]Our development is general in the sense that we do not constrain the number of video signals that can be used. We will write "videos" or "video signals" in the plural tense to emphasize this fact.

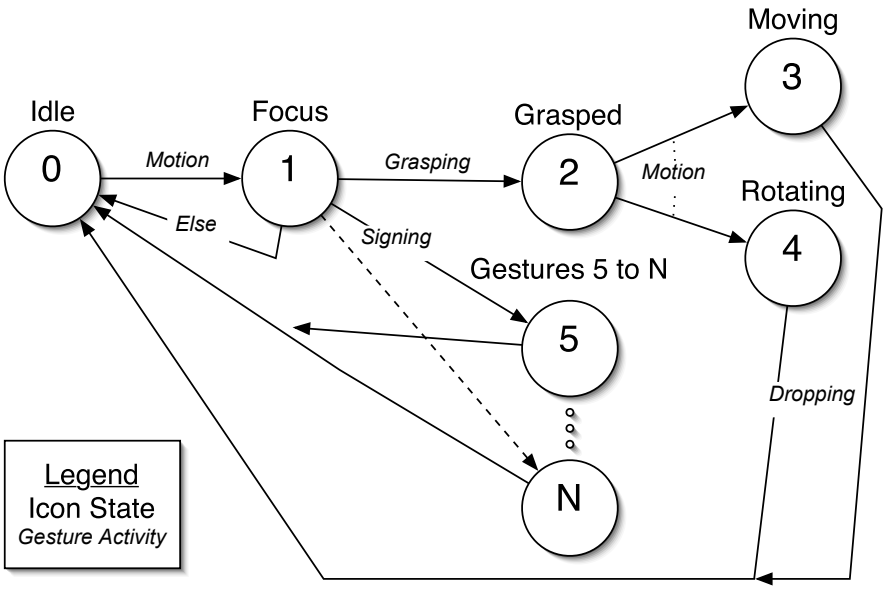


Figure 2.3: A possible post-WIMP icon state model.

model to better utilize the multi-modal nature of future interfaces. Here, we list the principles of our model:

1. There are two classes of **interface components** (Figure 2.4):

   (a) The "direct" objects (objects of interest) should be continuously viewable to the user and functionally rendered such that the interaction techniques they understand are intuitive to the observer. These objects should have a real-world counterpart, and their usage in the interface should mimic the real-world usage.

   (b) The "indirect" objects, or interface tools/components, may or may not have a real-world counterpart. These should be obvious to the user and a standard language of interaction should govern their usage. An example of such an interface tool would be grab-able tab at the corner of a window that can be used to resize the window.

2. **Sited-Interaction**: all physical[4] interaction with the system should be localized to specific areas (or volumes) in the interface to reduce the ambiguity of the user-

[4]We use the term "physical" here to describe the actions a user may perform with their physical body or with an interface device. Other interaction modalities would include speech-based interaction or even

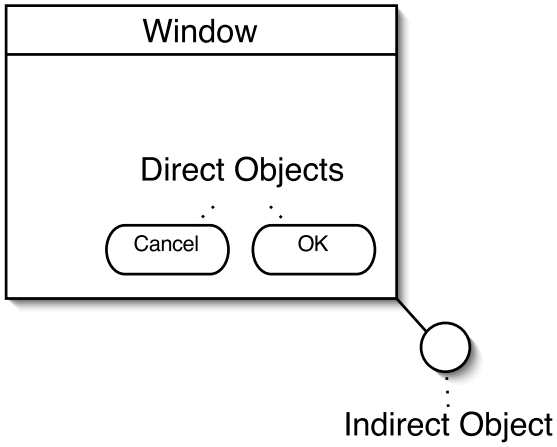


Figure 2.4: The direct interface objects in this example are buttons, which we find both in the real-world and in current computer interfaces. The indirect interface object is a small, grab-able tab at the boundary of a window. Here, the indirect object is an interface construct and not found outside of the computer system.

intention. Generally, the sited-interaction implies that all interaction is with respect to the interface elements, but it is not required to be as such.

3. **Feedback Reinforced Interaction**: since the interaction is essentially a dialog between the user and the computer system (with little or no mediation), it is necessary to supply continuous feedback to the user during the course of interactions as well as immediately thereafter.

4. The **learning** involved in using the system is separated into two distinct stages:

   (a) In the first stage, the user must learn the set of initial techniques and procedures with which to interact with the system. This initial language must be both simple and intuitive. Essentially, a new user should be able to apply their real-world experience to immediately begin using the "direct" interaction objects.

   (b) In the second stage, duplex learning will ensue where the system will adapt to the user and more complex interaction techniques can be learned by the user.

keyboard typing. Generally, the physical interaction will be of a manipulative nature while the non-physical interaction will be communicative.

The similarity to the direct manipulation model is immediately evident in the VICs interaction model. We add constraints to the model to enforce the naturalness of the interaction between the user and a vision-equipped computer system. The distinction between direct and indirect interface objects is made to avoid the aforementioned problem of the user manipulating the interface itself rather than the application objects and to simplify the learning required to use the interface. For example, a common *interface* object we find in the real world is a circular dial often used to adjust the volume in a stereo system. To use such a dial in the real-world, one would grasp it and rotate. Naturally, the user would expect the interface dial to operate in the same fashion.

2.2 The VICs Architectural Model

The architectural model describes the set of computer vision techniques we use to realize the VICs interaction model in post-WIMP interfaces and the core parts of the resulting interface. The sited-interaction principle is the basis of the architectural model. Since all physical interaction is with respect to an interface component, we use the video cameras to monitor these components. If a registration is known between the components of the interface and where they project in the video images, then the recognition problem is reduced to one of spatio-temporal pattern recognition. A gesture will appear as a sequence of visual cues in the local image regions near the components. In this section we discuss these three parts in more detail: the component mapping, the spatio-temporal pattern recognition, and the VICs interface component (VICon).

2.2.1 Interface Component Mapping

Let $\mathcal{W}$ be the space in which the components of the interface reside. In general, $\mathcal{W}$ is the 3D Euclidean space $\mathfrak{R}^3$ but it can be the Projective plane $\mathfrak{P}^2$ or the Euclidean plane $\mathfrak{R}^2$. Define an interface component mapping $M : \mathcal{C} \rightarrow \mathcal{J}$, where $\mathcal{C} \subset \mathcal{W}$ and $\mathcal{J} \doteq \{\mathcal{I} \vee A(\mathcal{I})\}$ with $\mathcal{I}$ the image pixel locations as defined in Section 1.5 and $A(\cdot)$ being an arbitrary function, $A\colon \mathfrak{R}^2 \mapsto \mathfrak{R}^2$. Intuitively, the mapping defines a region in the image to which an interface component projects (Figure 2.5).

If, for each interface component and the current image, a mapping is known, detecting a user action reduces to analyzing a local region in the image. We define the mapping in this way to enforce the sited-interaction principle.

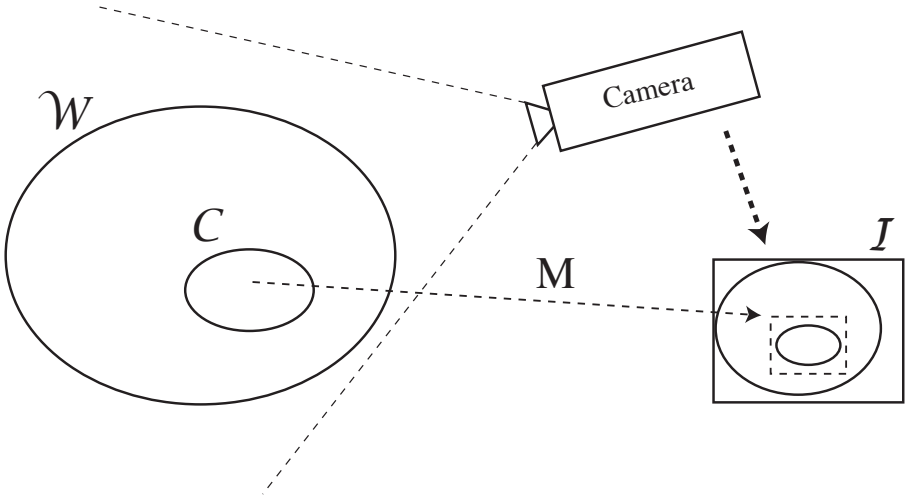


Figure 2.5: Schematic explaining the principle of local image analysis for the VICs paradigm: M is the component mapping that yields a region of interest in the image $\mathcal{I}$ for analyzing actions on component $\mathcal{C}$

2.2.2 Spatio-Temporal Pattern Recognition

The interface component mapping adds structure to the interaction problem. It serves to disambiguate the process of understanding user activity by reducing the hard problem of global user tracking and modeling to one of spatio-temporal pattern recognition. Each interface component will be capable of recognizing a function-specific set of gestures. For example, the circular dial component we introduce earlier would be required to recognize when the user grasps it, rotates, and releases it. The global user activity is irrelevant. The system is only interested in the local changes in the video stream near each of the interface components; such local changes will appear as a sequence of visual cues, or a *spatio-temporal signature*. We define a visual cue loosely as any salient event in the spatio-temporal video stream: for example, motion, color-change, or shape.

Thus, gesture recognition is solved by detecting this spatio-temporal signature in the local region surrounding an interface component. We will provide a brief introduction to this problem here and cover it in detail in the next chapter. The spatio-temporal signature of a gesture will present itself as a sequence of visual cues. We term the construct that detects these sequences of visual cues the *visual stream parser*. For example, consider a standard push-button with a known interface component mapping for a single, color camera. We break down the images of the user pushing the button into a set of discrete

stages (Figure 2.6). First, the user enters the local region. Visually, there is a notable disturbance in the appearance of the local region as it is disturbed by the finger. A simple thresholded, image-differencing algorithm would be sufficient to detect the disturbance. Second, the finger moves onto the button itself presenting itself as a large color-blob in the local region. Third, the finger actually pushes the button; from one-camera, it is impossible to spatially detect the pushing, but we can assume the pushing action has a certain, fixed duration.

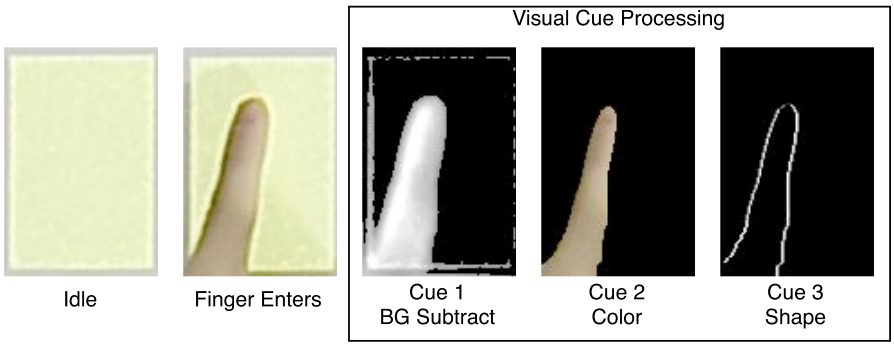


Figure 2.6: Cue parsing example: a button press can be represented as the sequence of three visual cues in the region-of-interest for an interface component. The three cues are background disturbance, color, and shape, which are detected with independent image processing modules.

One important point to note is the computational cost of the methods involved in analyzing the images. Each interface component defines a function-specific set of image processing components that are ordered in a simple-to-complex fashion such that each level of increasing interaction-detection precision (and increasing computational cost) is executed only if the previous levels have validated the likely existence of an expected object/activity in its region-of-interest. Such a notion of simple-to-complex processing is not novel; for example, in early image processing, pyramidal schemes were invented that perform coarse-to-fine analysis of images [2]. However, it is integral to the VICs paradigm.

2.3 The VICs Interface Component – VICon

We use the standard Model-View-Controller (MVC) design pattern [56] to define the VICs-based interface component. In a standard MVC design, the model contains all

logic of the application, the controller is the part of the interface with which the user can interact, and the view is the part of the interface that presents the current application state to the user. We apply this pattern in the following manner:

Model – The model contains the visual processing engine and all associated internal logic. Multiple visual cue parsing algorithms may exist for similar user-actions, but each may operate optimally under different circumstances: for example, in the presence of abundant lighting, a button-press parser may be defined as motion, color-blob, shape verification. Yet, if the lighting is inadequate, a second button-press parser may be defined as motion, coarse shape (edges), and shape verification. Thus, these two parsers provide similar functionality under different conditions thereby increasing the overall robustness of the system.

View – The VICon has the ability to continuously display itself to the user. The view will also provide all feedback reinforcement during an interaction session.

Controller – The VICons are function-specific, and the controller for each VICon comprises the set of gestures to which the VICon will respond.

In addition to these parts, the VICon also contains a set of application **hooks** which facilitate communication between the VICon, the application, and the system.

From an architectural standpoint, the two types of interface objects (direct and indirect) are equivalent. It is in their implementation and incorporation into the applications where the differences are evident. In general, such a vision-based interface component will have multiple exit conditions and a more complex state model (Figure 2.3) than the WIMP model (Figure 2.2). We defer a discussion on the gesture modeling to Chapter 3.

It should be clear that the presented architectural model can be implemented for both conventional 2D interfaces and future 2D/3D interfaces. In the next section we present our system implementation of the interaction and architectural models.

2.4 VICs System Implementation

In this section, we discuss the system implementation of the methodology at a high-level. The system architecture has a stratified design to partition component responsibilities and allow the integration of VICs into both current and future interfaces.

2.4.1 A Stratified Design

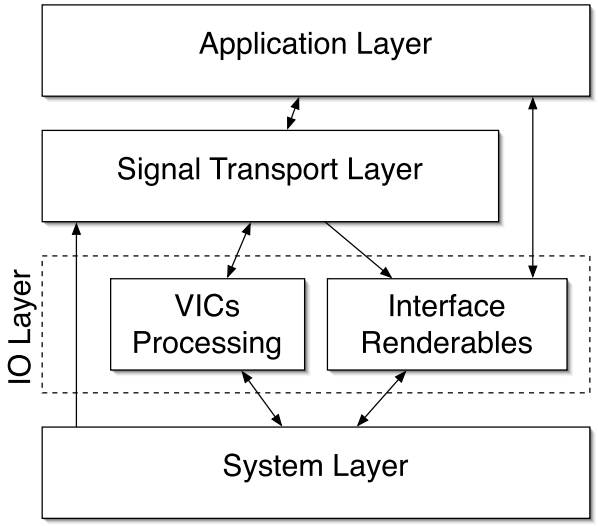


Figure 2.7: VICs system data flow graph.

The system architecture has a stratified design to partition component responsibilities. As depicted in Figure 2.7, there are four layers to the system:

Application Layer – The application layer contains all application logic.

Signal Transport Layer – The signal transport layer is responsible for transferring messages between the input/output systems and the application.

Input/Output (IO) Layer – The input/output layer comprises the vision processing, conventional input data passage (e.g. mouse, keyboard) and the graphical rendering.

System Layer – The system layer is an abstraction of the underlying physical computer system. It generates signals corresponding to mouse-clicks and key-presses, which will be passed through the IO layer to the signal transport. It also acquires the video signals that are used in the VICs processing.

Standard application code resides in the application layer. For instance, if the application is a word processor, all program code required to process and typeset text would be in this layer. Code for detecting user input (i.e. key presses) is not in this layer; instead,

the application waits on the system until it receives a signal that the user has pressed a key and then processes the inputted character. Similarly, the IO layer encapsulates all of the vision input functionality of the system. When *important* user-actions are detected by the VICs processing, the IO layer sends the appropriate signals to the application layer according to a predefined protocol. We use the word "important" to mean events that the application has instructed the vision system to trap, which is typically based on the types of VICons the application has instantiated. Thus, the vision system is inherently naive. It does not make any assumptions about important events. Likewise, in the above word-processing example, the system is continuously processing key presses, but the key press signals are only sent to the application if the application has requested such events.

2.4.2 Control VICons

The visual stream parsing of a VICon may be dependent on some global information application, system, or environment information. For example, in vision-based interfaces the lighting conditions may affect the underlying image analysis algorithms. The analog in the earlier word-processing example is the CAPS-LOCK key. Theoretically, this functionality code be incorporated directly into the visual stream parsers, but such a design would increase the complexity of the parsers making them more difficult to design, program, test, and evaluate.

Therefore, to better encapsulate the underlying vision processing from the application, we introduce the notion of control VICons. A control VICon is any VICon that directly affects the system state from the point-of-view of the vision processing. The only technical difference between the control VICons and standard VICons is the way in which the output signals are used: a standard VICon emits signals that are caught by the application while a control VICon emits signals that are caught in the VICs processing (IO layer). Additionally, the control VICons may be invisible to the user, but this distinction has no effect on the vision system.

2.4.3 Wrapping Current 2D Interfaces

There is a large body of 2D applications which are in daily use. We explain how the VICs paradigm can be seamlessly integrated into such applications without any modification to the existing application codes. We can complement the current set of user-events that

are mouse and keyboard generated with a standard set of hand gestures, which will be relatively small given the conventional 2D interfaces. In general, there are two classes of interface components that will be captured:

1. For clickable items, we define a natural pressing gesture that is performed by the user extending an outstretched finger near the interface component, touching the element and then removing the finger. A "button-press" event is generated when the visual parser observes the finger touching the element, and then a "button-release" event when the finger retracts.

2. For draggable items a natural gesture is again used: the user approaches the item with two opposing fingers open, and upon reaching the item, he or she closes the fingers as if grasping the item (there is no haptic feedback). Then, the user is permitted to drag the item around the workspace, and whenever he or she wishes to drop the item, the two grasping fingers are quickly released. The corresponding events we generate are "button-press," "motion-notify" (mouse motion), and "button-release."

Recalling Figure 2.7, from the perspective of the application, the events it receives from the vision-triggered actions are equivalent to those triggered by the mouse and keyboard. We will provide an approach to modeling these gestures and experimental results in Section 3.3.1.

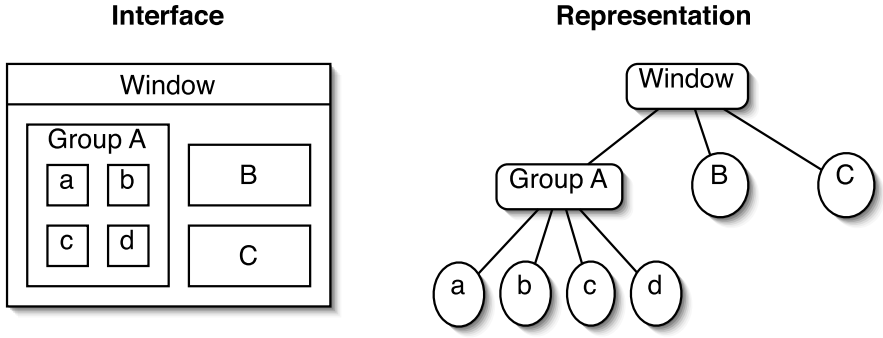


Figure 2.8: Example 2D interface component hierarchy.

To wrap an application, we analyze its component structure. In typical 2D windowing libraries, the interface components are stored in a hierarchical manner (Figure 2.8).

The hierarchical representation allows efficient, programmable analysis of arbitrary windowing applications. For each interface component, we check the events for which it is waiting (e.g. mouse clicks). If any of these events match those discussed above for clickable and draggable components, then we instantiate a VICon at the same location and with the same size as the original interface component. The VICon is created with a transparent visual representation to avoid modifying the application's visual representation.

2.5 Modes of Interaction

The notion of a VICs-based interface is broad and extensible to varying application domains. In this section we enumerate the set of interaction modes in which a VICon may be used.

2.5.1 2D-2D Projection

Here, one camera is pointed at a workspace, e.g. tabletop. One or more projectors is used to project interface components onto this surface while the video-stream is processed under the VICs paradigm. This mode has been proposed in [196] with the Visual Panel.

2.5.2 2D-2D Mirror

In this mode of interaction, one camera is aimed directly at the user and the image stream is displayed in the background of the user-interface for the user. Interface components are then composited into the video stream and presented to the user. This interface mode could also be used in a projection style display to allow for a group to collaborate in the shared space. The Sony I-Toy® and associated video games use this interaction mode.

Figure 2.9 shows some example applications of this model. First, we show a simple button-based VICon in a calculator setting (Figure 2.9-left). Next, we show multiple triggers based on user-input (Figure 2.9-middle). Here, the user can select the ball, drag it, and release. As mentioned earlier, motion and dynamics can be added to the VICons. Figure 2.9-right shows a *Breakout*^{TM} like program where the ball is a VICon. During play, the ball (the VICon) travels through the workspace. The user attempts to prevent the ball from falling through the bottom of the workspace while deflecting it toward the colored bricks at the

top of the workspace; notice the VICon is not anchored.

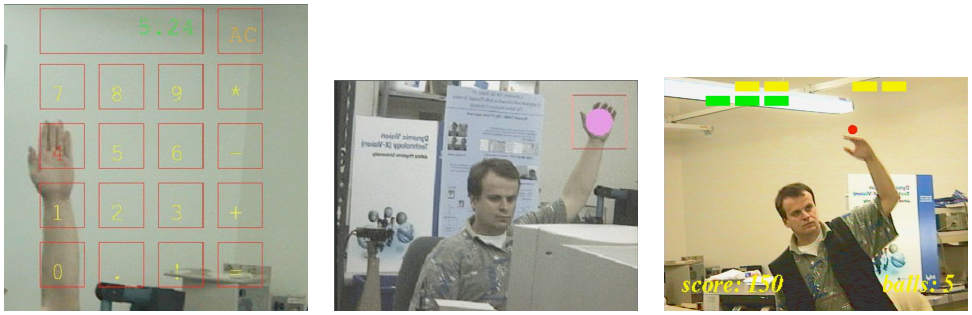


Figure 2.9: (left) A VICs-based calculator using a motion-color parser. (middle) Gesture-based demonstration of multiple interaction triggers for a single VICon. (right) VICs-based 2D-2D mirror mode interface for a *Breakout*^{TM} style game. We thank Prof. Dr. Darius Burschka for supplying these images.

2.5.3 3D-2D Projection

This mode is similar to the first (2D-2D Projection) except that 2 or more cameras will be aimed at the workspace and the set of possible gestures is increased to include more robust 3D geometry. The 4D-Touchpad (4DT) is an experimental platform based on the VICs framework; we will elaborate on this mode and the 4DT platform in Section 2.6.

2.5.4 2.5D Augmented Reality

Both video-see-through and optical-see-through augmented reality are possible if the user(s) wear stereo head-mounted displays (HMD) [4, 5]. With stereo cameras mounted atop the HMD, knowledge of a governing surface can be extracted from the view, e.g. planar surface [33]. All VICons can then be defined to rest on this governing surface and interaction is defined with respect to this surface.

2.5.5 3D Augmented Reality

In this mode, we remove the constraint that the interface is tied to one governing surface and allow the VICons to be fully 3D. Two example applications areas are (1) motor-function training for young children and (2) medical applications. Essentially, a

MACS, which was motivated in the introductory chapter, would be implemented using this interaction mode.

2.6 The 4D Touchpad: A VICs Platform

In this section, we introduce a VICs platform that has been constructed based on the 3D-2D projection interaction mode. A pair of wide-baseline cameras are directed at the interaction surface. There are two versions of the system: in the first version, the surface is a tabletop covered by a white-projection screen, and a projector is placed underneath the table. The cameras are positioned above to avoid user-occlusion in the projected images. In the second version, a standard flat-panel display is laid atop the table and used as the interaction surface. These setups are shown in Figure 2.11 and a schematic of the first version is shown in Figure 2.10. Unless otherwise noted, we assume the projector is present, which is the more complex case because there is an additional keystone distortion introduced by the projector.

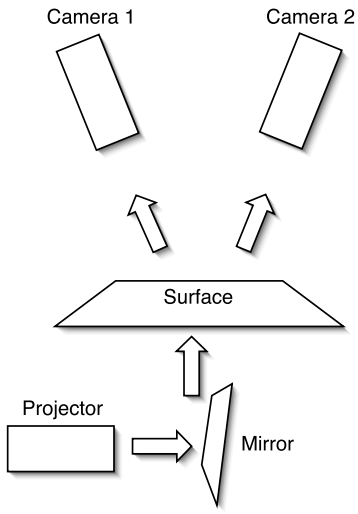


Figure 2.10: The schematics for the 4D-Touchpad (with projection).

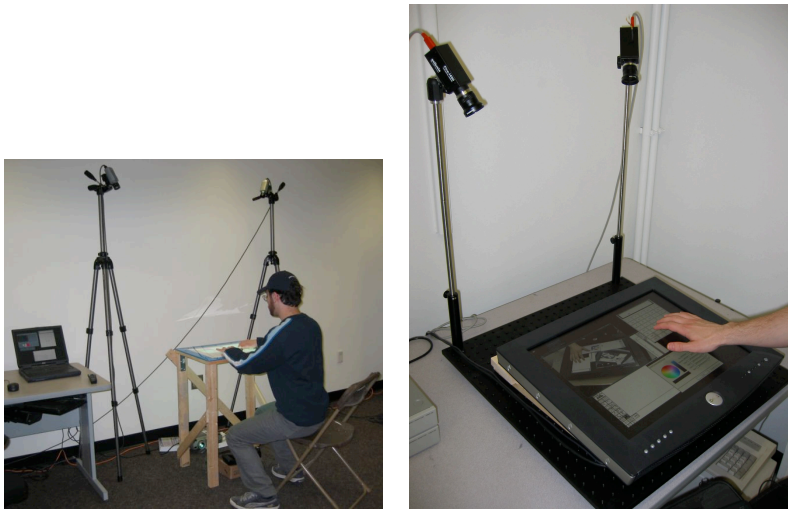


Figure 2.11: (left) 4D-Touchpad system with a rear-projection screen. (right) 4D-Touchpad system with a standard flat-panel display.

2.6.1 Image Rectification

In order for the visual processing and the graphical rendering of the interface to work correctly, the cameras and the projector must be calibrated relative to the interaction surface. Since each VICon is processing only input in its local ROI, the optimal placement of a camera is parallel to the plane of the interface above the center of the table, while the projector is placed ideally below the table (negating all keystone distortion) to project exactly the same extent as the cameras are capturing. In practice, this is not possible because of physical imperfections and misalignments.

The solution to this problem lies in the use of a well known homography that maps points on a plane to their image [45]. Similar to [161], the assumptions made in the system are that the intrinsic and extrinsic parameters of the cameras are unknown and that the camera and projector optics can be modeled by perspective projection. The projection $\begin{bmatrix} wx & wy & w \end{bmatrix}^{\mathsf{T}} \in \mathfrak{P}^2$ of a point $\begin{bmatrix} X & Y & Z \end{bmatrix}^{\mathsf{T}}$ in space into the camera-frame can be modeled with standard perspective projection:

$$\begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} m_1 & m_2 & m_3 & m_4 \\ m_5 & m_6 & m_7 & m_8 \\ m_9 & m_{10} & m_{11} & m_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{2.1}$$

where $m_1 \ldots m_{12}$ form the entries of the projection matrix, which has 11 degrees-of-freedom. Without loss of generality, we can say that the plane lies at $Z = 0$ yielding the following homography, $\mathfrak{P}^2 \mapsto \mathfrak{P}^2$:

$$\begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} m_1 & m_2 & m_4 \\ m_5 & m_6 & m_8 \\ m_9 & m_{10} & m_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}. \tag{2.2}$$

Numerous techniques exist for recovering this homography from points [161, 195], lines [115], and other image features. The homography is used to solve the following three problems: *(i)* rectification of the two image streams thereby *placing* the cameras directly above the table, *(ii)* keystone correction of the projected image, and *(iii)* projector-camera imaging area alignment. Thus, we have a two-step process that is now formalized; problems *(i)* and *(ii)* are solved in a manner that also satisfies *(iii)*. Note, we will apply the homography directly to the imaged points; there is a five-parameter linear transformation $\mathfrak{P}^2 \mapsto \mathfrak{P}^2$ that maps the camera-frame to the image-plane. We assume the parameters of this transformation are unknown and capture them directly in the computed set of homographies.

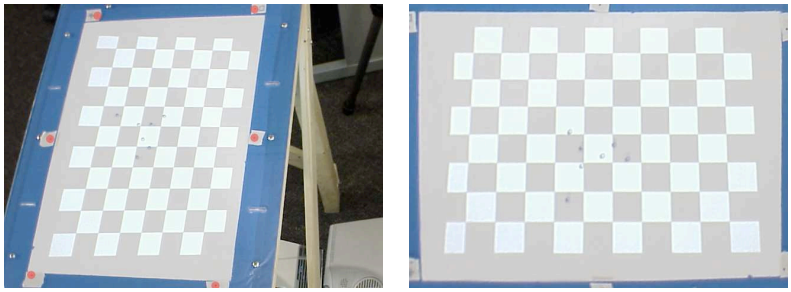


Figure 2.12: (left) The original image from one of the cameras. Calibration points are shown highlighted in red. (right) The same image after it has been rectified by H_i.

First, we compute $H_{i\in\{1,2\}} : \mathfrak{P}^2 \mapsto \mathfrak{P}^2$ that rectifies a camera image into model space (Figure 2.12). Let $\hat{p}_{j\in\{1\ldots n\}} \in \mathfrak{P}^2$ be an imaged point (shown in red in Figure 2.12-

left) and let $b_j \in \mathfrak{P}^2$ be the corresponding model point in rectified space. In our context, we assume that the model is constructed by a set of known points that form a rectangle atop the table. For each camera, we can write

$$b_j = H_i \hat{p}_j, \quad i \in \{1, 2\} \wedge j \in \{1 \dots n\} \tag{2.3}$$

Then, we compute the projector homography $H_p \colon \mathfrak{P}^2 \mapsto \mathfrak{P}^2$ that will keystone-correct the projected image and ensure projector-camera alignment. Let $q_{j \in \{1 \dots n\}} \in \mathfrak{P}^2$ be a point in the projector image, and $\hat{q}_j^i \in \mathfrak{P}^2$ be the corresponding point after it has been projected onto the table, imaged by camera i, and rectified by H_i. Thus, we define H_p in a way that it maps $\hat{q}_j$ to the corresponding model point b_j.

$$b_j = H_p \hat{q}_j^i, \quad i \in \{1, 2\} \wedge j \in \{1 \dots n\} \tag{2.4}$$

This homography corrects any keystone distortion and aligns the projected image with the rectified camera images:

$$H_p \hat{q}_j^i = b_j = H_i \hat{p}_j, \quad i \in \{1, 2\} \wedge j \in \{1 \dots n\} \tag{2.5}$$

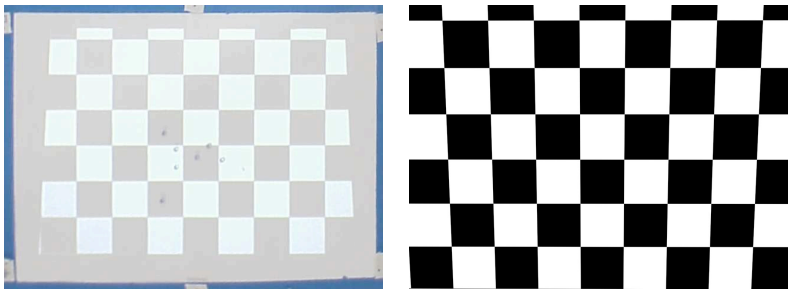


Figure 2.13: (left) The projected image with the keystone correction applied (the distorted one is Figure 2.12-right). (right) The pre-warped image with the keystone correction applied before it has been projected.

2.6.2 Stereo Properties

The vision system is responsible for the correct detection of press actions on the table and other gestures related to the VICons. Since the projection onto a camera plane

results in the loss of one dimension, we use two cameras to verify the contact of the object with the surface of the table.

