Exploiting Texture-Motion Duality in Optical Flow and Image Segmentation

by

Michael G. Ross

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Master of Science
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 2000

© Massachusetts Institute of Technology 2000. All rights reserved.
Exploiting Texture-Motion Duality in Optical Flow and Image Segmentation

by

Michael G. Ross

Submitted to the Department of Electrical Engineering and Computer Science on August 1, 2000, in partial fulfillment of the requirements for the degree of Master of Science

Abstract

This thesis develops an algorithm that combines color and motion information to produce superior optical flow and segmentation output. The algorithm’s main contribution is its ability to exploit the duality of segmentation and flow information. In areas where color/texture segmentation is well-defined, optical flow is under-constrained, and vice-versa. The thesis offers a review of relevant literature, a detailed description of the algorithm, and information about its performance on a variety of natural images.

Thesis Supervisor: Paul A. Viola
Title: Associate Professor
Acknowledgments

I would like to acknowledge the following people, without whom this thesis would not have been possible:

- Sophie Delano, whose constant love, support, and encouragement helped me get through the toughest times of this project.

- My parents, Carol and Gregory Ross, and my sister, Jessica Ross, who have always given me their support and love in all my academic and personal enterprises.

- Professor Paul Viola of M.I.T., my advisor, who allowed me the time and resources to learn about computer vision.

- Professor Daniela Rus of Dartmouth College, who introduced me to artificial intelligence and to research.

- Professor Thomas Cormen of Dartmouth College, who hooked me on computer science.

- John Winn, who listened patiently to my earliest ideas and provided a great data sequence.

- My many friends at the M.I.T. AI Lab who have helped make this thesis better through their advice and discussion. Erik Miller, Huizhen Yu, and Mike Oltmans went the extra mile and provided invaluable proofreading comments.
Contents

1 Introduction ......................................... 10
   1.1 Goals ........................................... 11
   1.2 What is Segmentation? .......................... 11
   1.3 What is Optical Flow? ........................... 12
   1.4 Color and Motion ............................... 13
   1.5 Algorithm ...................................... 14
   1.6 Applications .................................. 15
   1.7 Organization .................................. 16

2 Related Work ........................................ 17
   2.1 Optical Flow ................................... 17
      2.1.1 Gradient Methods ......................... 18
      2.1.2 Matching Methods ......................... 22
      2.1.3 Frequency Methods ......................... 23
      2.1.4 Matrix Methods ............................ 25
      2.1.5 Performance ............................... 26
   2.2 Segmentation ................................. 27
      2.2.1 Graphical Methods ......................... 27
      2.2.2 Clustering Methods ......................... 28
      2.2.3 Snakes and Bayes/MDL Methods ............ 29
      2.2.4 Tractability ............................... 30
   2.3 Constraints in Vision Algorithms .............. 30
3 The Flow-Segmentation Algorithm

3.1 Color/Texture Image Segmentation ........................................... 33
3.2 Segmentation-Constrained Optical Flow .................................... 35
  3.2.1 The Energy-Minimization Criterion ...................................... 36
  3.2.2 Minimization Method ...................................................... 38
  3.2.3 Minimization Correctness ............................................... 42
3.3 Merging Color Regions Based On Motion .................................. 43
3.4 Recomputing Optical Flow .................................................. 45

4 Results ................................................................................. 47

4.1 Evaluation of Optical Flow .................................................. 47
  4.1.1 Algorithms and Data Sets ................................................. 48
  4.1.2 Parameters ..................................................................... 52
  4.1.3 Results .......................................................................... 52
4.2 Evaluation of Segmentation .................................................. 62

5 Conclusions and Future Work .................................................. 66

5.1 Contributions ................................................................. 66
5.2 Future Work ..................................................................... 67
List of Figures

1-1 The dotted box indicates the visible part of the image. Even though the dark square moves diagonally, an observer examining the visible section only sees motion to the right, perpendicular to the edge of the box (and parallel to the image gradient). ........................................ 13

2-1 On the left, a first-order neighborhood system. On the right, a second-order system. .......................................................... 31

3-1 The Flow-Segmentation algorithm. ........................................ 34

3-2 An optical flow calculated without any segmentation information. The green areas indicate no motion, the dark gray indicates motion two pixels to the left, and the blue indicates motion one pixel to the left. Other colors indicate other motions. ........................................ 37

3-3 An optical flow calculated with segmentation information. The green areas indicate no motion, the dark gray indicates motion two pixels to the left, and the blue indicates motion one pixel to the left. Other colors indicate other motions. ........................................ 38

3-4 A simple example of the graph used in the Boykov, Veksler, and Zabih energy-minimization algorithm. The nodes labeled $\alpha$ and $\bar{\alpha}$ represent the labels; the other large nodes represent pixels. The small, purple nodes are auxiliary nodes separating differently-labeled pixel-nodes. 40
4-1 Ambiguity in an apparently "simple" optical flow. Is the white square moving? Or is it attached to a moving background? Or is the white patch a hole in a moving black object? ........................................ 48

4-2 Sample frames from the beginning, middle, and end of the JOHN, RUBIC, SIMPLE, and SQUARE sequences. ............................ 50

4-3 Sample frames from the beginning, middle, and end of the TAXI, TREES, and YOSEMITE sequences. ................................. 51

4-4 The optical flow of a middle frame of the JOHN sequence, analyzed by the six optical flow algorithms. From top-left: Ross, Anandan, Fleet & Jepson, Horn & Schunck, Lucas & Kanade, Nagel. ............... 54

4-5 The optical flow of a middle frame of the RUBIC sequence, analyzed by the six optical flow algorithms. From top-left: Ross, Anandan, Fleet & Jepson, Horn & Schunck, Lucas & Kanade, Nagel. ............... 55

4-6 The optical flow of a middle frame of the SIMPLE sequence, analyzed by the six optical flow algorithms. From top-left: Ross, Anandan, Fleet & Jepson, Horn & Schunck, Lucas & Kanade, Nagel. ............... 56

4-7 The optical flow of a middle frame of the SQUARE sequence, analyzed by the six optical flow algorithms. From top-left: Ross, Anandan, Fleet & Jepson, Horn & Schunck, Lucas & Kanade, Nagel. ............... 57

4-8 The optical flow of a middle frame of the TAXI sequence, analyzed by the six optical flow algorithms. From top-left: Ross, Anandan, Fleet & Jepson, Horn & Schunck, Lucas & Kanade, Nagel. ............... 58

4-9 The optical flow of a middle frame of the TREES sequence, analyzed by the six optical flow algorithms. From top-left: Ross, Anandan, Fleet & Jepson, Horn & Schunck, Lucas & Kanade, Nagel. ............... 59

4-10 The optical flow of a middle frame of the YOSEMITE sequence, analyzed by the six optical flow algorithms. From top-left: Ross, Anandan, Fleet & Jepson, Horn & Schunck, Lucas & Kanade, Nagel. .......... 60
4-11 On top, there are two consecutive frames from the JOHN sequence. The shadow of the moving figure can be seen on the wall behind him. Lower Left: The segmentation of the first frame - note that the shadowed part of the wall is in the wall segment. Lower Right: The resulting flow, which assigns the wall a small leftwards motion due to the shadow’s motion.

4-12 Each row contains a frame, its original segmentation, and a segmentation modified by motion information.
List of Tables

3.1 Graph Edge-Weights. From Boykov et al. ................. 41
Chapter 1

Introduction

The problem of image segmentation is fundamental to the field of computer vision and image understanding. Any intelligent system seeking information from a set of pixels needs to find sound grouping principles to aid in the process of moving from an external, noisy, world of photons to an internal, object-based representation. The computational and bandwidth demands of the raw visual information output by a high resolution camera or by the human eye require segmentation.

Although the term “segmentation” is most commonly used in the computer vision community to refer to a color or texture-based grouping of image elements, in its broadest sense it can also encompass the process known as “optical flow,” the determination of the motion present in a video sequence. While traditional segmentation tells us about the static similarities in an image, optical flow produces dynamic groupings; it tells us what things are moving together.

The past few decades of research have provided many algorithms designed to solve the optical flow and segmentation problems. Optical flow algorithms have been notably more successful than their segmentation counterparts, but this is partially because they are solving a better-defined problem. Neither field has developed truly general algorithms that perform well on all data sets.

Although they operate in different domains, optical flow and segmentation algorithms both try to extract information from the same reality - physical objects in the real world. Therefore, it seems likely that each could be improved with the addition
of data from the other's domain. In this thesis, we develop a framework that allows two distinct partitioning domains, the color domain and the motion domain, to be combined to produce improved optical flow and segmentation results.

1.1 Goals

The goal of this research is to examine methods of combining visual information from disparate low-level processing algorithms into representations that will be more useful to a higher-level system. This thesis takes a step in that direction by describing an algorithm that allows the production of optical flows that are strongly influenced by color segmentation information and color segmentations influenced by motion information. In the future, we hope to extend the system so that it fully integrates both color and motion cues into a single high-level segmentation that will more closely approximate a "general-purpose" segmentation.

1.2 What is Segmentation?

As we mentioned before, segmentation in the broadest sense can encompass optical flow. In the literature, however, it refers to color and texture-based algorithms that operate on individual, static, monocular images, and we will conform to this definition for the remainder of this thesis.

In the image segmentation literature, it is often unclear what is meant by the term "segmentation." Because most segmentation algorithms rely on color and texture properties to get their results, there is a confusion between the means of segmentation and its purpose. Segmentation’s purpose is to divide an image into regions that are meaningful to some higher-level process. Outside a specific application, such as medical imaging, “meaningful” regions are loosely defined as those regions that a human observer finds natural. So, a segmentation algorithm that divides the top of a desk from the desk drawers is usually considered to be behaving correctly, so long as the two elements are visually distinct, but an algorithm that places a door in the
same region as a person standing in front of it is in error. There can be no question that the subjective and non-mathematical nature of these definitions greatly impede progress in producing general-purpose segmentation algorithms. In fact, it could be fairly argued that in the segmentation domain the very concept of a "general purpose" is purely mythical.

