ECSE 6550 COMPUTER VISION PROJECT 4

Andrew Cunningham

May 2, 2016

1 Introduction

Given a sequence of images, it is possible to obtain motion information of the objects imaged. There are two ways to view the relationship between motion and image sequences: top down and bottom up. In the top down approach, we are interested in developing the models that take an object in motion and camera parameters to create a sequence of images. In the bottom up approach, we are interested in obtaining 3D structure, motion or camera parameters from a sequence of images with varying amounts of information. In this project, we explore a subset of the bottom up approach by calculating optical flow from image sequences to estimate 3D motion.

2 Technical Approach

The process of obtaining optical flow from a sequence of images includes taking image derivatives and applying OF estimation techniques. In this project, we use an analytical approach to computing image derivatives by fitting a cubic intensity function to a image sequence window. The Lucas-Kanade method is then used to estimate optical flow.

2.1 Computing Image Derivatives

Numerical methods are typically used in calculating image derivatives where

$$I_x(x,y,t) = I(x+1,y,t) - I(x,y,t) \tag{1}$$
$$I_y(x,y,t) = I(x,y+1,t) - I(x,y,t) \tag{2}$$
$$I_t(x,y,t) = I(x,y,t+1) - I(x,y,t) \tag{3}$$

I_X is the image derivative with respect to the x coordinates. Equations (1-3) are not necessarily smooth and are very sensitive to image noise. Analytically fitting an intensity function over the image can help to make derivatives smooth and make the estimate more robust against noise. In this project we fit a cubic function to a window of images. The cubic model is given by:

$$I(x,y,t) = a_1 + a_2x + a_3y + a_4t + a_5x^2 + a_6xy + ... + a_{19}xt^2 + a_{20}xyt \tag{4}$$

We fit this model to a $j \times j \times k$ image window where j is the length of the spatial image box and k is the length of the temporal image box. In this particular project, k is fixed to be 5 due to the fact that we are given 5 images per sequence. We can form a linear set of equations to fit the model with Da=J where a is the set of coefficients in Eq(4) To fit the model, we form D and J:

$$D = \begin{pmatrix} 1 & x_1 & y_1 & t_1 & \dots & x_1y_1t_1 \\ 1 & x_2 & y_1 & t_1 & \dots & x_2y_1t_1 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & x_k & y_1 & t_1 & \dots & x_ky_1t_1 \\ 1 & x_1 & y_2 & t_1 & \dots & x_1y_2t_1 \\ 1 & x_2 & y_2 & t_1 & \dots & x_2y_2t_1 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & x_k & y_2 & t_1 & \dots & x_ky_2t_1 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & x_1 & y_k & t_k & \dots & x_1y_kt_k \\ 1 & x_2 & y_k & t_k & \dots & x_2y_kt_k \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & x_k & y_k & t_k & \dots & x_ky_kt_k \end{pmatrix} \qquad J = \begin{pmatrix} I(x_1,y_1,t_1) \\ I(x_2,y_1,t_1) \\ \vdots \\ I(x_k,y_k,t_k) \end{pmatrix}$$

Figure 1: Taken from Qiang Ji's lecture notes on motion

We can then solve for the coefficients with

$$a = (D^tD)^1D^tJ \tag{5}$$

2.2 Lucas Kanade Optical Flow Estimation

The problem of estimating optical flow can be approached by viewing the problem from a top down perspective. To begin, optical flow is formally (V_x, V_y) the direction of displacement of a pixel from one image to the next. If we assume that the image intensity (brightness) is constant over time, then we can generate an equation that restricts the behavior of optical flow:

$$\frac{dI}{dt} = \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \tag{6}$$

Eq(6) does not provide enough information to solve for optical flow. The Lucas Kanade method adds in the assumption that image content displacement between frames is approximately constant within a neighborhood. Thus, a system of equations can be formed:

$$I_x(x_1, y_1)Vx + I_y(x_1, y_1)Vy = -I_t(x_1, y_1) \tag{7}$$

$$I_x(x_2, y_2)Vx + I_y(x_2, y_2)Vy = -I_t(x_2, y_2) \tag{8}$$

$$\vdots \tag{9}$$

$$I_x(x_N, y_N)Vx + I_y(x_N, y_N)Vy = -I_t(x_N, y_N) \tag{10}$$

For the N pixels that are in the neighborhood of the original pixel. The addition of this assumption resulted in a system of equations with more equations than unknowns so a linear least squares solution can be generated with:

$$\mathrm{v} = (A^T A)^{-1} A^T b \tag{11}$$

Where

$$A = \begin{bmatrix} I_x(x_1, y_1) & I_y(x_1, y_1) \\ I_x(x_2, y_2) & I_y(x_2, y_2) \\ \vdots & \\ I_x(x_N, y_N) & I_y(x_N, y_N) \end{bmatrix}, \quad b = \begin{bmatrix} -I_t(x_1, y_1) \\ -I_t(x_2, y_2) \\ \vdots \\ -I_t(x_N, y_N) \end{bmatrix} \tag{12}$$