The rectification process described in the previous section corrects both camera images in a way that all points in the plane of the interface appear at the same position in both camera images. For this homographic calibration, a simple, region-based stereo calculation can be used to detect contact with the surface (Figure 2.14). For a given VICon, we segment the color regions with a color skin detector. The intersection of the resulting color regions represents the part of the object that has actual contact with the plane of the interface. In Figure 2.15 we show a graph of the depth resolution of our system. The high depth discrimination we see is due to the wide baseline of the stereo system.

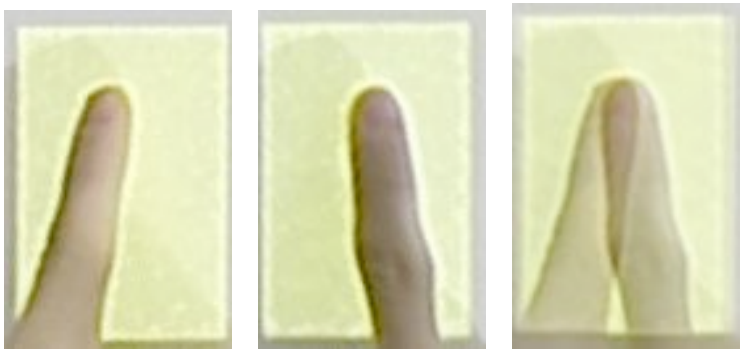


Figure 2.14: Disparity for a typical press action: (left) rectified image 1, (middle) rectified image 2, (right) overlayed images of the finger.

Since the interaction take places in the volume directly above the interface and the VICs system is aware of the rendered interface, we can perform a simple background subtraction in order to achieve robust segmentation of the objects above the plane. For both cameras, we can subtract the current frame from a stored background frame yielding mask of *modified* regions. Then, we can take the difference between two *modified* masks to find all pixels not on the plane and use it in more intensive computations like 3D gesture recognition. This method also reduces the influence of shadows that appear as part of the interface plane and get removed.

2.6.3 Color Calibration and Foreground Segmentation*

*The work in this section was performed by G. Ye. We include it here for completeness.

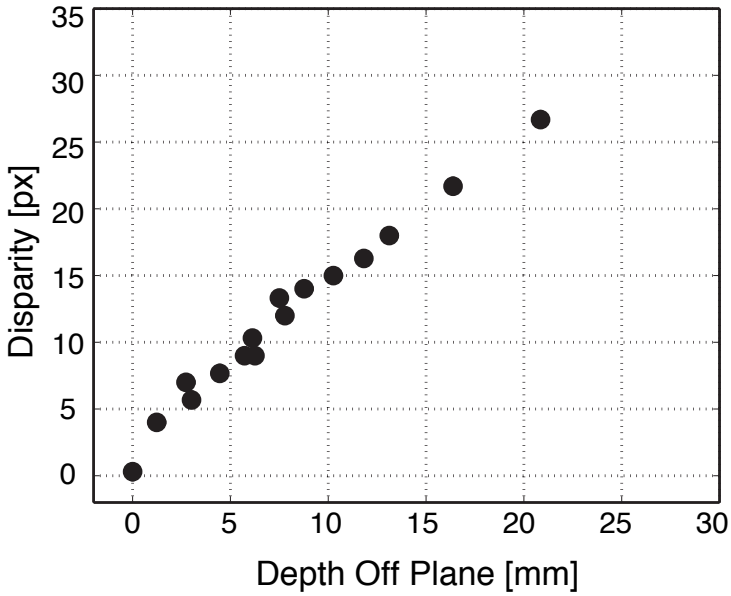


Figure 2.15: Graph showing the depth resolution of the system.

Since the interaction take places in the volume directly above the interface and the VICs system is aware of the rendered interface, we can perform background subtraction to segment objects of interest (e.g. gesturing hands) in the interaction volume. Robust hand detection is a crucial prerequisite for gesture capturing and analysis. Human hands demonstrate distinct appearance characteristics, such as color, hand contour, and geometric properties of the fingers. We carry out segmentation based on the appearance of the hand.

Background Modeling

The key is to find an efficient and robust method to model the appearance of the background and the hand. Background subtraction, gray-scale background modeling [74], color appearance modeling [163], color histogram [82] and combining of multiple cues [183] are among the most widely used methods to model the background and perform foreground segmentation. We propose to directly model the appearance of the background by color-calibrating the rendered scene and the images of the scene captured by the cameras.

We model the color appearance of the background using an affine model. The color images from the cameras and the rendered scene are represented in YUV format. An affine

model represents the transform from the color of the rendered scene, i.e., $s = \begin{bmatrix} Y_s & U_s & V_s \end{bmatrix}^\mathsf{T}$, to that of the camera image $c = \begin{bmatrix} Y_c, U_c, V_c \end{bmatrix}^\mathsf{T}$. Using a 3×3 matrix A and a vector t, we represent this model using the following equation.

$$c = As + t \tag{2.6}$$

The model parameters, A and t, are learned via a color calibration procedure. We generate a set of N scene patterns of uniform color, $\mathcal{P} \doteq \{P_1 \dots P_N\}$. To ensure the accuracy of the modeling over the whole color space, the colors of the set of the calibration patterns occupy as much of the color space as possible. We display each pattern P_i and capture an image sequence S_i of the scene. The corresponding image C_i is computed as the average of all the images in the sequence S_i. The smoothing process is intended to reduce the imaging noise. For each pair of P_i and C_i, we randomly select M pairs of points from the scene and the images. We construct a linear equation based on these $N \times M$ correspondences and obtain a least squares solution for the 12 model parameters.

We use image differencing to segment the foreground. Given background image I_B and an input image I_F, a simple way to segment the foreground is to subtract I_B from I_F. We compute the sum of absolute differences (SAD) for the color channels of each pixel. If the SAD is above a certain threshold, this pixel is set to foreground. Figure 2.16 shows an example of the segmentation.

Skin Modeling

To improve the robustness of the foreground segmentation, we include an additional skin color model. Many skin models have been proposed in the literature [127]. Here we choose a simple linear model in UV-space. Basically, we collect skin pixels from segmented hand images and train the model as a rectangle in the UV plane. Four parameters, i.e., $U_{min}, U_{max}, V_{min}, V_{max}$, are computed and used to classify image pixels.

Experiments

We have carried out a series of experiments to quantify the accuracy and stability of the segmentation. First, we examine the robustness and stability of the color calibration algorithm. We train the affine color model using 343 unicolor image patterns which are

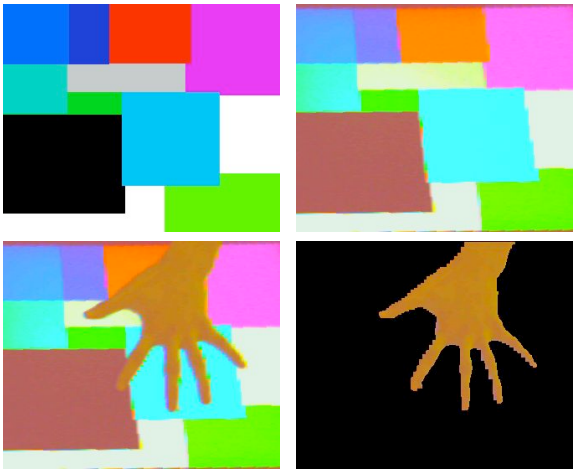


Figure 2.16: An example of image segmentation based on color calibration. The upper left image is the the original pattern rendered on the screen. The upper right image is the geometrically and chromatically transformed image of the rendered pattern. The lower left image shows the image actually captured by the camera. The lower right image is the segmented foreground image.

evenly distributed in the RGB color space. To test the accuracy of the learned affine model, we display over 100 randomly generated color images and examine the resulting segmentation. In this case, the ground truth is an image marked completely as background pixels. For both cameras, the system achieves segmentation accuracy of over 98%.

We also investigate the efficacy of the linear skin model. We learn the model by analyzing image sequences containing the hands of over 10 people. To test the model on our platform, the user is asked to place his or her hand on the flat-panel and keep it still. Next, we render a background which is known to perform well for skin segmentation and treat the resulting skin foreground segmentation as the ground truth. The user is asked to keep his or her hand steady while we render a sequence of 200 randomly generated patterns. For each image, we count the number of incorrectly segmented pixels against the true segmentation. The overall skin segmentation accuracy is over 93%.

2.7 Conclusion

In this chapter, we have presented the VICs methodology, which is a practical framework for building vision-enabled interfaces. The main contribution of this chapter is

the manner in which we approach the VBI problem: VICs relies on a shared perceptual space between the user and video cameras. In this space, interface components are mapped to the image projections and gestures are modeled as sequences of visual cues in the local image regions corresponding to the components. For the remainder of this conclusion section, we discuss this contribution in light of the current state-of-the-art in vision-based interface technology. In the next chapter, we will discuss the gesture recognition methods we have developed using the VICs paradigm.

2.7.1 Multiple Users

While in current interfaces, the interaction is typically restricted to one user, in simulation, training, collaborative, and large-scale applications multiple users may enter and exit the interaction dynamically (often unannounced). While this presents concern for techniques that focus on tracking the user(s) in a global sense, the VICs paradigm must make no adjustment for multiple users. Each VICs-enabled interface component will continue to parse its visual input stream in a consistent manner regardless of the number of users with the system. Clearly, the visual processing routines must be robust to perform the visual analysis of different users.

2.7.2 Applicability

The VICs approach is well-suited to any interaction task that is site-centric; i.e. the interaction happens with respect to some object or at some location in the interface. Since the gesture recognition is approached as generic spatio-temporal pattern recognition, the same or similar recognition routines can be used in both conventional and modern interface settings. In conventional settings, as discussed in Section 2.4.3, we can mimic the components (buttons, scrollbars, etc.) with vision-based components to provide fully-functional interfaces without requiring the user to learn new, complex interaction protocols. In general, this ability to re-use the same computer vision algorithms to handle interaction in a variety of settings is an advantage of the approach.

However, the paradigm is not universally applicable for there may be some interfaces better suited to user-centric interaction models. For example, eye gaze tracking may be the only way to enable interaction during automobile driving. In other settings, communication-based interaction protocols, like speech and language, may be more appli-

cable than the site-centric approach. The future of interface design lies in the integration of these different interaction modalities.

2.7.3 Robustness

We argue that the VICs approach to vision-based interaction adds structure to the gesture recognition problem. The additional structure makes it plausible to use advanced spatio-temporal pattern recognition techniques to perform gesture recognition, which results in reliable recognition. Thus, we avoid the difficult problem of globally tracking the articulated human, on which most conventional techniques rely. However, because the robustness is inherently dependent on the visual parsing, we defer further discussion to Chapter 3.

2.7.4 Efficiency

The efficiency of the VICs system is important since it will share computational resources with the rest of the computer (including the applications). For every frame of video, each VICon's parser operates on the image(s). A naively implemented parser could potentially waste computational resources resulting in dropped frames or queued parsers. However, the simple-to-complex processing (Section 2.2.2) will alleviate this problem. In the majority of situations, interaction is localized to a specific region in the mixed reality (and hence, the images), and only a small subset of the instantiated VICons will do any processing beyond simple algorithms (for example, motion detection). Thus, the amount of computational resources expended by the computer is directly proportional to the amount of activity occurring in the VICon image-regions. In contrast, the conventional approaches (Section 1.4.4) that rely on global user tracking and modeling will use computational resources even when no actual interaction is occurring; such unnecessary processing is wasteful.

Chapter 3

Gesture Modeling*

In this chapter, we discuss how we have performed the gesture modeling. We present our gesture parsing in the context of the Visual Interaction Cues paradigm, which was introduced in the previous chapter. First, we state a concrete definition of gesture as it relates to VBI, and we categorize different types of gestures. Second, we discuss deterministic cue parsing modeled with a fixed finite state machine. Then, we discuss learning-based extensions. We briefly discuss stochastic, probabilistic extensions of the parsers, and also present an approach to posture recognition and parametric gestures using neural networks. Last, we present the binary gesture, which integrates static, posture and parametric gesture tracking into a single model. Excluding the binary gesture, the gestures we consider in this chapter are atomic, low-level operations. Each low-level gesture corresponds to a single action, like "grasping" an interface object. In the next chapter, we will integrate these heterogeneous, low-level gestures in a coherent model that captures a *language of interaction*.

3.1 What is a Gesture?

Gestures are the foundation of natural interaction with a vision-enabled computer system. Like words in language, the gestures form the core of the interaction between human and computer. In Figure 3.1, we make the explicit link between speech and gesture recognition. The *gestemes* are low-level *features* from the video signals and form the basis

*We thank Ravi Mody for helping carry out some of the experiments in this chapter during his Research Experience for Undergraduates residence.

of the gesture recognition. While gestures can be made with many different body-parts, we focus on hand gestures exclusively.

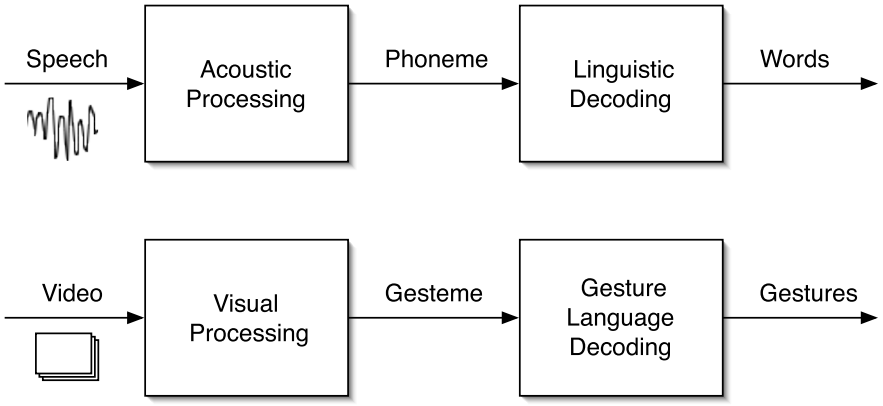


Figure 3.1: The analogy between gesture and speech recognition.

3.1.1 Definition

As discussed earlier, the conventional approach to VBI is based in tracking: build a model of the user, track the parameters of the model through time (in video), and model gestures through these parameters. In their review of hand gesture analysis, Pavlovic et al. [123, Def. 1] give a concrete definition of gesture taking such an approach: "Let $\mathbf{h}(t) \in \mathcal{S}$ be a vector that describes the pose of the hands and/or arms and their spatial position within an environment at time t in the parameter space $\mathcal{S}$. A hand gesture is represented by a trajectory in the parameter space $\mathcal{S}$ over a suitably defined interval $\mathcal{I}$."

Given the VICs paradigm, we take a different approach to gesture modeling and representation than is typical in the literature. We state a concrete definition:

> Given one or more video sequences $\mathbf{S}$ of the local region surrounding an interface component, a gesture is a subsequence $\mathbf{G} \subset \mathbf{S}$ with length $n \geq 1$ corresponding to a predefined (or learned) spatio-temporal pattern.

As will be shown in the results in the latter parts of this chapter, defining gestures in this manner leads to robust gesture recognition. The additional structure inherent in this definition avoids the difficult problem of articulated human tracking. However, in certain types of gestures (see the classification of gestures below) quantitative knowledge of the motion

is necessary. For example, recalling the circular dial component from the previous chapter, the rotation angle of the gesture motion is required to actuate the dial's functionality.

We claim that quantitative information can accompany the gesture as it is defined above. The solution lies in a temporary, function-specific parameterization of the spatio-temporal pattern. Therefore, still fewer motion parameters will be tracked than in typical global user-tracking methods, which can involve as many as 27 degrees-of-freedom for a single human hand [134]. To justify this claim, we will provide methods and results for quantitative gesture tracking.

3.1.2 Gesture Classification

Many gesture-based visual interfaces have been developed [34, 118, 183, 175, 190, 196]. All *important*[1] gestures in VBI are task-driven, but there is no single type of gesture. To best understand the different types of gestures in VBI, we introduce a classification. We adopt the previous gesture taxonomies of [123, 129] and integrate them into the VICs gesture recognition paradigm.

In VBI, gestures have two distinct modalities: manipulation and communication. A manipulative gesture is one that operates on an interface component or application object in the environment. The operation results in some change of the object, which is typically a function of the gesture instance. A communicative gesture, on the other hand, transfers some piece of information to the system via the gesture. The communicative gesture does not necessarily result in any immediate change to a particular interface component. Pavlovic et al. [123] introduce two distinct types of communicative gestures: symbols and acts. A symbolic, communicative gesture has a direct linguistic role to communicate a conceptual idea. The significant part of an act is in the motion: e.g. pointing in a particular direction is an act.

	Form	
Function	**Static**	**Dynamic**
Qualitative	Communicative (symbol)	Communicative (symbol)
Quantitative		Communicative (act) or Manipulative

Table 3.1: Classification of gestures according to form and function.

[1] We disregard any human motion or gesticulation that does not directly correspond to interaction.

According to form and function, we further classify the gestures into three categories (Table 3.1): static, dynamic-qualitative,[2] and dynamic-quantitative. Static postures [3, 98, 116, 129, 168, 185] model the gesture as a single key frame, thus discarding any dynamic characteristics. For example, in recent research on American Sign Language (ASL) [160, 197], static hand configuration is the only cue used to recognize a subset of the ASL consisting of alphabetical letters and numerical digits. The advantage of this approach is the efficiency of recognizing those gestures that display explicit static spatial configuration. However, it has an inherent shortcoming in handling dynamic gestures whose temporal patterns play a more important role than their static spatial arrangement.

Dynamic gestures contain both spatial and temporal characteristics, which presents a further challenge in the modeling. Many models have been proposed to characterize the temporal structure of dynamic gestures: including temporal template matching [17, 104, 121, 156], rule-based and state-based approaches [18, 129], hidden Markov models (HMM) [130, 160, 187, 189] and its variations [20, 118, 181], and Bayesian networks [155]. These models combine spatial and temporal cues to infer gestures that span a stochastic trajectory in a high-dimensional spatio-temporal space.

Most current systems model dynamic gestures qualitatively. That is, they represent the identity of the gesture, but they do not incorporate any quantitative, parametric information about the geometry or dynamics of the motion involved. To overcome this limitation, a parametric HMM (PHMM) [181] has been proposed. The PHMM includes a global parameter that carries an extra quantitative representation of each gesture. This parameter is included as an additional variable in the output probabilities of each state of the traditional HMM.

3.1.3 Gesture Recognition

Interaction is a dynamic, continuous process between the user and the computer system. Therefore, gesture recognition must be performed while the system is in operation in an on-line manner. We consider the gestures corresponding to the earlier definition as the atomic unit of interaction; accordingly, we will refer to them as low-level gestures, or gesture-words when we discuss them in the context of a complete gesture language (Chapter 4). We state the on-line, low-level gesture recognition problem as follows:

[2] We use the terminology dynamic-qualitative and dynamic-non-parametric interchangeably. Likewise, we use dynamic-quantitative and dynamic-parametric interchangeably.

> Given one or more video streams **S** of the local region surrounding an interface component, temporally segment the video subsequences whose spatio-temporal signature corresponds to a valid gesture in a predefined vocabulary of gestures. During segmentation, label the gesture identity and measure any quantitative information the gesture may carry.

There are many approaches to solve this problem. In the next two sections, we will discuss two classes of methods we have studied. First, we will introduce an algorithm that is based on procedurally defined visual cue detection and parsing. Second, we will discuss learning-based extensions to the visual cue parsing, which we have found to perform with much greater recognition accuracy than the deterministic methods.

3.2 Procedural Recognition

In general, the solution space for gesture recognition is extremely large. However, with the additional structure imposed on the problem by the VICs paradigm, we can solve the problem in a simple, procedural manner. Recall that the VICon monitors a region-of-interest in the video stream for recognizable user activity. For instance, if we model a simple push-button, the VICon might watch for something that resembles a human-finger in its ROI.

The obvious approach to detect user interaction is one of template-matching. Each frame of video is compared via template analysis to the appearance of a predefined set of possible gestures using standard image processing techniques. However, in practice, such a method is prone to false-positives by spurious template matches. Also, a template matching approach, alone, is potentially wasteful because it is more expensive than other simpler tasks like motion detection and color segmentation that may easily indicate a negative match.

If one observes the sequence of cues that precede a button-push, for instance, one notices that there are distinct cues comprising the actual button push: motion, color-blob, shape verification. This sequence of cues, ordered from simple to complex, can be used to facilitate efficient, robust user-input detection. Define a *selector* to be a vision component that computes some measure on a local region of an image, and returns either nothing, indicating the absence of a cue or feature, or values describing a detected feature [66]. For example, a motion selector might return nothing if there is no apparent image motion or a description of the size and magnitude of a region of detected motion.

We define a *parser* to be a sequence of selector actions. Multiple parsers may exist for similar user-actions, but each may operate optimally under different circumstances: for example, in the presence of abundant lighting, a button-press parser may be defined as above: motion, color-blob, shape verification. Yet, if the lighting is inadequate, a second button-press parser may be defined as motion, edges, and shape verification. Thus, these two parsers provide similar functionality under different conditions thereby increasing the overall robustness of the system.

3.2.1 Parser Modeling

Formally, we define a parser. It is modeled by a state machine with the following structure (Figure 3.2):

1 A finite set of discrete states $s_0, s_1, ...s_n$.

2 A distinguished initial state s_0 representing the inactive condition.

3 Associated with each state s_i, a function f_i comprised of a *bank* of selectors b_i that defines a continuous state variable x.

4 For each state s_i, a set of transition rules that associates an event $e_{i,j},\ j = 1 \ldots m \leq n$ (informally, the output of one or more selectors in the bank b_i) with either a state of a different index, or s_i (the null-transition). By convention, the first transition event to fire defines the transition for that state.

We explain the parser modeling of the example of a button press VICon from above. We create a possible sequence of selectors: (1) a simple motion selector defines the trigger condition to switch from the distinguished initial state s_0, (2) a coarse color and motion selector, (3) a selector for color and cessation of motion, and (4) template recognition. It is easy to see that processing under this framework is efficient because of the selector ordering from simple-to-complex wherein parsing halts as soon as one selector in the sequence is not satisfied.

While in its current state the topology of the parser and implementation of the selectors is the task of the developer, we are investigating learning methods (e.g. Bayesian networks) to make this process automatic. The example interface we present in this section demonstrate that this is not a hindrance to the use of procedural parsers.

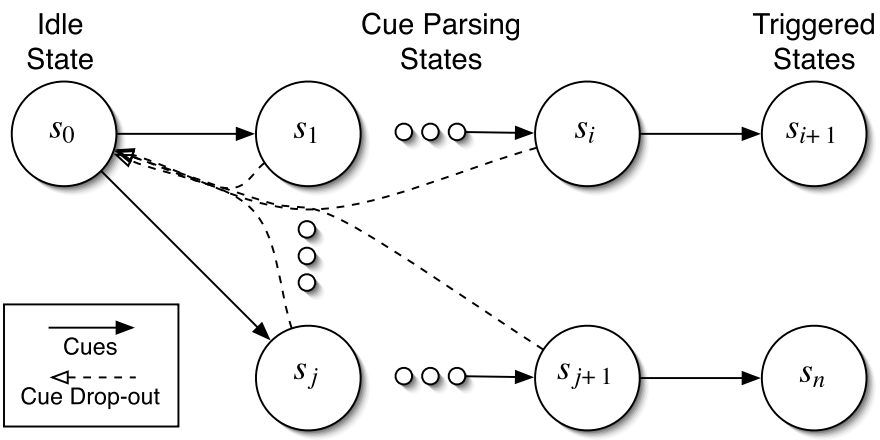


Figure 3.2: Parser state machine for procedural gesture recognition.

3.2.2 Dynamics

The intent of the framework is that a parser will not only accept certain input, but it may return other relevant information: duration, location, etc. We define a VICon history as a set $\{h_0 \dots h_m\}$, where the set length m implicitly represents the duration of the current interaction and $h_{j \in \{1 \dots m\}}$ is a snapshot of the behavior's current state including the measured quantitative information. When a behavior enters its distinguished initial state, its history is reset: $m \leftarrow 0$. The history begins to *track* user's actions when the behavior leaves its initial state. A snapshot is created for every subsequent frame and concatenated into the history thereby adding another dimension that can be employed by the VICon during parsing.

3.2.3 Example Button-Press Parser

We provide a concrete example of a procedural parser for the case of a standard button interface component. The parser (Figure 3.3) is modeled with 4 states with each state having an independent bank of cue selectors. Again, the solid arrows in the figure represent successful state jumps, and the dashed arrows represent cue drop-out. We assume we are working on the 4DT platform and have stereo imagery, but most of the selectors use only one channel to reduce computational cost.

Parsing begins when the initial state's *motion* selector passes. We build the motion

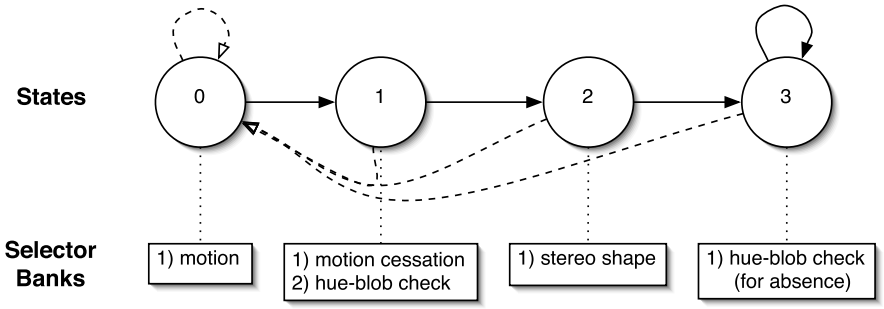


Figure 3.3: Example procedural state machine for a button press.

selector with very simple image-to-image differencing. For example, at time t_0, we store the image around the icon if no activity is occurring. Call this *background* image $\mathbf{B}$. Then, for all frames $\mathbf{I}_t, t = t_1 \ldots t_k$, we measure motion with a simple image difference:

$$m = \sum_{\mathcal{I}} |\mathbf{I}_t - \mathbf{B}| \tag{3.1}$$

If the motion estimate m passes a predefined threshold then the parser jumps to state 1. The threshold is heuristically determined based on the region size. The *background* image is updated every k frames.

There are two selectors in state 1. The *motion cessation* selector is measured by simply taking the opposite decision for the threshold in the previous selector. In the *hue-blob check*, we perform a simple connected-components analysis [78] on the hue representation of the image. Define a range of acceptable skin hue values. The selector passes if the hue of the largest connected component in the center of the local image region falls within this range.

The selector for state 2 will verify the object is actually near the plane of the interface. To do so, we use a coarse stereo test based on intersecting the hue-blobs from the two images (Section 2.6.2). First, the hue blob for the second image in the stereo pair must be segmented. Then, we intersect the two hue blobs with the following rule: a pixel is considered in the intersected set if it is in the valid hue-blob for both the images in the stereo pair. The number of pixels in the set is counted and thresholded giving a coarse estimate of how many pixels where on or near the interface place. In addition, the size and location of this blob is compared to a predefined *template*, which resembles a fingertip near

the center of the VICon.

Finally, the button is depressed in the switch from state 2 to state 3. The button will remain depressed until the hue-blob is absent from the video upon which time it switches back to the initial state and the button is released.

3.2.4 Robustness

We claim the procedurally defined parsers with a function-specific, simple-to-complex selector topology will perform robust gesture recognition according to the previously stated gesture recognition problem. We justify this claim in two ways. First, the individual selectors are built on well understood image processing techniques and operate on a small subset of the image. Second, the degree of false-positives is reduced due to the simple-to-complex parser structure operating on low-level visual cues. We provide experimental results to further justify this claim.

3.2.5 Experiments

To demonstrate the effectiveness of using the procedurally defined parsers within the VICs framework, we include some experiments. In these experiments, we use the first version of the 4D-Touchpad setup (Section 2.6). The system runs on a LinuxPC with dual CPUs operating at 1.4Ghz and 1GB of memory. The cameras are IEEE1394 Sony X-700; they are limited to 15Hz. However, we run the system in an asynchronous mode such that the VICons repeatedly process the same image until a new one is available. This allows multiple state transitions in the parser during one image-frame. Thus, our average processing rate is much greater than the input rate of the video-stream.

Efficiency

Since the vision processing system is sharing resources with the main computer system, it is necessary to minimize the amount of computational resources expended on the visual processing. In fact, this requirement is explicitly stated in the main vision-based interaction problem (Section 1.1). Each procedural parser analyses a local region of the image in a simple-to-complex fashion. This efficient manner of processing allows many VICons to be incorporated into the interface without using too much computation.

Table Table 3.2 shows the processor utilization for selectors of differing complexity (in this experiment, each selector was run independently and given full system resources).

Complexity	**Type**	**Rate [Hz]**
Coarse	Motion Detection	8300
Medium	Hue Segmentation	1000
Fine	Hue Blob Intersection	530
Fine	Shape Template	525

Table 3.2: Coarse-to-fine processing minimizes unnecessary computation. Rates for a 75x75 pixel region.

Recognition

We built a virtual piano-keyboard application. Figure 3.4-left shows the system in use. We include 8 keys in the piano that are green while off and blue when pressed. When the system is set to an image size of 320×240, the asynchronous visual processing operates at a mean rate of about 250 Hz with nearly 100% processor utilization and it operates at a rate of about 45 Hz with 100% utilization when using 640x480 images. Experimentation showed that 320×240 images were sufficient for robust interaction detection.

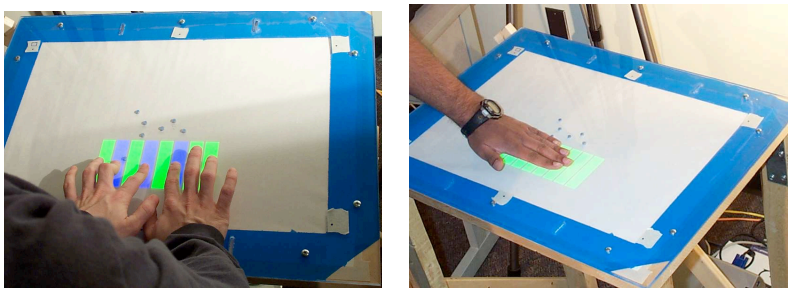


Figure 3.4: (left) The 8 key VICs piano-keyboard. The user is pressing-down the 3 blue keys. (right)The 8 key VICs piano-keyboard employs a fingertip shape matcher at its finest detection resolution.*

The piano key VICons were modeled exactly according to the example in Section 3.2.3. The piano key is considered pressed from the transition between states 2 and 3 until the transition from 3 to 0. Table 3.3 shows the accuracy rates of the system in usage;

*The dots in the center of the work-surface are artifacts of the piece of Plexiglas used for the first version–they are in no way intended or used by the system.

we performed an informal experiment with an unbiased onlooker judging user intention versus system response. Figure 3.4-right demonstrates the effectiveness of the system wherein the processing does not pass the shape-check in the stereo stage (a fingertip pressing the key).

	Percent
Correct	86.5
False Negative	11.45
False Positive	2.05

Table 3.3: Accuracy of the piano keys to normal user input. Each figure is the mean of a set of users playing for a half minute. The image size was 640×480.

3.3 Learning-Based Recognition

While the programmatically defined parsers are capable of accurate gesture recognition, there are a couple of drawbacks associated with their use. First, it is not clear that for any arbitrary gesture, an efficient and robust parser could be designed. Second, the design of procedural parsers can be both tedious and difficult. Therefore, an alternative approach is to introduce learning algorithms into the gesture recognition solution. As surveyed earlier, learning-based algorithms have been widely employed in gesture recognition. In this section, we discuss the learning-based techniques we have used to solve the recognition of low-level gestures.

Specifically, we use learning techniques to replace the manually constructed selectors and parsers. Learning techniques are data driven, which means that a corpus of *training* data is used to *learn* the parameters in the chosen technique. In the current development, we treat the resulting parser as a single black-box and assume a simple topology, which is induced as a function of the gesture being modeled and the modeling technique. We will extend the modeling in the next section (3.4); we will show how a complex gesture can be constructed by building a state machine with a simple topology that has a learned parser at each node.