The purpose of segmentation is vague, but the means employed by segmentation algorithms are concrete. Most image segmentation algorithms choose an image quality that is a strong indicator of a region's cohesion or division, and find the set of segments that best-fit it to the image. There are many reasonable choices in this domain and we will discuss more of them in Chapter 2.

In much of the segmentation literature, these single-criteria algorithms are put forward as solutions to the general-purpose segmentation problem. Our belief is that better segmentations can be achieved by combining information from different criteria. Therefore we develop a framework that enhances a color-based segmentation criterion with the input of a motion-based grouping criterion, such as optical flow.

1.3 What is Optical Flow?

According to Horn, the optical flow of a sequence is "the apparent motion of the brightness pattern" [15]. In most cases, this should correspond to the motion field, the two-dimensional projection of the actual three-dimensional movement of objects in the world. This supposition rests on the assumption of brightness constancy, the hypothesis that the pixels representing points on a moving object have the same intensity throughout the sequence.

Even without considering the information lost by dimensional reduction, optical flow algorithms are hampered by the aperture problem [15]. Consider a small image patch containing a sharp edge. If we watch the movement of that edge between two frames, it is only possible to determine the component of the optical flow that lies perpendicular to it (Figure 1-1). This generalizes to the restriction that optical flow can only be determined in the directions parallel to the local image gradient. Thus,
Figure 1-1: The dotted box indicates the visible part of the image. Even though the dark square moves diagonally, an observer examining the visible section only sees motion to the right, perpendicular to the edge of the box (and parallel to the image gradient).

In a highly textured region, it is easy to determine optical flow because there are high gradients in many directions at every location. In a low-texture region, such as a patch of constant color, there is no local flow information and we can only find a flow by imposing non-local constraints - importing information from more textured parts of the image plane.

Despite these problems, optical flow is a much better understood problem than segmentation. A number of algorithms exist for finding optical flows and we examine some of the best-known methods in Chapter 2. All of them rely on additional constraints to solve the aperture problem. The algorithm we will introduce in this thesis uses the additional constraints provided by a color segmentation algorithm to solve it.

1.4 Color and Motion

Color and motion algorithms are excellent candidates for algorithmic synergy. Consider a naive color-segmentation algorithm that places neighboring pixels in the same segment if and only if their RGB values are within a certain fixed distance of one another. An algorithm like this will perform well on “cartoon” images in which we
assume the segments are piecewise constant or piecewise smooth. But it will fail to group together pixels that are part of more complex patterns - such as the wood grain of a desk. If we expand the algorithm to include textured regions we find ourselves dealing with additional complexities. How can we determine the difference between a texture-caused variation and one caused by a segment boundary? We can employ local heuristics or smoothing to ease the problem, or force the algorithm to consider global metrics and complex texture models. No matter what strategy we adopt, it is clear that color-segmentation algorithms function best and produce results with the greatest degree of certainty in low-texture regions.

Optical flow algorithms suffer from the inverse problem. Optical flow is well-defined at precisely the locations where color segmentation is ill-posed, and optical flow methods fail where color segmentation succeeds easily. Complex textures and patterns have many large local gradients, which strongly constrain optical flow, but piecewise constant regions contain no local image gradient and, therefore, no motion data. We can try to spread information from well-defined locations to low-gradient areas, but this typically results in blurry motion boundaries. The alternative is to avoid classifying the motion of low texture regions, but this means the algorithm will perform poorly on simple artificial sequences, such as Figure 1-1. By incorporating segmentation, we can use strong flow information to solve areas with weakly constrained flow, but avoid smearing motion boundaries.

Because of this duality, it is possible to construct an algorithm which uses segmentation to assist in the determination of optical flow and vice-versa.

1.5 Algorithm

The algorithm is covered thoroughly in Chapter 3. It consists of the following steps:

1. Compute a color segmentation.

2. Using an energy-minimization criterion, calculate an optical flow by constraining flow within each color segment to be smooth.
3. Again using energy-minimization, join together similarly colored, neighboring segments that share similar optical flows.

This algorithm relies primarily on the observations made in the previous section about the duality of color and motion processing. We first segment the image by color, which, given a reasonable setting of the segmentation algorithm’s parameters, results in over-segmentation. Each object is represented by one or more color patches. At a minimum, these color patches have borders with significant image gradients, otherwise the algorithm would have joined them to their neighbors.

The second step computes the optical flow by calculating the local error values for each potential flow vector and finding a minimum for an energy function that favors smooth flow variations within each segment. Because each segment contains some areas with strong image gradients, there is enough information to constrain the flow. By matching the smoothness criteria to the segment boundaries, we avoid smoothing across object borders and can produce sharply defined flow fields.

Next, we return to the original segmentation and adjust it by merging similarly colored, neighboring segments that share motion, based on the hypothesis that they represent pieces of the same object.

When the algorithm is finished, we have produced two segmentations, each based on two image cues. One is primarily color-based and secondarily influenced by motion, the second is primarily motion-based with color support. Therefore, we have modified raw, low-level processing into a more robust form by mixing it with information from independent sources. These segmentations occupy a middle position on the continuum that stretches from low-level, single-cue segmentation to high-level, multi-cue, general-purpose segmentation. Thus they can be termed “intermediate” segmentations, or representations, of the scene.

1.6 Applications

Like all flow and segmentation algorithms, the work presented in this thesis is designed to be useful to higher-level vision tasks. These could potentially include object
recognition, machine-learning, or three-dimensional processing. When the output of hundreds of cameras needs to be combined into a single reconstruction, the ability to convert from pixels to reliable segments will be crucial in reducing the search spaces to manageable size. Optical flow algorithms that preserve motion boundaries are essential to the understanding of dynamic scenes and to applications such as object tracking.

1.7 Organization

The remainder of this thesis is organized into four chapters. Chapter 2 is an overview of related work in the fields of segmentation, optical flow, and constraint implementation. Chapter 3 gives a thorough overview of the algorithm and Chapter 4 covers the experimental results. Chapter 5 concludes the thesis with a look at potential future work.
Chapter 2

Related Work

The algorithm presented in this thesis is based on three areas of vision research: optical flow, image segmentation, and constraint implementation. Optical flow algorithms take two (or more) frames of a video sequence and try to determine the two-dimensional projection of the observed world's motion onto the image plane. In image segmentation, we seek to develop algorithms that can divide an image into regions consistent with the objects present in those images, usually based on intensity, color, or texture information. Finally, solution constraints, often represented with Markov Random Fields in computer vision, provides an essential framework for the design of vision systems and will form the glue between the problem domains of optical flow and segmentation.

2.1 Optical Flow

Optical flow is useful because it is a crucial part of many other perceptual tasks, such as recovering full three-dimensional motion, deducing structure, and segmenting objects. Apart from its utility, it has also been a focus of study because it presents serious computational challenges. All formulations require strong prior assumptions to make flow-recovery a well-posed problem, and these assumptions can lead to inaccurate results in some situations.

In this section, we provide an overview of several optical-flow algorithms. For
convenience they are divided into four families - gradient approaches, feature matching algorithms, frequency methods, and matrix formulations. Due to the nature of the flow problem, the difficulty of determining a ground truth flow in natural images, and the complexities of comparing these algorithms to one another, there is no consensus on a single best method.

2.1.1 Gradient Methods

One of the earliest, and still the best-known, optical flow methods was developed by Horn and Schunck in 1981 [16]. They laid the foundation for the gradient approach to optical flow. The method places two constraints on the flow field. The first is the brightness constancy constraint and is common to most optical flow algorithms. Assuming that the intensities of objects' images do not change, if an element at location \((x, y)\) is translated by velocity \((u, v)\), it must be the case that \(I_t(x, y) = I_{t_2}(x + u\delta t, y + v\delta t)\), where \(I_t(x, y)\) represents the brightness (irradiance) of location \((x, y)\) at time \(t\) and \(\delta t = t_2 - t_1\).

Horn and Schunck use the brightness constancy constraint and the assumption that all motion can be locally modeled by translations to derive the following equations, assuming that irradiance varies smoothly across space and time:

\[
\begin{align*}
\frac{dI}{dt} &= 0 \\
\frac{\partial I}{\partial x} \frac{dx}{dt} + \frac{\partial I}{\partial y} \frac{dy}{dt} + \frac{\partial I}{\partial t} &= 0 \\
\frac{\partial I}{\partial x} + v \frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} &= 0.
\end{align*}
\]

Unfortunately, this provides two unknowns \((u, v)\) and only one linear equation to constrain them. Therefore, Horn and Schunck introduce the second constraint - that the velocity field should vary smoothly across the image. This allows them to compute optical flow via an iterative error minimization which approximates solving the following integral equation, which combines brightness constancy with an
unsmoothness penalty term:

$$
\varepsilon^2 = \int_x \int_y \left( u \frac{\partial I}{\partial x} + v \frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} \right)^2 + \alpha(u_x^2 + u_y^2 + v_x^2 + v_y^2) dy dx.
$$

(2.4)

The under-constrained nature of Equation 2.3 is physically expressed as the aperture problem. In a gray-scale image, one can only recover the motion of a local patch in the direction parallel to the image gradient. So, local information can at best provide half of the information needed to determine \( u \) and \( v \) (and, in an untextured region, will be able to provide no information because there is no gradient). The Horn and Schunck iterative solution to Equation 2.4 propagates motion information from high-gradient locations into areas where it is locally unrecoverable due to the gradient.

Although the smoothness constraint is necessary to the Horn and Schunck algorithm, this assumption is not always correct and many authors have investigated how to improve performance in these cases. Some of the most common breakdowns occur at the boundaries of objects, where motion is likely to be discontinuous. Schunck and Horn investigated this problem with their original algorithm in a later paper [30] and argued for incorporating a measure of discontinuity likelihood based on similarities in the brightness constancy constraint lines.

Another early group of optical flow algorithms were developed by Nagel in the early and mid 1980s. Nagel proposed two methods of solving the aperture problem, one related to Horn and Schunck's smoothness constraint, and the other based on locating gray-scale corners.

