ELEC 6430 Semester Project: Optical Flow Computation Using the Horn & Schunck Algorithm

Josh Clanton

April 25, 2005

Contents

List of Figures

1 Introduction

In modern-day robotic work, one major task is making an on board computer system "see" its surroundings. If a robot can form a visual perception of its environment, this vision data can be combined with other sensors such as gyroscopes, accelerometers and ultra-sonic sensors to perform some type of autonomous task. The field of computer vision is a relatively new field with many different methods being explored. One of these methods is optical flow.

Optical flow can best be thought of as the movement of objects in a person (or machine's) field of view. Imagine riding down an interstate highway and looking at a large road sign up ahead. As you approach the sign, the object itself appears larger and larger, and its edges tend to move outward in your field of view. As you pass it and then look back, the sign appears smaller and smaller with the objects edges appearing to shrink. By imagining the edges of the object becoming larger or smaller, you are projecting a three dimensional world onto a two dimensional plane. As the edges move, they have apparent velocities relative to the other objects in the field of view. Optical flow techniques attempt to compute the direction and magnitude of these velocity vectors.

2 Optical Flow - Horn & Schunck's Method

The concept of optical flow has existed for quite some time, however, the first major milestone in attempting to calculate optical flow was done by Berthold K.P. Horn and Brian G. Schunck. Horn and Schunck define optical flow as "the distribution of apparent velocities of movement of brightness patterns in an image"[1]. If the velocities of the brightness patterns (objects within the image) are known, then a robot or vision system using the optical flow techniques will have some knowledge of how its surroundings are changing.

Horn and Schunck do point out some relative difficulties before setting up their optical flow calculations. The connection between optical flow in the image plane and object velocity in three dimension is not always accurate. For example if a sphere of constant color and texture is rotating in a sequence of images and the lighting upon the object does not change, no optical flow can be calculated. The object will appear the same when project on a two dimensional plane [1].

2.1 Assumptions

Horn and Schunck note that before computing the optical flow, the problem must be simplified by making several assumptions. First, it is assumed the the object being imaged is a flat surface. Second, it is assumed that the illumination on the image is constant and uniform. It is also assumed that the reflectance of the object varies smoothly and has no spatial discontinuities. These assumptions assure that the image brightness or intensity is differentiable [1].

2.2 Constraints

Horn and Schunck derive equations that relate the change in image brightness at a particular point to the motion of the brightness pattern. To do this, a few constraints are established.

2.2.1 Constant Brightness Constraint

The first constraint says that the intensity at a particular point in an object does not change over time:

$$\frac{dE}{dt} = 0$$

which can be expanded using the chain rule:

$$\begin{aligned} \frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} &= 0 \\ E_x u + E_y v + E_t &= 0 \end{aligned}$$

Rearranging the equation:

$$\begin{bmatrix} E_x & Ey \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -E_t$$

From this, it can be seen that movement component that lies in the direction of the brightness gradient is:

$$-\frac{E_t}{\sqrt{E_x^2 + E_y^2}}$$

This conclusion is very important in that it demonstrates the aperture effect. Because of the brightness constraint, the only component of velocity that can be computed is the portion that lies in the direction of the brightness gradient [1]. A classic example of this is the "barber shop pole" example. Imagine a barber pole rotation. The lines on the pole are actually moving left to right, however, the lines appear to be moving upward. Optical flow calculations would produce the same result because the brightness gradient lies in the vertical direction, not in the horizontal direction.

2.2.2 Smoothness Constraint

Another constraint placed on the optical flow problem is the smoothness constraint. Horn and Schunck note that neighboring points on a moving object have similar velocities, therefore the brightness patterns in the image vary smoothly [1]. Horn and Schunck implement this constraint by the minimization of the square of the gradient magnitudes:

$$\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 \textit{and} \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2.$$

The smoothness of the optical flow field can also be measured by the Laplacians of u and v:

$$\begin{aligned} \nabla^2 u &= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \\ \nabla^2 v &= \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \end{aligned}$$

2.3 Estimations

In order to perform the minimization and compute the velocity vectors, both the partial derivatives of E and the Laplacians of u and v must be estimated.

2.3.1 Partial Derivatives

Horn and Schunck estimate the partial derivatives E_x, E_y and E_t by computing the the average of four first differences in two adjacent slices in the image. The appropriate equations are seen below. They have been modified from Horn and Schunck notation to match matrix indexing in MATLAB.

$$
\begin{aligned}
Ex &\approx \frac{1}{4}[E_{ii+1,jj+1,kk} - E_{ii+1,jj,kk} + E_{ii,jj+1,kk} - E_{ii,jj,kk} \cdots \\
&\quad + E_{ii+1,jj+1,kk+1} - E_{ii+1,jj,kk+1} + E_{ii,jj+1,kk+1} - E_{ii,jj,kk+1}] \\
Ey &\approx \frac{1}{4}[E_{ii,jj,kk} - E_{ii+1,jj,kk} + E_{ii,jj+1,kk} - E_{ii+1,jj+1,kk} \cdots \\
&\quad + E_{ii,jj,kk+1} - E_{ii+1,jj,kk+1} + E_{ii,jj+1,kk+1} - E_{ii+1,jj+1,kk+1}] \\
Ey &\approx \frac{1}{4}[E_{ii+1,jj,kk+1} - E_{ii+1,jj,kk} + E_{ii,jj,kk+1} - E_{ii,jj,kk} \cdots \\
&\quad + E_{ii+1,jj+1,kk+1} - E_{ii+1,jj+1,kk} + E_{ii,jj+1,kk+1} - E_{ii,jj+1,kk}]
\end{aligned}
$$