A candidate approach is the Hidden Markov Model (HMM) [130]. The HMM is especially well-suited to modeling the non-parametric, dynamic gestures. However, we refer the reader to [188] as his dissertation research has focused on their use in solving the gesture recognition problem under the VICs paradigm. Instead, we will discuss the use of

neural networks, which is a standard machine learning technique for pattern recognition. We provide a brief introduction to multilayer neural networks, but we refer the reader to [15, 43, 144] for a more comprehensive presentation. There are alternative networks in the literature too: for example, the convolutional network [91] has been developed explicitly for imaging problems and has been shown to perform very well.

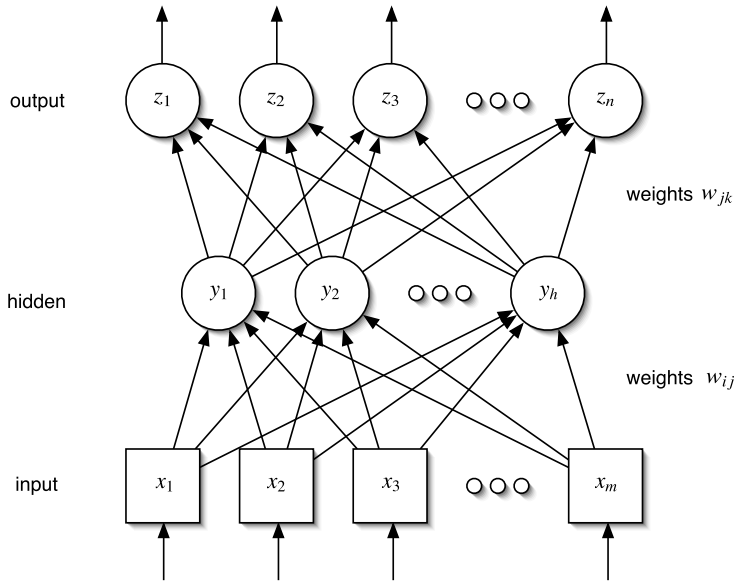


Figure 3.5: A standard three-layer neural network without bias. Nodes are labeled based on their output variable. Square nodes depict linear activation, and circular nodes depict non-linear activation. The weights between layer-nodes are represented by links in the figure.

We use standard three-layer neural networks without bias. In Figure 3.5, we show a typical network that is labeled with the notation we use. The three-layers consist of an input layer, hidden layer, and output layer. The middle layer is commonly termed the hidden layer because its activations are not exposed outside of the network. During network activation, a vector of inputs is passed to the input nodes. A linear combination of the inputs is passed to the hidden layer using the weights on the links between the two layers:

$$\hat{y}_j = \sum_{i=1}^{m} w_{ij} x_i. \tag{3.2}$$

Note, that we use networks without bias. A network with bias augments the input layer with a node that has constant activation. Each hidden unit outputs a value that is a non-linear function f of its input.

$$y_j = f(\hat{y}_j). \tag{3.3}$$

We use the common sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{3.4}$$

Again, linear combinations of the hidden node outputs are passed to the output nodes where the same non-linear function is applied to yield the final network activation result.

$$z_k = f\left(\sum_{j=1}^{h} w_{jk} y_j\right) \tag{3.5}$$

$$= f\left(\sum_{j=1}^{h} w_{jk} f\left(\sum_{i=1}^{m} w_{ij} x_i\right)\right) \tag{3.6}$$

In classification problems like static, qualitative gesture recognition, the outputs $z_1 \ldots z_k$ of the network activation are used as the discriminating functions. Intuitively, when used for classification, the network warps the input non-linearly and linearly discriminates it. Thus, such a network is capable of discriminating functions that are not linearly separable.

Before the network can be used, or activated, it must be trained, i.e. the weights of the network must be learned. In our work, learning is only concerned with the weights given a fixed network topology and sigmoid function. Here, the standard back-propagation algorithm is used. The network weights are initialized at random. During training, for each vector of inputs, a corresponding vector of *training* outputs is fed into the network. Any discrepancies, or *errors*, between the current network outputs and the training outputs is used to modify the parameters to reduce the error. The error function is the least-mean square error, and back-propagation implements a gradient descent error minimization. Again, the reader is referred to [15, 43, 144] for a thorough presentation of the back-propagation algorithm

In our work, we have used the multilayer neural network to solve two separate recognition problems. First, we use it in the classification of a set of static, nonparametric

gestures. Second, we perform quantitative gesture analysis for a known gesture with a real-valued network. For example, we track the location of the finger-tip in a VICon ROI for a pointing task. Next, we explain these two developments.

3.3.1 Classification of Static, Nonparametric Gestures

In the classification of static, nonparametric gestures, also called *postures*, we use a *one-of-X* three-layer neural network without bias. In this type of network, the number of nodes in the output layer of the network is either equal to the number of postures in the vocabulary. It is also possible to add an additional output node that is interpreted as *silence*, or the absence of a postures from the vocabulary. The network is termed one-of-X because each output node *points* to one of the postures in the vocabulary. The output node with the strongest activation is considered to be the network response.

A VICon with such a one-of-X neural network has a state machine as depicted in Figure 3.6, which is not to be confused with the network topology (the parser). We give the VICon state machine here for explanatory purposes as it will be extended later. As expected the VICon will be triggered by one of the postures in the vocabulary with the corresponding application event taking place.

In formatting the data for input to the network, we take advantage of the localized region processing of the VICs paradigm. We fix the size of the local region around the VICon to 128×128 and coarsely subsample the image in non-overlapping blocks with a parametric size. We perform no further feature extraction.

Experiments with a Four Posture Vocabulary

We choose a four posture vocabulary and include a silence posture (Figure 3.7). The set includes a finger pointing, a flat hand, a grasping gesture, a dropping gesture, and a silent gesture. The images are standard 640×480 YUV.[3] We include results for two different subsamping resolutions: 256 samples per channel (768 per image) and 64 samples per channel (192 per image). We have also experimented with both monocular and binocular video. In the Table 3.4 and Table 3.5, we show the training and recognition results for uncluttered and cluttered images, respectively. These experiments contained data from one user.

[3]YUV is a standard color format with the Y channel containing luminance information and the UV channels containing chrominance information. It is a linear colorspace.

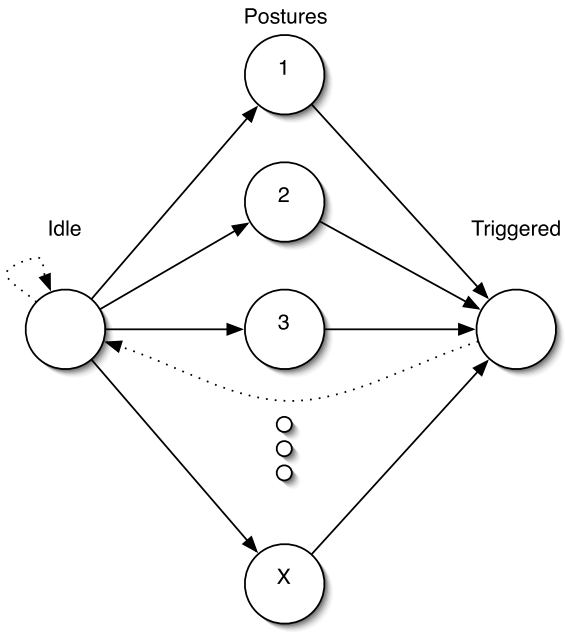


Figure 3.6: The state machine of a VICon with a one-of-X classification network. Such a VICon is capable of being triggered by any one of the postures in a given vocabulary.

In the second case of cluttered images, we have chosen to directly model the clutter in the neural network. The data shows that in the case of a static, uncluttered, background, a very simple network can be used to recognize the static postures in the gesture set. However, for more realistic scenarios (cluttered background), even complex networks (stereo with relatively good input resolution) have trouble *learning* the true discriminating features of the postures apart from the background clutter.

		Monocular		**Binocular**	
No. Hidden Nodes	**Input Size**	768	192	768	192
5		84|72	94|86	98|84	98|78
15		94|86	98|86	98|86	98|86
25		94|86	98|86	98|86	98|86

Table 3.4: Neural Network one-of-X posture classification for a four gesture plus silence vocabulary using uncluttered images. Results shown using training % | testing %.

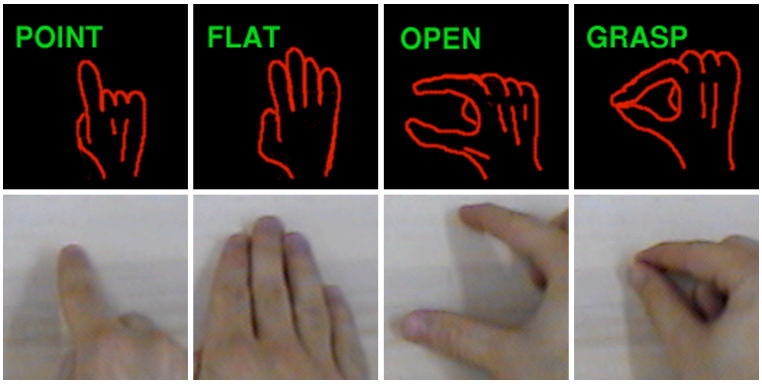


Figure 3.7: The gesture vocabulary in the one-of-X network classification.

		Monocular		Binocular	
No. Hidden Nodes	**Input Size**	768	192	768	192
5		77|70	82|73	90|66	86|63
15		77|72	90|72	95|68	99|72
25		81|72	87|69	83|67	99|71

Table 3.5: Neural Network one-of-X posture classification for a four gesture plus silence vocabulary using cluttered images. Results shown using training % | testing %.

Experiments Using the 4D-Touchpad Platform

We have also carried out a set of experiments that use the 4D-Touchpad platform. The gestures in these experiments have been integrated into applications that run on the platform and have been tested in a demonstration setting. In the experiments, we have implemented those gestures required for mapping the conventional interface actions; recall from Section 2.4.3 that these gestures are pressing, grasping, moving, and dropping an interface component. We include only the pressing and grasping here (Figure 3.8), and we leave the moving to the next section because it is a quantitative, dynamic gesture. As will be obvious, the dropping gesture can be handled in a manner similar to how we have handled the pressing and grasping.

We take a slightly different approach to implementing the classification networks when modeling the gestures in this experiment. A special case of the one-of-X network is constructed when the size of the gesture vocabulary is one. In such a network, only one output node is used per network, and it can take either an *on* or an *off* value, which is actually given as a real-value ranging from 0 to 1 interpreted as the network confidence.

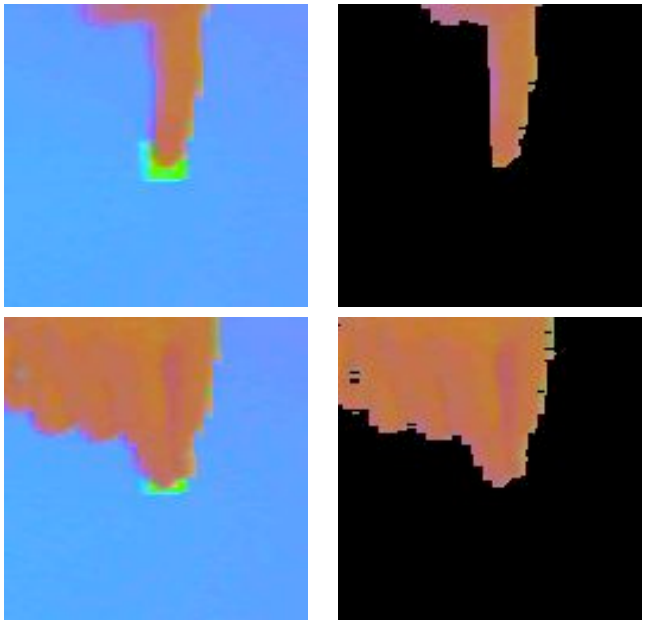


Figure 3.8: (Top) Pressing Gesture. (Bottom) Grasping Gesture. (Left) Original Image. (Right) Segmented Image.

Thus, we build a separate network for the pressing and grasping (and dropping) gestures which run in parallel on a given input stream. Such a protocol is slightly more computationally expensive since two network are running in place of one, but the extra expense is worth the added computation, as is shown in the recognition results (Table 3.6). In these experiments, we use the segmented image data that is available on the 4DT and subsample each local VICon region into a 16×16 image resulting in 512 inputs to the network (there is binocular image data). Example images are given in Figure 3.8. The networks both have 20 nodes in the hidden layer. In the results, we distinguish between positive cases and negative cases because such an on/off network can be more susceptible to false-positives. We note the recognition rate for these experiments is nearly 100% for the testing data, which is substantially better than in the previous one-of-X experiments. However, the two experiments are not directly comparable because in the earlier experiment we did not use the calibrated and segmented image data as we did in this case.

Next, we present an experiment testing the accuracy and spatial sensitivity of the button press gesture since this particular gesture is integral to the conventional interface technology we are wrapping via the 4DT. We display an array of square buttons that are

	Pressing		Grasping	
	Training	Testing	Training	Testing
Positive	2051/2051	2057/2058	366/367	364/367
Negative	1342/1342	1342/1342	480/480	483/483

Table 3.6: On/Off neural network posture classification for the pressing and grasping gestures. Input images are segmented in the standard 4D-Touchpad pipeline.

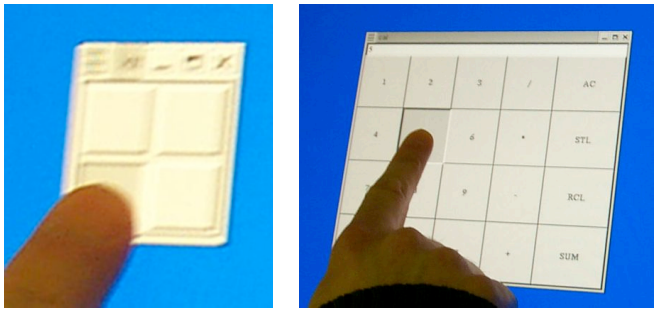


Figure 3.9: Left picture shows the scene of when the finger is trying to trigger a button of 40×40 pixels. Right picture shows a user is controlling an X window program using finger press gestures.

adjacent to each other with no buffer-space. Then, the system randomly chooses one of the buttons and instructs the user to press it. We vary the size of the button from 20×20 pixels to 75×75 pixels and repeat the experiment for each size. Figure 3.9 shows the scene when the size of the button is 40×40 pixels; here, we see that the size of the user's finger tip is about 40 pixels wide at the display resolution. Figure 3.10 shows the testing results. In the graph, we see the accuracy of the press recognition rapidly increases with the size the button. We also note that, as expected, for button sizes smaller than the fingertip resolution (about 40 pixels), the accuracy is much worse. Figure 3.9 also shows an application using the button press gesture to control a calculator program in the X-Windows system.

3.3.2 Neural Networks for Tracking Quantitative Gestures

The third class of gestures we defined in Section 3.1.2 is the class of parametric, dynamic gestures. For both the communicative and manipulative gesture cases, correct parameter calculation is key to the natural and efficient interaction with this gesture class. Many approaches are available for tracking in the vision literature, including template tracking [65] and mean-shift tracking [32], for example. We present a method that couples

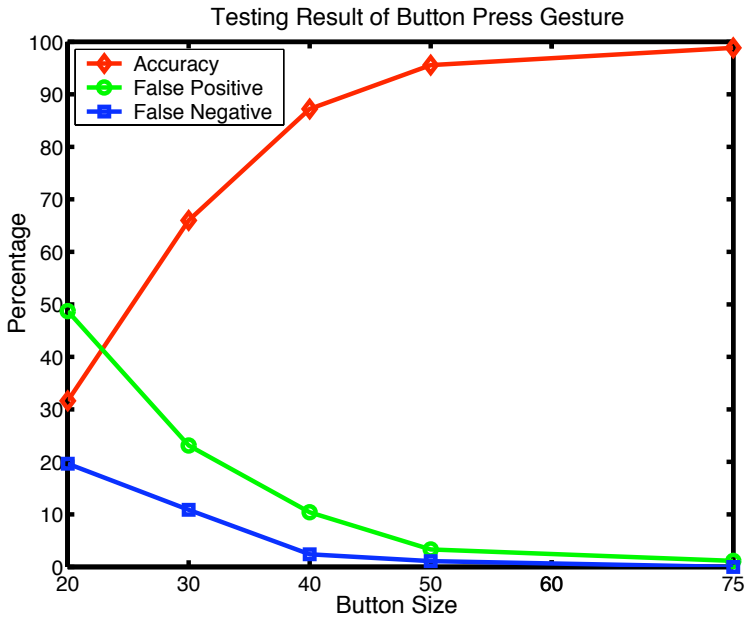


Figure 3.10: 4DT button-pressing accuracy experiment result.

multi-layer neural networks with linear prediction. We make the observation that given the local region of the image, to track gesture translation and rotation, we can approximate the motion by treating the gesture posture as a rigid object. Given the rigid object, we use a *filtered-detection* algorithm. In each frame of the video, we use a network to locate a small set of feature points in the local region, which are assumed to be repeatable from frame-to-frame. These feature points are filtered with a standard Kalman filter [84, 179] that assumes a constant velocity dynamics model. Under the assumption that the detector network will be able to continuously re-detect the same feature points, the result is smooth gesture motion with constant feedback to the user.

We manually annotate our training set of images (all containing the rigid posture at various locations in the local VICon region) with the location of the feature points. Then, using the standard back-propagation training algorithm, we train a real-valued network giving the locations of the feature points as the training information. Essentially, for a given posture, the network is learning to locate the set of key features, which will be used to estimate the gesture motion. Again, as input to the network we use a downsampled image with no further modification. We show a set of experiments where we have applied

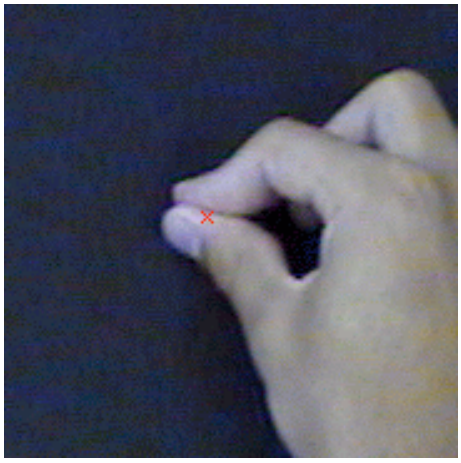


Figure 3.11: Image example with annotated grasping feature point.

this algorithm in tracking both the grasping gesture and the finger pointing gesture.

Tracking the Grasp

We show the results for an VICon that performs two degrees-of-freedom tracking (translation) that can be used to move an application object in the plane of the interface. An example image with the annotated feature point is shown in Figure 3.11. The input to the network is a coarsely subsampled image at 32×32, and the network had two outputs corresponding to the x and y pixel coordinates of the feature point.

In Table 3.7, we show the mean distance measured for both the training and testing data. The distance is calculated as the Euclidean distance between the human-annotated grasping tip pixel location and the detected location by the network. From these results, we find the location accuracy of the grasping feature point near the error expected in the human annotation, which we estimate to be around 5 pixels.

No. Hidden Nodes	**Tracking Error**	**Testing Error**
10	2.816	6.477
15	3.0336	6.347
25	3.072	6.546

Table 3.7: Mean pixel-distances for the real-valued grasping feature point locator. Data presented is the mean Euclidean error measured in pixels.

Pointing Stability on the 4DT

To investigate the stability of the location recognition and the underlying segmentation on the 4DT, we perform an experiment in which the user is asked to point and stay at an arbitrarily specified position on the screen. For this experiment, we have trained a neural network system to localize the spatial coordinates of the fingertip. Thus, for a given image region defined for an interface element, the images are processed by the network which yields a single (x, y) sub-pixel resolution tip location. Again, the input to the network is a coarse, sub-sampled image (256 total samples), and the network has 16 hidden nodes. We annotated each training datum by clicking on the fingertip location. Over 600 training samples are used and the average error for the training data is 0.0099 image pixels. We see a much more precise feature point locator in this experiment than the previous one because of the accurate segmentation we find on the 4DT platform.

To test the stability, we dynamically change the scene by rendering randomly generated background images and record the output of the pointing gesture recognizer. The stability of the system is measured by the standard deviation of the detected 2D position. On a standard flatpanel monitor with a resolution of 1280×1024 pixels, our system reports an average standard deviation of 0.128 and 0.168 pixels in the x and y direction, respectively.

3.4 Binary Gestures

Thus far, the gestures we have presented have all been so called low-level gestures, i.e. each parser is designed for one class of gestures. The power of vision-enhanced interaction lies in the high-dimensional video stream, which exposes the capability of continuous and rapidly changing interaction by the user. In this section, we present a simple extension of the low-level gestures presented earlier in this chapter. In the extension, which we term *binary gestures*, we integrate static postures and parametric gestures into a single model. Such a model would allow the user to grasp an application object, rotate it, and then drop it wherein all the processing is happening in the same VICon engine. This model represents a simple, higher-level gesture — we will extend the model even further in the next chapter with a complete gesture language model.

The proposed gesture state machine (Figure 3.12) has two states, hence the term *binary gesture*. The first state is *idle*, and in this state, the icon is simply waiting for

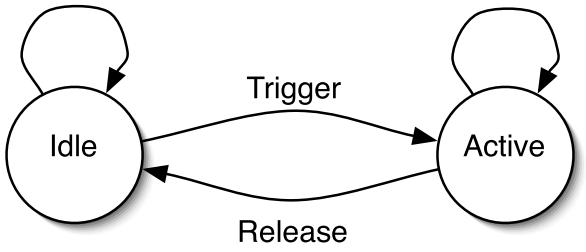


Figure 3.12: Binary gesture state model.

the trigger-posture. Upon being triggered, the icon switches into the active state. Here, both a quantitative gesture tracker and a release-posture recognizer work in parallel. While in active state, the binary gesture VICon tracks the gesture motion parameters until the dropping posture is detected. When the detection is made, the VICon stops tracking and switches back to idle state. The three gesture processors used in the construction of a binary gesture VICon are task-specific. For example, in a standard draggable item, we would use a grasping posture recognizer, the translational, parametric gesture tracker, and a dropping posture recognizer.

3.5 Conclusion

In this chapter, we presented a set of techniques for solving gesture recognition under the VICs paradigm of local, spatio-temporal pattern recognition. We focused on recognizing low-level, atomic gestures in this chapter. We presented both procedurally defined parsers and learning based parsers. We presented two methods for using neural networks in gesture analysis: (1) multi-class posture recognition and (2) parametric gesture tracking with a filtered-detection algorithm. In addition, we have given a reference to [188] for a VICs-based approach to modeling dynamic, non-parametric gestures with HMMs. In the previous section, we integrated two of the low-level gesture classes into a so-called *binary gesture*. We extend this idea in the next chapter with an approach to high-level gesture language modeling that integrates all three types of low-level gestures.

Chapter 4

A High-Level Gesture Language Model*

It seems clear that to fully harness the representative power of human gestures, static postures and non-parametric and parametric, dynamic gestures must be integrated into a single, coherent gesture model. For example, visual modeling of ASL is still limited by the lack of capabilities to handle the composite nature of gestures. To that end, we present a novel framework that integrates static postures, unparameterized dynamic gestures and dynamic parameterized gestures into a single model.

In this framework, a probabilistic graphical model is used to model the semantics and temporal patterns of different parts of a complex gesture; essentially, the model captures a high-level language (or behavioral) model. In the model, each stage of the gesture is represented as a basic language unit, which we call a *gesture word* (GWord). A GWord can be modeled as either a static posture, unparameterized dynamic gesture or a parameterized gesture. A composite gesture is composed of one or more GWords with semantic constraints. These constraints are represented in the graphical model, with nodes denoting GWords and edges describing the temporal and linguistic relationship between GWords. The parameters of the model can be learned based on heuristics or via a probabilistic framework based on recorded training data. Online gesture recognition is carried out via greedy inference on the model. Here, online means that the algorithm does not have access to future video frames.

Our proposed framework is related to work in the field of activity modeling. Bre-

*The material in this chapter is joint work with G. Ye

gler [22] abstracted human activity in a three-layered model. In the data-driven approach, regions of coherent motion are used as low-level features. Dynamic models capture simple movements at the mid-level, and HMMs model the high-level complex actions. Pentland and Liu [125] proposed Markov Dynamic Models which couple multiple linear dynamic models (e.g. Kalman filters) with a high-level Markov model. Ivanov and Bobick [77] proposed a probabilistic syntactic approach to activity modeling. In their two-layered model, a discrete symbol stream is generated from continuous low-level detectors and then parsed with a context-free grammar. Galata et al. [55] proposed an approach to learn the size of structure of the stochastic model for high-level activity recognition.

The main contribution of this chapter is to investigate a high-level language model to integrate the three different low-level gesture forms in a coherent manner. We extend the state-of-the-art in gesture modeling by relaxing the assumption that the low-level gesture primitives have a homogeneous form: e.g. all can be modeled with an HMM.

4.1 Modeling Composite Gestures

Probabilistic graphical models (PGM) are a tool for modeling the spatial and temporal characteristics of dynamic processes. For example, HMMs and Bayesian networks are commonly used to model such dynamic phenomena as speech and activity. PGMs provide a mathematically sound framework for learning and probabilistic inference. We use a relatively simple PGM structure: ergodic, causal, discrete first-order hidden Markov models.

However, most previous works in gesture and activity recognition assume a consistent model for all low-level processes (GWords). We propose to integrate multiple, heterogeneous low-level gesture processes into a high-level composite gesture. Intuitively, we combine multiple GWords to form a *gesture sentence* that corresponds to a complete interaction task. For example, grasping a virtual object $\rightarrow$ rotating it $\rightarrow$ dropping the object (Figure 4.1). The *binary gestures* presented in Section 3.4 are a simple, heuristic case of such composite gestures.

In the remainder of this section, we define notation in Section 4.1.1 and present our construction of the composite gestures using PGMs in Section 4.1.2. In Section 4.1.3 we discuss different types of GWords. We formulate the learning of the PGM in Section 4.2. The gesture inference is discussed in Section 4.3.

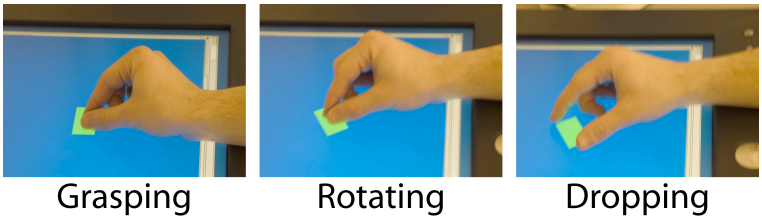


Figure 4.1: Three-stage composite gesture example

4.1.1 Definitions

Let $G = \{\mathcal{V}, \mathcal{E}\}$ be a directed graph representing the gesture language model. Each node $v \in \mathcal{V}$ in the graph corresponds to a GWord which belongs to a vocabulary $\mathcal{V}$ of size $|\mathcal{V}|$. Associated with each node v is a probability function $P(\mathcal{S}|v)$, which measures the observation likelihood of $\mathcal{S}$ for a given GWord v. Each edge $e \in \mathcal{E}$ is a probability function $P(v_j|v_i)$, where $v_j, v_i \in V$. Intuitively, the edge models the temporal relationship between successive gesture units in the composite gesture.

4.1.2 The Gesture Language

We use a *bigram model* to capture the dynamic nature of the gesture language. The bigram model represents the linguistic relationship between pairs of GWords. Formally, given a vocabulary $\mathcal{V}$, define a GWord sequence $\mathcal{W} = \{s, v_1, \ldots, v_k, t\}$ where $v_i \in \mathcal{V}$ and s, t are two special nodes (*dummy* gestures) that act as the graph source and sink. The node s is the initial state, and the node t is the accepting state of the model. Thus, a gesture is a path through the PGM starting at the source node and ending at the sink node. In Figure 4.2, we give an example PGM that can model 6 gestures. For example, the path $s \rightarrow 1 \rightarrow 3 \rightarrow 6 \rightarrow t$ is a candidate gesture sentence.

We embed the bigram language model into the PGM by associating nodes with individual GWords and assigning transition probabilities from the bigram model. For convenience, let $P(v_1) \doteq P(v_1|s)$, which can be considered as the prior of a GWord. Then the probability of observing the sequence in the bigram model is

$$P(\mathcal{W}) \doteq P(s, v_1, \ldots, v_k, t) = P(v_1) \prod_{i=2}^{k} P(v_i|v_{i-1}) \tag{4.1}$$

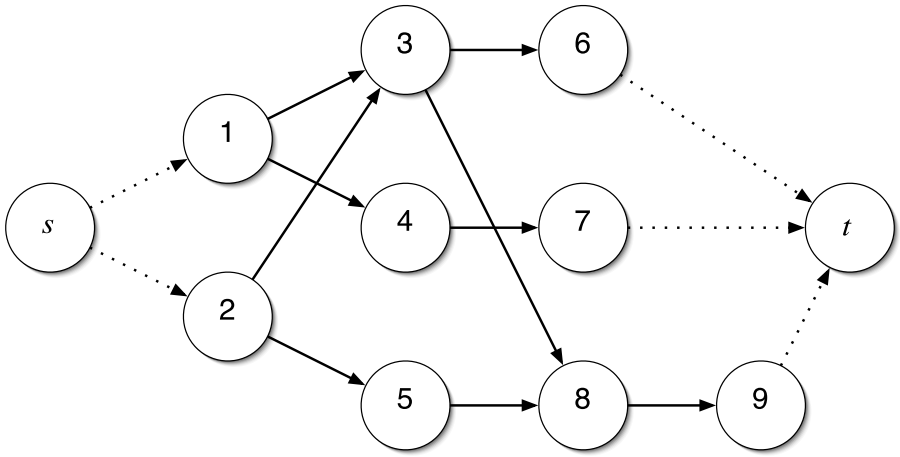


Figure 4.2: Example PGM used to represent the gesture language model. Each path beginning at node s and ending at node t is a valid gesture sentence.

As defined in Section 4.1.1, each node of the PGM models a specific GWord with its corresponding observation likelihood. Given an image sequence $\mathcal{S}$ we can construct a candidate segmentation (Section 4.3) that splits the sequence into p subsequences $\{\mathcal{S}_1 \dots \mathcal{S}_p\}$. We establish a correspondence between each of the subsequences to a GWord thus creating a gesture sentence $\mathcal{W}$. Assuming conditional independence of the subsequences given the segmentation and the observation likelihood of a subsequence only depends the corresponding GWord, the observation likelihood of the sequence is

$$P(\mathcal{S}|\mathcal{W}) = \prod_{i=1}^{p} P(\mathcal{S}_i|v_i) \tag{4.2}$$

Then, the overall probability of observing the gesture sentence is

$$\begin{aligned} P(\mathcal{W}|\mathcal{S}) &\propto P(\mathcal{W}) \cdot P(\mathcal{S}|\mathcal{W}) \\ &= P(v_1) \prod_{i=2}^{p} P(v_i|v_{i-1}) \cdot \prod_{i=1}^{p} P(\mathcal{S}_i|v_i) \end{aligned} \tag{4.3}$$

with the special source and sink node probabilities defined as $P(s) = 1$, $P(t|v \in \mathcal{V}) = 1$, $P(v \in \mathcal{V}|s) = P(v)$.

4.1.3 The Three Low-level Gesture Processes

Recall from Section 3.1.2, there are three main approaches to modeling gestures: static postures, non-parametric dynamic, or parametric dynamic. Each node corresponds to one of these three types of gesture processes. We repeat a short description of each low-level gesture here.

Static Posture

Static postures are based on the recognition of single discriminative frames of video. Hence, static postures simplify gesture processing by discarding all temporal information. For example, in the current literature, most alphanumeric symbols in ASL are represented as static postures [197]. Commonly used approaches to model the postures include appearance-based templates, shape-based models, and 3D model-based methods.