3 Code

The code for this project is organized in functions. The main.m file calls the functions to find the optical flow for each sequence of images

3.0.1 Code

Listing 1: main.m

```
% Read spheres
sphere2 = imread('sphere/sphere2.pgm');
sphere3 = imread('sphere/sphere3.pgm');
sphere4 = imread('sphere/sphere4.pgm');
sphere5 = imread('sphere/sphere5.pgm');
sphere6 = imread('sphere/sphere6.pgm');
spheres = cat(3,sphere2,sphere3,sphere4,sphere5,sphere6);

% Read grid
grid1 = imread('grid/center1.jpg');
grid2 = imread('grid/center2.jpg');
grid3 = imread('grid/center3.jpg');
grid4 = imread('grid/center4.jpg');
grid5 = imread('grid/center5.jpg');
grids = cat(3,grid1,grid2,grid3,grid4,grid5);

% Read people
people1 = imread('people/1.pgm');
```

```
people2 = imread('people/2.pgm');
people3 = imread('people/3.pgm');
people4 = imread('people/4.pgm');
people5 = imread('people/5.pgm');
peoples = cat(3,people1,people2,people3,people4,people5);


%% OF FOR SPHERES
boxSize = 2;
figure; hold on;
for c=15:5:185
    for r=15:5:185
        centralPixel = [c;r];
        A = fitCubicRegion(spheres,centralPixel,boxSize);
        v=lucasKMethod(A,2);
        quiver(r,c,v(2),v(1),'b','MaxHeadSize',8)
    end
end

%% OF FOR GRID
boxSize = 16;
figure; hold on;
for c=335:20:535
    for r=275:20:475
        centralPixel = [c;r];
        A = fitCubicRegion(grids,centralPixel,boxSize);
        v=lucasKMethod(A,12);
        quiver(r,c,v(2),v(1),'b','MaxHeadSize',8)
    end
end

%% OF FOR PEOPLE
boxSize = 2;
figure; hold on;
for c=10:2:230
    for r=10:2:310
        centralPixel = [c;r];
        A = fitCubicRegion(peoples,centralPixel,boxSize);
        v=lucasKMethod(A,1);
        if norm(v) > 0
            quiver(r,c,v(2),v(1),'b','MaxHeadSize',8)
        end
    end
end
```

Listing 2: fitCubicRegion.m

```
function [ a ] = fitCubicRegion( images, centralPixel, boxLength)
    D = [];
    J = [];
    % For each image
    for t1=1:5 % time index
        for xOff=-boxLength:boxLength % columns
            for yOff=-boxLength:boxLength % rows
                d= [1 xOff yOff t1 xOff^2 xOff*yOff yOff^2 ...
                yOff*t1 t1^2 xOff*t1 xOff^3 xOff^2*yOff ...
```

```
                    xOff*yOff^2 yOff^3 yOff^2*t1 ...
                    yOff*t1^2 t1^3 xOff^2*t1 xOff*t1^2 xOff*yOff*t1];
                    D = [D;d]; x1 = xOff + centralPixel(1); y1 = yOff +...
                    centralPixel(2);
                    J = [J;images(x1,y1,t1)];
                end
            end
        end
        a = pinv(D)*double(J);

        % Check the quality of fit
        if norm(mean(D*a - double(J))) > .001
            norm(mean(D*a - double(J)))
        end
    end
```

Listing 3: lucasKMethod.m

```
function [v] = lucasKMethod( A, boxLength )
I = []; b = [];
    for c=-boxLength:boxLength
        for r=-boxLength:boxLength
            Ix = dIdx(r,c,3,A); %Estimate optical flow of central frame
            Iy = dIdy(r,c,3,A); %Estimate optical flow of central frame
            It = dIdt(r,c,3,A); %Estimate optical flow of central frame

            I = [I; Ix Iy]; b = [b;-It];
        end
    end

    %If there is a very small change, treat it as if there were no change
    if norm(b) < .0001
        b = zeros(size(b));
    end

    v=pinv(I)*b;

    % Filter badly conditioned matrices or large v's
    if cond(I) > 200 || cond(A) > 200 || norm(v) > 15 || norm(v) < .5
        v = [0;0];
    end


end
```

4 Results and Analysis

4.1 Sphere

The sphere sequence appears to be rotating to the right in 3D. The optical flow vectors confirm this because they point rightward. The gaps in the sphere correspond to the lighter regions. The jump in colors resulted in large optical flow vectors that were filtered out. The change in magnitude of the optical flow vectors around the sphere can be explained by the fact there is image movement around the axis of rotation that the sphere is rotating about.