2.3.2 Laplacians

For the Laplacian estimates, Horn and Schunck use the following approximation:

$$\nabla^2 u \approx k(\bar{u}_{i,j,k} - u_{i,j,k}) \quad and \quad \nabla^2 u \approx k(\bar{v}_{i,j,k} - v_{i,j,k})$$

where $\bar{u}$ and $\bar{v}$ are local averages of the velocity vectors. They are estimated by subtracting the value at a point from a weighted average of neighboring values. Equations for the calculations are:

$$
\begin{aligned}
\bar{u}_{i,j,k} &= \frac{1}{6}[u_{ii-1,jj} + u_{ii,jj+1} + u_{ii+1,jj} + u_{ii,jj-1}] \cdots \\
&\quad + \frac{1}{12}[u_{ii-1,jj-1} + u_{ii-1,jj+1} + u_{ii+1,jj+1} + u_{ii+1,jj-1}] \\
\bar{v}_{i,j,k} &= \frac{1}{6}[v_{ii-1,jj} + v_{ii,jj+1} + v_{ii+1,jj} + v_{ii,jj-1}] \cdots \\
&\quad + \frac{1}{12}[v_{ii-1,jj-1} + v_{ii-1,jj+1} + v_{ii+1,jj+1} + v_{ii+1,jj-1}]
\end{aligned}
$$

2.4 The H&S Algorithm

2.4.1 Minimization

Now that the proper estimations have been made, Horn and Schunck attempt to minimize both the error in the brightness equation and the error in the departure from smoothness.

$$
\begin{aligned}
\varepsilon_b &= E_x u + E_y v + Et \\
\varepsilon_c^2 &= \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2
\end{aligned}
$$

The total error to be minimized is then:

$$\varepsilon^2 = \int\int (\alpha^2 \varepsilon_c^2 + \varepsilon_b^2) dx dy$$

Horn and Schunck then develop a constrained least squares minimization using variational calculus techniques and substituting the partial derivative and Laplacian approximations. (see [1] for a detailed derivation)

2.4.2 Iterative Solution

A direct solution to the constrained minimization is computationally expense, therefore an iterative solution is suggested. This method computes a new set of velocity estimates (u^{n+1}, v^{n+1}) based off of the estimated derivatives and the average of previously estimated velocities [1]. The iterative solution is expressed in the equations below.

$$u^{n+1} = \bar{u}^n - E_x \left(\frac{E_x \bar{u}^n + E_y \bar{v}^n + E_t}{\alpha^2 + E_x^2 + E_y^2} \right)$$
$$v^{n+1} = \bar{v}^n - E_x \left(\frac{E_x \bar{u}^n + E_y \bar{v}^n + E_t}{\alpha^2 + E_x^2 + E_y^2} \right)$$

2.4.3 The Algorithm

Now that iterative solution is in place, an generic algorithm can be set up to perform the optical flow calculation. The actual MATLAB code can be seen in *hsof.m* in Appendix A.

```
%======Optical Flow Algorithm==============
obtain image sequence
get size of image sequence, M(x), N(y), number(t)
set alpha
num_its = number of images

for kk = 1:number
    for ii = 1:M
        for jj = 1:N
            compute Ex, Ey and Et
        end
    end


    for nn = 1:num_its
        for ii = 1:M
            for jj = 2:N
                compute ubar
                compute vbar
                update u
                update v
            end
        end
    end
end
%==========================================
```

3 Experimental Results

The Horn and Schunck algorithm was tested in MATLAB with several different image sequences. This report will show the results from two different image sequences. The first is a sequence generated from a synthetic image. Next, the algorithm is tested on a sequence of real images.

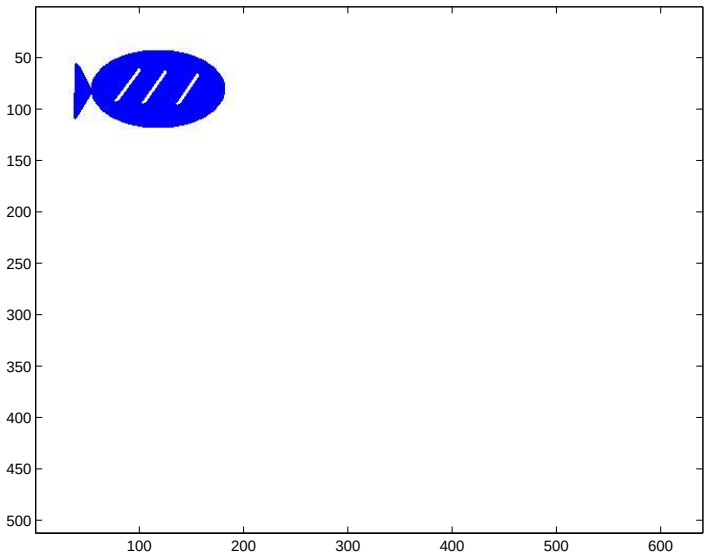


Figure 1: Original synthetic image used to create sequence

3.1 Synthetic Image Test

A synthetic image was created in Microsoft paint. The original synthetic image is seen in Figure 1. Using this image, a sequence was created in which the blue object (shaped like a blimp) in the image is moving down and to the right from its original position. All of the images are converted to gray-scale. The MATLAB code used to create this synthetic image sequence can be seen in Appendix A. Figures 2(a) through 4(a) show the frames of the synthetic image sequence.