Non-parametric Dynamic

Non-parametric dynamic gestures capture temporal processes that carry only qualitative information; no quantitative information (e.g. length of hand-wave) is present. Hence, these gestures are potentially more discriminative than static postures because of the additional temporal dimension. For example, the 'j' and 'z' letters in the ASL have a temporal signature; i.e. the spatial trajectory of the finger over time is used to discriminate between the 'i' and the 'j'. Hidden Markov models [130, 160, 187, 189] and motion history images [17] are common methods used to model non-parametric dynamic gestures.

Parametric Dynamic

Parametric dynamic gestures are the most complex among the three types because they not only incorporate a temporal dimension but also encode a set of quantitative parameters. For example, in explaining the height of a person using an outstretched hand, the distance between the ground and the hand gives a height estimate. Parametric hidden Markov models [181] have been proposed to model a single spatial variable. Most of the techniques are based on visual tracking. However, we use the filtered-detection algorithm that was presented in Section 3.3.2.

The parametric dynamic gestures bring an added degree of difficulty to the recognition process because they can have too high a degree of temporal variability to be captured

by a standard models and inference algorithms. For example, Figure 4.1 shows a composite gesture for grabbing, rotating, and dropping a virtual object. In general, the moving gesture will appear quite arbitrary because the user has the freedom to navigate the entire workspace and also pause for variable amounts of time before dropping the object. To handle the variability in dynamics, we couple the parametric gesture recognition with posture recognition that is used to *stop* the parametric tracking. The binary gestures (Section 3.4) are a simple case of this; we use this idea in the more complex language model in this chapter.

4.2 Learning the Language Model

In this chapter, we assume that the learning and the implementation of the individual low-level gesture units are handled separately (in Section 4.4 we discuss the implementations used in our experiments) and the observation probabilities of these units are normalized on the same scale. Here, we address the problem of learning and inference on the high-level gesture model. Specifically, we learn the parameters of the bigram language model (Equation 4.1). We describe three basic techniques to learn the bigram model: supervised, unsupervised, and hybrid.

4.2.1 Supervised Learning

Given a set of n labeled GWord sequences $\mathcal{L} = \{\mathcal{W}_1 \dots \mathcal{W}_n\}$ with $\mathcal{W}_i = \{s, v_{(i,1)}, \dots, v_{(i,m_i)}, t\}$ where $m_i + 2$ is the length of sequence $\mathcal{W}_i$ and $v_{(i,j)} \in \mathcal{V}$. The GWord prior is given by

$$P(v_k) = \frac{\sum_{i=1}^{n} \delta(v_k, v_{(i,1)})}{n} \tag{4.4}$$

where $\delta(\cdot)$ is the Kronecker delta function and $v_k \in \mathcal{V}$. The prior computes the probability that a gesture sentence begins with a certain GWord. The bigram transition probability is given by the following equation:

$$P(v_l|v_k) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m_i-1} \delta(v_k, v_{(i,j)}) \cdot \delta(v_l, v_{(i,j+1)})}{\sum_{i=1}^{n} \sum_{j=1}^{m_i-1} \delta(v_k, v_{(i,j)})}. \tag{4.5}$$

Intuitively, Equation 4.5 measures the transition probability from a GWord v_k to another GWord $v_l \in \mathcal{V}$ by accumulating the number of bigram pairs $v_k \rightarrow v_l$ and normalizing by

the number of bigrams beginning with v_k.

4.2.2 Unsupervised Learning

Given a set of n unlabeled image sequences $\mathcal{U} = \{U_1 \dots U_n\}$. We generate an initial bigram model M_0 in a uniform fashion based on the PGM. We can use additional heuristics based on the specific application to refine the uniform initialization. We train the bigram model using an EM-like [108] iterative algorithm.

1. $M \leftarrow M_0$.
2. Compute the best labeling (Section 4.3) for each sequence in $\mathcal{U}$ based on the current bigram model M.
3. Using the supervised learning algorithm (discussed previously), refine the bigram model M.
4. Repeat until a fixed number of iterations is reached or the change of the bigram model in successive iterations is small.

4.2.3 Hybrid Learning

Given a set of labeled GWord sequences $\mathcal{L}$ and a set of unlabeled image sequences $\mathcal{U}$. We generate an initial bigram model M_0 using the labeled sequences with the supervised learning algorithm discussed above. Then, we refine the bigram model in an iterative manner similar to the one used in unsupervised learning.

1. $M \leftarrow M_0$.
2. Compute the best labeling (Section 4.3) for each sequence in $\mathcal{U}$ based on the current bigram model M. Call the labeled sequences $\hat{\mathcal{U}}$.
3. $\mathcal{T} = \bigcup(\mathcal{L}, \hat{\mathcal{U}})$.
4. Using the data $\mathcal{T}$ perform the supervised learning algorithm (discussed previously) to refine the bigram model M.
5. Repeat until a fixed number of iterations is reached or the change of the bigram model in successive iterations is small.

4.2.4 Discussion

The core equations of these learning algorithms are (4.4) and (4.5). Since the learning is essentially histogramming the training data into the bigram model, the relative amount of training data between different gesture sentences is important. Any bias toward a particular sequence present in the training data will be present in the resulting bigram model.

In the unsupervised case, the learning relies heavily on the observation likelihoods of the low-level gesture recognizers, which drive the labeling algorithm (Section 4.3) in the case of a uniform bigram model. Thus, spurious labelings may result. However, the unsupervised learning removes the need for tedious data-labeling. The hybrid learning bootstraps the bigram model with relatively few labeled video sequences making it less likely to learn an incorrect model while also allowing unsupervised data to be used.

4.3 Inference on the PGM

Given an image sequence $\mathcal{S}$ of length m and a PGM with an embedded bigram model, we construct the inference problem as the search for the best labeling $\mathcal{L}$ of $\mathcal{S}$ that maximizes the overall probability given in Equation 4.3. Formally, the inference problem is stated as

$$\{v_1^* \dots v_p^*\} = \arg \max_{\mathcal{W}=f(\mathcal{S})} P(\mathcal{W}) \cdot P(\mathcal{S}|\mathcal{W}) \tag{4.6}$$

where $\mathcal{S} \doteq \{\mathcal{S}_1 \dots \mathcal{S}_p\}$, $f(\mathcal{S}) = \{v_1 \dots v_p\}$ is a one-to-one mapping from a sequence segmentation to a gesture sentence, and p is unknown. Let $g(\cdot)$ be the mapping from subsequence $\mathcal{S}_i$ to a GWord v_i; it is computed using the maximum-likelihood criterion:

$$g(\mathcal{S}_i) = \arg \max_{v_j \in \mathcal{V}} P(\mathcal{S}_i|v_j) \tag{4.7}$$

Theoretically, the inference problem in Equation 4.6 could be solved by an exhaustive search. However, the combinatorial complexity is prohibitive. Furthermore, the fundamental differences in the three types of low-level gesture processors makes the optimization more difficult. In addition, online processing is a prerequisite for human-computer interfaces. Recall that each node in the model is a low-level gesture. Thus, for each step through the model, there is an associated reaction in the computer interface. Also, since the

processing is online, any delay in choosing a path is immediately evident to the user in the form of interface latency. Therefore, many powerful algorithms, like dynamic programming, are implausible for our model. Thus, we propose a sub-optimal, greedy algorithm.

Initialize the algorithm by setting $v_0 = s$ and $\mathcal{S}_0 = \emptyset$. At stage t in the algorithm processing, we search for the transition from v_t to v_{t+1} that maximizes the product of the transition probability $P(v_{t+1}|v_t)$ and the observation probability $P(\mathcal{S}_{t+1}|v_{t+1})$. Call this product, the *step probability*. The beginning of subsequence $\mathcal{S}_{t+1}$ is set as the end of $\mathcal{S}_t$. To determine the end of the subsequence S_{t+1} and thus make the greedy path choice, we incrementally increase the length of the subsequence until the path to one of the children c meet both of the following two conditions.

1. The observation probability of the child passes a threshold τ_c. We discuss a supervised technique for learning the node thresholds below.

2. The step probability of c is highest among all of the children of node v_t. Formally, $c = \arg\max_{v_{t+1}} P(v_{t+1}|v_t) \cdot P(\mathcal{S}_{t+1}|v_{t+1})$.

In Figure 4.3 we show a graphical depiction of a stage in the middle of the greedy algorithm. In the figure, at stage $t+1$, child c_2 of node v_t is chosen. We see that at the end of stage $t+1$ the end of sequence $\mathcal{S}_{t+1}$ has been determined.

We learn the individual node thresholds using a supervised technique. Given a set of labeled GWord sequences and segmented image sequence pairs $(\mathcal{W}_i, \mathcal{S}_i)$. We pose the problem of determining the threshold τ_v for GWord $v \in \mathcal{V}$ as finding the minimum observation probability for all occurrences of v:

$$\tau_v = \min_{(\mathcal{W}_i, \mathcal{S}_i)} \min_{v_i \in \mathcal{W}_i \text{ and } \delta(v_i, v)} P(\mathcal{S}_i|v). \tag{4.8}$$

First, we initialize all the thresholds to 0, $\tau_v = 0, \forall v \in \mathcal{V}$, to handle the case where v does not occur in $\mathcal{L}$. Then, for all GWords $v \in \mathcal{V}$ we compute τ_v according to Equation 4.8.

4.4 Experimental Setup

We analyze the proposed model for recognizing composite gestures by constructing a gesture set and the corresponding PGM. We use the Visual Interaction Cues (VICs) paradigm (Chapter 2) to structure the vision processing and use the 4D Touchpad (Section 2.6) as the experimental platform.

4.4.1 Gesture Set

The goal of the proposed framework is to facilitate the integration of different types of gestures (Section 4.1.3) and thus, natural interaction in virtual environments. To that end, we present an experimental gesture set with ten elements (GWords) with each of the three gesture types represented.

Low-Level Gwords

The gesture set is designed to be used in general manipulative interfaces where actions such as selecting, grasping, and translating are required. Table 4.1 contains graphical depictions of each GWord. For dynamic gestures, we show three example images during the progress of the gesture.

- **Press** is the static posture of a single finger activating the interface component.
- **Left** is a dynamic, non-parametric motion of a finger to the left with respect to the interface component.
- **Right** is a dynamic, non-parametric motion of a finger to the right with respect to the interface component.
- **Back** is a dynamic, non-parametric retraction of the finger off the interface component.
- **Twist** is a clockwise twisting motion of a finger atop the interface component (dynamic, non-parametric).
- **Grab 1**. The first grabbing gesture is the dynamic, non-parametric motion of two fingers approaching the interface component open and closing once they have reached it.
- **Grab 2**. The second grabbing gesture is the dynamic, non-parametric motion of two fingers approaching the interface component open and remaining open upon reaching it.
- **Track** is a parametric gesture that tracks two translational degrees-of-freedom.
- **Rotate** is a parametric gesture that tracks one rotational degree-of-freedom.

- **Stop** is a static posture represented by an open hand atop the interface component.

Probabilistic Graphical Model

We construct and train a probabilistic graphical model to be the *interaction language*. Figure 4.4 is a graphical depiction of the PGM; for clarity, we have not drawn any edges with zero probability in the bigram language model from supervised learning. A simple gesture sentence is thus **Press** → **Left**: the user approaches an interface component with an outstretched finger and then swipes his or her finger to the left. For example, such composite gesture could be used to delete an interaction component. A more complex gesture sentence involving all three types of low-level GWords is **Grab 1** → **Track** → **Stop**. This gesture sentence could be widely used in virtual reality to grab and move virtual objects.

Implementation of Low-Level GWords

As discussed in Section 4.1.3, we include three types of low-level gesture processing: static posture, non-parametric dynamic, or parametric dynamic. In this section we discuss the construction of these low-level processors for our experimental setup. However, from the perspective of the PGM framework, the specific construction of the low-level processors is arbitrary.

Static Posture. Static postures are based on the recognition of single discriminative frames of video. A multitude of potential methods exist in the literature for such recognition: SIFT keys [96] and Shape Contexts [12] for example. We implement the static postures using the on/off discriminative neural network approach presented in Section 3.3.1.

Non-parametric Dynamic. We model the dynamics of the motion of the finger using discrete forward HMMs. For a complete discussion of our technique, refer to [188, 189, 190]. Instead of directly tracking the hand, we take an object-centered approach that efficiently computes the 3D appearance using a region-based coarse stereo matching algorithm in a volume around the interaction component. The appearance feature is represented as a discrete volume with each cell describing the similarity between corresponding image patches of the stereo pair. The motion cue is captured via differentiating the appearance feature between frames. A K-means based vector quantization [79] algorithm is used to learn the cluster structure of these raw visual features. Then, the image sequence

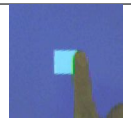
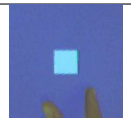
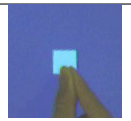
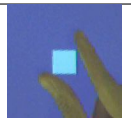
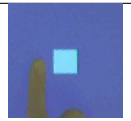
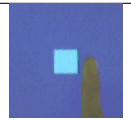
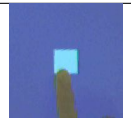
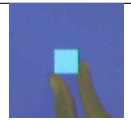
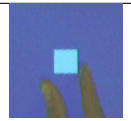
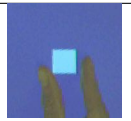
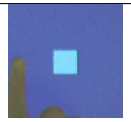
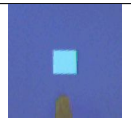
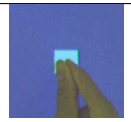
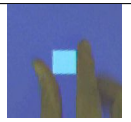
GWord	Press	Left	Right	Back	Twist	Grab 1	Grab 2	Track	Rotate	Stop
Stage 1										
Stage 2										
Stage 3										

Table 4.1: Example images of basic GWords.

of a gesture is converted to a series of symbols that indicate the cluster identities of each image pair. A 6-state forward HMM is used to model the dynamics of each gestures. The parameters of the HMM are learned via the standard forward-backward algorithm based on the recorded gesture sequences. The gesture recognition is based on the probability that each HMM generates the given gesture image sequence.

Parametric Dynamic. The implementation of a parametric, dynamic processor is dependent on the task for which it is to be used. For example, in our gesture set, we require both a translational and a rotational processor. Again, many potential techniques exist for tracking the local motion of an image patch or pair of image patches. In these experiments, we used the *filtered-detection* approach that was presented in Section 3.3.2.

4.5 Experimental Results

Figure 4.4 shows our vocabulary of 6 possible composite gestures. To quantitatively analyze the approach, we recorded a training set of 100 video sequences each corresponding to one of the 6 gestures. The length of the sequences vary from 30 to 90 frames (at 10 frames-per-second). These sequences were not used in training the low-level gesture units. For the supervised training, we manually labeled each frame of the video sequences with a GWord. For unsupervised learning, we initialized a uniform language model and used the algorithm in Section 4.2.2 to refine the model. After 2 iterations, the bigram model converged.

We compare the language models after supervised and unsupervised learning in Tables 4.2 and 4.3, respectively. The bigram models are presented as adjacency matrices such that each row represents the probability of transitioning from a GWord (leftmost column) to other GWords (or itself). It can be seen that the 2 PGM bigram models have similar structure. It shows that even without good heuristics or labeled data, our unsupervised learning algorithm can still capture the underlying language model from raw gesture sequences.

However, there are differences worth mentioning. For example, the prior for **Stop** from unsupervised learning is 0.03, but there are no sequences in the training corpus that begin with it. This is caused by the failure of the inference algorithm given a uniform bigram language model. Second, we see a difference in the self-transition probability for the **Press** GWord. In the labeled data, we fixed the duration of **Press** to one frame, but

Prior											
GWord	Left	Right	Back	Twist	Grab 1	Grab 2	Track	Rotate	Press	Stop	t
	0	0	0	0.167	0.167	0.167	0	0	0.5	0	0
Bigram Model											
GWord	Left	Right	Back	Twist	Grab 1	Grab 2	Track	Rotate	Press	Stop	t
Left	0.94	0	0	0	0	0	0	0	0	0	0.06
Right	0	0.93	0	0	0	0	0	0	0	0	0.07
Back	0	0	0.84	0	0	0	0	0	0	0	0.16
Twist	0	0	0	0.93	0	0	0	0	0	0	0.07
Grab 1	0	0	0	0	0.94	0	0.06	0	0	0	0
Grab 2	0	0	0	0	0	0.94	0	0.06	0	0	0
Track	0	0	0	0	0	0	0.96	0	0	0.04	0
Rotate	0	0	0	0	0	0	0	0.95	0	0.05	0
Press	0.33	0.33	0.33	0	0	0	0	0	0	0	0
Stop	0	0	0	0	0	0	0	0	0	0.7	0.3
t	0	0	0	0	0	0	0	0	0	0	1

Table 4.2: Language Model (Priors and Bigram) using supervised learning.

Prior											
GWord	Left	Right	Back	Twist	Grab 1	Grab 2	Track	Rotate	Press	Stop	t
	0	0	0	0.1	0.11	0.16	0	0	0.6	0.03	0
Bigram Model											
GWord	Left	Right	Back	Twist	Grab 1	Grab 2	Track	Rotate	Press	Stop	t
Left	0.91	0	0	0	0	0	0	0	0	0	0.09
Right	0	0.88	0	0	0	0	0	0	0	0	0.12
Back	0	0	0.83	0	0.01	0	0	0	0	0	0.16
Twist	0	0	0	0.95	0	0	0	0	0	0	0.05
Grab 1	0	0	0	0	0.82	0	0.14	0	0	0.02	0.02
Grab 2	0	0	0	0	0	0.77	0.04	0.15	0	0.04	0
Track	0	0	0	0	0.02	0	0.77	0.03	0	0.16	0.02
Rotate	0	0	0	0	0	0.01	0.03	0.90	0	0.06	0
Press	0.02	0.02	0.03	0	0	0	0	0	0.91	0	0.02
Stop	0	0	0	0	0	0	0	0	0	0.77	0.23
t	0	0	0	0	0	0	0	0	0	0	1

Table 4.3: Language Model (Priors and Bigram) using unsupervised learning.

Gesture Sentence	Supervised %	Unsupervised %
Press → Left	97.3	97.3
Press → Right	85.7	78.6
Press → Back	88.9	90.4
Twist	96.4	96.4
Grab 1 → Track → Stop	93.3	82.1
Grab 2 → Rotate → Stop	97.9	97.9

Table 4.4: Recognition accuracy of the PGM used in our experimentation.

with a uniform bigram model, a static posture can last for several consecutive frame via self-transition.

During testing, we used the proposed greedy inference algorithm to analyze the video sequences. In Table 4.4, we present the recognition accuracy for the gestures for both language models. For each sequence, we compared its known composite gesture identity with the GWord output of the PGM. We consider the output correct if it matches the GWord sentence at every stage.

We can see from the results that the proposed high-level gesture language modeling can recognize compositions of heterogeneous low-level gestures. These composite gestures would be impossible to recognize using a single, uniform model-type for each gesture. Our formulation takes advantage of high-level linguistic constraints to integrate fundamentally different low-level gesture units in a coherent probabilistic model.

However, the recognition accuracy for gesture **Press → Right** and gesture **Press → Back** are relatively poor. From visual inspection of the recognition algorithm's output, we find that this is due to the greedy algorithm. The **Left**, **Right**, and **Back** are modeled with HMMs and trained with relatively long sequences (e.g. 20 frames). However, during inference, the greedy algorithm jumps to a conclusion based on an shorter subsequences (e.g. 7 frames). In our experiments, we see a bias toward the **Left** GWord for these incomplete subsequences.

The recognition results from the supervised and the unsupervised learning are comparable. This suggests that our linguistic approach to gesture recognition can perform well without a heuristic prior or manually labeled data. Hence, our method is less susceptible to the curse of dimensionality which, in our case, is that the amount of data (labeled, for supervised learning) required for learning generally increases exponentially with the number

of GWords.

4.6 Conclusion

We have presented a linguistic approach to recognize composite gestures. The composite gestures consist of three different types of low-level atoms (GWords): static, posture-based primitives; non-parametric dynamic gestures; and parametric, dynamic gestures. We construct a coherent model by combining the GWords and a high-level language model in a probabilistic framework that is defined as a graphical model. We have proposed unsupervised and supervised learning algorithms; our results show that even with a uniform initialization, the PGM can learn the underlying gesture language model. By combining the PGM and the greedy inference algorithm, our method can model gestures composed of heterogeneous primitives. To the best of our knowledge, this is the first composite gesture model comprised of different, low-level gestures primitives.

Our approach allows the inference of composite gestures as paths through the PGM and uses the high-level linguistic constraints to guide the recognition of composite gestures. However, the proposed greedy inference algorithm will make locally optimal decisions since it is operating online. Our experimentation shows that this greedy path selection does not present a problem for the majority of the composite gestures in a moderate-sized vocabulary. Furthermore, even in the offline case, the heterogeneous, low-level gesture processes make an exhaustive search through all composite gesture sequences computationally prohibitive.

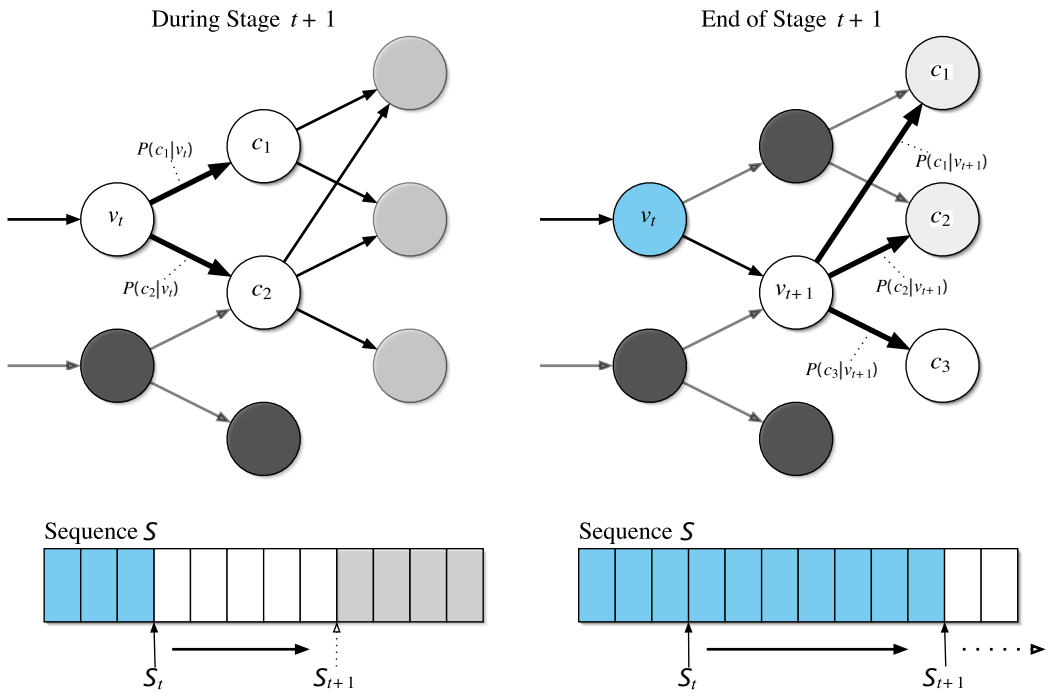


Figure 4.3: Graphical depiction of two stages of the proposed greedy algorithm for computing the inference on the PGM. Dark gray nodes are not on the best path and are disregarded, and blue represents past objects on the best path.

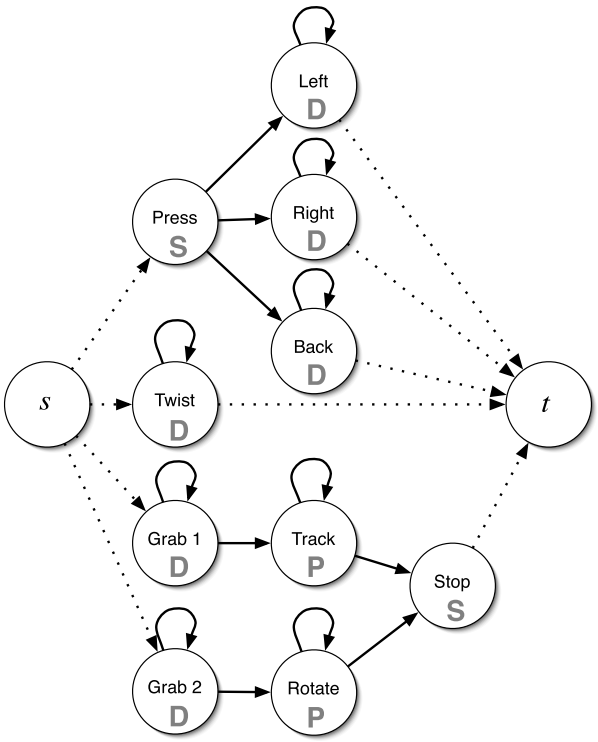


Figure 4.4: The probabilistic graphical model we constructed for our experimental setup. Edges with zero probability are not drawn. The nodes are labeled as per the discussion in Section 4.4.1. Additionally, each node is labeled as either **P**arametric, **D**ynamic, non-parametric, or **S**tatic posture.

Chapter 5

Region-Based Image Analysis

In this chapter, we study a region-based approach to image analysis that is motivated by the large-scale interaction problem that was presented in Section 1.1.2. With the VICs approach and the gesture modeling presented in the previous chapters, we have laid the foundation for such large-scale interaction. Since the complete problem is beyond the scope of this dissertation, we focus on one of the remaining problems: localization in *a priori* unknown environments using passive, video sensing exclusively. Here, localization is loosely defined because the problem is posed in the context of large-scale interaction, and the definition of localization varies depending on the approach.

A well-studied approach (Section 1.4.5) is to build a complete metric reconstruction of the scene, which would enable later localization queries and permit a variety of methods to track the user motion (pose estimation) in the environment. Such methods can be traced back to Marr's primal sketch theory [100]. Recent examples of the pose estimation methods are Iterative Closest Points [13, 26], the normal flow constraint [111, 174], a hybrid of the two [110], disparity-space tracking [41], and structure-from-motion [10, 164]. However, these approaches suffer from a few drawbacks:

1. In general, the methods make an underlying assumption that adequate texture exists in the scene. The texture facilitates either precise, discriminative point-feature extraction and matching, or dense image gradient fields.

2. Point-based geometry methods tend to have very high storage requirements.

3. They cannot easily re-initialize. In many large-scale interface settings, the tracking system can fail for a few frames due to a large occlusion, dropped video, or motion

that violates a differential assumption. In such cases, re-initialization could mean searching through a very large scene structure database.

4. The correspondence problem is difficult, in general. Methods that rely exclusively on the correct matches tend to lack robustness. However, this difficulty has been widely recognized and recently, there are more robust approaches in correspondence: e.g. random sample and consensus [48] and the SIFT descriptor [96].

To avoid these issues, we propose a strategy that does not perform a full scene reconstruction. We claim that, in the context of large-scale interfaces, the localization problem can be approached from an appearance-based perspective. Thus, we model an environment as a set of *special* surfaces (or volumes), which we will more precisely define in the chapter text. We use the appearance signature of these surfaces to index into the scene database. In order to be used for interaction, we need to define interface component mappings (Section 2.2.1), which will use a strictly local set of geometric information.

In this chapter, we discuss a novel approach to the image modeling required for such an approach to localization. We make the observation that *large, homogeneous* regions are good image elements for such image description and relation. Homogeneous regions can be described with few parameters (on the order of 10) thereby replacing the redundancy in thousands of pixels with a few parameters. In addition, we focus on *large* regions because they are stable in the sense that they will be viewable from many of the images and their low-dimensional description will not change drastically when viewed from different viewpoints. Note, we will define *homogeneous* and *large* more concretely.

The large, homogeneous regions are the representative features we use in comparing images. Two images are said to be *overlapping* if there is any scene content common to both of the images. While one can quantify this overlap by counting the number of pixels in common between the two images, we currently treat it as a "yes" or "no" question. In either case, the overlap between pairs of images induces an ordering on a set of images taken from an environment. In the "yes" or "no" case, for a given image $\mathbf{I}$ in the database, the associated ordering is the list of images that have overlap followed by the list of images that do not have overlap. We make a concrete problem statement for this chapter:

> Given a set of unordered images $\mathcal{E}$ of an *a priori* unknown environment, construct the ordering of the images. The ordering should facilitate later *mapping* queries such as, for a given image of the environment $\mathbf{I} \in \mathcal{E}$ return the subset of images which contain overlapping scene content $\mathcal{O} \subset \mathcal{E}$.

In the chapter, we begin with a survey of the image modeling and description techniques (Section 5.1). Then, we cover our proposed image modeling algorithms, which is composed of two parts: defining a set of image feature spaces (Section 5.3) and extracting the large, homogeneous regions (Section 5.2). We include a set of experimental results (Section 5.7).

5.1 Related Work in Image Modeling

We consider the problem of ordering a set of images of the same environment. Matching (or registering) differing views of a scene to each other has received much attention over the last decade. When matching the images, one must incorporate some distance measure to use in matching pairs of images. A distance measure has two parts: data representation and distance function. In most cases, the appropriate distance function is implied by the data representation. Thus, we will focus on the data representation in the early parts of this chapter and leave the distance function to the experiments section. In this section, we discuss two of the basic approaches at modeling images: local, pixel-level modeling and global, entire-image modeling.

5.1.1 Local Methods

Local, pixel-level methods focus on finding salient points in images. The general idea of such approaches is to locate regions of high texture content using an interest operator, and to then create indices for matching. The key to good performance is to create interest operators and match indices that are insensitive to geometric and photometric image distortions. The advantages of such approaches are the robustness to occlusion, changes in lighting, and moderate changes of viewpoint. The disadvantages are the need to identify such local image regions, they often require a large amount of storage, and they (typically) the use of only gray-scale image projections. In particular, large areas of the image are potentially discarded as "untextured" and therefore unusable by the method.

The first so-called interest operator was proposed by Moravec [109], which detects points with high-contrast neighborhoods. The Moravec operator is rotationally invariant. Another rotationally invariant interest operator is the Harris corner detector [69], which performs a local gradient eigen-analysis to select points with neighborhoods whose gradient is varying in both image dimensions. Since the interest points will be matched across images,

repeatability is an important characteristics; the Harris detector has been shown to have a high repeatability rate [148].

The pioneering work of Schmid and Mohr [149] emphasizes the invariance properties of both detection and description of the interest points and the local regions surrounding them. They use local, differential grayvalue invariants in a multiscale representation at a number of interest points (Harris corners) in the image for description. The invariants are the local differential *jets* from Koenderink and van Doorn [88]. Their representation is robust to similarity transforms and partial visibility. Additionally, they include information from multiple scales yielding a scale-invariant (up to the discrete scale-quantization) representation. To match, they propose a fast multidimensional hash-table voting algorithm that is robust to mismatches and outliers.

Schmid and Mohr's work gave rise to numerous related techniques, which we summarize here. Lowe [95, 96] proposed a scale-invariant feature transform (SIFT). The interest point, or key, locations are identified with a staged filtering approach that searches through a discrete scale-space [93] for minima and maxima of a difference-of-Gaussian function. For representation, the image neighborhood around each key location is assigned a canonical orientation in accordance with the local image gradients. Then, the feature vector is constructed by *orientation planes*. A local image region can be separated into a set of orientation planes each consisting of only the gradients corresponding to that orientation. The keys are invariant to image translation, scaling and rotation, and partially invariant to illumination changes. Since the keys require a relatively high storage footprint, Ke and Sukthankar [85] propose an extension of Lowe's SIFT [95] method, PCA-SIFT. While SIFT patch descriptors are constructed by smoothed orientation histograms, the PCA-SIFT patch descriptors is based on the projection of the patch gradient maps into a low-dimensional eigenspace.

In recent years, many researchers have proposed affine-invariant interest points and features. Lazebnik, Schmid and Ponce [90] detect interest points in the images by local maxima of the Laplacian in scale-space. Then, for each maxima, the local image region, at the appropriate scale, is then normalized based on the second-moment matrix resulting in affine-invariant patches. The normalized patches are represented using intensity-domain spin images, a two-dimensional histogram of brightness values. The two dimensions of the histogram are the distance from the patch-center and the intensity value.