In the first method, Nagel suggests replacing the general smoothness term of the Horn and Schunck equation with an "oriented" smoothness measure [25]. Nagel's smoothness constraint takes the form of a matrix of the second derivatives of the gray-scale values. For each image location \( X \), with gray-scale value \( g(X) \), he defines:

$$
W = \begin{bmatrix}
g_{yy} & -g_{xy} \\
-g_{xy} & g_{xx}
\end{bmatrix}
$$

(2.5)
\[
U = \begin{bmatrix}
    u_x & v_x \\
    u_y & v_y
\end{bmatrix}.
\] (2.6)

Then he replaces the Horn and Schunck smoothness term with a weighted curvature term:

\[
E_{\text{curvature}} = \text{trace} \left[ U^T(W)(W)^T U \right].
\] (2.7)

Nagel's modification is easily visualized if the coordinates have been aligned with the principal curvatures, which sets \(g_{xy} = 0\). In this case, the \(x\)-motion terms \((u_x, v_x)\) are multiplied by the \(y\) curvature and the \(y\)-motion terms are multiplied by the \(x\) curvature. In other words, because motion is highly constrained in the direction of high gray-scale curvature, smoothness in velocity is only imposed in the direction perpendicular to the gradient-constrained direction. This preserves some motion boundaries that are smoothed out by the Horn and Schunck algorithm because it preserves the strong local information along the image gradient direction. Nagel notes that if \(W\) is replaced with the identity matrix, his method reduces to the Horn and Schunck unoriented smoothness constraint.

In his second paper, Nagel combines this method with the constraint of gray-scale corners [26]. This approach is similar in spirit to oriented smoothness because it focuses on avoiding smoothing when local information is sufficient to constrain the flow vectors. In the neighborhood around a gray-scale corner, there exist two perpendicular image edges. If we assume that the corner represents a unified feature that moves along a constant \((u, v)\) vector between frames, we can avoid the aperture problem, since the motion perpendicular to the gradient of one edge equals the unknown motion along the gradient of the other edge. Nagel relies on preprocessing to detect the location of these edges and uses these assumptions to calculate the optical flow at their locations first. Then he uses oriented smoothness and Horn and Schunck-style iterative propagation to fill in the under-constrained regions. Because his methods use a more restricted smoothness criterion, they are more successful at preserving motion boundaries than the general smoothness constraint of Horn and Schunck.
Bergen et al. [3] attack the problem of inappropriate smoothing by viewing it in terms of the single motion assumption. They observe that over-smoothing occurs when we assume that a single motion vector can describe the motion of pixels within a local neighborhood. They calculate a motion field using standard methods and then use it to form two difference images from a three-frame sequence:

\[ D_1 = I_2(x,y) - I_1^p(x,y) \]  \hspace{1cm} (2.8)
\[ D_2 = I_3(x,y) - I_2^p(x,y). \]  \hspace{1cm} (2.9)

\( I_k^p \) is the intensity that results from warping a frame \( k \) by motion field \( p \) and is calculated using a modified Horn-Schunck method that assumes only one motion at each area in the image. If there are actually two motion fields present, \( D_1 \) and \( D_2 \) should only contain differences that are not explained by motion field \( p \) and we can calculate another field \( q \) by examining them.

Bergen et al. claim that after 3 to 5 cycles of alternately computing \( p \) and \( q \), the results converge to within 1% of ground truth in the artificial examples they analyzed. Not only does this method allow direct modeling of two-motion areas, such as object boundaries, but it solves some cases that neither Horn’s nor Nagel’s algorithm is well-equipped to handle. One of these is two textured, partially transparent surfaces sliding over each other. The paper only partially addresses the one-motion case, however. The authors claim that if only one field is present in an area, the other field converges to tracking noise and that this can be detected via statistical tests.

Black and Anandan deal with over-smoothing by using robust statistics [5]. In place of squared error terms, they use truncated quadratic functions that ignore error contributions that are above a predetermined threshold. Applying these functions to the brightness constraint equation results in reduced noise sensitivity, while applying them to the smoothness function prevents smoothing over sharp motion boundaries. In other work, they include a temporal constraint - a tendency for acceleration to be zero between frames [4]. In this paper, they also reformulate the equations as Markov
Random Fields, which can be efficiently solved using a combination of simulated annealing and motion prediction. In both approaches excellent results are achieved on artificial data. Black and Anandan also apply the MRF algorithm to a complex helicopter flight sequence and find qualitatively good results.

2.1.2 Matching Methods

Although gradient methods have been very popular and received a great deal of attention since Horn and Schunck's paper first appeared, many other algorithms are based on the concept of feature-matching. Instead of looking at the gradients at every point in the image, these algorithms attempt to match entire features - such as distinctive patterns or precomputed edges - between frames.

Although originally developed for stereo matching, many people use the Lucas and Kanade registration algorithm as a method for calculating optical flow [23]. This algorithm presumes the existence of spatially localized features in one frame and attempts to match them in the next frame. Assuming translational motion, it finds $h$ such that $\sum_{x \in X} I_1(x + h) = \sum_{x \in X} I_2(x)$ where $X$ is the set of pixel locations that make up the feature in the second frame.

Lucas and Kanade note that searching the entire state space of possible $h$ values is $O(m^2 n^2)$ where the search region is of size $n \times n$ and the possible values of $h$ are of size $m \times m$. Therefore, they propose a hill-climbing algorithm based on the linear approximation of spatial derivatives.

They note that for a small displacement, one can approximate $h$ using spatial derivatives:

$$h = \frac{I_2(x) - I_1(x)}{I_1'(x)}.$$  \hspace{1cm} (2.10)

This approximation forms the basis for their iterative matching algorithm, which estimates $h$, displaces the feature in $I_1$ by that value, and then reestimates $h$ from the new starting point. This method converges if the displacement of the feature is less than half the wavelength of the smallest wavelengths present in the search window.
Therefore, they suggest implementing a multi-resolution matching scheme - starting the process on low-resolution, low-frequency versions of the image and then using the results from that level to find subsequent, higher-resolution matches.

Anandan [1] also advocates a multi-resolution, pyramid matching scheme. He notes that large-scale feature matching gives large-displacement, low-confidence flow information, while small-scale matching provides small-displacement, high-confidence flow. He constructs a Laplacian pyramid to decompose the image into multiple levels. This pyramid is constructed by creating a Gaussian pyramid (in which each level is the Gaussian-smoothed version of the prior level) and then subtracting each level’s predecessors from it.

Starting at the top of the pyramid, Anandan uses a sum of squared differences (SSD) metric and a 5 x 5 pixel search window to exhaustively search for the best possible match. He also applies a confidence measure to the result - rating it as low-confidence if it lies in a canyon or plane on the SSD surface, since that indicates that there is little information available to differentiate the minimum from other possibilities. These matches are then smoothed, with a bias towards maintaining high-confidence features, and used to initialize the next finer level of matching.

Anandan’s pyramid is closely related to earlier work by Burt, Yen, and Xu [7]. They create a pyramid of band-pass filtered representations of each image and combine motion data from each level. They also compute the reliability of the motion hypotheses at each location given the available image data.

The multi-resolution (coarse to fine) methods discussed in these three papers represent a common technique in many optical flow algorithms.

2.1.3 Frequency Methods

It is not intuitively obvious that motion could be elegantly expressed in the frequency domain, but formulating it in this way allows for another interesting class of optical flow algorithms. If we consider all the frames of a video sequence simultaneously, the two-dimensional images sweep out a three-dimensional volume over time. Heeger [14] points out that in space-time, the translational movement of a simple feature, such
as a vertical bar, forms a plane. Similar relations hold in spatiotemporal frequency space, a Fourier space composed of spatial ($\omega_x$ and $\omega_y$) and temporal ($\omega_t$) frequencies for which:

$$\omega_t = u\omega_x + v\omega_y.$$  

(2.11)

This holds for all the frequency components of a small, translating pattern. The fundamental observation supporting this formulation is easiest to understand in terms of a single, one-dimensional sine-wave with frequency $\omega_{x_1}$, translating with velocity $v$. This situation produces a temporal frequency $\omega_t = \omega_{x_1}v$. Now consider a wave moving at the same velocity, but with twice the spatial frequency ($\omega_{x_2} = 2\omega_{x_1}$). If we observe a fixed location in space, the peaks of the wave pass through it twice as quickly, so $\omega_t = \omega_{x_2}v$. Generalizing this to two spatial dimensions gives Heeger’s planar constraint.

Heeger convolves the space-time sequence with a 3-d generalization of the Gabor filter, which selects oriented frequency responses in spatiotemporal frequency space. By measuring the amplitudes of the response generated by a set of filters spanning a discrete set of velocities, the algorithm estimates the orientation of the plane that corresponds to the motion of the pattern. Heeger notes that his results seem to correspond well with psychophysical analysis of human vision, even with respect to the aperture problem. Many scientists who study the visual cortex have observed frequency-selective cells operating in the visual path, so the frequency optical flow method has some basis in biology as well as in computation and mathematics.

Fleet and Jepson [12] follow a similar line of reasoning, but base their solution on the phase contours returned by the Gabor filters, rather than on the amplitude component of their response. They demonstrate that a pattern following a particular motion field will produce curves of constant phase response from the Gabor filters that corresponds to that motion, while in some situations the curves of constant amplitude response will indicate an incorrect motion pattern. Experimentally, their method provides higher accuracy in many situations than Heeger’s amplitude method,
often because phase-response is more robust than amplitude-response in the presence of common photometric effects, such as lighting changes and noise.

2.1.4 Matrix Methods

Matrix-based optical flow methods use a careful formulation of the flow problem in an attempt to render it soluble with relatively weak additional constraints. Irani [17] demonstrates that the set of all flow vectors in many scenes form a very low dimensional subspace and are therefore over-constrained, rather than under-constrained.

Irani defines matrices $U$ and $V$ to contain the horizontal and vertical flow components for all image points in all frames. If there are $F$ frames and $N$ points, this gives us two matrices of size $F \times N$. Irani then assumes that either the scene is imaged under orthographic projection, or that camera rotation is small and forward translation is small relative to scene depth. Under these assumptions, she finds that the matrices $[U|V]$ and $[V]$ have a rank less than or equal to 9. This work is related to Tomasi and Kanade’s similar analysis of shape and motion recovery [34]. There they found a rank of less than 3 for each matrix, but required that the tracking information of image features be precomputed. Irani’s derivation, on the other hand, only requires image-based measurements. Because it is possible to use these matrices to compose versions of either Horn and Schunck’s brightness constraint or Lucas and Kanade’s window-matching equations, rank-reduction allows us to solve equations that appear under-constrained in a naive analysis.