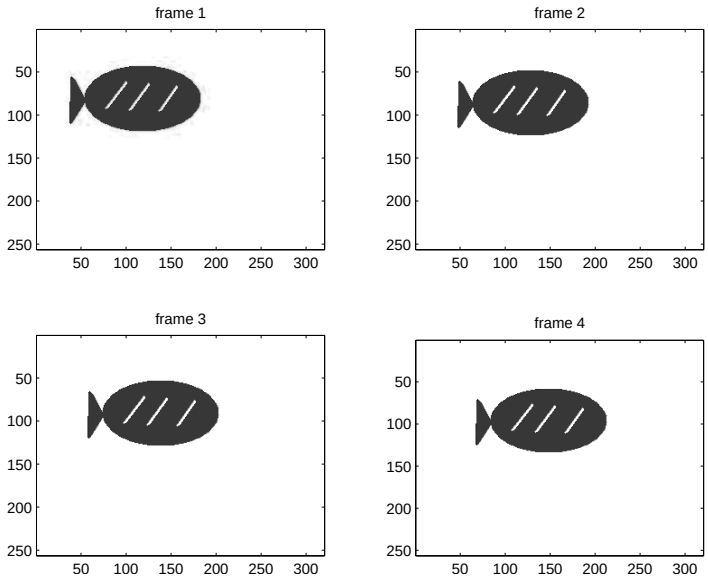
3.1.1 Experiment Setup

The synthetic sequence created consists of twelve successive image frames. For simplicity, the sequence is broken into chunks of four frames. Each chunk will undergo a separate optical flow calculation. By trial and error, the weighting factor alpha is chosen to be 20, and the number of iterations for each frame is 32. Figures 2 through 4 show the frame chunks and the corresponding optical flow calculations.

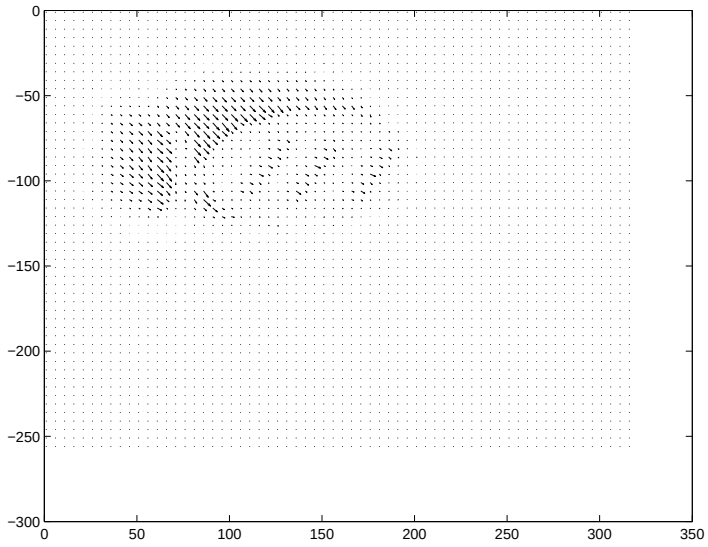
3.1.2 Results

Examining the motion vectors computed by the optical flow algorithm reveals that the vectors are in fact pointing in the true direction of motion of the blimp. The grouping of vectors that exists around the trailing end of the object correspond to the field of motion. However, the "empty" area that exists in front of the vector field constitutes a problem. This empty area is a result of the structure of the original synthetic image. Because the object was created in a paint program, the color of the object is constant throughout the image. Therefore, there is no intensity gradient within the object. This empty area was completely covered by the object over the four frames. For this reason, the small white stripes in the original image were added to produce some gradient content within the object.

Another issue brought up in these simulation is the presence of vectors that exist where no motion was occurring. This, in part, is due to the smoothness constraint that Horn and Schunck used to formulate this optical flow method. For occluding objects and objects moving in different direction within an image, the Horn and Schunck algorithm does a poor job of calculating optical flow in the regions where these situations occur.