Tuytelaars and van Gool [170] propose detection of regions by finding intensity-

based local extrema, constructing an irregularly shaped blob, and then fitting an affinely invariant ellipse (equivalent to the irregularly shaped blob up to the second-order moments). The regions are described by *Generalized Color Moments*, which implicitly characterize the shape, intensity, and color distribution of the region pattern in a uniform manner. They couple the image description with Harris corners and apply it to the wide-baseline stereo problem. Related work in wide-baseline stereo using interest region based techniques include the Maximally Stable Extremal Regions by Matas et al. [102, 103] and the scale-invariant, normalized affine-corner pairs of Tell and Carlsson [166].

The last set of techniques uses a maximization of the Shannon entropy measure [36] in the image signal to detect and characterize salient points. The idea is to define saliency in terms of local signal complexity. Gilles [59] uses salient image patches to register aerial images; he builds intensity histograms of local image patches. He uses the entropy of these histogram to characterize their saliency. However, Gilles fixed a global-scale for the size of the patches per image. For the case of aerial satellite imagery, the fixed scale is acceptable, but in the general case, it is not. Kadir and Brady's scale-saliency technique [83] extended Gilles's saliency detector to incorporate patch scale. They search for clusters of high-entropy in scale-space and use them as the interest points. Hare and Lewis [68] use the scale-saliency interest points for image matching.

[50] argue that one should fuse the characterization with the detection because if they are separate, then it is possible the detector may find regions whose description will not be salient. [50] use geometric features (specifically, Harris corners [69]) to describe the local image patch. Hence, they improve the robustness of the patch description to photogrammetric and geometric changes [150]. They incorporate Kadir and Brady's [83] scale-selection technique. They compare their proposed descriptor against the standard descriptor and find that their method significantly reduces the number of false-matches.

Mikolajczyk and Schmid [106] evaluated the performance of several local image descriptors. Their evaluation tested the descriptors' stability to rotation, scaling, affine transformations, and illumination changes. The study showed that SIFT [96] features performed the best over all conditions.

5.1.2 Global Methods

The global methods attempt to capture the most important content of the image as a whole. Such methods, in general, attempt to form a low-order summary of the image. This approach is particularly appealing for image retrieval problems where the goal is to find similar, rather than exactly matching images. The advantages are that large areas of the image tend to be stable across large changes in viewpoint, spatially approximate matching is possible, and the methods generally require a small amount of storage. The disadvantages are that it is necessary to have an efficient yet stable segmentation process, it is necessary to find image cues that are themselves stable over variations in pose and lighting, and occlusion can be an issue.

One set of approaches represent the images through segmentation. The literature on segmentation is very dense, we survey a subset that includes only the image retrieval application area since it relates directly to our work. This approach is exemplified by Malik et al. [99, 24]. They attempt to group pixels that roughly correspond to objects therefore allowing image to image matching at the object level. In the modeling, they incorporate color, texture, and position features into a Gaussian mixture model (GMM). They use the Expectation-Maximization [42] with the Minimum Description Length principle [64, 137] for GMM estimation and model selection. Similarly, Greenspan et al. [63, 62] use GMM modeling in a 5D color and spatial feature space to model images and video. For image to image comparison, they directly compare the GMM using the Kullback-Leibler distance, which is an information theoretic measure of the distance between two distributions [36]. Mo and Wilson [107] extend these ideas in a multiresolution GMM.

Ruiz et al. [139] generalize the notion of the scale-space blob [93] to include color information. The scale-space blobs are analogous to a full image segmentation. They use the automatic scale-selection principle based on extrema of the normalized Laplacian [94]. Neural networks are used to learn the image categories.

Schaffalitzky and Zisserman [143] describe a texture-region descriptor that is invariant to affine geometric and photometric transformations and insensitive to the shape of the texture region. In the affine-normalized region, they compute a rotationally and scale invariant description using a statistical approach that creates a histogram of the dominant gradient at each pixel (for each scale). In their approach, *detection* of regions is solved by standard texture-based image segmentation [99].

The second set of approaches uses whole-image histograms as the modeling scheme. Swain and Ballard [163] presented the color indexing technique, which showed that color histograms of multi-colored objects can be used for efficient and accurate database querying. They introduced a new distance measure between histograms called *histogram intersection*. The color indexing can be susceptible to spatial or spectral variation in illumination. Funt and Finlayson [53] extended the color indexing technique by histogramming ratios of colors from neighboring locations, which are relatively insensitive to changes in the incident illumination. Schiele and Crowley [145] presented a different extension to the color indexing approach. They characterize appearances by joint statistics of pixel-neighborhood filter responses (e.g. gradients). Their approach will work in cases where color alone cannot be used to sufficiently describe the objects.

5.2 Image Modeling

In our work, we consider a "middle ground." Namely, our goal is to create interest operators that focus on large, homogeneous regions, and local image descriptors for these regions. A *coherent* region in an image is a connected set of (relatively) homogeneous pixels. Coherency is indifferent to the character of the homogeneity. For example, the image of a plaid shirt with its colorful, checkered pattern is considered coherent. Our image modeling is based on detecting the homogeneous regions in scalar projections of the image, i.e. the coherent regions in the image. Recall, we denote the image $\mathbf{I} \doteq \{\mathcal{I}, I, t\}$ where $\mathcal{I}$ is a finite set of pixel locations (points in $\Re^2$), I is the map $\mathcal{I} \rightarrow \mathcal{X}$ (here, $\mathcal{X}$ is some arbitrary value space), and t is a time parameter, which is disregarded in the following discussion. We describe a region, spatially, with an anisotropic Gaussian kernel:

$$K(i,r) = \frac{1}{2\pi|\mathbf{\Psi}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(i-\boldsymbol{\mu})^{\mathsf{T}}\mathbf{\Psi}^{-1}(i-\boldsymbol{\mu})\right), \tag{5.1}$$

where $i \in \mathcal{I}$ and $r \doteq \{\boldsymbol{\mu} \in \Re^2, \mathbf{\Psi} \in \mathrm{GL}(2), \mathbf{\Psi} = \mathbf{\Psi}^{\mathsf{T}}\}$ fully describe the anisotropic Gaussian. When this function is to be interpreted as a likelihood function, we will use the notation $K(i|r) \doteq K(i,r)$ to make explicit the condition that the region parameters are known. We use the Gaussians to spatially describe the region because they have relatively few parameters, are sufficient to estimate regions of arbitrary size and orientation, and do

not require the precise estimation of object boundaries. Pixels are weighted based on their proximity to the spatial mean of the Gaussian kernel.

The spatial kernels are applied to scalar projections of the image: a scalar projection is the scalar image resulting from applying a projection function $S\colon \mathcal{X} \times \mathcal{I} \mapsto \mathfrak{R} \times \mathcal{I}$ to the input image $\mathbf{I}$. We give a complete discussion of the scalar projections in Section 5.3. In a scalar projection, the appearance of a region is modeled as a constant value α with additive, zero-mean Gaussian noise $\mathcal{N}(0, \nu^2)$. Thus, an appearance model A comprises a projection function $S\colon \mathcal{X} \times \mathcal{I} \mapsto \mathfrak{R} \times \mathcal{I}$, the constant value α, and the variance of the noise ν^2. We use a one-dimensional Gaussian function to capture the region appearance:

$$F(J(i), A) = \frac{1}{\sqrt{2\pi\nu^2}} \exp\left(-\frac{(J(i) - \alpha)^2}{2\nu^2}\right), \tag{5.2}$$

where we have written $J(i)$ to mean the value of pixel i in the scalar projection image $\mathbf{J} = S(\mathbf{I})$. Again, when this function is to be interpreted as a likelihood function, we will use the notation $F(J(i)|A) \doteq F(J(i), A)$ to make explicit the condition that the appearance model A is known.

Let a set of appearance models $\mathcal{A}$, a set of spatial regions $\mathcal{R}$, and a mapping from regions to appearance $Y\colon \mathcal{R} \mapsto \mathcal{A}$ define the model $\mathbf{\Theta}$. For a given image $\mathbf{I}$, the maximum *a posteriori* solution is

$$\mathbf{\Theta}^* = \arg\max_{\mathbf{\Theta}} P(\mathbf{\Theta}|\mathbf{I}). \tag{5.3}$$

Using the standard Bayes rule, we can write the maximization in terms of the likelihood and the prior:

$$\mathbf{\Theta}^* = \arg\max_{\mathbf{\Theta}} P(\mathbf{I}|\mathbf{\Theta})P(\mathbf{\Theta}). \tag{5.4}$$

We assume the pixels are conditionally independent given the model, $\mathbf{\Theta}$. Thus, we can write the maximization over the product of pixel likelihoods and the model prior. Although a prior distribution on the model can be evaluated for a given domain of images, modeling such a distribution is, in general, very difficult. Thus, we assume the prior is uniform, which yields a maximum likelihood solution. The resulting function is the product of pixel likelihoods:

$$\mathbf{\Theta}^* = \arg\max_{\mathbf{\Theta}} \prod_{i \in \mathcal{I}} P(i, I(i)|\mathbf{\Theta}). \tag{5.5}$$

We use a mixture model [15, Ch. 2.6] where each region in the model is a component in the mixture model. We write ϕ_r to denote the mixing coefficient for component $r \in \mathcal{R}$. Recall that, given Y, the appearance models are a deterministic function of the region.

$$P(i, I(i)|\mathbf{\Theta}) = \sum_{r \in \mathcal{R}} \phi_r P(i, I(i)|r, Y(r)) \tag{5.6}$$

The following two constraints must be satisfied in order for Equation 5.6 to be considered a proper mixture distribution.

$$\sum_{r \in \mathcal{R}} \phi_r = 1 \tag{5.7}$$

$$\sum_{i \in \mathcal{I}} P(i, I(i)|r, Y(r)) = 1 \tag{5.8}$$

We take insight from the kernel-based density estimation literature [158] to expand the pixel likelihood given a region, $P(i, I(i)|r, Y(r))$. Here, K represents the spatial component and F measures the appearance component. We assume that the pixel location, i, and the pixel intensity, $I(i)$, are conditionally independent given the region, r, and the region's appearance model, $Y(r)$:

$$P(i, I(i)|r, \mathcal{A}, Y) = K(i|r)F(I(i)|r, Y(r)). \tag{5.9}$$

Thus, the spatial parameters of the region serve as the kernel-weighting in the density estimate for the appearance parameters.

5.2.1 Final Image Model

We define a set of projection functions $\mathcal{B} = \{S \colon \mathcal{X} \times \mathcal{I} \mapsto \mathfrak{R} \times \mathcal{I}\}$. Each projection defines a feature space (the resulting scalar image) in which we analyze the image (Section 5.3). The final image model is a set of Gaussian mixture models (GMM) with one per

projection, which implicitly encodes the mapping Y. Each GMM is represented in a joint feature-spatial space of three dimensions. In Figure 5.1, we give a demonstrative example using a toy 1D "image." The bold, black, solid line is the image signal, and there are three appearance models and five spatial regions. The appearance models are drawn using a green solid, blue dashed, or red dotted horizontal line at the signal intensity α for the model. The five spatial regions are drawn using the scaled, spatial kernel functions. Each region is rendered using the corresponding appearance model (green solid, blue dashed, or red dotted). Notice the weighting for each pixel is based on its proximity to the center of one of the regions.

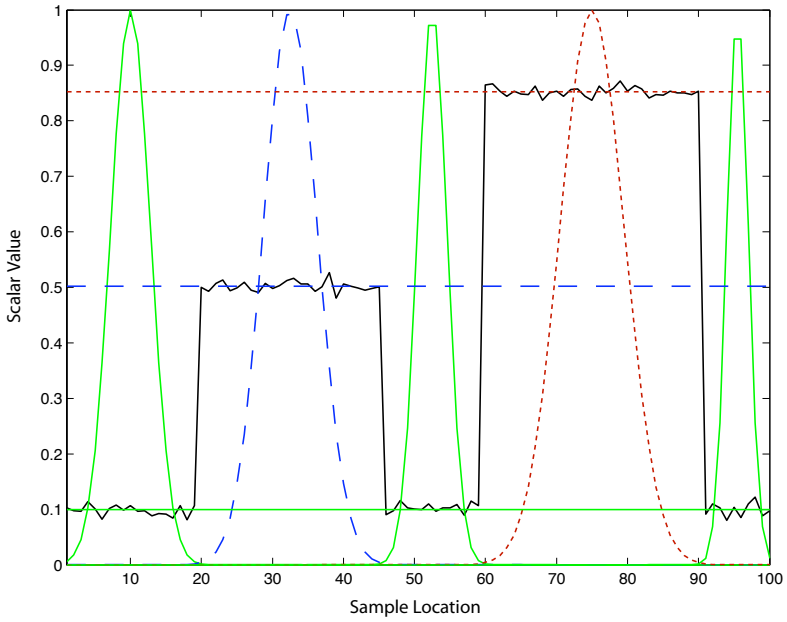


Figure 5.1: Toy 1D image example to demonstrate the parts of the model (see text for detailed explanation).

5.2.2 Estimating the Model

In this development, we assume that a set of projection functions is given. The free parameters of the model are the number of regions and both the spatial and appearance region parameters. We estimate the model independently in each projection. The most popular [105] technique to estimate the mixture model is the Expectation-Maximization

method [42]. While it has guaranteed convergence, it is very sensitive to initialization and requires the number of components as input. In our formulation, the number of components corresponds to the number of coherent regions, which is a data dependent variable. As in [24, 63], one can apply the minimum description length principle [64, 137]. However, there remains a problem due to the regular sampling of the pixels, which violates the assumption that the location samples are normally distributed.

Instead of taking this global[1] approach to estimating the model, we propose a local approach that defines an interest operator for coherent regions. Our approach approximates the maximum likelihood solution. We assume the number of regions is unknown and propose a novel objective function on the five parameters of the spatial anisotropic Gaussian.

Scale is a crucial parameter in the analysis of objects in images. There are two essential notions of scale: the *integration* scale[2] of the image content (e.g. texture or edges), and the scale of an associated spatial kernel function used to summarize image content. In both cases, there is no universally accepted method for choosing an optimal scale. Lindeberg proposed a set of scale selection principles [94] for feature detection and image matching, and a technique [93] for building a gray-level blob and scale-space blob representation of an image. Comaniciu et al. [31] proposed the variable bandwidth mean shift algorithm to solve this problem (in the context of kernel-based density estimation [30]). Collins [28] applied Lindeberg's general scale selection principles [94] to extend the kernel-based mean shift tracking to refine the scale of the object being tracked. Okada et al. [119] presented a method for the creation of an anisotropic, Gaussian scale-space by extending Lindeberg's [94] isotropic scale-space methods. In our work, we focus primarily on determining the correct scale of a spatial kernel for clustering regions of similar content.

Define a set of n fixed pixel locations $\mathcal{I}$, and a scalar image $I\colon \mathcal{I} \rightarrow \Re$. We model a region as a scaled rectangle function $\alpha\Pi(i), i \in \mathcal{I}$ with additive i.i.d. noise conforming to a zero-mean Normal distribution $\mathcal{N}(0, \nu^2)$. Let $\boldsymbol{\mu}, \boldsymbol{\Psi}$ be the spatial mean and covariance (2D) and α, ν^2 be the appearance mean and variance (1D) of the region, r. We estimate the region parameter α by minimizing the following error term

[1] In this context, the term *global* is used to mean the joint estimation of all model parameters at once. Likewise, *local* is used to mean the estimation of a single region independently from the other components of the model.

[2] A more concrete definition of integration scale is given in Section 5.3.

$$\arg\min_{\alpha} \sum_{i \in \mathcal{I}} K(i,r) \left(I(i) - \alpha\right)^2 . \tag{5.10}$$

This term follows directly from (5.2), (5.9), and the assumption of additive, normally distributed noise about α; it is the kernel-weighted estimate for α. The minimum is found by differentiating and setting the result equal to zero (we drop the subscript on the summation to simplify notation):

$$\begin{aligned} 0 &= \sum -2K(i,r)(I(i) - \alpha) \\ \sum K(i,r)\alpha &= \sum K(i,r)I(i) \\ \alpha &= \frac{\sum K(i,r)I(i)}{\sum K(i,r)}. \end{aligned} \tag{5.11}$$

Since we consider only normalized kernels ($\sum K(i,r) = 1$, proved in Appendix A.1), the expression in Equation 5.11 reduces to the *kernel-weighted mean* of the signal:

$$\alpha = \sum K(i,r)I(i). \tag{5.12}$$

We plug this kernel-weighted mean back into the original equation, and arrive at the kernel weighted variance. This function can be used to compute the spatial parameters of the region:

$$\begin{aligned} &\arg\min_{\boldsymbol{\mu},\boldsymbol{\Psi}} \sum K(i,r) \left(I(i) - \alpha\right)^2 \\ &\arg\min_{\boldsymbol{\mu},\boldsymbol{\Psi}} \sum K(i,r) \left(I(i) - \left[\sum K(i,r)I(i)\right]\right)^2 \\ &\arg\min_{\boldsymbol{\mu},\boldsymbol{\Psi}} \sum K(i,r)I(i)^2 - \left[\sum K(i,r)I(i)\right]^2 . \end{aligned} \tag{5.13}$$

However, this error function has its minimum with zero variance, which is a *degenerate* Gaussian. Physically, the function would be minimized when the region is only sampling the value at a single pixel. To regularize the error function, we include a second, additive term. While many regularizing choices are possible, we choose one that requires no parameter tuning and has a physical meaning. We minimize the squared distance between

the kernel and the signal. Since we consider normalized kernels, we include a scale factor β for the kernel. The additive, "template-matching" term is

$$\sum \left[\beta K(i,r) - I(i)\right]^2 . \tag{5.14}$$

We include a normalizing factor $\frac{1}{n}$ where n is the number of pixels and a multiplier γ that is set by hand to weight the two terms of the function. We combine (5.13) and (5.14) to yield the final objective function:

$$\arg\min_{\beta,\boldsymbol{\mu},\boldsymbol{\Psi}} \sum K(i,r)I(i)^2 - \alpha^2 + \frac{\gamma}{n}\sum \left[\beta K(i,r) - I(i)\right]^2 . \tag{5.15}$$

By taking a binomial expansion and discarding both constant and higher order terms, we use the following function to approximate (5.15) in our implementation:

$$\arg\min_{\boldsymbol{\mu},\boldsymbol{\Psi}} \frac{\sum K(i,r)I(i)^2}{\alpha^2} + \frac{\tau}{|\boldsymbol{\Psi}|^{\frac{1}{2}}}, \tag{5.16}$$

where τ is a weighting factor between the two terms. We show the derivation and proof in Appendix A. We note the appealing form of this function. It is the sum of a homogeneity term and a scale term, which are precisely the two characteristics we wish to focus on in the coherent regions. Since the kernels are defined continuously, standard optimization methods can be used to minimize (5.16).

5.2.3 Initialization

We initialize the regions using conventional blob-finding techniques. Marr and Hildreth [101] first proposed the use of the Laplacian of a Gaussian (LoG) for distinguishing homogeneous regions from the drastic changes in intensity that separate them. More recently, Lowe [96], among others [90], used a Difference of a Gaussian (DoG) to approximate the LoG filter. They construct a dense, discrete scale-space of DoG responses and then perform an explicit search for stable points (local extrema in space and scale).

To detect seed points, we likewise create a coarse, discrete scale-space of isotropic DoG responses by sampling a few (in our experiments, just 2) large scales. This coarse sampling is sufficient for seed detection because we later refine each candidate seed and

localize it in both space and scale. Similar to Lowe, we look for local extrema in the DoG response to detect seeds. However, since we are coarsely sampling scale-space, we analyze each 2D DoG-response separately (Lowe searches for extrema in 3D scale-space). Our search will result in many spurious seed extrema, which will converge to the nearest true extrema in the refinement process.

We define a seed with three parameters: $\boldsymbol{\mu}$ is set to the spatial location of the extrema point, and the $\boldsymbol{\Psi}$ is set to the product of the 2×2 identity and one-third of the scale of the LoG filter. Intuitively, this one-third scale factor shrinks the kernel to the homogeneous region at the filter's center. In contrast, Lowe scales the region by a factor of 1.5 because the SIFT keys function best in regions of high contrast (the region including its surrounding areas, for example). Figure 5.2 shows a comparison of our scaling and Lowe's scaling with respect to the LoG function.

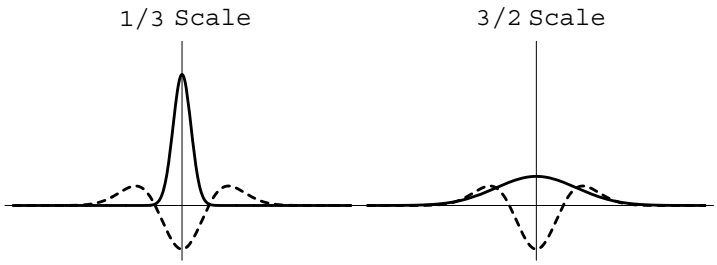


Figure 5.2: A comparison of the region scaling between our homogeneous regions (one-third) and Lowe's SIFT keys (1.5). The LoG kernel is shown as a dotted line with the region size as a solid line.

5.2.4 Merging

Different seed points may converge to the same minimum of the objective function (5.16), and since the optimization is independent for each seed point, we must account for this issue in a post-processing step. It is possible to do a more sophisticated initialization procedure that would reduce or remove the need for a merging, post-process. Essentially, there is a trade-off between the complexity of the initialization and the necessity of a merging post-process. Since we have a continuous objective function that can be minimized with efficient techniques, we are conservative and choose a simpler initialization that will result

in multiple seeds converging to the same minimum.

Let $\mathcal{R}$ denote the set of active regions in an image. For a region $R \in \mathcal{R}$, denote the parameters by $\boldsymbol{\theta}(R) \doteq \{\boldsymbol{\mu}, \boldsymbol{\Psi}\}$. Since the regions are described by anisotropic Gaussian functions, we use the Kullback-Leibler (KL) distance function. For any two probability density functions $p(x)$ and $q(x)$, the KL divergence is written:

$$KL(q,p) = \int q(x) \log \frac{q(x)}{p(x)} dx \tag{5.17}$$

The KL divergence is always non-negative and is zero if and only if $p = q$, but it is not symmetric [36]. For a symmetric distance, we sum the two directions: $KL(q,p) + KL(p,q)$. For the 2D Gaussians we use to spatially describe a region, the closed-form KL divergence is [124]:

$$KL(\boldsymbol{\theta}(R_1), \boldsymbol{\theta}(R_2)) = \frac{1}{2}\left(\log \frac{|\boldsymbol{\Psi}_2|}{|\boldsymbol{\Psi}_1|} + \operatorname{tr}\left(\boldsymbol{\Psi}_2^{-1}\boldsymbol{\Psi}_1\right) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^{\mathsf{T}}\boldsymbol{\Psi}_2^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\right) + \text{const.} \tag{5.18}$$

Define the symmetric KL distance between a pair of regions $R_1, R_2 \in \mathcal{R}$ as

$$d(R_1, R_2) = KL(\boldsymbol{\theta}(R_1), \boldsymbol{\theta}(R_2)) + KL(\boldsymbol{\theta}(R_1), \boldsymbol{\theta}(R_2)). \tag{5.19}$$

Fix a threshold τ and let two regions be *equivalent* if their KL distance is less than τ. Define an empty set of merged regions $\hat{\mathcal{R}} = \emptyset$, and merge with the following algorithm:

1. For each region $R \in \mathcal{R}$.
2. For each region $S \in \mathcal{R} \setminus R$
3. If $d(R,S) < \tau$, remove S from $\mathcal{R}$
4. Add R to $\hat{\mathcal{R}}$.

We have found the number of regions was significantly reduced (about 25% on average) after the merging procedure.

5.3 Scalar Projections

Images are complex entities; they are the result of numerous physical and stochastic processes and live in a very high dimensional space. To make image analysis tractable, we project the images into a lower dimensional space. Each dimension in the projected space captures a single image character like red-ness, or stripy-ness. This idea is related to descriptive modeling techniques like using a bank of Gabor filters as the basis (e.g. [132]), or dimensionality reduction with principle components analysis (e.g. [169]). However, we differ in that there is an underlying assumption that the image has been generated by a set of unknown scalar image processes. The goal in our approach is to define a set of projections that can approximate these unknown underlying processes.

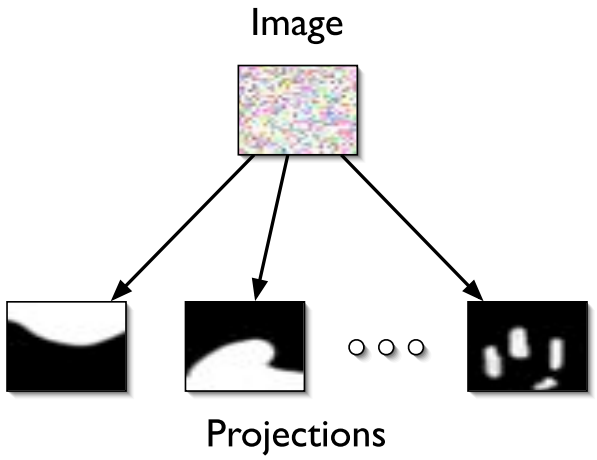


Figure 5.3: Explanation of data-flow in image dimensionality reduction.

Essentially, each projection defines a new *feature space* in which to analyze the input image. The intuition is that various projection functions will map a region of consistent image content to a homogeneous image patch in the scalar field: for example, there is some texture and/or color projection function such that an image of a plaid shirt will be mapped to a relatively homogeneous scalar field. Thus, by choosing appropriate scalar projections, we can capture coherent image content of varying character. To that end, define a function $S\colon \mathcal{X} \times \mathcal{I} \mapsto \mathfrak{R} \times \mathcal{I}$ that *projects* the d-dimensional image $\mathbf{I}$ to a one-dimensional scalar field $\mathbf{J}$. The scalar image $\mathbf{J}$ is indexed by the same pixel locations $\mathcal{I}$ and is thus comprised of $\{\mathcal{I}, J\}$, where $J\colon \mathcal{I} \mapsto \mathfrak{R}$. We will simply write $J(i)$ instead of $S(\mathbf{I})(i)$ in the following

examples.

There are two classes of projections: those that operate independently on single pixels and those that operate over neighborhoods of pixels. The methodology we propose is general and the construction of these projections is application dependent. We do not restrict the projections: they may be non-linear, and they may be dependent. In the experiments, we give a simple set of projections (Section 5.7).

5.3.1 Pixel Projections

A pixel projection is one that operates individually on the pixels without considering any neighborhood information. Such functions tend to be quite simple because they use only one value. However, a benefit to the pixel projections is that they do not affect any of the invariance properties of the detection.

Figure 5.4: Example pixel projections. (left) Original image. (middle) RGB linear combination with coefficients $(-1, 1, -1)$. (right) RGB pixel likelihood with color $(0, 1, 0)$.

Linear Combinations of Pixel Color

A simple linear combination of image bands can generate a useful feature space. Given three coefficients $\{c_r, c_g, c_b\}$ on the pixel color components, the definition is

$$\mathbf{J}(i) = c_r \mathbf{I}_r(i) + c_g \mathbf{I}_g(i) + c_b \mathbf{I}_b(i), \quad \forall i \in \mathcal{I}. \tag{5.20}$$

Figure 5.4-middle shows an example. Such a discrete set of linear combinations is used by Collins and Liu [29] in a tracking framework. Each vector of coefficients creates a feature space, and they propose a Fisher discriminant-like ratio to choose the best feature space for the current image frame.

Pixel Color Likelihood

A second pixel projection models a feature space as a Gaussian process in color-space having a mean (the color) and a covariance (the estimated error). Then, the projection computes the likelihood that each pixel, independently, has been generated by this process. Given a color, $\mathbf{c}$ and an estimated covariance $\mathbf{\Sigma}$, the likelihood function is written:

$$\mathbf{J}(i) = \exp\left(-\frac{1}{2}(\mathbf{I}(i) - \mathbf{c})^{\mathsf{T}}\mathbf{\Sigma}^{-1}(\mathbf{I}(i) - \mathbf{c})\right), \quad \forall i \in \mathcal{I}. \tag{5.21}$$

Figure 5.4-right gives an example of the likelihood function.

5.3.2 Neighborhood Projections

Neighborhood projections can be more powerful than the pixel projections because they incorporate information from multiple pixels in a single calculation. However, the neighborhood projections greatly affect the invariance properties of the detection. For example, for the detection to be scale invariant, we would need to know the per-pixel *integration* scale (i.e. the size of the local neighborhood needed to completely model the local image texture). While some heuristic methods have been presented to estimate this local scale [24], its calculation is error-prone, especially near object boundaries.

In addition to the two neighborhood projections we discuss below, various linear filters can be used as projection functions as well. These include gradient operator, Gabor [54] functions, and even template-matching kernels.

Graylevel Variance

The neighborhood variance is a very simple texture operator. Let $\mathcal{N}(i) \subset \mathcal{I}$ define the set of neighborhood pixels for $i \in \mathcal{I}$ with cardinality n. For pixel i, the variance is

$$J(i) = \frac{1}{n}\sum_{j\in\mathcal{N}(i)}\left(I(j) - \frac{1}{n}\sum_{j\in\mathcal{N}(i)} I(j)\right)^2. \tag{5.22}$$

Local Orientation Coherency

A second texture projection measures the local orientation coherency, or the *stripy-ness*, of the neigborhood. Jahne [78, p. 357] suggests a ratio of a linear combination of the

eigenvalues of the structure tensor. Denote the local image gradients of $\mathbf{I}$ in the x and y direction as $\mathbf{I}_x$ and $\mathbf{I}_y$ respectively. Then, for pixel i and its neighborhood $\mathcal{N}$, the structure tensor T is

$$T(i) = \begin{bmatrix} \sum_{\mathcal{N}(i)} I_x(i)^2 & \sum_{\mathcal{N}(i)} I_x(i)I_y(i) \\ \sum_{\mathcal{N}(i)} I_x(i)I_y(i) & \sum_{\mathcal{N}(i)} I_y(i)^2 \end{bmatrix}. \tag{5.23}$$

Since the tensor samples pixel gradients in an image neighborhood, it suffers from a resolution—localization problem: we need the window large enough to capture the periodicity of a stripy texture but small enough that it can accurately localize the boundary.

Let $\lambda_1(i) \geq \lambda_2(i)$ be the eigenvalues of $T(i)$. We drop the pixel index in the notation for clear presentation. Then, if there is a ideal local orientation, one eigenvalue is zero, $\lambda_1 > \lambda_2 = 0$. Considering image noise, the ideal case will never happen. For dominant local orientation $\lambda_1 \gg \lambda_2$. Otherwise, if there is isotropic local orientation (including the case of little gradient information), $\lambda_1 \approx \lambda_2$. Thus, this suggests using the following ratio to analyze the presence of dominant local gradient:

$$J(i) = \left(\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2} \right)^2. \tag{5.24}$$

For the case of dominant local orientation, then this ratio is near 1, and conversely, for the case of no dominant local orientation, this ratio tends to 0. Care must be taken in the implementation to avoid division-by-zero; in our implementation, we threshold on very low gradients. We give three examples of this stripy-ness projection in Figure 5.5.