Irani uses the matrices composed of $U$ and $V$ to derive generalized versions of both the Horn and Schunck and the Lucas and Kanade equations:

\[
[U|V]\begin{bmatrix}
\frac{F_x}{F_y}
\end{bmatrix} = F_t
\]

\[
[U|V]\begin{bmatrix}
\frac{A}{B} | \frac{C}{D}
\end{bmatrix} = [G|H].
\]

In both of these equations, all matrices except $U$ and $V$ can be calculated directly from the image data. She makes an initial estimate of the motion and decomposes the
results in order to compute the basis for the low-rank subspace that all the motion vectors will reside in. Then she uses that to reduce the rank of $U$ and $V$ and calculate them in either the modified Horn and Schunck or Lucas and Kanade equations. This process is repeated iteratively to refine the estimates produced for $U$ and $V$.

2.1.5  Performance

Unfortunately, performance comparison remains a weakness in the optical flow literature. It is difficult to determine the ‘true’ optical flow of natural images, especially since the human visual system is susceptible to many of the same ambiguities that plague computer systems. Barron et al. [2] compared several different methods, including Horn and Schunck, Lucas and Kanade, Anandan, Heeger, and Fleet and Jepson. They concluded that the Lucas and Kanade matching method and the Fleet and Jepson phase shift methods were the most accurate, but comparison was made difficult because the densities of the motion fields provided by each algorithm varied considerably. For example, the Horn and Schunck method, which was found to be the least accurate, provided a motion estimate for 100% of the pixels in each example, while Fleet and Jepson and Lucas and Kanade only estimated flow for 30% - 70% of the pixels, depending on the input and the settings used in the algorithms.

Clearly, the optical flow problem has not been completely and convincingly solved by any of these algorithms and it is still a very open research topic. While it is certainly well-defined compared to image segmentation, it sits on the boundary between the physical and interpretive visual processes, which is reflected in the need for strong prior assumptions to make it soluble. It seems unlikely that the aperture problem can be solved by a single “correct” constraint. Many of the papers examined here noted that the human visual system can calculate incorrect flows under various circumstances.
2.2 Segmentation

Many techniques have been developed for image segmentation. In the literature, the task of segmentation is sometimes treated as part of object recognition. Here, we will use the traditional definition of segmentation as a division of the image into color or textured-based regions without the benefit of a database of object models.

2.2.1 Graphical Methods

Many segmentation algorithms operate on graph-based representations of the images, arranging the pixels as a lattice of vertices connected using either a first or second-order neighborhood system. Felzenszwalb and Huttenlocher [10] connect vertices with edges weighted by the intensity or RGB-space distance between the vertices' pixel values. Then they sort the edges and merge the pixels together into larger and larger regions by considering each edge in order. Further details can be found in Chapter 3.

Shi and Malik's [32] well-known method also employs a graphical model, but instead of growing regions, it starts with every pixel part of a single region. Then the image is divided by successive “normalized cuts” until a cut-value threshold is reached. The normalized cut criteria is based on the familiar minimum cut problem from algorithmic graph theory. In min-cut, we attempt to divide a graph into two unconnected subgraphs by removing edges such that the combined weight of the removed edges is minimized. Therefore, if the weight of edges varies proportionally with the similarity between the pixels it joins, finding the min-cut of a graph should separate it into dissimilar regions. Unfortunately, min-cut favors separating the graph into one small region and one large region. To correct for this, Shi and Malik use the following criteria:

\[ Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}. \]  

Here \( cut(A, B) \) is the value of the cut separating \( A \) and \( B \) and \( assoc(A, V) \) tells the total weight of all connections from \( A \) to all nodes of the graph. This method
can also be used for motion segmentation and for tracking [33], which is interesting considering the focus of the work presented in this thesis.

Wu and Leahy’s algorithm [36] is a precursor to the work of Shi and Malik. They partition by taking repeated minimum cuts, but without a normalization factor. Their algorithm employs edge nodes, which are analogous to the MRF line-process. Between every pair of graph vertices representing neighboring pixels, they insert a vertex whose connections will represent the possibility of an edge occurring in that location.

Although Shi and Malik develop their methods from a graphical perspective, they use an eigenvector decomposition of a matrix model to find the solution in practice. Weiss [35] notes that normalized cuts are part of a family of algorithms that operate on the affinity matrix. This matrix is defined by \( W(i, j) = e^{-\frac{d(x_i, x_j)}{s^2}} \), where \( d(x, y) \) measures the difference in value between a pair of features and \( s \) is a free parameter. Other methods that operate on this representation include the work of Perona and Freeman [28] and the algorithm of Scott and Longuet-Higgins [31]. Weiss emphasizes that the normalization performed by Shi and Malik is crucial for adequate performance on natural images.

### 2.2.2 Clustering Methods

Because we are concerned with grouping similar picture elements together, a clustering approach to segmentation is very natural. Instead of embedding the image and interpixel relationships in a graph, they are transformed into a feature space and grouped together. Methods that employ the clustering methodology have to decide how to perform the clustering and how to structure the feature space to preserve texture/color and locational similarity.

Jain and Farrokhnia [18] compute features by passing the image through a set of differently-oriented Gabor filters and non-linear functions to compose features of the magnitude of each region’s response. Mao and Kain [24], on the other hand, create features by describing each pixel’s value in terms of the local mean gray-scale value and the values of its neighbors. Both use the k-means clustering algorithm to compute clusters and rely on the modified Hubert (MH) index to determine how
many clusters should be present in the image. The MH value measures the correlation of the distances between pairs of features and the distances between their current labels. Mao and Kain employ additional feature terms to enforce the spatial locality of each segment. Pauwels and Frederix \[27\] implement a non-parametric clustering that considers the similarity of a pixel's nearest feature-space neighbors to its own and penalizes the existence of voids between similarly categorized pixels.

Schroter and Bigun \[29\] use a multi-resolution approach similar to those popular in optical flow in order to avoid adding spatial locality as a feature. By initiating clustering at extremely low resolution and propagating the results downwards, spatial locality is enforced at the coarse levels and persists into the higher-resolution pyramid base.

Felzenszwalb and Huttenlocher \[11\] have developed a hybrid approach that combines elements of graphical and clustering segmentation. They use their original graph algorithm, but connect the pixels to their nearest neighbors in an \((R, G, B, x, y)\) feature space, rather than to their neighbors in the image lattice.

### 2.2.3 Snakes and Bayes/MDL Methods

Another significant subset of segmentation algorithms can be grouped as derivatives of the pioneering "snakes" algorithm developed by Kass et al.\[19\]. In this algorithm one or more closed region boundaries are defined and given initial positions. Then they are modified by the following energy function:

\[
E_{\text{snake}} = \int_0^1 E_{\text{int}}(v(s)) + E_{\text{image}}(v(s)) + E_{\text{con}}(v(s)) ds. \tag{2.15}
\]

The first energy term measures the bending in the curve, the second computes its desirability as a boundary of the image data it is located around, and the third represents an externally imposed constraint - a user-specified initial location in the case of the original implementation. Typically, the snakes are encouraged to be smooth and to favor high-gradient locations that might indicate segment boundaries.

Another significant approach is the minimum description length segmentation pro-
posed by LeClerc [21]. LeClerc creates a series of increasingly complex language models for images. First he describes them as a collection of piecewise constant regions, next he examines piecewise smoothness, and then he adds noise models of varying complexity. The best segmentation of an image is the one that gives the smallest description of it using the chosen language. This criterion is based in information theory and the relationship between entropy, information, and data compression.

Zhu and Yuille unify snakes, MDL, and region-growing segmentation algorithms such as Felzenszwalb and Huttenlocher's method under their own segmentation formulation [37]. This method uses an EM-style formulation in which the algorithm alternates between classifying pixels into probabilistic regions and then uses the classifications to modify the region parameters. It also makes an attempt to classify based on surface albedo and thus avoid artificial, illumination-produced boundaries, but this requires a number of simplifying assumptions that do not hold in the general case.

2.2.4 Tractability

For a problem as complex as segmentation, the issue of computational tractability naturally arises. Cooper proves that while computing a segmentation based on local uniformity criteria is a polynomial time operation, segmentations which combine uniformity and a bias towards fewer output regions are NP-complete [9]. The algorithm we employ in Chapter 3, designed by Felzenszwalb and Huttenlocher, runs in $O(n \log n)$ by locally approximating a bias towards fewer regions [10].

2.3 Constraints in Vision Algorithms

Over the past two decades, Markov Random Fields (MRFs) have provided an important framework for constraint-based vision algorithms. The most influential work in this area is Geman and Geman's work on Bayesian image restoration [13]. A good overview of MRF techniques in vision problems can be found in Li [22].
A Markov Random Field is a set of labels (Ω) for a data set (X) that follow the following two restrictions:

\[
P(X = \omega) > 0 \quad \forall \omega \in \Omega \quad (2.16)
\]
\[
P(X_s = x_s | X_r = x_r, \forall r \neq s) = P(X_s = x_s | X_r = x_r, r \in N_s). \quad (2.17)
\]

The first property states that there must be a positive probability that the data conforms to any potential set of labels. The second equation, the "Markovianity" property, tells us that the probability of a particular location receiving a label, given all the other label assignments, is only conditioned on the labels given to a subset of the other location. This subset, N_s, is referred to as the neighborhood of the location.

A significant part of using an MRF-style algorithm successfully is determining the appropriate neighborhood structure. Two very commonly used neighborhoods are the first-order system, which is often referred to as "four-connected," and the second-order system, often called "eight-connected." They are shown in Figure 2-1.