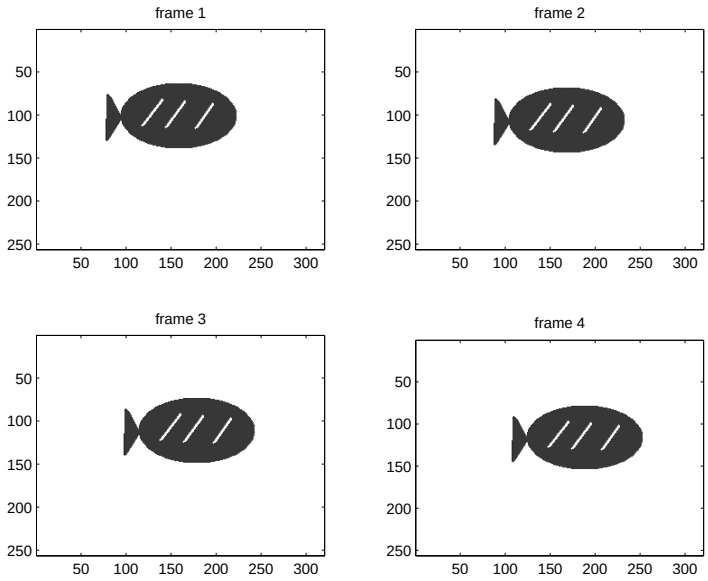


(a) Frames 1 - 4

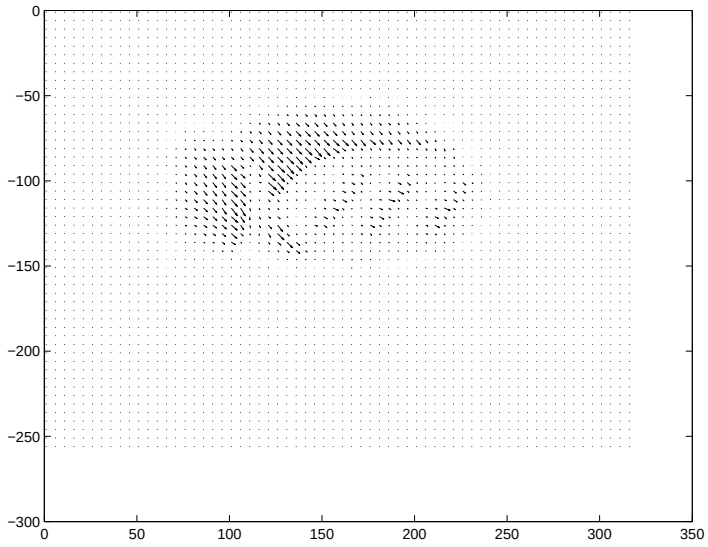


(b) Optical Flow

Figure 2: First four frames of the synthetic blimp image sequence and the computed optical flow

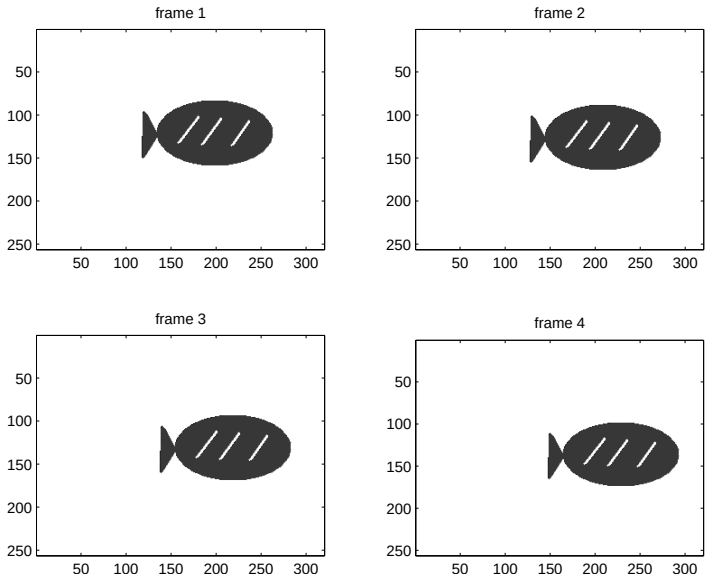


(a) Frames 5 - 8

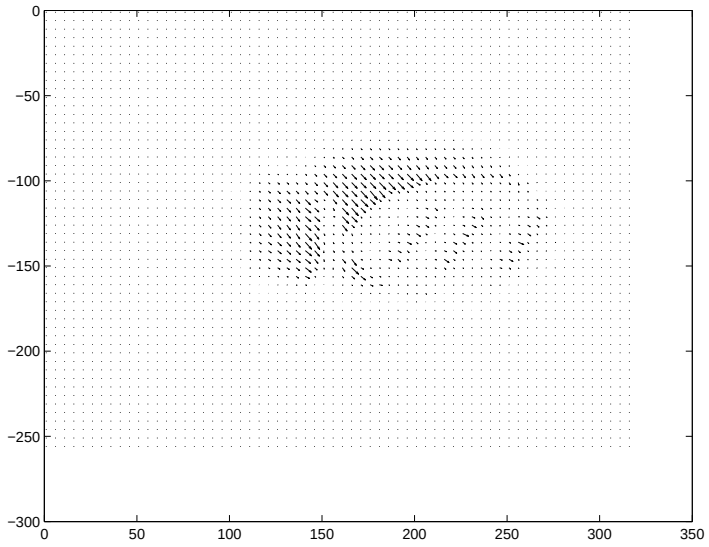


(b) optical Flow

Figure 3: Next four frames of the synthetic blimp image sequence and the computed optical flow



(a) Frames 9 - 12



(b) Optical Flow

Figure 4: Last four frames of the synthetic blimp image sequence and the computed optical flow