5.4 The Complete Algorithm

In this section, we summarize the complete algorithm for extracting coherent regions. It uses the local minima of a continuous scale-space as representative coherent regions in the image description. For a given input image $\mathbf{I}$, define a set of scalar projections $\mathcal{B} = \{S_1, \ldots S_b\}$. For each projection $p \in \mathcal{B}$, define an initial, empty set of regions $\mathcal{R}_p$ and carry out the following steps:

1. Detect seeds. (Section 5.2.3).

2. Independently, minimize the function in Equation 5.16 to refine each seed.

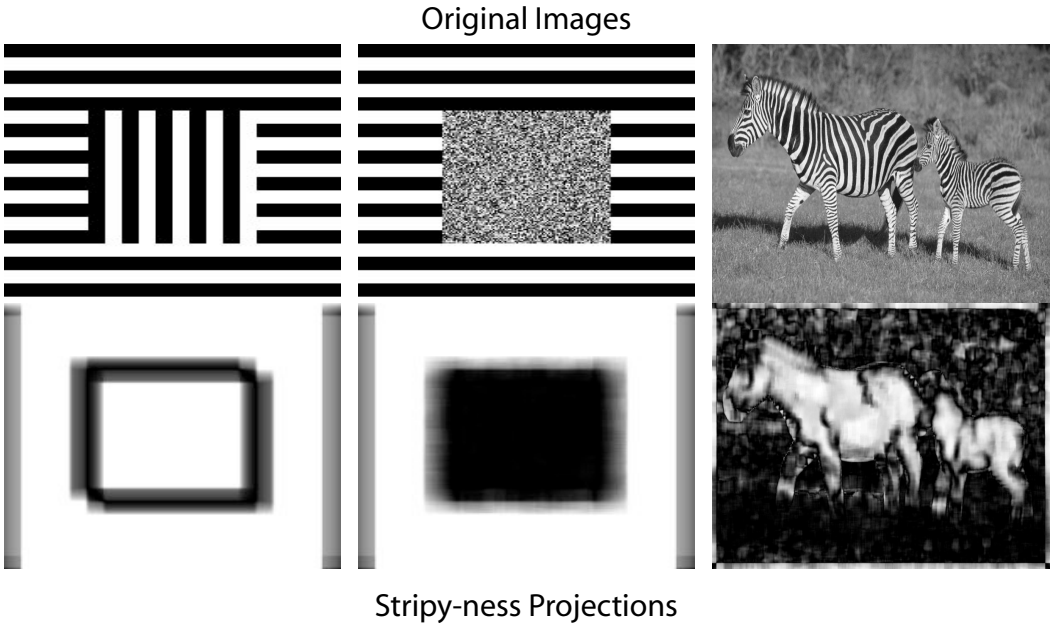


Figure 5.5: Examples of the stripy-ness (local orientation coherency) projection. The grayscale images are on top with the corresponding projections below. In the projections, white means more stripy.

3. Add convergent regions to $\mathcal{R}_p$.

4. Merge $\mathcal{R}_p$ (Section 5.2.4).

After computing the b region sets, we compute region descriptions (Section 5.5). In Figure 5.6, we show examples of the algorithm running on four different image projections.

5.5 Region Description

The method used to compute a description for each region will be dependent on the application domain. Earlier in Section 5.2, we described a simple appearance model using a single Gaussian in one scalar projection. This appearance model helped in the design of the coherent region segmentation, but it may not be discriminative enough for some applications. In this section we discuss a few potential approaches to region description; the description approaches are compared in the Section 5.7.3. Recall that at this step in the algorithm, we have b sets of regions $\mathcal{R}_1 \ldots \mathcal{R}_b$.

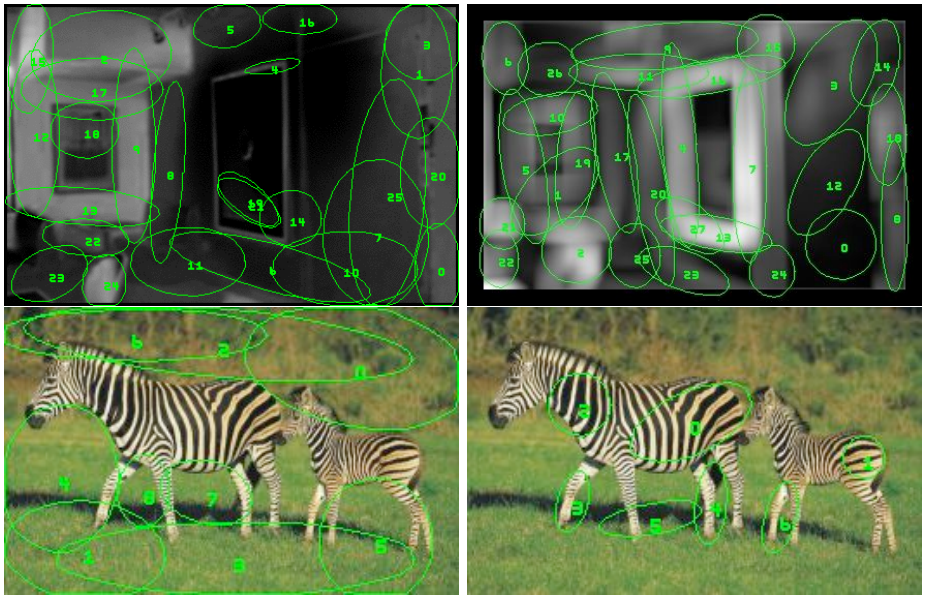


Figure 5.6: Examples of the algorithm running on four different image projections. Top-left is the green color projection introduced in Figure 5.4(right). Top-right is a neighborhood variance projection on the same image from Figure 5.4(left). Bottom-left is a "grassy" color projection. Bottom-right is the stripy-ness projection.

5.5.1 Single Appearance with Cooccurence

The first description is the appearance model we have already discussed. Here, we assume that the homogeneous character is a constant value in one of the projected images with zero-mean, normally distributed noise. We can either fix or estimate the variance. Because we have not restricted the development to independent projections, we must explicitly model the cooccurence information relationship between different projections. In this context, cooccurence simply means that the same region (spatial) has been detected in two or more different projections.

We augment each region with a parameter-list indicating from which projection it was detected. Denote the parameter-list of projection(s) for a region R by $L(R)$. Then, we define a new region set that is the union of all detected regions detected $\mathcal{R} = \bigcup_{p=1}^{b} \mathcal{R}_p$. To evaluate the cooccurence information, we enumerate through each pair of regions in $\mathcal{R}$. For each pair of regions (R, S), we evaluate their spatial proximity with the KL distance (5.19). If $d(R, S) < \tau$, the regions are said to cooccur. In this case, we define a new region T that

inherits its parameters from R and S: the spatial parameters set to those of R^3 and the projection list is combined by taking the union $L(T) = L(R) \bigcup L(S)$. We remove R and S from $\mathcal{R}$ and add T to $\mathcal{R}$.

After checking each region pair in $\mathcal{R}$, we compute the appearance description for the remaining regions. For each region $R \in \mathcal{R}$, describe it by computing the kernel-weighted mean (5.12) under each projection in its list $L(R)$.

5.5.2 Appearance in all Projections

The first method for describing the appearance of the regions creates a summarization of the image based on the exact projections used in the segmentation. However, such a representation may not be discriminative enough for certain problems, like image-to-image matching. A slightly more discriminative approach is to sample the appearance properties in all projections regardless of a region's originating projection.

Take the union of all regions detected $\mathcal{R} = \bigcup_{p=1}^{b} \mathcal{R}_p$. Then, for each region in $\mathcal{R}$ sample the kernel-weighted mean over all projections $\mathcal{B}$ creating a b dimensional vector. This is the representation we have used in [35]. Although this description yields good results (Section 5.7), it is invalid to assume a single, homogeneous value in each projection. Therefore, one can also measure the kernel-weighted variance in each projection, which is shown to improve matching results (Section 5.7.3).

5.6 Properties

The coherent regions we present in this paper have a number of good properties: **stability/invariance**, **conciseness**, and **scalability**. Since the image description is composed of a number of independent regions, like other local descriptor methods [149], it is robust to occlusion. In addition, using the kernel functions to weight the region statistics increases the robustness since it weights pixels based on their distance from the kernel center.

We claim that the detection is robust to affine distortions in the image. In Figure 5.7 we show the extracted regions using the RGB projection for exposition. To qualitatively analyze the detection, we have transformed by different affine maps: (top-right)

[3]The spatial parameters of R and S are equivalent since $d(R, S) < \tau$, which uses exclusively the spatial information of R and S.

Figure 5.7: The coherent regions extracted are robust to affine distortion of the image. The top-left is the original image, top-right is a rotation, bottom-left is an increase in the aspect ratio, and bottom-right is a reduction in the aspect ratio.

is rotation, (bottom-left) increasing the aspect ratio, (bottom-right) reducing the aspect ratio. From the figure, we see that roughly the same regions are extracted. We have also performed two experiments to quantitatively measure the detection invariance. For these experiments, a point projection is used, which is rotation, translation, and scale invariant. In the first experiment (Figure 5.8), we analyze the detection repeatability under rotations in the image. To detect if a region is re-detected, we use only the spatial parameters. We compare it against the SIFT method on the same images. We find the two methods perform comparably. In the second experiment (Table 5.1), we distort the image by an affine transformation chosen at random with varying complexity (5 grades). The simplest transformations, grade 1, included scale changes (1 ± 0.1) and rotations ($\pm \frac{\pi}{16}$ radians). The most complex transformations, grade 5, included scale changes (1 ± 0.5), rotations ($\pm \frac{\pi}{2}$ radians), and skew (1 ± 0.3).

Grade	**CRE**	**SIFT**
1	95%	93%
2	92%	91%
3	88%	89%
4	89%	88%
5	87%	86%

Table 5.1: Detection repeatability under random affine transformations of varying complexity. Our method is CRE (Coherent Region Extraction).

As discussed in Section 5.3, both the detection and the description invariance properties are dependent on the specific scalar projections employed. If a neighborhood

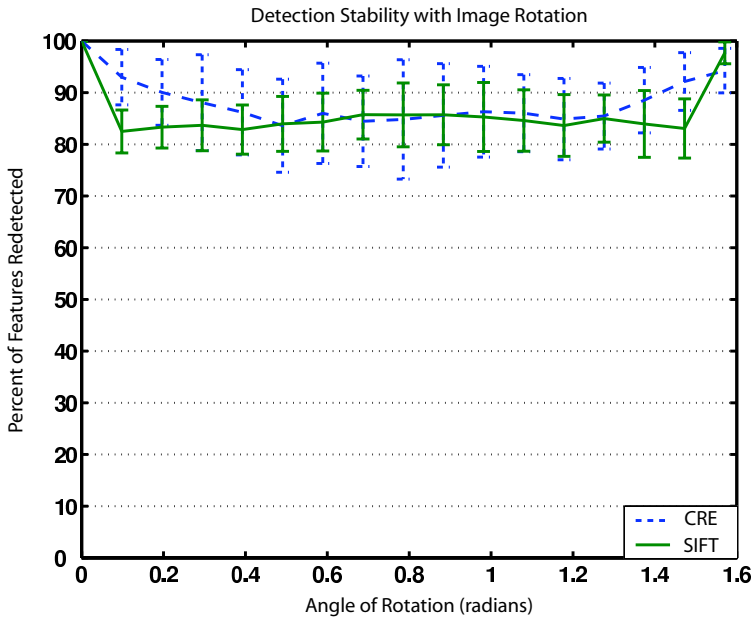


Figure 5.8: Detection repeatability experiment for rotated images. Our method is labeled CRE (Coherent Region Extraction).

based projection is used, then the detection is sensitive to scale because the underlying projection function is sensitive to scale. Likewise, if the scalar projection is designed to extract *vertical* texture (*y-gradient* in the image), then the region's description under this projection is no longer rotationally invariant or robust to affine distortion. A rotated image will yield a completely different region description under this projection.

The region description is implicitly invariant to rotation and translation in the image because it is simply a set of kernel-weighted statistics. Given each of the description methods in Section 5.5, the maximum number of appearance parameters per regions is $2b$ for b projections. It is clear that the image description is concise. Thus, the storage requirement for our technique will not prohibit its scaling to large or very large databases.

5.7 Experiments

In this section, we discuss the experiments we have done to demonstrate the coherent region based image description solves the image ordering problem. We measure the correctness of the ordering by querying the image database in a manner similar to image re-

trieval systems and checking the retrieved images. Thus, for a given database of images, we apply our coherent region extraction to each image independently. The image descriptions are then stored in a database and a querying protocol is established. To perform retrieval, for each image in the dataset, we query the database, and a sorted list of *matching* images is returned with the best match first.

For most of the experiments, we use a moderate sized dataset of 48 images[4] taken of an indoor scene from widely varying viewpoints and with drastic photometric variability (a subset of the dataset is shown in Figure 5.9). We also include the retrieval results for a second dataset of 91 outdoor images in Section 5.7.4. We hand-labeled the datasets; two images are said to be *matching* if there is any area of overlap between them.

Figure 5.9: A subset of the indoor dataset (chosen arbitrarily) used in the retrieval experiments.

We use the standard precision-recall graphs to present the matching results. The precision is defined as the fraction of true-positive matches from the total number retrieved and the recall is the fraction of matching images that are retrieved from the total number of possible matches in the database. Thus, in the ideal case, the precision-recall graph is a horizontal line at 100% precision for all recall rates.

Denote the three bands of the input image as R, G, B and S as their grayscale projection. Unless otherwise noted, we use a set of 5 projections: the 3 opponent color axes

[4]The complete datasets can be found on the world-wide-web at `http://www.cs.jhu.edu/~jcorso/r/regions/`.

$\{(R+G+B)/3$,$(R-B)/3$, and $(2G-R-B)/4\}$ which have been experimentally shown by [117] to perform well in color segmentation, a neighborhood variance measure in S with a window size of 16, and a orientation coherency measure in S with a window size of 16.

5.7.1 Matching

We take a nearest neighbor approach to compute the similarity score between two images. While many more sophisticated methods are viable given our coherent region based description, we use this simple approach to allow a valid comparison between our method and two other methods. As we defined earlier, two images are said to be matching if there exists any overlapping scene content. This matching problem has two levels. First, we address image content similarity, i.e. based on the two region sets, how similar are the two images. Second, if the two images are deemed similar in content, then how much spatial coherence exists between them. In this case, spatial coherence is defined as the pixel-area in each image where matching regions overlap. We can then, for instance, maximize the amount of overlap region to compute the parameters of a geometric transformation relating the two images. In these experiments, we focus only on the first level in matching.

Given a pair of images $\mathbf{I}_1, \mathbf{I}_2$ and their corresponding region sets $\mathcal{R}_1, \mathcal{R}_2$ computed from the same set of projections $\mathcal{B}$, the matching score between the two images is defined as the number of consistent nearest neighbor region-pairs. A consistent nearest neighbor region-pair is defined as a pair of regions with each being mutual nearest neighbors in a brute force search through both region sets. To be concrete, for region $R \in \mathcal{R}_1$, solve the following function

$$R_2^* = \arg \min_{R_2 \in \mathcal{R}_2} D(R, R_2), \tag{5.25}$$

where D is a distance function between the two region descriptions (we discuss candidate distance functions next). Then, for the nearest neighbor R_2^*, solve the following function

$$R_1^* = \arg \min_{R_1 \in \mathcal{R}} D(R_1, R_2^*). \tag{5.26}$$

The match $\{R, R_2^*\}$ is considered consistent match if and only if $R_1^* = R$.

From the possible descriptions previously discussed (Section 5.5), there are two candidates for the distance function. First, if the kernel-weighted means are used to de-

scribe the regions, then a simple sum of squared distance measure is sufficient. Second, if the kernel-weighted variances are included in the description, then the more appropriate measure is the KL distance. Additionally, if the cooccurence information is maintained, and the descriptions stored are dependent on the projection in which the regions were extracted, then the distance is only valid between regions that have been extracted from an equivalent set of projections. The distance between regions that have been extracted from different projections is set to infinity.

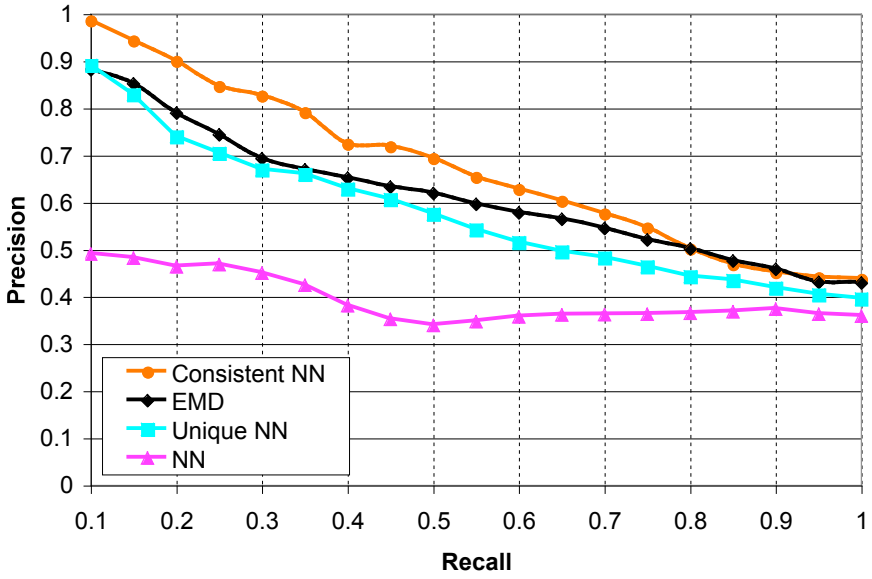


Figure 5.10: Comparison of different matching functions.

In Figure 5.10, we compare the matching function we just discussed, which is labeled Consistent NN, to other potential matching functions. The NN matching function simply accumulates the number of nearest neighbors for each regions. Since a nearest neighbor will always exist, this function is simply counting the number of regions per images. Thus, we expect it to perform quite poorly as the graph shows. The Unique NN function approximate Consistent NN. Basically, a nearest neighbor match is accepted only if it matches *much* better than the second nearest neighbor. Define a scalar α. If the match distances for the nearest and second nearest neighbors are m_1 and m_2, respectively, then the match is accepted only if $m_1 < \alpha m_2$. This is the matching function suggested by Lowe [96]. We

use $\alpha = 0.6$. The earth mover's distance [138], EMD, function reflects the minimal amount of work that must be done to transform one distribution into another. It is suitable in the case of feature matching because it permits partial matching. We find the Consistent NN yields the highest precision-recall rate on our dataset.

5.7.2 Projections and Kernels

The projections define the feature spaces in which the image will be analyzed for coherent content. In the dissertation, the set of projections is fixed independent of the dataset. The representative power of the resulting feature spaces is dependent on the projections used. In the following two experiments we analyze the matching sensitivity to the projections we have used in the experiments.

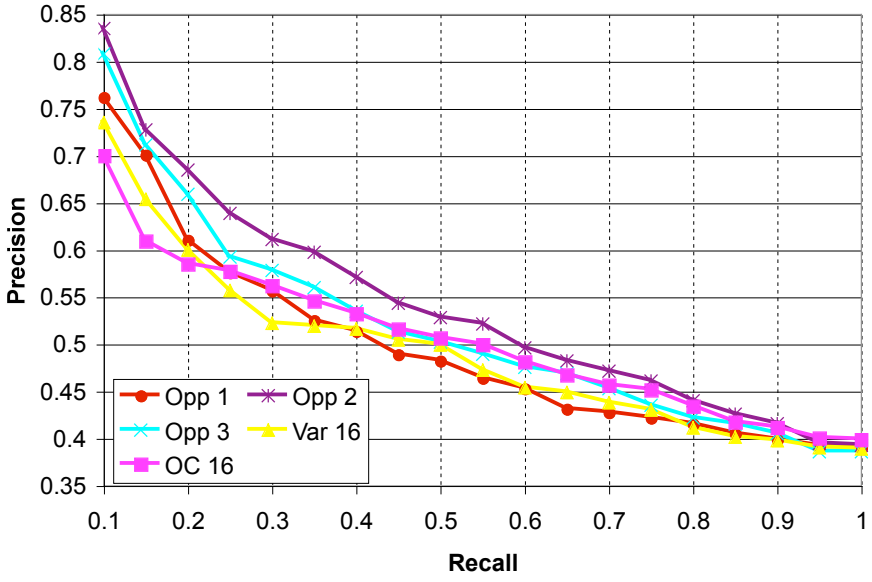


Figure 5.11: Graph showing precision-recall for each of the five projections used in the experiments (independently).

In Figure 5.11, we show the representative power of each of the five projections that we use in the experiments. The three opponent axes are labeled Opp #, the variance projection Var 16, and the orientation coherency projection OC 16. The graph indicates that the color projections are more representative of the image content in the test database than the two texture projections. The orientation coherency projection performs the worst

initially, but, for greater recall rates, it improves with respect to the other projections. This change is because the images we use have very few regions of stripy texture, and thus, the color is more discriminative for low recall rates. However, for higher recall rates, the stripy region information is less ambiguous than the remaining color information. In Figure 5.11, the Opp 1 projection is, essentially, the grayscale image; it is interesting to note that while it performs better than the variance and orientation coherency for recall rates up to 20%, for the remaining recall rates, it performs the worst. This degradation is due to the high variation in the lighting conditions between the images, and the raw grayscale data is not very robust to such photometric variation.

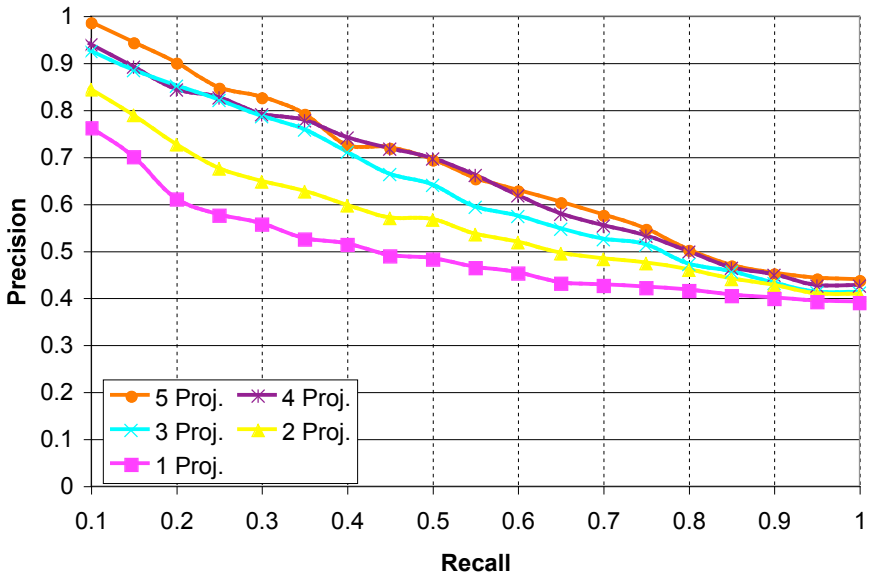


Figure 5.12: Graph showing precision-recall as the number of projections (feature spaces) is varied.

In Figure 5.12, we show the effect of varying the number of projections used in the image description. For Proj. 1, we just use the grayscale image. For Proj. 2, we use the grayscale image and the variance projection with a neighborhood size of 16. For Proj. 3, we use the 3 opponent color axes, and for Proj. 4, we add the variance with neighborhood size 16. Proj. 5 is the same set of projections used in all the other experiments. We find that the addition of multiple projections greatly improves the retrieval accuracy. However, we note that it this improvement is not always the case. Note for Proj. 5 at a recall of 40%,

the precision drops below Proj. 4. This variation is due to the unconstrained projections; there are cases, like this one, where two projections can be conflicting.

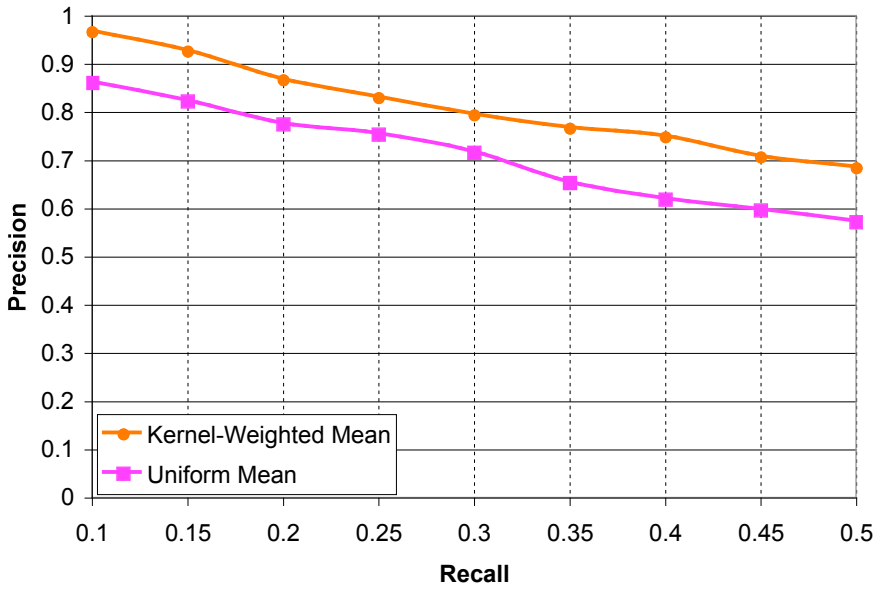


Figure 5.13: Graph showing precision-recall using kernel-weighted means in the projections versus uniform means.

In Figure 5.13, we show the effect of using kernel-weighted means for region description versus standard, uniformly weighted means. As expected, the kernel-weighted means greatly outperform the uniform means (by about 10% on average).

5.7.3 Description Comparison

In this experiment, we compare the candidate region descriptions from Section 5.5. In computing the precision-recall, we use the distance function that is appropriate for the given description algorithm. The three descriptions we compare are:

MC - Kernel-weighted mean and cooccurence modeling. Here, we make explicit use of the projections from which the regions are extracted.

M - A kernel-weighted mean from each projection.

MV - A kernel-weighted mean and variance from each projection.

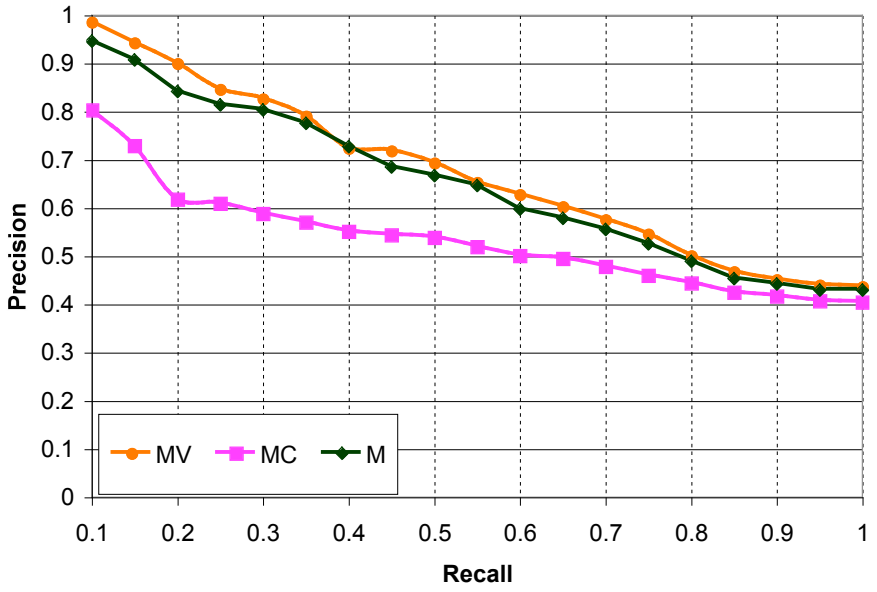


Figure 5.14: Graph showing precision-recall for different region description algorithms.

Figure 5.14 shows that the kernel-weighted mean and variance from each projection performs the best of the three techniques. One would expect the **MC** description to perform better since it explicitly incorporates the projections from which the regions are extracted. However, the resulting description is not as discriminative as those resulting from other two description algorithms, and the nearest neighbor matching algorithm inherently relies on discriminative features.

5.7.4 Retrieval Comparison

Figure 5.15: Image representation for the three methods on the same image. For our technique (left) and Blobworld (middle), a color representation is used. For SIFT (right), the key locations, scales, and orientations are rendered by arrows.

We compare our technique to two representative techniques[5] for local and global image description: SIFT keys [96] and Blobworld [24], respectively. Figure 5.15 gives a visualization of the different representations. SIFT is an example of a local, affine-insensitive and scale-invariant interest point descriptor. For matching, we use the unique nearest-neighbor scheme as discussed in Section 5.7.1. Note, that additional geometric constraints are plausible for both our method and SIFT key matching, but we do not employ any of them in order to keep the comparisons between methods fair. Blobworld is an example of using segmented image regions as the description. To measure matches using their provided source code, we used blob-to-blob queries. For a query image $\mathbf{I}$ with regions $r_1, \ldots r_n$, we queried the database independently for each region r_i and maintained accumulators for each image. The final matches for the query image were those images with the highest accumulators after queries for all n regions had been issued.

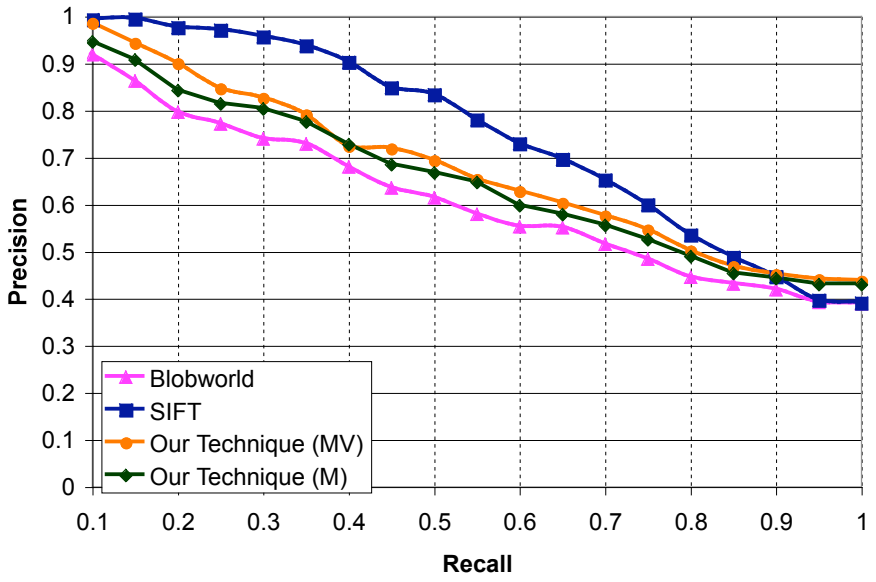


Figure 5.16: Comparison between our technique and other published techniques.

Figure 5.16 presents the precision-recall graph (average for querying on all images in the database) for each of the methods. For retrieval, we find the SIFT keys outperform the other two methods. This result agrees with the study by Mikolajczyk and Schmid [106].

[5]We are grateful to David Lowe and the group at Berkeley for making their source code available on-line.

Our method (**MV**) outperforms the Blobworld technique by about 6% precision on average.

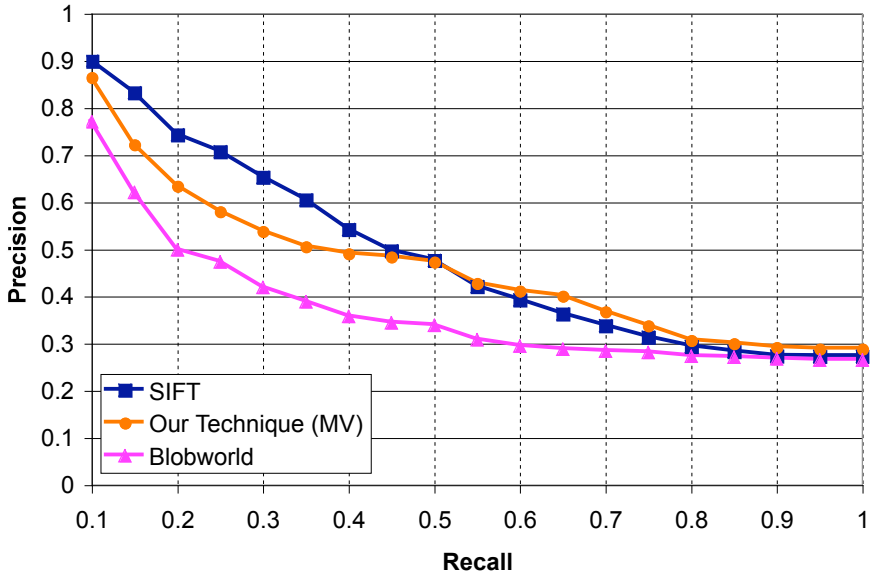


Figure 5.17: Comparison between our technique and other published techniques for a larger, outdoor dataset.