![Figure 2-1: On the left, a first-order neighborhood system. On the right, a second-order system.](image)

In MRFs, and in many energy equations, the neighborhoods represent statistical dependence. Graph-based segmentation algorithms also define neighborhood systems, but these usually represent topological connectedness. It is important to note that there is no agreement among image researchers over what neighborhood models are best, even in the domain of image segmentation. Although it appears to be more
robust, a naive eight-neighbor model treats two sets of neighbors as equivalent - the nearby horizontal and vertical neighbors and the more distant diagonal neighbors. These difficulties are an unfortunate consequence of using discrete, rectangular pixel lattices to represent images.

MRFs allow a great deal of flexibility in designing solutions to image problems. Geman and Geman [13] demonstrated that successful image restoration could be accomplished by coupling two MRFs together - one representing the labels on image pixels, the other, called a “line process,” representing the probability of an edge between any pair of neighboring pixels. In our algorithm for finding optical flow, described in Chapter 3, we will use the output of a segmentation algorithm in a role analogous to that of a line process MRF. Both prevent smoothing across scene-based discontinuities.

Unfortunately, in their most general form, the best algorithm for minimizing an MRF is the simulated annealing algorithm first described by Kirkpatrick et al. [20], which is extremely slow for large solution spaces. Solutions for a reasonably sized image (320 x 200 pixels) often take several hours. Thus, although the MRF framework is well-proven, its most general form is unsuited for use in processing video, where dozens or hundreds of frames might be considered during a single processing job.

The minimization framework proposed by Boykov et al. [6] employs an energy function form that is similar in form and spirit to an MRF, but it allows for approximate minimization that is several orders of magnitude faster than simulated annealing. This method represents optimization as a graph-cutting problem. Its details are described thoroughly in Chapter 3, which details its use in our optical flow algorithm.
The algorithm presented in this thesis can be divided into three steps.

1. Color image segmentation.
2. Computation of optical flow, constrained by the color segmentation.
3. Further merging color regions based on optical flow results.

Originally, we incorporated a fourth step, the recomputation of optical flow based on the new segmentation. It has been eliminated because it was discovered to be insignificant. A brief discussion of the reasons for this is included at the end of this chapter.

3.1 Color/Texture Image Segmentation

The first step of the algorithm uses the Felzenszwalb-Huttenlocher [10] image segmentation algorithm. The algorithm was selected for its speed, its clear theoretical formulation, and its demonstrated performance on natural images.

Unlike some of the algorithms described in Chapter 2, the Felzenszwalb algorithm uses a simple model that allows it to perform segmentations in $O(n \log n)$ time, but still capture a reasonably general class of textures. It is very similar to the uniformity-criterion based algorithm described in Cooper [9]. Every pixel in the image is initially...
foreach $f_i \in Frames$

\[ S_{f_i} = \text{Segment} \ (f_i) \]

\[ OF_{f_i} = \text{OpticalFlow} \ (S_{f_i}, f_i, f_{i+1}) \]

\[ \text{SegmentWithOptFlow} \ (f_i, S_{f_i}, OF_{f_i}) \]

Figure 3-1: The Flow-Segmentation algorithm.

labeled as a unique segment. The pixels form the vertices in an eight-neighbor connected graph - every pixel $(i, j)$ is connected to $(i - 1, j - 1), (i - 1, j), (i - 1, j + 1), (i, j - 1), (i, j + 1), (i + 1, j - 1), (i + 1, j), \text{and} (i + 1, j + 1)$.

Every edge is associated with two pixels and is given a weight according to the difference in their RGB values. In the case of gray-scale images, this is a simple intensity difference. For color images, Felzenszwalb and Huttenlocher suggest combining three independent intensity segmentations - one each for the red, green, and blue channels. We have found that good results can be obtained by using the Euclidean distance between the pixels' RGB vectors and chose that approach for efficiency reasons.

The edges are sorted in nondecreasing order by their weights and then each edge is used to compute the following merge criterion:

\[
W(e_{i,j}) \leq \text{MaxWeight}(\text{Seg}(i)) + \frac{\rho}{\text{Size}(\text{Seg}(i))} \]  
(3.1)

\[
W(e_{i,j}) \leq \text{MaxWeight}(\text{Seg}(j)) + \frac{\rho}{\text{Size}(\text{Seg}(j))}. \]  
(3.2)

If the edge weight is less than or equal to the heaviest edge already included in each of the segments it connects, then the two segments are merged. The $\frac{\rho}{\text{Size}(\text{Seg}(i))}$ factor encourages small segments to grow until they reach a size such that their MaxWeight values are significant.

Felzenszwalb and Huttenlocher have proven that this method of segmentation will produce a result that is neither too fine nor too coarse, according to their definitions. An overly fine segmentation contains neighboring regions that would pass the merge
criterion, while an overly coarse result contains segments that could be divided into subregions that are not too fine.

There are several things worth noting about the practical performance of this algorithm. The first is that despite its efforts to discourage the formation of small regions, segmenting most natural images inevitably produces some tiny regions. Felzenszwalb and Huttenlocher suggest cleaning these up by merging them into neighboring segments, and we have followed that suggestion. Their implementation of the algorithm also uses Gaussian filtering as a pre-processing step to remove noise, as does our program.

Although this segmentation algorithm is designed to allow the creation of regions with high-weight edges, it is important to note that it is still difficult for them to appear. Because the edges are considered in non-decreasing order, it is the term that controls the degree to which high-weight edges are included. If we ignore that term, at any point in the algorithm, a segment can only expand by including edges equal to those already included in it because of the ordering of the edges. Because the \( \rho \) factor is divided by the segment size, it selects the sample size for determining the internal variability of a segment. Therefore, the selection of \( \rho \) can have a significant impact on the output of a segmentation and may result in over- or under-segmentation of an image. In our work, we try to err on the side of over-segmentation. Because motion relies on texture, it should be possible to give two regions of an over-segmented object the correct motion because they will each contain some texture information. Determining the motions if the regions are under-segmented is likely to result in error if the two objects joined together are moving separately.

### 3.2 Segmentation-Constrained Optical Flow

The next stage in our algorithm uses the output from the segmentation phase to compute an optical flow. Our optical flow formulation uses an energy-based approach that is similar to Markov Random Field theory, but which introduces constraints that make it more tractable than an MRF formulation. This enables us to avoid the
costs involved in general MRF minimization, which typically requires computationally
intensive methods such as simulated annealing [13] [22] [20].

3.2.1 The Energy-Minimization Criterion

As discussed in Chapter 2, an optical flow algorithm requires a source of local informa-
tion and a larger-scale constraint in order to solve the aperture problem. According
to Horn and Schunck [16], the following energy function should be minimized:

\[ E = \int \int (u \frac{\partial I}{\partial x} + v \frac{\partial I}{\partial y} + \frac{\partial I}{\partial t})^2 + \alpha(u_x^2 + u_y^2 + v_x^2 + v_y^2) dydx. \] (3.3)

The first term encourages the \( u \) and \( v \) components of the flow vectors to match
the local image information and the second requires \( u \) and \( v \) at a pixel to be similar
to the \( u \) and \( v \) flows for its neighbors. This is a global smoothness criterion and is
similar to the following discrete criterion:

\[ E = \sum_{x} \sum_{y} |I_1(x, y) - I_2(x + u, y + v)| + \alpha \sum_{i \in N(x,y)} \sqrt{(u - u_i)^2 + (v - v_i)^2}. \] (3.4)

The two images, \( I_1 \) and \( I_2 \), represent two consecutive frames in a video sequence.
Just as in the Horn and Schunck formulation, \( u \) represents the \( x \)-component of motion
and \( v \) the \( y \)-component. Each pixel can be assigned a different \( u \) and \( v \). The energy
criterion imposes two requirements - that the proposed motion match the local image
values \( (I_1(x, y) = I_2(x + u, y + v)) \), and that the motion at any pixel \( (x, y) \) be similar
to the motion of any pixel \( i \) that is a neighbor of \( (x, y) \).

If we use the obvious neighborhood system, the eight-connected neighborhood
used in the segmentation algorithm, we will get a result that does not match well
with our perception of the motion in the sequence. Figure 3-2 shows the result of
minimizing Equation 3.4 using the algorithm described in Section 3.2.2 and an eight-
connected neighborhood.

Although it is clear that motion in the frame is sharply defined, the general neigh-
Figure 3-2: An optical flow calculated without any segmentation information. The green areas indicate no motion, the dark gray indicates motion two pixels to the left, and the blue indicates motion one pixel to the left. Other colors indicate other motions.

The neighborhood system causes a great deal of blurring at motion boundaries (Figure 3-2). We will focus on improving the result by altering the neighborhood definition. Instead of using eight-connected neighborhoods, we will connect a pixel to its neighbors according to the color segmentation produced in the first step.

Referring to the original, eight-connected neighborhood as $N_8$, we will define a new neighborhood for use in motion-smoothing:

$$N_{\text{seg}}(x, y) = \{(i, j) | (i, j) \in N_8(x, y) \text{ and } \text{Seg}(i, j) = \text{Seg}(x, y)\}.$$  \hspace{1cm} (3.5)

This is the eight-neighborhood with the additional constraint that neighbors must be in the same segment. When this modified neighborhood definition is used with our optical flow criterion, the result is significantly improved (Figure 3-3).

Now the motion boundaries are sharp and clear. Why does the use of segmentation information produce such clear motion boundaries? Most segmentation algorithms, including Felzenszwalb and Huttenlocher’s method, form regions composed of a low-variance interior surrounded by a high-variance edge that separates the inte-
rior from neighboring regions. In addition, the Felzenszwalb algorithm can produce regions with high-variance interiors. Therefore, at a minimum, optical flow information is available all around the edges of a segment. So long as the segment does not contain pieces of more than one independently moving object, it stands to reason that it forms a good smoothing region for optical flow calculations. Given this justification, it is important to note that over-segmentation is preferable to under-segmentation. If under-segmentation is a significant possibility, the choice of $\alpha$ in the energy-minimization equation is more difficult, since we must prevent over-smoothing within segments. This also negatively impacts our ability to remove noise via the use of a large $\alpha$.

### 3.2.2 Minimization Method

To minimize the energy function, we use an algorithm developed by Boykov, Veksler, and Zabih [6]. The algorithm imposes a few minimal constraints on the form of our energy function and provides a fast, near-optimal solution in return.