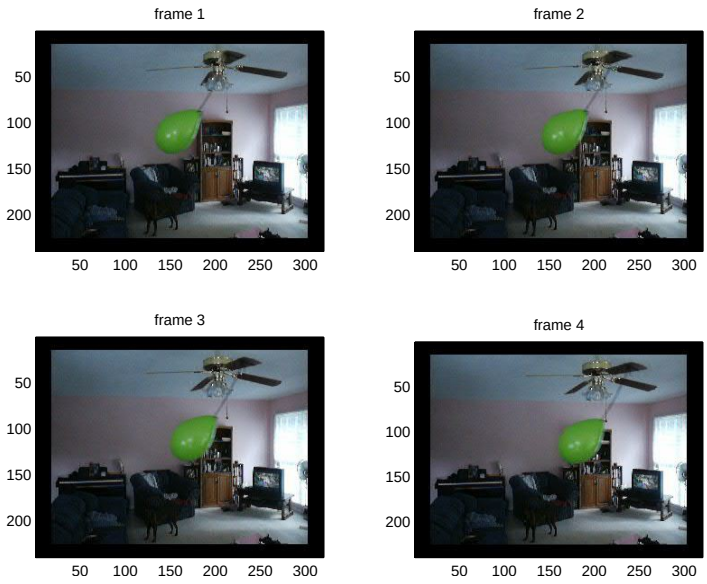


Figure 5: First four images in original balloon sequence

3.2 Real Image Sequence

The next step is testing the Horn and Schunck algorithm on a sequence of real images.

3.2.1 Experiment Setup

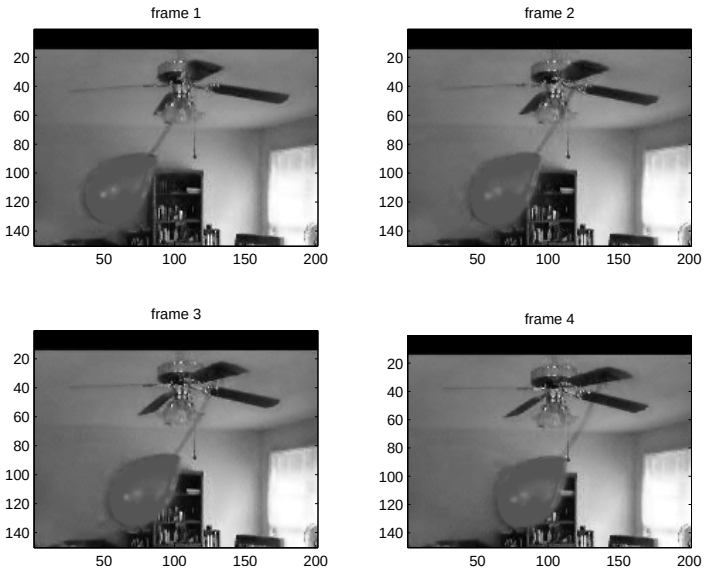
Images for this experiment are of a green balloon attached to a living room fan, which is turned on at its medium setting. Again, twelve sequential images were extracted from approximately 100 that were taken during the experiment. Figure 5 shows the first four of the selected images. Like the synthetic image, the region of motion was extracted from the image sequence, simply for the sake of saving processing time. Again, the sequence was broken into 3 separate chunks, each with four images, all of which are converted to gray-scale. The same settings for alpha (20) and the number of iterations (32) are used in this experiment. Figures 6 through 4 show the frame chunks and the corresponding optical flow calculations.

3.2.2 Results

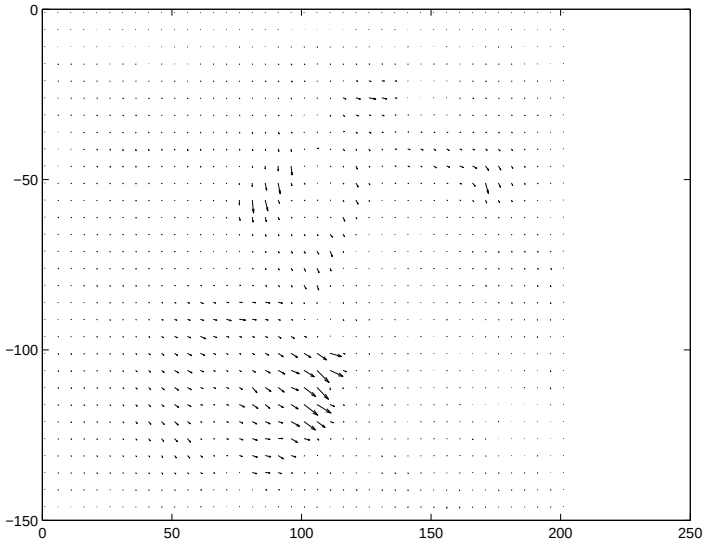
Results generated from the real image sequence produce mixed results. The motion vectors for Frames 1 - 4 appear to be the most accurate. The outline of the balloon is very visible, and its motion vectors are pointing in the right direction. The motion of the fan blades are also visible, and the corresponding motion vectors appear to be accurate. Frames 5 - 8 produce moderate results, as the motion of the balloon and the fan blade are detected, but not as accurately as the first four frames. The optical flow algorithm has difficulty in detecting the motion in the last set of image frames. One possible reason for the difficultly encountered in the last frames is the aperture effect. In the last set of images, the balloon is beginning to move away from the camera. Due to the close proximity of the camera to the balloon, as the balloon moves away, its size relative to the surroundings in the project image plane does not change. Therefore, the gradient in all three dimensions is small, so the motion of the balloon will not be detected.