We have also experimented with the image retrieval on a larger dataset consisting of 91 outdoor images. Figure 5.17 shows the precision-recall graph for this dataset. The relative retrieval rates are comparable to the indoor dataset. As the results show, the second dataset is considerably more difficult due to the outdoor lighting conditions and widely varying viewpoints.

5.7.5 Storage Comparison

In this section, we compare the required storage of the methods. As an image or a scene database grows, querying it becomes more difficult and better indices or searching algorithms are required. In Table 5.2, we compare the storage efficiency for the three methods. We see that our method generate a data-size on the same order as Blobworld, which is far less than the SIFT approach. This data reflects the available source code for Blobworld and SIFT. It should be noted that the SIFT keys store 128 1-byte elements while the other two methods use 4-byte (1-word) floating point elements. We have not experimented with quantizing the storage for our technique to further reduce the size.

	Average Number of Elements	Size per Element (in Words)	Average Size (in Words)
Our Technique (M)	333	5	1665
Our Technique (MV)	333	10	3330
Blobworld	9	239	2151
SIFT	695	32	22260

Table 5.2: Comparison of average per-image storage for the three techniques.

Next, we show the results of an experiment that compares the retrieval rates for the SIFT method with our method when they store an equivalent (or nearly equivalent) number of features. In this case, we are still not storing the same amount data since the length of the SIFT keys are 128 bytes and the length of our descriptors is 5, which is dependent on the number of projections (we use the standard 5 projections and the **M** description). As suggested by Lowe [96], the larger (in spatial scale) SIFT keys are generally more stable and robust to noise. Thus, to reduce the number of SIFT keys stored, we keep the largest.

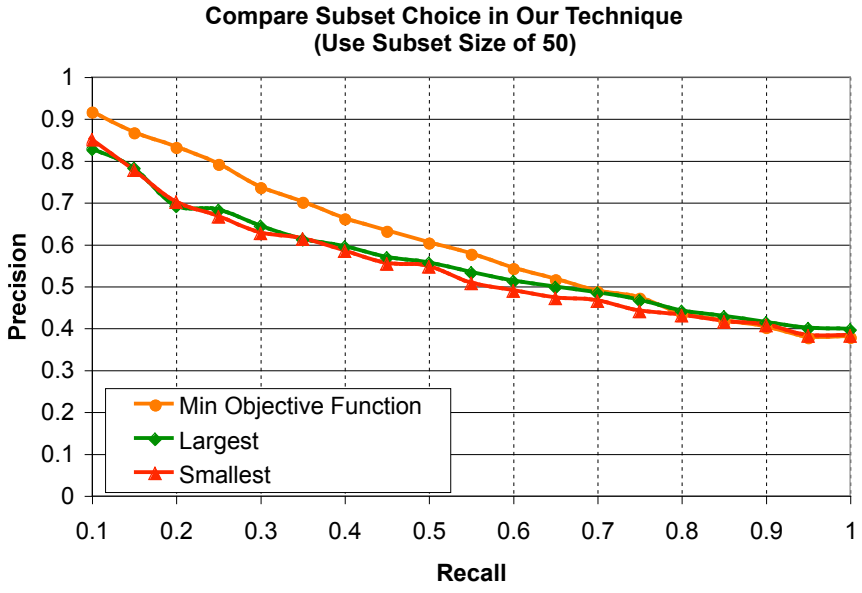


Figure 5.18: Comparing the three different subset choices for our technique.

In choosing the subset of features for our method, we rely on the value of the objective function for each region, which incorporates both scale and homogeneity. To show this method is better than relying on the scale alone, we compare the three potential subset choice methods in Figure 5.18.

In Figure 5.19, we show the precision-recall graph for four different subset sizes: 150, 100, 50, and 20. The two methods perform comparably at 150 feature with the SIFT method slightly outperforming the coherent regions. However, in the next three graphs, we find our technique drops in precision slightly for smaller subset sizes, but the SIFT method drops at a much faster rate.

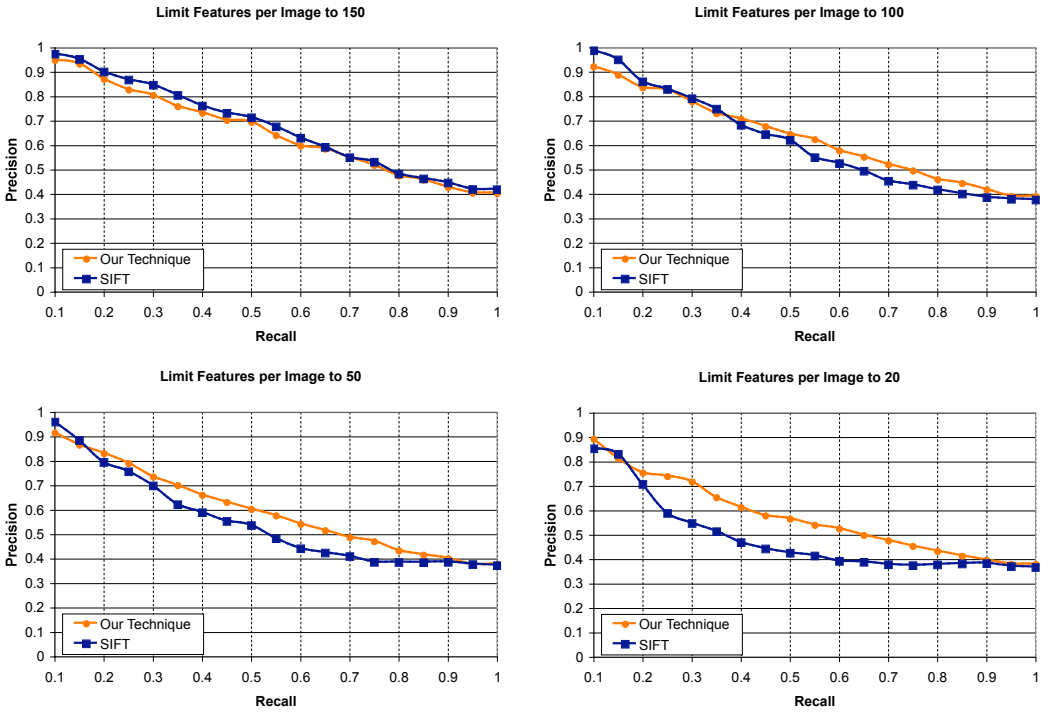


Figure 5.19: Retrieval comparison for four different feature subset sizes.

5.7.6 Robustness to Affine Distortion

In Section 5.6 we discussed the properties of our representation, and we claimed that it is robust to affine transformations of the image. To test this claim, we changed the aspect ratio of each image in the entire dataset and re-computed the coherent regions and

SIFT keys. We performed a complete dataset query (same as above) and measured the precision-recall (Figure 5.20) when querying with these distorted images. We used the **MV** description method. We experimented with aspect ratio changes of 0.5, 0.75, 1.25, and 1.5. From the graphs, we see that our method is very robust to the image distortion. At the extreme cases, it outperforms the SIFT method, which drops substantially.

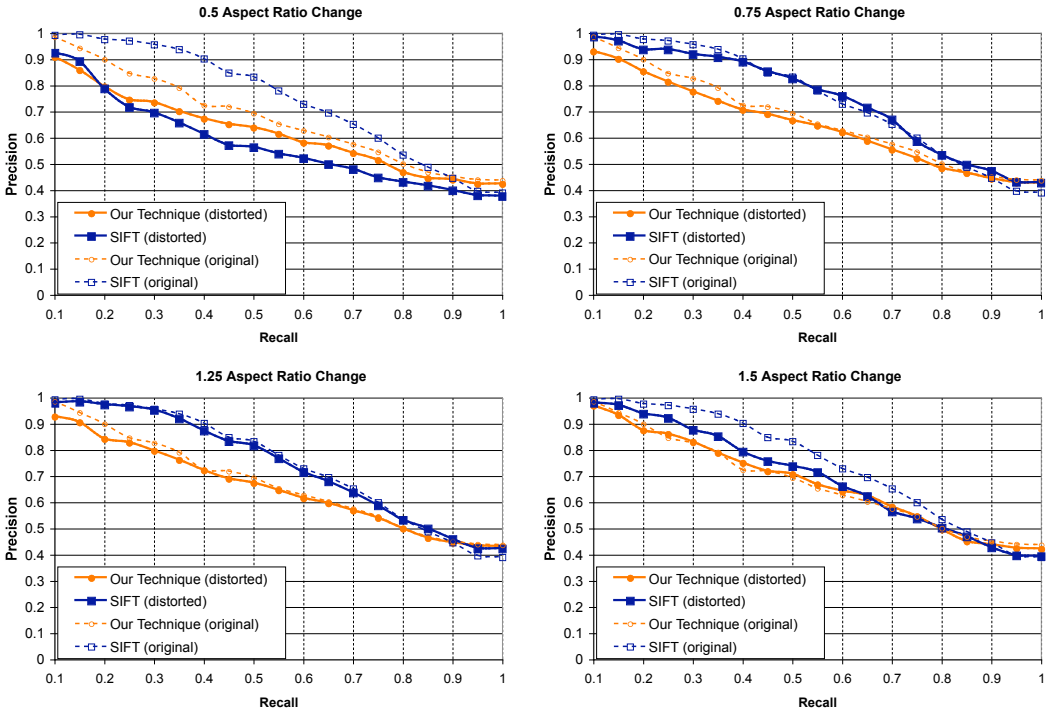


Figure 5.20: Graph showing precision-recall for our technique and the SIFT method when querying with distorted images from the database.

5.7.7 Robustness to Occlusion

In this section, we discuss the set of experiments we have performed to demonstrate robustness to occlusion. As mentioned earlier, one of the benefits of the local interest-operator techniques is their robustness to occlusion since an entire image (or object) is represented as a set of independent (and local) measurements. Likewise, our method summarizes an image as a set of independent regions. To simulate the occlusion, we choose a rectangle independently at random in each image and turn all the pixel intensities in that region to 0. In Figure 5.21, we show the precision-recall rate as we vary the size of the

rectangle between 0% and 25% of the image pixels. The graph shows good robustness to occlusion for 5% and 15%, and a slight degradation in the precision for 25%.

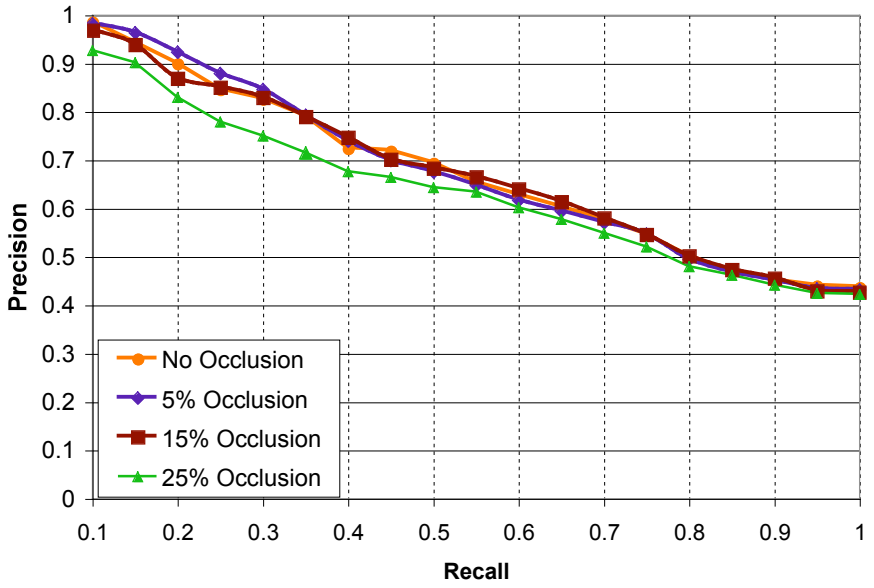


Figure 5.21: Graph showing precision-recall for retrieval under simulated occlusion.

In Figure 5.22, we compare the robustness to occlusion of our method and the SIFT method. To compute the change in precision we subtract the precision with occlusion from the precision without occlusion. Thus, 0 change in precision means the occlusion has no effect on the retrieval, negative change in precision means the occlusion actually improved the rate, and positive change means the occlusion caused the retrieval rates to degrade. An improvement is possible for small partial occlusions when the occluder masks an ambiguous image region. Essentially, a "smaller" change in precision means more robustness to the occlusion. We compare the same occlusion sizes: 5%, 15%, and 25%. We find that our technique is more robust to the occlusion in this experiment than the SIFT technique for the same respective occluders.

5.8 Conclusion

We have presented a novel method for image representation using a kernel-based, sparse image segmentation and description method. The method is general in that it permits

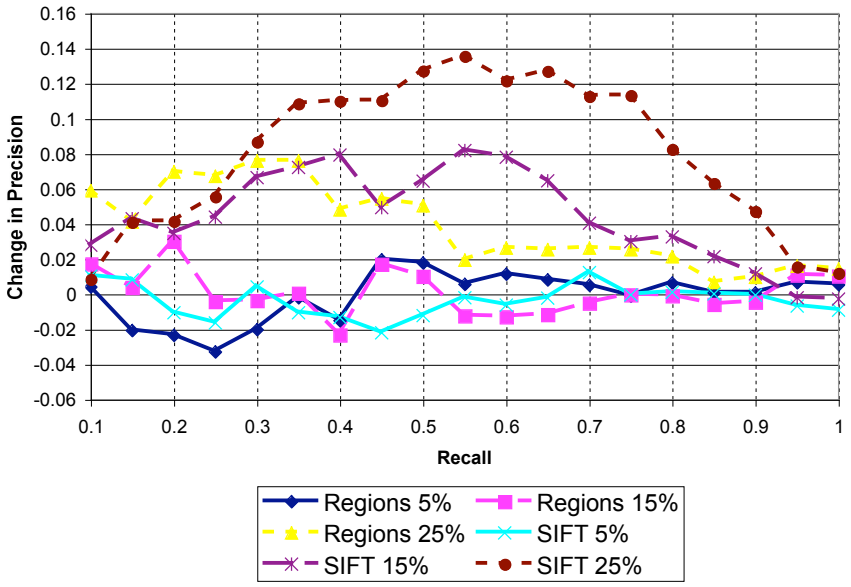


Figure 5.22: Graph showing the change in precision under partial occlusion for our technique and the SIFT method. 0 change in precision means the occlusion has no effect on the retrieval, negative change in precision means the occlusion actually improved the rate, and positive change means the occlusion caused the retrieval rates to degrade.

a variety of feature spaces which are represented as scalar image projections. We create a continuous scale-space of regions with coherent image content. The regions are robust under drastic viewpoint changes and varying photometric conditions. Our experiments indicate that the methods are stable, reliable, and efficient in terms of both computation and storage. In particular, the use of spatial kernels admits efficient, optimization-based methods for segmentation and image matching.

Concretely, the contribution we make is two-fold: First, the extraction of coherent regions in an anisotropic scale-space is novel. In comparison to other related works, which we presented in the first part of the chapter, our method differs from other interest point methods in that we focus on large homogeneous regions while the more conventional approaches search for points with rich image gradients. In this sense, we integrate ideas from traditional segmentation into the interest point concepts thereby creating an interest region operator. Additionally, we show that the coherent regions yield a very concise and robust image description for matching.

Second, we show that the detection of the regions is independent of the descrip-

tion; one can use a variety of different techniques to describe the local appearance of the extracted coherent regions. In the chapter, we have presented and compared three alternative description algorithms. Through the comparison, we show that the description method arising directly from the detection approach is not as discriminative for image matching as alternative description methods. Thus, one can use the same extraction algorithms for differing tasks that require a very concise summarization or a storage-expensive discriminative representation by simply varying the description method used.

Chapter 6

Conclusions

We have presented a set of techniques for vision-based human-computer interaction in the context of a novel methodology, which we term Visual Interaction Cues (VICs). In the VICs paradigm, we approach gesture recognition without globally tracking and modeling the user. Instead, the interface components are localized based on their projections in one or more video cameras. In the local regions of the video images, gesture recognition is solved by modeling the spatio-temporal pattern of visual cues that correspond to gestures. We show that the paradigm is applicable in both conventional 2D interface settings and unconventional 3D virtual/augmented reality settings. Through the VICs paradigm, we have extended the state of the art in reliable gesture recognition and vision-based interfaces due to the added structure given in the methodology.

We consider human-computer interaction to be a dialog between the user and the computer. With the presented techniques, the restrictive mediation through devices like the mouse is replaced with a direct, natural language of interaction. In the language of interaction, each atomic gesture corresponds to a word. Sequences of the gestures are analogous to sentences communicated to the computer. We have developed a coherent, linguistic model that integrates heterogeneous forms of the low-level gestures into a single framework; to the best of our knowledge, this is the first model to provide such an integration. The model has been integrated into our interaction platform, the 4D Touchpad.

In the last part of the dissertation, we discuss a region-based image modeling technique. We form a concise and stable image description that is based on detecting coherent regions in space, scale and appearance. The image is analyzed in a set of predefined feature spaces, which are designed to measure single appearance attributes like red-ness

or stripy-ness. We conceptualize the model as a set of mixture models taking attributes from both space and appearance. We present a novel approach to estimating the mixture model (segmenting the regions) that is a local, approximation to the MAP solution. Using techniques from kernel-based optimization and a novel objective function, we extract the regions and build the model. The modeling technique is applied to the image matching problem associated with mapping and localization in unknown environments for large-scale interaction. Our experiments show the modeling technique consistently orders image based on overlapping pixel content and is robust to viewpoint change and occlusion.

Among other applications, the region-based image modeling technique can be applied to the where-am-I problem as we have presented in the dissertation. In future work, we plan to implement this application. Recall that an interface component mapping must be established for each of the interface components, and given this mapping, the recognition techniques presented in Chapters 3 and 4 can be used. Thus, the novelty in this line of research is in the study of robust methods to dynamically establish the interface component mapping given a set of overlapping images from the environment. Creating the mapping is trivial for textured, planar surfaces: a homography, similar to the one used to calibrate the 4D Touchpad, can be used. However, for higher-order and untextured surfaces, it is more difficult.

A second line of future research would focus on establishing a standard language of interaction for large-scale, mobile interfaces. In particular, we have emphasized the importance of the human being able to bring their real-world domain knowledge to the interaction problem. We are interested in understanding what real-world protocols are directly transferable to HCI, and under what circumstances would a user prefer to learn a novel interaction technique to replace a previously used method.

We are also interested in extending the image modeling work from Chapter 5. In the current work, the projections are defined using heuristics and are fixed. However, in the original derivation of the modeling approach, the projections are a part of the region appearance model. One can jointly optimize over the projections and the region parameters. A simple case for such an approach is the pixel-likelihood projection, which has a convenient parametric form that makes its plausible to include it directly in the optimization.

The problem of registration using segmentation has been addressed by Schaffalitzky and Zisserman [143]. One advantage of kernel-based methods is that the registration problem can be posed as a continuous optimization defined directly on images. We intend

to investigate this approach.

In the image modeling, we assume the regions have a uniform prior distribution. While even with this assumption we are able to adequately summarize image content for matching and retrieval, the assumption is clearly not true. For example, in natural, outdoor images, large, blue regions at the top of images are more common than large red regions at the top. Explicitly modeling the prior distribution could serve multiple purposes: first, it could be directly incorporated into the model estimation. Second, in classification problems, a prior distribution can be modeled on the regions for each image class. Then, a novel image is segmented and the resulting regions are tested against the different model classes.

Appendix A

Derivation of Objective Function in Equation 5.16

We show that Equation 5.16

$$\arg\min_{\boldsymbol{\mu},\boldsymbol{\Psi}} \frac{\sum K(i,r)I(i)^2}{\alpha^2} + \frac{\tau}{|\boldsymbol{\Psi}|^{\frac{1}{2}}},$$

approximates Equation 5.15

$$\arg\min_{\beta,\boldsymbol{\mu},\boldsymbol{\Psi}} \sum K(i,r)I(i)^2 - \alpha^2 + \frac{\gamma}{n}\sum \left[\beta K(i,r) - I(i)\right]^2.$$

Recall $\boldsymbol{\mu}, \boldsymbol{\Psi}$ are the spatial mean and covariance (2D), α, ν^2 are the appearance mean and variance (1D) of the region r, β is a scale factor, and $|\boldsymbol{\Psi}|$ is the determinant of $\boldsymbol{\Psi}$.

A.1 Gaussian Integrals

We use a 2D Gaussian weighting function to spatially represent a region. The Gaussian function is written

$$K(r,i) = \frac{1}{2\pi|\boldsymbol{\Psi}|^{\frac{1}{2}}} \exp -\frac{1}{2}(i-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Psi}^{-1}(i-\boldsymbol{\mu}), \tag{A.1}$$

for region $r \doteq \{\boldsymbol{\mu}, \boldsymbol{\Psi}\}$ and pixels $i \in \mathcal{I} \subset \Re^2$. We approximate the discrete sum

$$\sum_{i \in \mathcal{I}} K(r, i) \tag{A.2}$$

with the integral

$$\int \int K(r, \mathbf{x}) d\mathbf{x}, \tag{A.3}$$

where the integral is over $\Re^2$ and $\mathbf{x} \in \Re^2$. This integral is computed in closed-form, and, since the regions are generally smaller than and contained by the image, the approximation captures the majority of the region-area.

First, consider the one-dimensional Gaussian integral

$$G = \int_{-\infty}^{\infty} e^{-ax^2} dx. \tag{A.4}$$

This integral is solved by squaring it and changing the variables to polar coordinates.

$$\begin{aligned} G^2 &= \int_{-\infty}^{\infty} \exp\left(-ay^2\right) dy \int_{-\infty}^{\infty} \exp\left(-ax^2\right) dx \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp\left(-a(x^2 + y^2)\right) dy dx \\ &= \int_0^{2\pi} d\theta \int_0^{\infty} r \exp\left(-ar^2\right) dr \\ &= \pi \int_0^{\infty} \exp\left(-au\right) du \\ &= \frac{\pi}{a} \end{aligned}$$

We have used the substitutions $x = r\cos\theta$, $y = r\sin\theta$, and, later, $u = r^2$. By taking the square root, we have the solution to (A.4):

$$G = \int_{-\infty}^{\infty} \exp\left(-ax^2\right) dx = \left(\frac{\pi}{a}\right)^{\frac{1}{2}} \tag{A.5}$$

Next, consider the 2D Gaussian integral:

$$G_2 = \int \int \exp\left(-b\mathbf{x}^{\mathsf{T}} \mathbf{\Psi}^{-1} \mathbf{x}\right) d\mathbf{x} \tag{A.6}$$

where the integral is over $\Re^2$ and $\mathbf{x} \in \Re^2$. Since, $\mathbf{\Psi}^{-1}$ is real and symmetric, it can be decomposed into $\mathbf{\Psi}^{-1} = UDU^\mathsf{T}$, where U is an orthogonal matrix of eigenvectors of $\mathbf{\Psi}^{-1}$ and D is the diagonal matrix of eigenvalues of $\mathbf{\Psi}^{-1}$, by the spectral theorem [73, Th. 4.1.5]. Let λ_1, λ_2 be the elements (eigenvalues) on the diagonal of D.

$$\begin{aligned}\int\int \exp\left(-b\mathbf{x}^\mathsf{T}\mathbf{\Psi}^{-1}\mathbf{x}\right) d\mathbf{x} &= \int\int \exp\left(-b\mathbf{x}^\mathsf{T}UDU^\mathsf{T}\mathbf{x}\right) d\mathbf{x} \\ &= \int\int \exp\left(-b\mathbf{y}^\mathsf{T}D\mathbf{y}\right) |J| d\mathbf{y},\end{aligned}$$

where we have substituted $\mathbf{y} = U^\mathsf{T}\mathbf{x}$ and J is the Jacobian matrix of the substitution. Denoting the coordinate vectors $\mathbf{x} \doteq \{x_1, x_2\}$ and $\mathbf{y} \doteq \{y_1, y_2\}$,

$$J = \begin{bmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_2}{\partial y_1} \\ \frac{\partial x_1}{\partial y_2} & \frac{\partial x_2}{\partial y_2} \end{bmatrix}.$$

By construction, the elements of J are the components of the matrix U, which is an orthogonal matrix. Thus, $|J| = |U| = 1$. Since D is a diagonal matrix, we can decouple the two integrals.

$$\begin{aligned}\int\int \exp\left(-b\mathbf{y}^\mathsf{T}D\mathbf{y}\right) d\mathbf{y} &= \prod_{k=1}^{2} \int \exp\left(-b\lambda_k y_k^2\right) dy_k \\ &= \prod_{k=1}^{2} \left(\frac{\pi}{b\lambda_k}\right)^{\frac{1}{2}} \\ &= \frac{\pi}{b}\left(\frac{1}{|D|}\right)^{\frac{1}{2}} \\ &= \frac{\pi}{b}|\mathbf{\Psi}|^{\frac{1}{2}} \end{aligned} \tag{A.7}$$

The last line follows because $|\mathbf{\Psi}^{-1}| = |UDU^\mathsf{T}| = |U||D||U^\mathsf{T}| = |D|$.

With (A.5) and (A.7), we can solve (A.3). Without loss of generality assume a zero mean Gaussian kernel.

$$\begin{aligned}
\int\int K(r,\mathbf{x})d\mathbf{x} &= \int\int \frac{1}{2\pi|\mathbf{\Psi}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{\Psi}^{-1}\mathbf{x}\right) d\mathbf{x} \\
&= \frac{1}{2\pi|\mathbf{\Psi}|^{\frac{1}{2}}} \int\int \exp\left(-\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{\Psi}^{-1}\mathbf{x}\right) d\mathbf{x}, \\
&= \frac{2\pi|\mathbf{\Psi}|^{\frac{1}{2}}}{2\pi|\mathbf{\Psi}|^{\frac{1}{2}}} \\
&= 1
\end{aligned}$$

Last, we consider the squared kernel

$$\begin{aligned}
\sum_{i\in\mathcal{I}} K(i,r)^2 &\approx \int\int K(r,\mathbf{x})^2 d\mathbf{x} \\
&= \int\int \left[\frac{1}{2\pi|\mathbf{\Psi}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{\Psi}^{-1}\mathbf{x}\right)\right]^2 d\mathbf{x} \\
&= \frac{1}{4\pi^2|\mathbf{\Psi}|} \int\int \exp\left(-\mathbf{x}^{\mathsf{T}}\mathbf{\Psi}^{-1}\mathbf{x}\right) d\mathbf{x} \\
&= \frac{\pi|\mathbf{\Psi}|^{\frac{1}{2}}}{4\pi^2|\mathbf{\Psi}|} \\
&= \frac{1}{4\pi|\mathbf{\Psi}|^{\frac{1}{2}}} \qquad \text{(A.8)}
\end{aligned}$$

A.2 Derivation

First, we solve for the scale factor in the template term of Equation 5.15.

$$\begin{aligned}
\arg\min_{\beta} \sum \left[\beta K(i,r) - I(i)\right]^2 \qquad & \text{(A.9)} \\
0 &= \sum \left[\beta K(i,r) - I(i)\right] K(i,r) \\
\beta \sum K(i,r)^2 &= \sum I(i) K(i,r) \\
\beta &= \frac{\sum I(i) K(i,r)}{\sum K(i,r)^2} \\
\beta &\approx 4\pi|\mathbf{\Psi}|^{\frac{1}{2}} \sum I(i) K(i,r) \\
&\approx 4\pi\alpha|\mathbf{\Psi}|^{\frac{1}{2}}. \qquad \text{(A.10)}
\end{aligned}$$

Next, we simplify Equation 5.15:

$$\begin{aligned}
&\arg\min_{\beta,\boldsymbol{\mu},\boldsymbol{\Psi}} \sum K(i,r)I(i)^2 - \alpha^2 + \frac{\gamma}{n}\sum \left[\beta K(i,r) - I(i)\right]^2 \\
&\arg\min_{\beta,\boldsymbol{\mu},\boldsymbol{\Psi}} \sum K(i,r)I(i)^2 - \alpha^2 + \frac{\gamma}{n}\left[\beta^2 \sum K(i,r)^2 - 2\beta\alpha\right] + \text{const} \\
&\arg\min_{\boldsymbol{\mu},\boldsymbol{\Psi}} \sum K(i,r)I(i)^2 - \alpha^2 + \frac{\gamma}{n}\left[4\pi|\boldsymbol{\Psi}|^{\frac{1}{2}}\alpha^2 - 8\pi|\boldsymbol{\Psi}|^{\frac{1}{2}}\alpha^2\right] + \text{const} \\
&\arg\min_{\boldsymbol{\mu},\boldsymbol{\Psi}} \sum K(i,r)I(i)^2 - \alpha^2 - \frac{\gamma}{n}4\pi|\boldsymbol{\Psi}|^{\frac{1}{2}}\alpha^2 + \text{const} \\
&\arg\min_{\boldsymbol{\mu},\boldsymbol{\Psi}} \frac{\sum K(i,r)I(i)^2}{\alpha^2} - \gamma\frac{4\pi|\boldsymbol{\Psi}|^{\frac{1}{2}}}{n} + \text{const} \qquad \text{(A.11)}
\end{aligned}$$

We assume that $\alpha^2 \neq 0$. Finally, we show that Equation 5.16 is a first-order approximation to Equation 5.15. We use a special case of the binomial series expansion [178]:

$$\begin{aligned}
(1-x)^{-r} &= \sum_{k=0}^{\infty} \frac{(r)_k}{k!}(-x)^k \\
&= 1 + rx + \frac{1}{2}r(r-1)x^2 + \frac{1}{6}r(r-1)(r-2)x^3 + \ldots
\end{aligned}$$

We have used the Pochhammer symbol $(r)_k = r(r+1)\ldots(r+k-1)$. The series converges for $|x| < 1$. For the case $r = 1$, we have

$$(1-x)^{-1} = 1 + x + x^2 + \ldots \qquad \text{(A.12)}$$

Let $\tau = \frac{n}{4\pi\gamma}$, and write $B = \frac{4\pi\gamma}{n}|\boldsymbol{\Psi}|^{\frac{1}{2}}$.

$$\begin{aligned}
\frac{\tau}{|\boldsymbol{\Psi}|^{\frac{1}{2}}} &= \frac{n}{4\pi\gamma|\boldsymbol{\Psi}|^{\frac{1}{2}}} \\
&= B^{-1} \\
&= (1-(1-B))^{-1} \\
&= 1 + (1-B) + (1-B)^2 + \ldots \\
&\approx -B \\
&= -\frac{4\pi\gamma}{n}|\boldsymbol{\Psi}|^{\frac{1}{2}} \qquad \text{(A.13)}
\end{aligned}$$

For the binomial expansion, we must ensure $|(1-B)| < 1$. We derive bounds for γ to ensure $0 < B < 2$. Note $|\mathbf{\Psi}|^{\frac{1}{2}} > 0$, and $n > 0$. The lower bound is clearly $\gamma > 0$. The upper bound derivation follows:

$$\begin{aligned} B &< 2 \\ \frac{4\pi\gamma}{n}|\mathbf{\Psi}|^{\frac{1}{2}} &< 2 \\ \gamma &< \frac{n}{2\pi}|\mathbf{\Psi}|^{\frac{1}{2}}. \end{aligned} \tag{A.14}$$

Assuming (A.14), (A.13) shows that the objective function defined in Equation 5.16 is an first-order approximation of Equation 5.15.