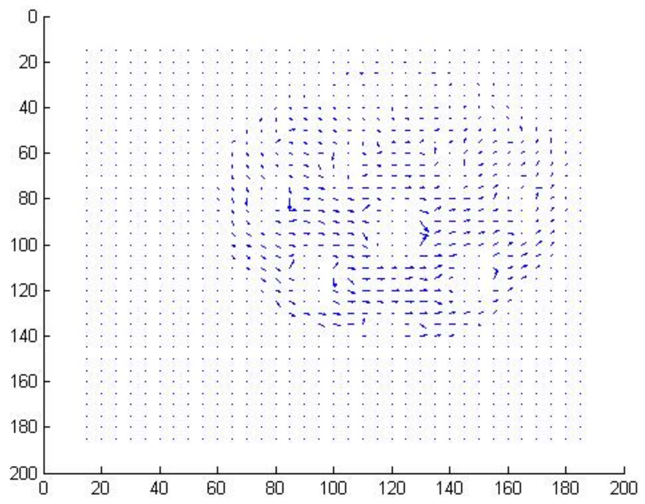

Figure 2: Sphere results

4.2 Grid

The grid optical flow vectors took a little work to get a result that made sense. The jump between black and white squares originally produced many optical flow vectors that pointed in different directions depending on where they originated. After extending the cubic model and Lucas-Kanade windows, an averaging effect was achieved. The remaining vectors point rightward indicating that the object was translating to the right in 3D.

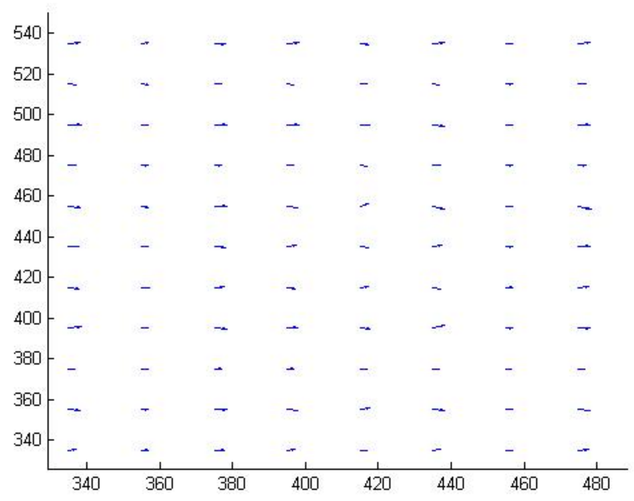

Figure 3: Grid results

4.3 People

The people example required more optical flow vectors to make sense of what was going on. Once enough vectors were added, it became possible to distinguish the shapes of the people in the original sequence of images. The resulting optical flow vectors show the person on the left moving leftward (for the most part) and the person on the right moving to the right and away from the camera (as indicated by the optical flow vectors pointing toward each other)

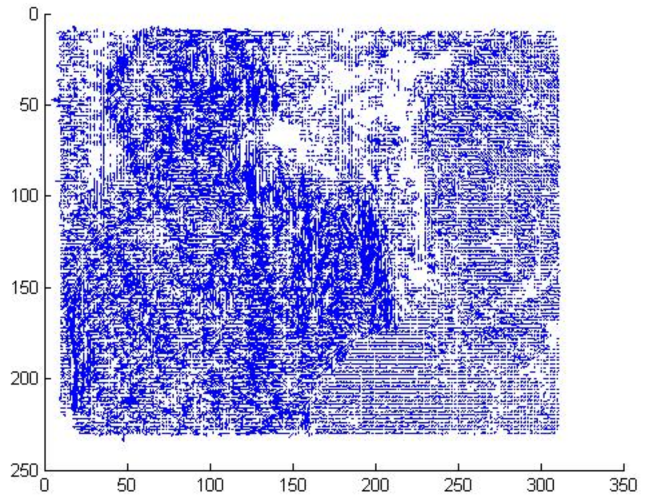

Figure 4: People results

5 Conclusion

This project focused extracting optical flow from a sequence of images. A cubic model was used to describe the intensity of images. The cubic model allowed for clean derivatives. These clean derivatives were then fed into the Lucas-Kanade method to obtain an estimate of optical flow. The parameters of window size for both the cubic model and the Lucas-Kanade estimator were varied on a set of test images to produce a set of optical flow vectors. In the end, it was found that larger windows were needed in cases where images had jumps in intensity to smooth out the transition.