4 Conclusions

The Horn and Schunck algorithm for computing optical flow was one of the first methods used to compute motion vectors within an image sequence. Many of the newer and more complex optical flow algorithms have

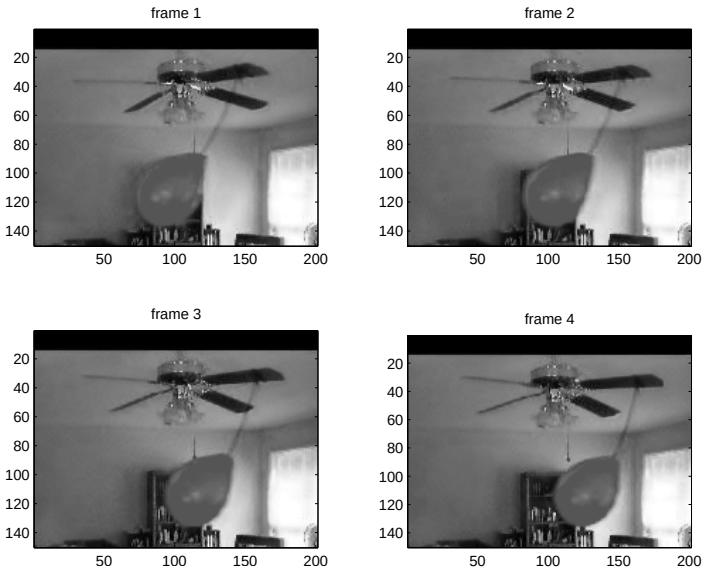


(a) Frames 1 - 4

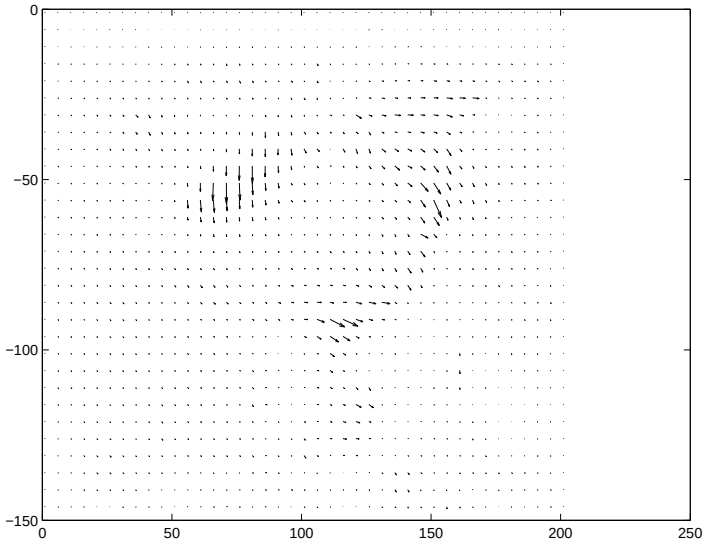


(b) Optical Flow

Figure 6: First four frames of the balloon image sequence and the computed optical flow

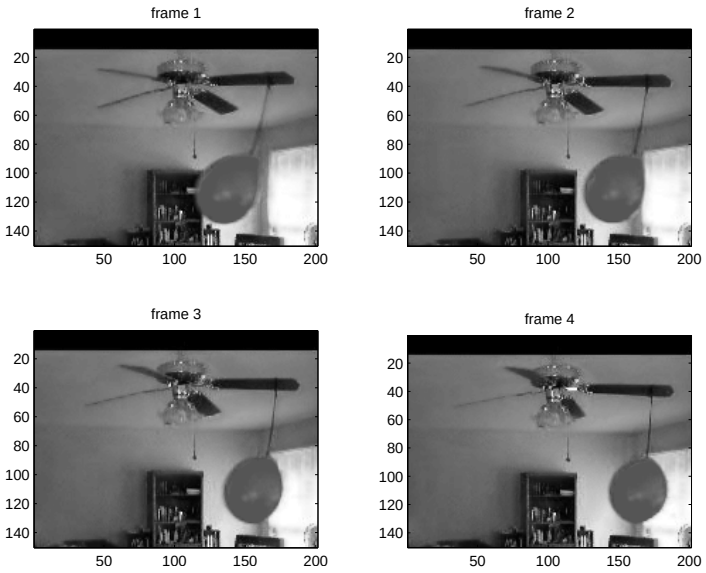


(a) Frames 5 - 8

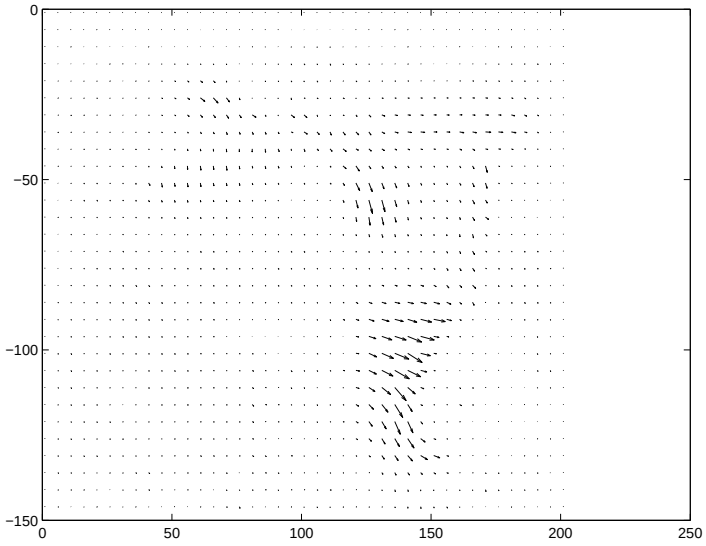


(b) Optical Flow

Figure 7: Next four frames of the balloon image sequence and the computed optical flow



(a) Frames 9 - 12



(b) Optical Flow

Figure 8: Last four frames of the balloon image sequence and the computed optical flow

stemmed from modifications of this method. In order to test the newer methods against this benchmark, the strengths and weaknesses of the method must be analyzed.