Let $f$ be an assignment of labels to every pixel, and let $f_p$ be the label assigned
to a particular pixel $p$. $N$ is the set of all pairs of neighboring pixels. The Boykov et al. algorithm is designed to minimize energy functions of the form:

$$E(f) = E_{\text{data}}(f) + E_{\text{smooth}}(f)$$  \hfill (3.6) \\
$$E_{\text{data}}(f) = \sum_{p \in P} D_p(f_p)$$  \hfill (3.7) \\
$$E_{\text{smooth}}(f) = \sum_{\{p,q\} \in N} V_{\{p,q\}}(f_p, f_q).$$  \hfill (3.8)

The $E_{\text{data}}$ term, which measures the distance between a pixel and its current label, corresponds to the first term in Equation 3.4. The label $f_p$ assigned to a pixel $p$ represents its current motion hypothesis $(u, v)$. The $E_{\text{smooth}}$ term, which calculates a penalty for differences in labeling between neighboring pixels, corresponds to the second part of Equation 3.4.

The algorithm imposes no constraints on the $D_p$ functions, which measure local error, so long as they are always greater than or equal to 0. The version of the Boykov method that we employ requires that the $V_{\{p,q\}}$ family of functions are all metric measures. This means that $V(A, B) = V(B, A)$, $V(A, B) = 0$ if and only if $A = B$, and that the measure obeys the triangle inequality ($V(A, B) \leq V(A, C) + V(C, B)$). These constraints are clearly satisfied by the energy function in Equation 3.4.

Generalized energy-minimization is computationally very expensive, especially when one relies on an algorithm such as simulated annealing [20], which only allows for small steps towards the minimum. The Boykov algorithm allows for large steps and therefore is fast and well-suited to dealing with the large data sets typically seen in image processing.

The algorithm is as follows:

1. Assign an initial set of labels to the graph nodes.

2. For each label $\alpha$, find the lowest energy label-expansion. If the post-expansion energy is lower than the original system energy, relabel the nodes with its results.

3. If any relabeling has taken place, repeat Step 2. Otherwise, end.
Clearly, the most difficult step is the computation of the minimum $\alpha$-expansions in Step 2. The process of label-expansion is a search to determine if increasing the number of pixels under a particular label will lower the energy of the system. Boykov et al. have demonstrated that this can be accomplished efficiently by constructing a graph that represents the energy function, and performing a min-cut on the graph. The source of the graph is a special node representing the current label $\alpha$, and the sink is a node representing all the non-$\alpha$ labels. If the minimum cut separates a node from the source, it is $\alpha$-labeled by the minimizing expansion, otherwise it retains its original label.

The specifics of the graph's construction are best illustrated by the following figure and chart, both adapted from [6].

![Figure 3-4: A simple example of the graph used in the Boykov, Veksler, and Zabih energy-minimization algorithm. The nodes labeled $\alpha$ and $\bar{\alpha}$ represent the labels; the other large nodes represent pixels. The small, purple nodes are auxiliary nodes separating differently-labeled pixel-nodes.](image)

The links are given weights as indicated in the table below, taken from Boykov et al. [6].

The t-links represent the likelihood that a particular pixel node will be labeled $\alpha$ or $\bar{\alpha}$, while the e-links represent the likelihood that neighboring nodes will have the same label. The weight of a t-link to a label increases as the compatibility between that label and the data at a particular node decreases. Very high-weight t-links are unlikely to be broken by the minimum-cut. Therefore, a pixel attached to $\alpha$ by a strong link will likely remain on the $\alpha$ side of a minimum cut and retain its original
Table 3.1: Graph Edge-Weights. From Boykov et al.

<table>
<thead>
<tr>
<th>Edge</th>
<th>Weight</th>
<th>For</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e_{p, P} )</td>
<td>( \infty )</td>
<td>( p \in P \alpha )</td>
</tr>
<tr>
<td>( e_{p, P} )</td>
<td>( D_p(f_p) )</td>
<td>( p \in P \alpha )</td>
</tr>
<tr>
<td>( e_{p} )</td>
<td>( D_p(\alpha) )</td>
<td>( p \in P )</td>
</tr>
<tr>
<td>( e_{i(p,q)} )</td>
<td>( V_{i(p,q)}(f_p, \alpha) )</td>
<td>( {p,q} \in N, f_p \neq f_q )</td>
</tr>
<tr>
<td>( e_{i(a,q)} )</td>
<td>( V_{i(p,q)}(\alpha, f_q) )</td>
<td>( {p,q} \in N, f_p \neq f_q )</td>
</tr>
<tr>
<td>( e_{a} )</td>
<td>( V_{i(p,q)}(f_p, f_q) )</td>
<td>( {p,q} \in N, f_p = f_q )</td>
</tr>
</tbody>
</table>

Labeling. The effect of the e-links is not as intuitive, but they serve to prevent cuts that would result in local unsMOOTHNESS.

In order to minimize our energy function, we use the segmentation-enhanced neighborhood constraint, as discussed before, and assign the following functions to \( D_p \) and \( V_{i(p,q)} \), where \( p \) and \( q \) represent two pixels’ \((x, y)\) coordinates in the image, and \( M \) is the set of possible motion vectors. \( R, G, \) and \( B \) are the intensities of the three color channels. In the second equation, \( m_p \) indicates the motion vector currently assigned to pixel \( p \) and \( m_{p(x)} \) and \( m_{p(y)} \) are the \( x \) and \( y \)-components of that vector. The functions are:

\[
D_p(m \in M) = \sqrt{I_{\text{diff}}(R_2, R_1, p, m) + I_{\text{diff}}(G_2, G_1, p, m) + I_{\text{diff}}(B_2, B_1, p, m)} \quad (3.9)
\]

\[
I_{\text{diff}}(I_2, I_1, p, m) = (I_2(p + m) - I_1(p))^2 \quad (3.10)
\]

\[
V_{i(p,q)}((m_{p}, m_{q}) \in \{M \times M\}) = \beta \sqrt{(m_{p(x)} - m_{q(x)})^2 + (m_{p(y)} - m_{q(y)})^2} \quad (3.11)
\]

In our experiments, \( M \) was composed of the discrete motion vectors covering a two-pixel radius from the original location of any pixel. This gives us 25 possible labels for every pixel in the Boykov algorithm. The initial assignment of labels places each pixel under a label that minimizes the local motion error. If one of the locally minimal labels is the null-motion label, that label is preferred over other possibilities.

Although the \( D_p \) equation allows for color images, all of our experiments were with
gray-scale pictures because the implementations of the other optical flow algorithms did not allow color input.

On a Pentium III processor running at 866 MHz, our implementation computes one frame (288 x 216 pixels) of optical flow in approximately 2.5 minutes. Cutting a graph representing all the pixels takes approximately 4 seconds, using a push-relabel algorithm originally described and implemented by Cherkassky and Goldberg [8]. A significant percentage of this cost is incurred by the graph-construction routines; reuse of portions of the graph structure between iterations might improve running times significantly, but implementing it would be decidedly non-trivial. Between any two iterations, the graph can undergo significant change even if no relabeling occurs.

The algorithm is highly parallelizable. Graph-cutting via the push-relabel algorithm is \( O(V^3) \), where \( V \) is the number of vertices in the graph. But, because the segments are independent of one another, it is possible to compute the optical flow of each segment separately. Therefore, significant savings can be found by splitting the problem amongst \( N \) processors, where \( N \) is the number of segments. This gives \( O(V_{\text{max}}^3) \) where \( V_{\text{max}} \) is the number of vertices in the largest segment. Even on a single processor, significant speed improvements are available by processing the segments in batches. Empirically, a maximum batch size of 10000 pixels per batch has been found to give the minimum running time.

### 3.2.3 Minimization Correctness

The Boykov et al. algorithm does not guarantee a globally minimal result. The variant described in the previous sections is guaranteed to find a state such that the energy is no more than \( 2 \times \frac{\max(V(\alpha, \beta) : \alpha \neq \beta)}{\min(V(\alpha, \beta) : \alpha \neq \beta)} \) times larger than the global minimum [6]. More importantly for our purposes, the algorithm will not stop at any local minimum that is only one label-expansion away from a lower-energy state. This can be proven to give us a global minimum under certain constraints.

**Theorem 1** If the global minimum energy corresponds to the correct optical flow and every segment denotes a region undergoing a single translational motion, then...
the Boykov et al. algorithm will find the correct and globally minimum labeling.

Consider a labeling which is not the global minimum. In this case, there are some pixels in the image that are mislabeled. If an entire segment is mislabeled, which indicates that the local error term is a source of the problem, the energy can be reduced by $\alpha$-expanding a label that better fits the data to the entire segment. The smoothness term in the energy function will not change because the algorithm's neighborhoods never cross segment boundaries. If some pixels within a segment have a label that conflicts with their neighbors, the energy can be reduced by $\alpha$-expanding the label of their neighbors to cover them as well. Because we assume that each segment has a single motion which fits its data best, these are the only two cases that apply if the labeling is not globally optimal. Since both cases can be improved in a single $\alpha$-expansion, the algorithm will not become stuck in either of them and will successfully reach global minimization.

Unfortunately, in many real-world circumstances, these assumptions are incorrect. Although rotations correspond to local translations if the frame rate is high enough, the translation vectors will not be identical across a rotating segment. This violates the "single translational motion" assumption. Another real-world problem involves the consideration of shadows. It can be very difficult to segment shadows out of the objects they are projected on, especially in well-lit environs. A shadow moving across a stationary background object might cause that object to be labeled with a motion. To some degree this can be alleviated by reducing the coefficients of the smoothing term in the energy equation, but it is nearly impossible in practice to choose a value that will smooth non-shadowed segments adequately, but not cause problems with shadows. An example of this problem is given in Chapter 4.

3.3 Merging Color Regions Based On Motion

As described in the introduction, the benefits of combining color and motion constraints should cut both ways. This next step corrects over-segmentation in the
original color segmentation by merging similar adjoining regions that share the same motion.