A.3 Discussion

We discuss the bound (A.14) given in the previous section to show that the approximation holds in our experiments (Section 5.7). Recall n is the number of pixels in the image, which is on the order of 10^5. We implement (5.16) with the τ multiplier set to 1. Then, by definition, $\gamma = \frac{n}{4\pi}$. Given the bound (A.14), we can determine for what size regions this approximation holds, i.e. we get a bound for $|\mathbf{\Psi}|^{\frac{1}{2}}$:

$$\begin{aligned} \gamma &< \frac{n}{2\pi}|\mathbf{\Psi}|^{\frac{1}{2}} \\ \frac{n}{4\pi} &< \frac{n}{2\pi}|\mathbf{\Psi}|^{\frac{1}{2}} \\ \frac{1}{2} &< |\mathbf{\Psi}|^{\frac{1}{2}} \end{aligned} \tag{A.15}$$

Therefore, the approximation holds for all but extremely small regions. Since the unit is the pixel, the lower bounds roughly corresponds to regions that are smaller than a pixel.

Bibliography

[1] J. K. Aggarwal and Q. Cai. Human Motion Analysis: A Review. *Computer Vision and Image Understanding*, 73(3):428–440, 1999.

[2] P. Anandan. A Computational Framework and an Algorithm for the Measurement of Visual Motion. *International Journal of Computer Vision*, 2(3):283–310, 1989.

[3] Vassilis Athitsos and Stan Sclaroff. Estimating 3D Hand Pose from a Cluttered Image. In *Computer Vision and Pattern Recognition*, volume 2, pages 432–439, 2003.

[4] Ronald Azuma. A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments*, 6(11):1–38, 1997.

[5] Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, and Blair MacIntyre. Recent Advances in Augmented Reality. *IEEE Computer Graphics and Applications*, 21(6):34–47, 2001.

[6] Chandrajit L. Bajaj, Fausto Bernardini, and Guoliang Xu. Automatic Reconstruction of Surfaces and Scalar Fields from 3D Scans. *Computer Graphics*, 29:109–118, 1995.

[7] M. Bajura, Henry Fuchs, and R. Ohbuchi. Merging Virtual Objects with the Real World: Seeing Ultrasound Imagery Within the Patient. In *SIGGRAPH*, volume 26, pages 203–210, 1992.

[8] Rafael Ballagas, Meredith Ringel, Maureen Stone, and Jan Borchers. iStuff: A Physical User Interface Toolkit For Ubiquitous Computing Environments. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 537–544, 2003.

[9] S. Basu, I. Essa, and A. P. Pentland. Motion Regularization for Model-Based Head Tracking. In *International Conference on Pattern Recognition*, 1996.

[10] Paul Beardsley, Phil Torr, and Andrew Zisserman. 3D Model Acquisition from Extended Image Sequences. *European Conference on Computer Vision*, 0(0), 1996.

[11] M. Beaudouin-Lafon. Instrumental Interaction: An Interaction Model for Desigining Post-WIMP User Interfaces. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 446–453, 2000.

[12] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape Context: A New Descriptor for Shape Matching and Object Recognition. In *Neural Information Processing*, pages 831–837, 2000.

[13] Paul J. Besl and Neil D. McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[14] Oliver Bimber, Bernd Frohlich, Dieter Schmalstieg, and L. Miguel Encarnacao. The Virtual Showcase. *IEEE Computer Graphics and Applications*, 21(6):48–55, 2001.

[15] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[16] M. J. Black and Y. Yacoob. Tracking and Recognizing Rigid and Non-Rigid Facial Motions using Local Parametric Models of Image Motion. *International Journal of Computer Vision*, 25(1):23–48, 1997.

[17] Aaron Bobick and James Davis. The Recognition of Human Movement Using Temporal Templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3):257–267, 2001.

[18] Aaron F. Bobick and Andrew Wilson. A State-based Approach to the Representation and Recognition of Gesture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(12):1325–1337, 1997.

[19] G. R. Bradski. Computer Vision Face Tracking for Use in a Perceptual User Interface. *Intel Technology Journal*, 1998.

[20] M. Brand, N. Oliver, and A.P. Pentland. Coupled Hidden Markov Models for Complex Action Recognition. In *Computer Vision and Pattern Recognition*, pages 994–999, 1997.

[21] C. Bregler and J. Malik. Tracking People with Twists and Exponential Maps. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8–15, 1998.

[22] Christoph Bregler. Learning and Recognizing Human Dynamics in Video Sequences. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[23] Wolfgang Broll, Eckhard Meier, and Thomas Schardt. The Virtual Round Table - A Collaborative Augmented Multi-User Environment. In *International Conference on Collaborative Virtual Environments*, pages 39–46, 2000.

[24] Chad Carson, Serge Belongie, Hayit Greenspan, and Jitendra Malik. Blobworld: Image Segmentation using Expectation-Maximization and Its Application to Image Querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1026–1038, 2002.

[25] Y. Chen and G. Medioni. Surface Description of Complex Objects from Multiple Range Images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 153–158, 1994.

[26] Y. Chen and G. Medioni. Object Modeling by Registration of Multiple Range Images. *Image and Vision Computing*, 10(3):145–155, 1999.

[27] C. Chinnock. Holographic 3D Images Float In Free Space. *Laser Focus World*, 31(6):22–24, 1995.

[28] Robert Collins. Mean-Shift Blob Tracking Through Scale Space. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[29] Robert Collins and Yanxi Liu. On-Line Selection of Discriminative Tracking Features. In *International Conference on Computer Vision*, volume 1, pages 346–352, 2003.

[30] Dorin Comaniciu and Peter Meer. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[31] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. The Variable Bandwidth Mean Shift and Data-Driven Scale Selection. In *International Conference on Computer Vision*, volume 1, pages 438–445, 2001.

[32] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-Based Object Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003.

[33] Jason J. Corso, Darius Burschka, and Gregory D. Hager. Direct Plane Tracking In Stereo Images for Mobile Navigation. In *International Conference on Robotics and Automation*, pages 875–880, 2003.

[34] Jason J. Corso, Darius Burschka, and Gregory D. Hager. The 4DT: Unencumbered HCI with VICs. In *IEEE Workshop on Human Computer Interaction at Conference on Computer Vision and Pattern Recognition*, 2003.

[35] Jason J. Corso and Gregory D. Hager. Coherent Regions for Concise and Stable Image Description. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[36] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley and Sons, 1991.

[37] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. In *SIGGRAPH*, pages 135–143, 1993.

[38] Y. Cui, D. Swets, and J. Weng. Learning-based Hand Sign Recognition using SHOSLIF-M. In *International Conference on Computer Vision*, 1995.

[39] Y. Cui and J. Weng. View-Based Hand Segmentation and Hand-Sequence Recognition with Complex Backgrounds. In *International Conference on Pattern Recognition*, 1996.

[40] Y. Cui and J. Weng. Appearance-Based Hand Sign Recognition from Intensity Image Sequences. *Computer Vision and Image Understanding*, 78:157–176, 2000.

[41] D. Demirdjian and Trevor Darrell. Motion Estimation from Disparity Images. In *International Conference on Computer Vision*, volume 1, pages 205–212, 2001.

[42] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood From Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society – Series B*, 39(1):1–38, 1977.

[43] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley and Sons, 2 edition, 2001.

[44] D. Ezra, G. Woodgate, B. Omar, N. Holliman, J. Harrold, and L. Shapiro. New Auto-Stereoscopic Display System. In *SPIE*, volume 0, 1995.

[45] Olivier Faugeras. *Three-Dimensional Computer Vision*. The MIT Press, 1993.

[46] Steven Feiner, Blair MacIntyre, Tobias Hollerer, and T. Webster. A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Environment. *International Symposium on Wearable Computers*, 1997.

[47] Steven Feiner, Blair MacIntyre, and Doree Seligmann. Knowledge-Based Augmented Reality. *Communications of the ACM*, 36(7):53–62, 1993.

[48] Martin A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[49] George W. Fitzmaurice, Hiroshi Ishii, and Bill Buxton. Bricks: Laying the Foundations for Graspable User Interfaces. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 442–449, 1995.

[50] F. Fraundorfer and H. Bischof. Detecting Distinguished Regions By Saliency. In *Scandinavian Conference on Image Analysis*, pages 208–215, 2003.

[51] Pascal Fua. From Multiple Stereo Views to Multiple 3-D Surfaces. *International Journal of Computer Vision*, 1(24):19–35, 1997.

[52] Henry Fuchs, S.M. Pizer, L.C. Tsai, and S.H. Bloomberg. Adding a True 3-D Display to a Raster Graphics System. *IEEE Computer Graphics and Applications*, pages 73–78, 1982.

[53] B. V. Funt and G. D. Finlayson. Color constant color indexing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):522–529, May 1995.

[54] D. Gabor. Theory of Communication. *Journal of the I.E.E.*, 3(93):429–457, 1946.

[55] Aphrodite Galata, Neil Johnson, and David Hogg. Learning Variable-Length Markov Models of Behavior. *Computer Vision and Image Understanding*, 83(1):398–413, 2001.

[56] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Addison-Wesley Professional, 1995.

[57] D. M. Gavrila. The Visual Analysis of Human Movement: A Survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.

[58] D. M. Gavrila and L. S. Davis. Towards 3-D Model-Based Tracking and Recognition of Human Movement: A Multi-View Approach. *International Conference on Automatic Face and Gesture Recognition*, 1995.

[59] Sebastien Gilles. *Robust Description and Matching of Images*. PhD thesis, University of Oxford, 1998.

[60] L. Goncalves, E. Di Bernardo, E. Ursella, and P. Perona. Monocular Tracking of the Human Arm in 3D. In *International Conference on Computer Vision*, pages 764–770, 1995.

[61] D.O. Gorodnichy and G. Roth. Nouse 'Use your nose as a mouse' Perceptual Vision Technology for Hands-Free Games and Interfaces. *Image and Vision Computing*, 2004.

[62] Hayit Greenspan, Jacob Goldberger, and A. Mayer. Probabilistic Space-Time Video Modeling via Piecewise GMM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):384–397, 2004.

[63] Hayit Greenspan, Jacob Goldberger, and L. Ridel. A Continuous Probabilistic Framework for Image Matching. *Computer Vision and Image Understanding*, 84:384–406, 2001.

[64] P. Grunwald. A Tutorial Introduction to the Minimum Description Length Principle. In P. Grnwald, I.J. Myung, and M. Pitt, editors, *Advances in Minimum Description Length: Theory and Applications*. MIT Press, 2005.

[65] Gregory D. Hager and Peter N. Belhumeur. Efficient Region Tracking with Parametric Models of Geometry and Illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.

[66] Gregory D. Hager and Kentaro Toyama. Incremental Focus of Attention for Robust Visual Tracking. *International Journal of Computer Vision*, 35(1):45–63, 1999.

[67] Daniella Hall and James L. Crowley. Estimating the Pose of Phicons for Human Computer Interaction. In *International Conference on Multimodal Interfaces*, 2000.

[68] J. S. Hare and P. H. Lewis. Scale Saliency: Applications in Visual Matching, Tracking and View-Based Object Recognition. In *Distributed Multimedia Systems / Visual Information Systems*, pages 436–440, 2003.

[69] Chris Harris and M.J. Stephens. A Combined Corner and Edge Detector. In *ALVEY Vision Conference*, pages 147–151, 1988.

[70] D. Austin Henderson and Stuart Card. Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention in a Window-based Graphical User Interfaces. *ACM Transactions on Graphics*, 5(3):211–243, 1986.

[71] S. Holland and D. Oppenheim. Direct Combination. In *ACM Human Factors in Computing Systems*, pages 262–269, 1999.

[72] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface Reconstruction from Unorganized Points. *Computer Graphics*, 26(2):71–78, 1992.

[73] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[74] T. Horprasert, D. Harwood, and L. S. Davis. A Robust Background Subtraction and Shadow Detection. In *Asian Conference on Computer Vision*, 2000.

[75] B. Insko, M. Meehan, M. Whitton, and F. Brooks. Passive Haptics Significantly Enhances Virtual Environments. Technical Report 10, Department of Computer Science, UNC Chapel Hill, 2001.

[76] Hiroshi Ishii and Brygg Ullmer. Tangible Bits: Towards Seemless Interfaces Between People, Bits, and Atoms. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 234–241, 1997.

[77] Y. A. Ivanov and Aaron F. Bobick. Recognition of Visual Activities and Interactions by Stochastic Parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):852–872, 2000.

[78] B. Jahne. *Digital Image Processing*. Springer, 5 edition, 2001.

[79] F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, 1999.

[80] Brad Johanson, Armando Fox, and Terry Winograd. The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. *IEEE Pervasive Computing*, 1(2):67–74, 2002.

[81] J. Johnson, T.L. Roberts, W. Verplank, D.C. Smith, C.H. Irby, M. Beard, and K. Mackey. The Xerox Star: A Retrospective. *Computer Graphics*, 22(9):11–26, 1989.

[82] Michael J. Jones and James M. Rehg. Statistical color models with application to skin detection. *International Journal of Computer Vision*, 46(1):81–96, 2002.

[83] Timor Kadir and Michael Brady. Saliency, Scale and Image Description. *International Journal of Computer Vision*, 43(2):83–105, 2001.

[84] Rudolph E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASMA; Journal of Basic Engineering*, 82:35–45, 1960.

[85] Yan Ke and Rahul Sukthankar. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. Technical Report 15, Intel, 2003.

[86] R. Kjeldsen and J. Kender. Interaction with On-Screen Objects using Visual Gesture Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 788–793, 1997.

[87] Rick Kjeldsen, Anthony Levas, and Claudio Pinhanez. Dynamically Reconfigurable Vision-Based User Interfaces. In *International Conference on Computer Vision Systems*, pages 323–332, 2003.

[88] J. J. Koenderink and A. J. van Doorn. Representation of Local Geometry in Visual Systems. *Biological Cybernetics*, 55:367–375, 1987.

[89] Hideki Koike, Yoichi Sato, Yoshinori Kobayashi, Hiroaki Tobita, and Motoki Kobayashi. Interactive TextBook and Interactive Venn Diagram: Natural and Intuitive Interfaces on Augmented Desk System. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 121–128, 2000.

[90] Svetlana Lazebnik, C. Schmid, and Jean Ponce. Affine-Invariant Local Descriptors and Neighborhood Statistics for Texture Recognition. In *International Conference on Computer Vision*, pages 649–656, 2003.

[91] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, November 1998.

[92] Christophe LeGal, Jerome Martin, Augustin Lux, and James L. Crowley. SmartOffice: Design of an intelligent Environment. *IEEE Intelligent Systems*, 1:60–67, 2001.

[93] Tony Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, 1994.

[94] Tony Lindeberg. Principles for Automatic Scale Selection. In *Handbook on Computer Vision and Applications*, volume 2, pages 239–274. Academic Press, Boston, USA, 1999.

[95] David Lowe. Object Recognition from Local Scale-Invariant Features. In *International Conference on Computer Vision*, 1999.

[96] David Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[97] P. Maes, T. J. Darrell, B. Blumberg, and A. P. Pentland. The ALIVE System: Wireless, Full-body Interaction with Autonomous Agents. *ACM Multimedia Systems*, 5(2):105–112, 1997.

[98] S. Malassiotis, N. Aifanti, and M.G. Strintzis. A Gesture Recognition System Using 3D Data. In *International Symposium on 3D Data Processing, Visualization and Transmission*, pages 190–193, 2002.

[99] Jitendra Malik, Serge Belongie, Jianbo Shi, and Thomas Leung. Textons, Contours, and Regions: Cue Combination in Image Segmentation. In *International Conference on Computer Vision*, 1999.

[100] David Marr. *Vision*. W. H. Freeman and Company, 1982.

[101] David Marr and E. Hildreth. Theory of Edge Detection. In *Royal Society of London B*, volume 290, pages 199–218, 1980.

[102] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. In *British Machine Vision Conference*, 2002.

[103] J. Matas, S. Obdrzalek, and O. Chum. Local Affine Frames for Wide-Baseline Stereo. In *International Conference on Pattern Recognition*, 2002.

[104] Stephen J. Mckenna and Kenny Morrison. A Comparison of Skin History and Trajectory-Based Representation Schemes for the Recognition of User-Specific Gestures. *Pattern Recognition*, 37:999–1009, 2004.

[105] G. J. McLachlan and K. E. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, Inc., 1988.

[106] K. Mikolajczyk and C. Schmid. A Performance Evaluation of Local Descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 257–264, 2003.

[107] X. Mo and R. Wilson. Video Modelling and Segmentation using Gaussian Mixture Models. In *International Conference on Pattern Recognition*, volume 3, pages 854–857, 2004.

[108] Todd K. Moon. The Expectation-Maximization Algorithm. *IEEE Signal Processing Magazine*, pages 47–60, 1996.

[109] H. P. Moravec. Visual Mapping by a Robot Rover. *Internation Joint Conference on Artificial Intelligence*, pages 598–600, 1979.

[110] Louis-Philippe Morency and Trevor Darrell. Stereo Tracking Using ICP and Normal Flow Constraint. In *International Conference on Pattern Recognition*, 2002.

[111] Louis-Philippe Morency and Rakesh Gupta. Robust Real-Time Egomotion From Stereo Images. In *Internation Conference On Image Processing*, 2003.

[112] Louis-Philippe Morency, A. Rahimi, and Trevor Darrell. Fast 3D Model Acquisition from Stereo Images. In *International Symposium on 3D Data Processing, Visualization and Transmission*, 2002.

[113] C. S. Myers and L. R. Rabiner. A Comparative Study of Several Dynamic Time-Warping Algorithms for Connected Word Recognition. *The Bell System Technical Journal*, 60(7):1389–1409, 1981.

[114] Anonymous n/a and Apple Computer. *Macintosh Human Interface Guidelines*. Addison-Weslet Publishing Company, 1995.

[115] N. Navab and A. Shashua. Algebraic Derivations of Relative Affine Structure and Applications to 3D Reconstructions from 2D Views. Technical Report 270, M.I.T. Media Laboratory Perceptual Computing Sections, 1994.

[116] Kai Nickel and Rainer Stiefelhagen. Pointing Gesture Recognition based on 3D-Tracking of Face, Hands and Head Orientation. In *Workshop on Perceptive User Interfaces*, pages 140–146, 2003.

[117] Y. Ohta, Takeo Kanade, and T. Sakai. Color Information for Region Segmentation. *Computer Graphics and Image Processing*, 13(3):222–241, 1980.

[118] Kenji Oka, Yoichi Sato, and Hideki Koike. Real-Time Fingertip Tracking and Gesture Recognition. *IEEE Computer Graphics and Applications*, 22(6):64–71, 2002.

[119] Kazunori Okada, Dorin Comaniciu, and Arun Krishnan. Scale Selection for Anisotropic Scale-Space: Application ot Volumetric Tumor Characterization. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 594–601, 2004.

[120] Daniel R. Olsen, Sean Jefferies, Travis Nielsen, William Moyes, and Paul Fredrickson. Cross-modal Interaction Using XWeb. In *ACM Symposium on User Interface Software and Technology*, pages 191–200, 2000.

[121] Vasu Parameswaran and Rama Chellappa. View Invariants for Human Action Recognition. In *Computer Vision and Pattern Recognition*, volume 2, pages 613–619, 2003.

[122] Sylvain Paris, Francois Sillion, and Long Quan. A Volumetric Reconstruction Method from Multiple Calibrated Views Using Global Graph-Cut Optimization. Technical Report 4348, The French National Institute for Research in Computer Science and Control, 2003.

[123] V. I. Pavlovic, R. Sharma, and T. S. Huang. Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):677–695, 1997.

[124] W. D. Penny. KL-Divergences of Normal, Gamma, Dirichlet and Wishart densities. Wellcome Department of Cognitive Neurology, University College Longon, Lecture Notes, March 2001.

[125] Alex Pentland and Andrew Liu. Modeling and Prediciton of Human Behavior. *Neural Computation*, 11(1):229–242, 1999.

[126] Ken Perlin and David Fox. Pad - An Alternative to the Computer Interface. *Computer Graphics*, 27:57–64, 1993.

[127] S. L. Phung, A. Bouzaerboum, and D. Chai. Skin Segmentation Using Color Pixel Classification: Analysis and Comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):148–154, 2005.

[128] Claudio Pinhanez. The Everywhere Displays Projector: A Device To Create Ubiquitous Graphical Environments. Technical report, IBM Thomas Watson Research Center, 2002.

[129] F. Quek. Unnencumbered Gesture Interaction. *IEEE Multimedia*, 3(3):36–47, 1996.

[130] Lawrence Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[131] Ravi Ramamoorthi and James Arvo. Creating Generative Models From Range Images. In *SIGGRAPH*, pages 195–204, 1999.

[132] T. Randen and J. H. Husoy. Filtering for Texture Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):291–311, 1999.

[133] Ramesh Raskar, Greg Welch, Matt Cutts, Adam Lake, Lev Stesin, and Henry Fuchs. The Office of the Future: A Unified Approach to Image-Based Modeling and Spatially Immersive Displays. In *SIGGRAPH*, pages 179–188, 1998.

[134] J. M. Rehg. *Visual Analysis of High DOF Articulated Objects with Application to Hand Tracking*. PhD thesis, School of Computer Science, Carnegie Mellon University, April 1995.

[135] J. M. Rehg and Takeo Kanade. Visual Tracking of High DOF Articulated Structures: An Application to Human Hand Tracking. In *European Conference on Computer Vision*, volume 2, pages 35–46, 1994.

[136] Jun Rekimoto, Brygg Ullmer, and Haruo Oba. DataTiles: A Modular Platform for Mixed Physical and Graphical Interactions. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 269–277, 2001.

[137] J. Rissanen. Modeling by Shortest Data Description. *Automatica*, 14(5):465–471, 1978.

[138] Yossi Rubner, Carlo Tomasi, and L. J. Guibas. A Metric for Distributions with Application to Image Databases. In *International Conference on Computer Vision*, 1998.

[139] David Gavilan Ruiz, Hiroki Takahashi, and Masayuki Nakajima. Ubiquitous Image Categorization using Color Blobs. In *Nicograph*, 2003.

[140] Szymon Rusinkiewicz. *Real-Time Acquisition and Renering of Large 3D Models*. PhD thesis, Stanford University, 2001.

[141] M. Salada, J. E. Colgate, M. Lee, and P. Vishton. Validating a novel approach to rendering fingertip contact sensations. In *Proceedings of the 10th IEEE Virtual Reality Haptics Symposium*, pages 217–224, 2002.

[142] Philip M. Sauter. VR2Go: A New Method For Virtual Reality Development. *Computer Graphics*, 37(1):19–24, 2003.

[143] F. Schaffalitzky and Andrew Zisserman. Viewpoint Invariant Texture Matching and Wide Baseline Stereo. In *International Conference on Computer Vision*, volume 2, pages 636–643, 2001.

[144] Robert J. Schalkoff. *Artificial Neural Networks*. The McGraw-Hill Companies, Inc., 1997.

[145] Bernt Schiele and James L. Crowley. Recognition without Correspondence using Multidimensional Receptive Field Histograms. *International Journal of Computer Vision*, 36(1):31–50, 2000.

[146] Steven Schkolne, Hiroshi Ishii, and Peter Schroder. Tangible + Virtual = A Flexible 3D Interface for Spatial Construction Applied to DNA. Technical report, California Institute of Technology, 2003.

[147] C. Schmandt. Spatial Input/Display Correspondence in a Stereoscopic Graphic Workstation. *Computer Graphics*, 17(3):253–261, 1983.

[148] C. Schmid. *Appariement d'images par invariants locaux de niveaux de gris*. PhD thesis, Institut National Polytechnique de Grenoble, France, 1996.

[149] C. Schmid and R. Mohr. Local Grayvalue Invariants for Image Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–535, 1997.

[150] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of Interest Point Detectors. *International Journal of Computer Vision*, 37(2):151–172, June 2000.

[151] Oliver Schonbrunner. Human-Computer Interface in the CAVE. In *Central European Seminar on Computer Graphics*, pages 1–10, 2000.

[152] J. Segen and S. Kumar. Fast and Accurate 3D Gesture Recognition Interface. In *International Conference on Pattern Recognition*, volume 1, pages 86–91, 1998.

[153] J. Segen and S. Kumar. GestureVR: Vision-Based 3D Hand Interface for Spatial Interaction. In *ACM Interactional Multimedia Conference*, 1998.

[154] O. Shaer and R. J. K. Jacob. Toward a Software Model and a Specification Language for Next-Generation User Interfaces. In *ACM CHI 2005 Workshop on The Future of User Interface Softwarer Tools*, 2005.

[155] Yifan Shi, Yan Huang, David Minnen, Aaron Bobick, and Irfan Essa. Propagation Networks for Recognition of Partially Ordered Sequential Action. In *Computer Vision and Pattern Recognition*, volume 2, pages 862–869, 2004.

[156] Min C. Shin, Leonid V. Tsap, and DMitry B. Goldgof. Gesture Recognition Using Bezier Curvers for Visualization Navigation from Registered 3-D Data. *Pattern Recognition*, 37(0):1011–1024, 2004.

[157] B. Shneiderman. Direct Manipulation: A Step Beyond Programming Languages. *IEEE Computer*, 16(8):57–69, 1983.

[158] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London and New York, 1986.

[159] Greg Slabaugh, Bruce Culbertson, Tom Malzbender, and Ron Schafer. A Survey Of Methods for Volumetric Scene Reconstruction From Photographs. Technical report, Hewlett-Packard Laboratories, 2001.

[160] Thad Starner and Alex Pentland. Real-Time American Sign Language Recognition from Vision using Hidden Markov Models. Technical Report 375, MIT Media Laboratory, 1996.

[161] Rahul Sukthankar, R. G. Stockton, and M. D. Mullin. Smarter Presentations: Exploiting Homography in Camera-Projector Systems. In *International Conference on Computer Vision*, volume 1, pages 247–253, 2001.

[162] I. Sutherland. A Head-Mounted Three Dimensional Display. *FJCC*, pages 757–764, 1968.

[163] M. J. Swain and D. H. Ballard. Color Indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[164] Richard Szeliski and Sing Bing Kang. Direct Methods for Visual Scene Reconstruction. *IEEE Workshop on Representation of Visual Scenes*, pages 26–33, 1995.

[165] Richard Szeliski and David Tonnesen. Surface Modeling with Oriented Particle Systems. In *SIGGRAPH*, pages 185–194, 1991.

[166] D. Tell and S. Carlsson. Wide Baseline Point Matching Using Affine Invariants Computed From Intensity Profiles. In *European Conference on Computer Vision*, pages 814–828, 2000.

[167] Demetri Terzopoulos and M. Vasilescu. Sampling and Reconstruction with Adaptive Meshes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 70–75, 1991.

[168] Carlo Tomasi, Slav Petrov, and Arvind Sastry. 3D Tracking = Classification + Interpolation. In *Proc. Int'l Conf. Computer Vision*, pages 1441–1448, 2003.

[169] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[170] T. Tuytelaars and Luc Van Gool. Wide Baseline Stereo Matching Based on Local, Affinely Invariant Regions. In *British Machine Vision Conference*, 2000.

[171] A. van Dam. Post-WIMP User Interfaces. *Communications of the ACM*, 40(2):63–67, 1997.

[172] A. van Dam. Beyond WIMP. *IEEE Computer Graphics and Applications*, 20(1):50–51, 2000.

[173] A. van Dam. User Interfaces: Disappearing, Dissolving, and Evolving. *Communications of the ACM*, 44(3):50–52, 2001.

[174] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-Dimensional Scene Flow. In *International Conference on Computer Vision*, volume 2, pages 722–729, 1999.

[175] Christian von Hardenberg and Francois Berard. Bare-Hand Human-Computer Interaction. In *Perceptual User Interfaces*, 2001.

[176] K. Wang. SALT: A Spoken Language Interface for Web-Based Multimodal Dialog Systems. In *Proceedings of International Conference on Spoken Language Processing*, 2002.

[177] M. Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):94–104, 1991.

[178] E. W. Weisstein. Binomial Series, From Mathworld – A Wolfram Web-Resource, `http://mathworld.wolfram.com/BinomialSeries.html`.

[179] Greg Welch and Gary Bishop. An Introduction to the Kalman Filter. Technical report, University of North Carolina at Chapel Hill, 95.

[180] Pierre Wellner. Interacting with Paper on the Digital Desk. *Communications of the ACM*, 36(7):86–97, 1993.

[181] Andrew Wilson and Aaron Bobick. Parametric Hidden Markov Models for Gesture Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):884–900, 1999.

[182] Andrew Wilson and N. Oliver. GWindows: Towards Robust Perception-Based UI. In *IEEE Workshop on Human Computer Interaction at Conference on Computer Vision and Pattern Recognition*, 2003.

[183] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-Time Tracking of the Human Body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[184] C. R. Wren and A. P. Pentland. Dynamic Models of Human Motion. In *International Conference on Automatic Face and Gesture Recognition*, 1998.

[185] Ying Wu and Thomas S. Huang. View-independent Recognition of Hand Postures. In *Computer Vision and Pattern Recognition*, volume 2, pages 88–94, 2000.

[186] Ying Wu and Thomas S. Huang. Hand Modeling, Analysis, and Recognition. *IEEE Signal Processing Magazine*, 18(3):51–60, 2001.

[187] Junji Yamato, Jun Ohya, and Kenichiro Ishii. Recognizing Human Actions in Time-sequential Images Using Hidden Markov Model. In *Computer Vision and Pattern Recognition*, pages 379–385, 1992.

[188] G. Ye. *Learning 3D Dynamics gestures for Vision-Based Human Computer Interaction.* PhD thesis, Dept. of Computer Science, The Johns Hopkins University, 2005. (expected).

[189] Guangqi Ye, Jason J. Corso, Darius Burschka, and Gregory D. Hager. VICs: A Modular HCI Framework Using Spatio-Temporal Dynamics. *Machine Vision and Applications*, 16(1):13–20, 2004.

[190] Guangqi Ye, Jason J. Corso, and Gregory D. Hager. Gesture Recognition Using 3D Appearance and Motion Features. In *Workshop on Real-Time Vision for Human-Computer Interaction at the Conference on Computer Vision and Pattern Recognition*, 2004.

[191] Guangqi Ye, Jason J. Corso, Gregory D. Hager, and Allison Okamura. VisHap: Augmented Reality Combining Haptics and Vision. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 3425–3431, 2003.

[192] Y. Yokokohji, J. Kinoshita, and T. Yoshikawa. Path planning for encountered-type haptic devices that render multiple objects in 3d space. *Proceedings of IEEE Virtual Reality*, pages 271–278, 2001.

[193] T. Yoshikawa and A. Nagura. A touch/force display system for haptic interface. *Presence*, 10(2):225–235, 2001.

[194] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes. Sketch: An Interface for Sketching 3D Scenes. In *23rd Annual Conference on Computer Graphics and Interactive Techniques, ACM Press*, pages 163–170, 1996.

[195] Z. Zhang. A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[196] Zhengyou Zhang, Ying Wu, Ying Shan, and Steven Shafer. Visual Panel: Virtual mouse, keyboard, and 3D Controller With an Ordinary Piece of Paper. In *Perceptual User Interfaces*, 2001.

[197] H. Zhou, D.J. Lin, and Thomas S. Huang. Static Hand Postures Recognition based on Local Orientation Histogram Feature Distribution Model. In *Proceedings of CVPR Workshop on Real-Time Vision for Human-Computer Interaction*, 2004.

Vita

Jason Joseph Corso was born in Queens, New York, USA on August 6, 1978. He achieved his high school diploma at Chaminade High School in Mineola, New York in 1996. He studied computer science at Loyola College in Baltimore, Maryland, USA and received the Bachelor of Science degree in 2000. He received the James D. Rozics medal from Loyola College, which is awarded to the first ranked student in computer science upon graduation. From 2000 to 2005, he was pursuing his Doctor of Philosophy degree in the field of computer vision in the Department of Computer Science at The Johns Hopkins University in Baltimore, Maryland, USA. During this time, he received the Masters of Science in Engineering in 2002, and the Doctor of Philosophy in Computer Science in 2005. His research interests include vision-based human-computer interaction, image understanding, and artificial intelligence.