4.1 Strengths

The major strength of the Horn and Schunck method is that it works well when the constraints used to derive the method are met. For example, in the synthetic image sequence used in this project, when the intensity of the object within in image varies smoothly, the computed motion vectors are more reliable. Another strength lies in the method's simplicity. The derivative estimates of intensity are intuitive, in the manner that they are obtained by taking a difference between intensity values in the x, y, and time dimensions. Thus, the algorithm is asking what is different about this image when compared to the last one?

4.2 Weaknesses

In contrast, the strengths of the Horn and Schunck method can also be weaknesses. The simple approximation of motion using the estimated derivatives places the restriction on performance (Aperture effect). Also, the smoothness constraint is not a good estimation of objects in motion within an image plane. For most applications, a vision system would encounter multiple moving objects, including the background if the vision system itself is mounted on some type of vehicle. In this situation moving objects would occlude one another, creating a severe problem for the Horn and Schunck algorithm.

4.3 Final Remarks

The Horn and Schunck algorithm for computing optical flow has proven to be the benchmark against which all other optical flow methods are tested. It's ease of implementation make it a good option to learn the concepts of computing optical flow. However, for real time applications, the method's constraints reduce its effectiveness.

References

[1] B.K.P Horn and B.G. Schunk. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

A Project m-files

A.1 hsof.m

```
function [u,v] = hsof(E,num_its,alpha,u,v)

[M, N, sizek]=size(E);
%M = number of rows (indexed by ii)
%N = number of columsn (indexed by jj)
% u = zeros(M,N);
% v = zeros(M,N);

LaplKern = [1/12,1/6,1/12;1/6,-1,1/6;1/12,1/6,1/12];

for kk = 1:sizek-1
    Ex = zeros(M-1,N-1,sizek-1);
    Ey = Ex;
    Et = Ey;
    for ii = 2:M-1
        for jj = 2:N-1
            Ex(ii,jj,kk) = (1/4)*(E(ii+1,jj+1,kk)-E(ii+1,jj,kk)+E(ii,jj+1,kk)-E(ii,jj,kk)...
                +E(ii+1,jj+1,kk+1)-E(ii+1,jj,kk+1)+E(ii,jj+1,kk+1)-E(ii,jj,kk+1));
            Ey(ii,jj,kk) = (1/4)*(E(ii,jj,kk)-E(ii+1,jj,kk)+E(ii,jj+1,kk)-E(ii+1,jj+1,kk)...
                +E(ii,jj,kk+1)-E(ii+1,jj,kk+1)+E(ii,jj+1,kk+1)-E(ii+1,jj+1,kk+1));
            Et(ii,jj,kk) = (1/4)*(E(ii+1,jj,kk+1)-E(ii+1,jj,kk)+E(ii,jj,kk+1)-E(ii,jj,kk)...
                +E(ii+1,jj+1,kk+1)-E(ii+1,jj+1,kk)+E(ii,jj+1,kk+1)-E(ii,jj+1,kk));

        end
    end


    for nn = 1:num_its
        for ii = 2:M-1
            for jj = 2:N-1

                ubar = (1/6)*(u(ii-1,jj)+u(ii,jj+1)+u(ii+1,jj)+u(ii,jj-1))+...
                    (1/12)*(u(ii-1,jj-1)+u(ii-1,jj+1)+u(ii+1,jj+1)+u(ii+1,jj-1));
                vbar = (1/6)*(v(ii-1,jj)+v(ii,jj+1)+v(ii+1,jj)+v(ii,jj-1))+...
                    (1/12)*(v(ii-1,jj-1)+v(ii-1,jj+1)+v(ii+1,jj+1)+v(ii+1,jj-1));

                %-------------------------
                %compute alpha
                lambda = (Ex(ii,jj,kk)*ubar+Ey(ii,jj,kk)*vbar+Et(ii,jj,kk))/...
                    (alpha^2 +(Ex(ii,jj,kk))^2+(Ey(ii,jj,kk))^2);
                %update u(ii,jj) and v(ii,jj)
                u(ii,jj) = ubar-lambda*Ex(ii,jj,kk);
                v(ii,jj) = vbar-lambda*Ey(ii,jj,kk);
            end
        end
    end
```

```
end
```

A.2 make_blimps.m

```
clear
clc
close all

xoff = 5;
yoff = 10;
num = 12;

[input,map] = imread('blimp_original.jpg','jpeg');
imagesc(input)
print -depsc blimp_original
close all
original = rgb2ind(input,gray(256));
dummy = original(1:256,1:320);
[M,N] = size(dummy);
circle = 255*ones(M,N,num);
circle(:,:,1) = dummy;

for kk = 2:num
    dummy = circle(:,:,kk-1);
    [R,C] = find(dummy<200);
    nn = size(R,1);
    for ii = 1:nn
        circle(R(ii)+xoff,C(ii)+yoff,kk) = dummy(R(ii),C(ii));
    end
end

%make jpeg sequence for blimp_of.m
for ii = 1:num
    A = mat2gray(circle(:,:,ii));
    [A,map] = gray2ind(A,256);
    filename = sprintf('dori%d.jpg',ii);
    imwrite(A,map,filename,'jpeg')
end

%make  group figures for report
for ii = 1:4:9
    figure(ii)
    subplot(2,2,1)
    imagesc(circle(:,:,ii))
    title('frame 1')
    subplot(2,2,2)
    imagesc(circle(:,:,ii+1))
    title('frame 2')
    subplot(2,2,3)
```

```
    imagesc(circle(:,:,ii+2))
    title('frame 3')
    subplot(2,2,4)
    imagesc(circle(:,:,ii+3))
    title('frame 4')
    colormap(gray(256))
    truesize
    eval(sprintf('print -depsc blimp%dthru%d',ii,ii+3))
end
```