Just as in the previous section, we will combine two sources of information - color and motion - with the Boykov et al. energy-minimizer. In the previous section, the labels represented motions, and smoothness was constrained by segmentation, but now the labels will represent color segments and smoothness neighborhoods will be based on the motion vectors computed in the prior step.

The full energy equation for this operation is:

\[ E = \sum_{p \in P} D_p(s_p) + \sum_{p_i \in N_p} V_{\{p,p_i\}}(s_p, s_{p_i}). \] (3.12)

In this instance, \( P \) is the set of segments and \( S \) is the set of segment labels. Initially, every \( p \in P \) has a unique segment label \( s \). We want to reduce the energy of the system by merging segments - giving adjacent segments that have similar motions and colors the same segment labels. Consider two segments, \( p_1 \) and \( p_2 \), initially labelled with \( s_1 \) and \( s_2 \). If they are similarly colored and are undergoing the same motion, we can lower the energy of the system by assigning both the \( s_1 \) label.

As before, we must define an appropriate local error term and a neighborhood smoothness constraint. The local error, \( D_p \) measures the appropriateness of a segment label for segment \( p \) using information from the Felzenszwalb-Huttenlocher algorithm. The smoothness function, \( V_{\{p,q\}} \), assigns a penalty to neighboring segments that share the same motion, but different segment labels. The neighborhood, \( N_p \), used in the smoothness summation requires that neighboring segments share the same velocities and contain pixels that are second-order neighbors across their shared segment boundary. \( M \) is the set of motion vectors, equivalent to the set of motion-labels used in the previous step. The functions are:

\[ D_p(s \in S) = \text{MinWeight}(s, s_p) - \min(\text{MaxWeight}(s), \text{MaxWeight}(s_p)) \] (3.13)
\begin{equation}
V_{(p,q)}(s_p, s_q) = \begin{cases} 
0 & s_p = s_q \\
\alpha & m_p = m_q \neq \text{no motion} \\
\beta & m_p = m_q = \text{no motion} 
\end{cases} \quad (3.14)
\end{equation}

Here, MinWeight(s_p, s_q) refers to the weight of the lightest edge connecting segments s_p and s_q. Segments that do not share any boundaries in the image are unconnected or have an infinite value MinWeight. MaxWeight(s) is the heaviest edge included in the segment s by the segmentation algorithm. Therefore, subtracting one weight from the other gives an indication of how close the two segments were to being connected by the original segmentation algorithm.

It is important to note the different smoothing values assigned to the motion-hypothesis case and the no-motion hypothesis case. In a sequence filmed on a fixed camera, it is likely that most of the contents of the scene will be at rest. Therefore, two adjoining segments that share a no-motion tag are only slightly more likely to be joined than two segments that have different motions. Sharing any other motion-label, however, is a much stronger indication that two neighboring segments belong to the same object.

In a future implementation, we might attempt to determine the camera motion so we could designate the no-motion hypothesis with respect to the scene and not just with respect to the images.

The running time of this step is typically only a few seconds on a Pentium III 866 MHz machine. Instead of working with 10,000-60,000 individual pixels, the graph used for this step contains only 100-200 segment-nodes in practice.

### 3.4 Recomputing Optical Flow

Using the motion-adjusted segmentation, we could recompute the optical flow with the same algorithm used in step 2. In practice this does not significantly improve the flow results. Two segments are not merged unless they share the same average motion.
In most cases, all the pixels in a particular segment will be labeled with exactly the same motion, and merging them with another set of identically labeled pixels will have no effect on the smoothing or the local-fit terms of the energy function.
Chapter 4

Results

4.1 Evaluation of Optical Flow

It is difficult to evaluate optical flow algorithms on a quantitative basis. In the well-known study by Barron et al. [2], several optical flow algorithms are evaluated by using artificially generated video sequences. These sequences are devoid of noise and the flow fields that produced them are used to determine “correct” values. Unfortunately, even a simple case can admit multiple, incompatible, flow interpretations. It is not right to penalize an algorithm if it produces a plausible flow field that differs from the field used to generate the images, but which explains the data just as well.

One of the simplest Barron examples involves a white square moving diagonally across a black background. Psychologically, this usually suggests that the background is stable and all the pixels contained in the square are following the same flow vector. However, it is possible that the sequence represents a white square attached to a black background object, which is itself in motion. The white square could be a hole in a moving piece of black material, or perhaps only the edges of the hole are moving. All of these possibilities produce radically different flow patterns (Figure 4-1), for one of the simplest artificial sequences imaginable.

Therefore, we will evaluate the optical flow results by looking at several sequences, natural and artificial, and presenting the flow fields produced by several different algorithms, including our own.
Figure 4-1: Ambiguity in an apparently “simple” optical flow. Is the white square moving? Or is it attached to a moving background? Or is the white patch a hole in a moving black object?

4.1.1 Algorithms and Data Sets

We compared our algorithm to five well-known optical flow algorithms, which were originally compared to one another in Barron et al.[2]. The implementations are provided via anonymous ftp at ftp://ftp.csd.uwo.ca/pub/vision/, along with many of the test data sets used in that paper. We have used 7 sequences, 5 of which come from the Barron data set.

1. JOHN - a noisy video sequence of a man walking to the left.

2. RUBIC - (from Barron) a rubic cube rotating counter-clockwise on a turntable. Created by Richard Szeliski at DEC.

3. SIMPLE - a black square on a white background, traveling one pixel to the right each frame.
4. SQUARE - (from Barron) a white square on a black background, traveling diagonally up and to the right.

5. TAXI - (from Barron) the well-known “Hamburg taxi” sequence, showing several cars moving in different directions on a road. Created at the University of Hamburg.

6. TREES - (from Barron) camera motion past a foreground and background group of trees. Taken from the IEEE Motion Workshop database at the David Sarnoff Research Center.

7. YOSEMITE - (from Barron) an artificial mountain fly-through sequence. Created by Lynn Quam at SRI.

The JOHN, RUBIC, TAXI, and TREES data sets are “natural” - real scenes captured with real cameras. The other data sets, SIMPLE, SQUARE, and YOSEMITE, are artificially generated. All the natural sequences contain noise and the JOHN sequence also contains JPEG compression artifacts. The Barron-implemented algorithms processed the images as gray-scale Sun Rasterfiles, our algorithm read them as gray-scale PGM/PPM files. These formats provide uncompressed, per-pixel information, so each algorithm worked with exactly the same data. Sample images from each sequence are displayed in Figures 4-2 and 4-3.
Figure 4-2: Sample frames from the beginning, middle, and end of the JOHN, RUBIC, SIMPLE, and SQUARE sequences.
Figure 4-3: Sample frames from the beginning, middle, and end of the TAXI, TREES, and YOSEMITE sequences.
4.1.2 Parameters

Each sequence was processed with our algorithm using the same set of input parameters. The images were Gaussian-blurred by a 5x5-pixel filter with a standard deviation of 1.0 pixels. The $\rho$ value in the Felzenszwalb and Huttenlocher segmentation algorithm was set to 300 and all segments smaller than 20 pixels in size were merged into their most similar neighbors. The $\alpha$ value in Equation 3.4, which regulates the degree of smoothing, was a constant in our software and set to 20 in our experiments.

Parameters for the other algorithms were taken from those suggested by the scripts provided with the Barron software. If a script was provided for a particular data set, we used its values, otherwise we used values from a script for similar data. The SIMPLE data was processed with the values for the Barron SQUARE sequence, since they are so similar in content. The JOHN data was processed using the values from the Barron TREES sequence because they are both noisy, natural video sequences. All the software and scripts for generating the output will be available via anonymous ftp at ftp://ftp.ai.mit.edu/people/mgross/pub/.

4.1.3 Results

As can be seen in Figures 4-4 to 4-10, our algorithm performed very well in comparison with the other five on all the data sets, with the exception of the RUBIC data set, as seen in Figure 4-5. It may be significant that this was the only data set that focused on rotation, rather than translation. The strong smoothness assumption, which served us well with the translational sequences, might need to be relaxed in the case of rotation. It could be that our algorithm's performance was impeded by its discretization of the vector space during flow determination. The other optical flow methods use floating-point numbers to represent motion vectors and allow for sub-pixel motion discrimination. Our reliance on a small, discretized set of motion vectors is an unavoidable consequence of employing the Boykov et al.\cite{6} algorithm, which requires a predetermined, finite set of labels.

Our algorithm provided very strong performance on all of the other sequences. It
is interesting to note that it was clearly the most successful of the six algorithms on the two simplest sequences, SIMPLE and SQUARE, shown in Figures 4-6 and 4-7. The Nagel and Horn & Schunck algorithms blurred the edges of the square motions, due to their relatively general smoothness constraints and the untextured nature of the images. The Fleet & Jepson algorithm was only able to determine motion at the corners, where two strong image gradients were present, and all the algorithms, except for ours, were unable to fill in the square. Note that in the SQUARE sequence, our algorithm labelled the entire picture as sharing the motion, which, although slightly contrary to psychological intuition, is a legitimate and reasonable interpretation of what is occurring in the sequence.