A.3 extract.m

```
clear
clc
close all

nn = 12; %number of images in the sequence
for ii = 1:nn
    s = sprintf('gb%d.jpg',ii);
    [input,map] = imread(s,'jpeg');
    original(:,:,:,ii) = input;
    dummy = rgb2ind(input,gray(256));
    large(:,:,ii) = dummy;
    dummy2= large(1:150,100:300,ii);
    small(:,:,ii) = dummy2;
    A = mat2gray(dummy2);
    [A,map] = gray2ind(A,256);
    filename = sprintf('gbe%d.jpg',ii);
    imwrite(A,map,filename,'jpeg')
end

figure(2)
subplot(2,2,1)
imagesc(original(:,:,:,1))
title('frame 1')
subplot(2,2,2)
imagesc(original(:,:,:,2))
title('frame 2')
subplot(2,2,3)
imagesc(original(:,:,:,3))
title('frame 3')
subplot(2,2,4)
imagesc(original(:,:,:,4))
title('frame 4')
truesize
print -depsc balloon_sequence

%make  group figures for report
for ii = 1:4:9
```

```
    figure(ii)
    subplot(2,2,1)
    imagesc(small(:,:,ii))
    title('frame 1')
    subplot(2,2,2)
    imagesc(small(:,:,ii+1))
    title('frame 2')
    subplot(2,2,3)
    imagesc(small(:,:,ii+2))
    title('frame 3')
    subplot(2,2,4)
    imagesc(small(:,:,ii+3))
    title('frame 4')
    colormap(gray(256))
    truesize
    eval(sprintf('print -depsc gbe%dthru%d',ii,ii+3))
end
```

A.4 blimp_of.m

```
%Josh Clanton
%ELEC 6430 Semester Project on Optical Flow
%4/4/05
%blimp image sequence

clear
clc
close all

dd = 5;
alpha = 20; %weighting factor
num_its = 16; %number of iterations

img_num = 12; %number of images in sequence
numperof = 4; %number of images per optical flow calculation

%==========blimp sequence====================
filename ='dori%d.jpg';
nn = 0;
fignum = 1;
while nn < img_num
    for ii = 1:numperof
        p = nn+ii;
        s = sprintf(filename,p);
        [input,map] = imread(s,'jpeg');
        dummy = rgb2ind(input,gray(256));
        E(:,:,ii) = double(dummy);
```

```
    end
    [M, N, sizek]=size(E);
    u = zeros(M,N);
    v = zeros(M,N);
    firstrun = 0;
    [uu,vv]=hsof(E,num_its,alpha,u,v);
    figure(fignum)
    [xx,yy] = meshgrid(1:dd:N,1:dd:M); %creates grid
    a = uu(1:dd:M,1:dd:N);
    b = vv(1:dd:M,1:dd:N);
    axis ij
    quiver(xx,-yy,-b,a,'k')
    nn = nn + numperof;
    eval(sprintf('print -depsc blimp_of%d',fignum))
    fignum = fignum+1;
end
```

A.5 balloon_of.m

```
clear
clc
close all

nn = 12; %number of images in the sequence
for ii = 1:nn
    s = sprintf('gb%d.jpg',ii);
    [input,map] = imread(s,'jpeg');
    original(:,:,:,ii) = input;
    dummy = rgb2ind(input,gray(256));
    large(:,:,ii) = dummy;
    dummy2= large(1:150,100:300,ii);
    small(:,:,ii) = dummy2;
    A = mat2gray(dummy2);
    [A,map] = gray2ind(A,256);
    filename = sprintf('gbe%d.jpg',ii);
    imwrite(A,map,filename,'jpeg')
end

figure(2)
subplot(2,2,1)
imagesc(original(:,:,:,1))
title('frame 1')
subplot(2,2,2)
imagesc(original(:,:,:,2))
title('frame 2')
subplot(2,2,3)
imagesc(original(:,:,:,3))
```

```
title('frame 3')
subplot(2,2,4)
imagesc(original(:,:,:,4))
title('frame 4')
truesize
print -depsc balloon_sequence

%make  group figures for report
for ii = 1:4:9
    figure(ii)
    subplot(2,2,1)
    imagesc(small(:,:,ii))
    title('frame 1')
    subplot(2,2,2)
    imagesc(small(:,:,ii+1))
    title('frame 2')
    subplot(2,2,3)
    imagesc(small(:,:,ii+2))
    title('frame 3')
    subplot(2,2,4)
    imagesc(small(:,:,ii+3))
    title('frame 4')
    colormap(gray(256))
    truesize
    eval(sprintf('print -depsc gbe%dthru%d',ii,ii+3))
end
```