The algorithm was also very successful in interpreting complex natural sequences. In the JOHN sequence it picked up on the motion of the walker and correctly labeled most of the background as immobile (Figure 4-4). Because of its strong inter-segment smoothness constraints, the algorithm was very successful at separating background from foreground. The Fleet & Jepson, Lucas & Kanade, and Anandan algorithms can skip labeling points that lack enough information for a flow determination, which can include stable background locations. But, as demonstrated in the natural image sequences, these algorithms often label parts of the moving object as indeterminate as well. Our algorithm is robust to noise and is very successful at labeling both the motion of foreground objects and the non-motion of fixed backgrounds. This has the side-effect of making the results easier to visually interpret, as can be seen in the results for JOHN (Figure 4-4) and TAXI (Figure 4-8). In the case of the results for TREES, our algorithm produced three planes of motion, a fast rightwards plane in the foreground, a slower rightwards plane in the background, and a still plane in the deep background. This corresponds well to human perceptions of the scene, which was created with a translating camera.
Figure 4-4: The optical flow of a middle frame of the JOHN sequence, analyzed by the six optical flow algorithms. From top-left: Ross, Anandan, Fleet & Jepson, Horn & Schunck, Lucas & Kanade, Nagel.
Figure 4-5: The optical flow of a middle frame of the RUBIC sequence, analyzed by the six optical flow algorithms. From top-left: Ross, Anandan, Fleet & Jepson, Horn & Schunck, Lucas & Kanade, Nagel.
Figure 4-6: The optical flow of a middle frame of the SIMPLE sequence, analyzed by the six optical flow algorithms. From top-left: Ross, Anandan, Fleet & Jepson, Horn & Schunck, Lucas & Kanade, Nagel.
Figure 4-7: The optical flow of a middle frame of the SQUARE sequence, analyzed by the six optical flow algorithms. From top-left: Ross, Anandan, Fleet & Jepson, Horn & Schunck, Lucas & Kanade, Nagel.
Figure 4-8: The optical flow of a middle frame of the TAXI sequence, analyzed by the six optical flow algorithms. From top-left: Ross, Anandan, Fleet & Jepson, Horn & Schunck, Lucas & Kanade, Nagel.
Figure 4-9: The optical flow of a middle frame of the TREES sequence, analyzed by the six optical flow algorithms. From top-left: Ross, Anandan, Fleet & Jepson, Horn & Schunck, Lucas & Kanade, Nagel.
Figure 4-10: The optical flow of a middle frame of the YOSEMITE sequence, analyzed by the six optical flow algorithms. From top-left: Ross, Anandan, Fleet & Jepson, Horn & Schunck, Lucas & Kanade, Nagel.
Figure 4-11: On top, there are two consecutive frames from the JOHN sequence. The shadow of the moving figure can be seen on the wall behind him. Lower Left: The segmentation of the first frame - note that the shadowed part of the wall is in the wall segment. Lower Right: The resulting flow, which assigns the wall a small leftwards motion due to the shadow's motion.

Apart from the problems in handling the slow rotation present in the RUBIC sequence, our algorithm’s reliance on the assumption of smoothness within a segment can occasionally cause errors. Sometimes, as seen in Figure 4-11, the motion of a shadow can cause some static objects to be assigned a false motion. The shadow from the moving figure crosses the background, but both the shadow and unshadowed regions of the wall are included in the same segment. Therefore, the lowest-energy motion assignment for the wall assigns a slight leftwards motion to the wall behind the walker.

This problem is essentially one of over-smoothing and can be alleviated by decreasing $\alpha$, the intersegment smoothing parameter. Unfortunately, lowering it enough to avoid the wall effect unacceptably decreases the quality of the result for the figure.
This problem with shadows is a good target for future work and research.

In summary, the algorithm performs well on translational cases, is robust to noise, allows for distinct motion boundaries, and works very well in artificial and natural cases. In several of the sequences, its performance was notably superior to the output of the other 5 algorithms examined. Its weaknesses are its discretization of the flow vectors, its difficulty with rotation, and its performance when the fundamental assumption of coherent intra-segment motion is violated (as in the case of the moving shadow on the wall segment).

In running time, it was slower than all of the other algorithms except for the Fleet algorithm. For two frames of the JOHN sequence, 288x216 pixels in size, our implementation of the algorithm took 151 seconds to calculate the flow on a Pentium III processor running at 866 MHz.

4.2 Evaluation of Segmentation

If determining ground truth in optical flow evaluation problems is difficult, in image segmentation it is impossible. As discussed in Chapter 1, there is no agreed upon criterion for evaluating the coherence of visual regions. If such criteria existed, segmentation research could take a large step forwards.

It is not the purpose of this thesis to argue that the segmentation part of our algorithm is superior to other forms of segmentation, but only to compare the quality of segmentation available before and after the addition of motion information. In some cases, we find that motion information had no appreciable effect on segmentation results. In other cases, it is clear that the segmentation produced with the addition of motion data conforms to our notion of objects in the picture better than the original segmentation. In a few cases, the flow information caused segmentation mistakes not present in the original color segmentations. Overall we conclude that improving color-based segmentation algorithms with motion information is promising, but it requires a more subtle algorithm than the one we employed.

The figures that follow provide a selection of segmentation examples from the
seven data sets. They have been chosen to illustrate successes and failures of the segmentation part of our algorithm.

In the JOHN sequence, we generally were able to improve the segmentation of the walking figure. In the top row, we see that the many of the small segments in the figure have been merged into their larger neighbors because they share the same leftwards motions. This hasn’t happened in all cases, but some pieces of the figure are visually distinct, such as the head and hands. The motion of the feet segments are not similar enough to be merged into the body and there are still some parts of the figure’s shirt that are unmerged, perhaps due to imaging noise or JPEG artifacts. But the changes have also introduced some mistakes. The segments composing the post to the right of the figure have joined together, but have erroneously merged with some of the wall at the post’s base. Both segmentations combine the wall behind the figure with pieces of the floor and the hallway, but the second segmentation does so to a greater degree than the first. One of the weaknesses in our formulation is that if an image is under-segmented, it will tend to become more so - there is no way to use motion information to divide segments.

The TREES segmentations underwent significant improvements after they were augmented with motion information. The foreground tree trunks are clearly one object, but the patterns of light and shadow on them make them difficult to segment. The addition of motion information allowed good integration of many trunk segments from the original segmentation. Also, the ground is made up of fewer segments in the motion-modified segmentation and some of the background trees are better segmented. The addition of the motion information allowed us to overcome some of the intra-object variability and get a better segmentation.

The RUBIC frame undergoes some improvements, mainly in the background segmentation. But some of the squares on the cube, formerly distinct segments, are merged together in this sequence, and the design on the base of the turntable is smeared together in the second segmentation. This is not necessarily an error, but it is not very useful - either the pattern should divide the turntable into segments, or the entire turntable should be one segment. The algorithm does incorporate the
shadowed part of the turntable top into its neighboring segments correctly, which is a nice improvement.

Another weakness of the algorithm is highlighted in the TAXI sequence. In the original segmentation, the taxi is segmented into many pieces, but all of them are separate from the surrounding road. But, because the optical flow only marked the top of the cab as moving, and marked the segments making up the sides as standing still, the shadowed parts of the cab side are incorrectly joined with the road in the second segmentation.

We believe that these results show that improvements to color/texture segmentations via motion information are possible, but we need to develop a better way of integrating the information. Here we have just attempted to reduce over-segmentation by a previously developed segmentation algorithm. In the future, we would like to develop a segmentation algorithm similar to the optical flow algorithm - incorporating motion information into the neighborhood functions used. Such an algorithm should be able to correct for under-segmentation as well as for over-segmentation.
Figure 4-12: Each row contains a frame, its original segmentation, and a segmentation modified by motion information.
Chapter 5

Conclusions and Future Work

5.1 Contributions

This thesis proposes that combining the results of different low-level visual processes can result in natural synergies, whereby weaknesses in the information provided by one process can be compensated by a complimentary process’ information. Optical flow and segmentation are two such processes. Optical flow is strongly determined in high texture regions and weakly constrained in low texture areas. The inverse situation holds true for most color-based segmentation algorithms.

Using this insight, we have developed an algorithm that produces both an optical flow constrained by segmentation information and a segmentation influenced by optical flow data. The optical flow results are very compelling and demonstrate that the algorithm is robust to noise, works well on artificial and natural images, and correctly detects sharp motion boundaries. It has limited success with rotational motion, however, perhaps due to the intra-segment smoothness constraints it imposes or its discretization of the motion vectors.

The segmentation results are also promising and demonstrate that incorporating optical flow data can correct the problem of object over-segmentation. However, it can exacerbate under-segmentation in some cases, which suggests that a redesigned segmentation algorithm will be necessary to put its results on par with the optical flow output.
5.2 Future Work

In the realm of optical flow, it is clear that the algorithm needs to handle rotation better. Although it performed well on all of the other sequences, our method produced some of the worst results on the RUBIC data. There are a few possibilities that require further exploration here. The rotation in the RUBIC sequence was quite slow, so the algorithm might have done better if it had considered velocity vectors with sub-pixel magnitudes. The corresponding problem is that the energy-minimization algorithm is $O(l)$, where $l$ is the number of labels examined, but $l$ increases at $O(n^2)$ where $n$ is the number of discrete distances considered from the center of each flow-search window to its edge. Therefore, if we went from considering full-pixel motion in a 5x5 window to considering half-pixel motion, we would have to go from considering 25 possible labels to considering 81. Since each graph-cutting operation is expensive, this would be a significant increase to the computational burden imposed by the algorithm for even medium sized images.

On the other hand, the RUBIC problems might be alleviated by relaxing the intra-segment smoothness constraint, or by modeling rotation explicitly, rather than relying on translational approximations.

Clearly, there is a need for a more sophisticated approach to modifying the segmentation to conform to the optical flow results. In the future, we would hope to replace the Felzenszwalb-Huttenlocher [10] segmentation algorithm with one more directly compatible with the Boykov et al.[6] energy-minimization methods. Ideally, segmentation would operate in a manner completely analogous to optical flow - flow boundaries would define regions over which color-similarities and locality would determine how likely neighboring pixels are to share the same segment labels.

Another possibility is suggested by Felzenszwalb and Huttenlocher’s revised algorithm [11], which compares $(R,G,B,x,y)$ feature vectors assigned to each pixel to determine connectedness. It might be interesting to examine an $(R,G,B,x,y,u,v)$ neighborhood function, where $u$ and $v$ are the $x$ and $y$ motion vectors.

The algorithms in this thesis produce results that are primarily based on one cue.
with secondary support from a complimentary process. A true integration of cues requires that we produce a single segmentation with equal input from both domains. Accomplishing this may require a new framework, perhaps entirely based on feature-space clustering or on probabilistic methods.

Fundamentally, vision algorithms of the type explored in this thesis should be designed to provide the most useful and reliable information to higher level processes, such as object recognition or stereo-matching. It would be worthwhile to study how well the algorithms presented in this paper can merge with higher-level vision processing and investigate what adaptations might improve their utility. This would clearly include attempts to improve the speed of the optical flow processing. Unless the graph-cutting step can be usefully approximated or otherwise sped-up, the algorithm cannot be used in any real-time systems.
Bibliography


