Optical Flow Estimation

Royston Rodrigues - 16307R004
Arka Sadhu - 140070011

April 3, 2017

Contents

Problem Statement

The objective of the lab project is to estimate the Optical Flow between a pair of two images where the motion is very small. The motion can be of the camera itself in which case it is called ego motion, or the motion can be of the scene or a combination of the two. We take into account both cases. Once optical flow is estimated, the user defined parameter λ is varied in orders of ten to understand the importance of setting a good value of λ. The optical flow uses a squared function which usually weighs the larger derivatives very highly, and it is beneficial to avoid them for robustness. So we re-derive the Euler-Lagrange equations for a more robust cost function and then recompute the Optical Flow. Lastly, it is of interest to get the motion parameters of the camera using the Optical flow data which is basically the translation and rotation of the camera. Here only ego motion is considered. It is difficult to do it in the case of both translation and rotation, and hence we adopt an iterative scheme of considering only one of the two translation and rotation to find the camera parameters, and use the new values to get a better approximation.

Optical Flow

Basic Idea

It is of high interest to know how the individual rigid objects are moving in the real world from a sequence of images. Of course this can be done by feature matching methods, but these tend to involve a lot of computation and quite often we cannot be very sure about the results obtained from feature matching. A much simpler and elegant method to understand the scene is Optical Flow. It computes the image velocity of each point and gives an estimate of how every object in the image is moving in the real world, by plotting a needle diagram at every point. It implicitly assumes that the objects do not move a lot from initial positions and assumes the image as a continuous function and derives a constraint equation.

Constraint Equation

We assume that the sequence of images is a continuous function $f(x, y, t)$, where (x, y) denote the spatial coordinates, and t is the time coordinate. We assume that the pixels wouldn't move a lot from their initial positions. As a result we can say,

$$f(x, y, t) = f(x + \delta x, y + \delta y, t + \delta t)$$

Now we use Taylor series approximation

$$f(x + \delta x, y + \delta y, t + \delta t) = f(x, y, t) + \delta x \frac{\partial f}{\partial x} + \delta y \frac{\partial f}{\partial y} + \delta t \frac{\partial f}{\partial t} = f(x, y, t)$$

Denoting $\frac{\partial f}{\partial t}$ as f_t. We get

$$f_x \delta x + f_y \delta y + f_t \delta t = 0$$

Dividing by δt we get

$$f_x u + f_y v = -f_t \tag{1}$$

where $u = \frac{\delta x}{\delta t}$ and $v = \frac{\delta y}{\delta t}$ and determines the velocity of the scene in the image plane.

Equation 1 can also be interpreted as

$$(u, v) \cdot (f_x, f_y) = -f_t$$

Therefore we only have information about (u, v) in the direction of (f_x, f_y) and no information about its perpendicular direction. This is known as the **aperture problem**.

We get around this by assuming that the image velocity (u, v) is smooth. So our cost function becomes

$$\varepsilon = \int\int_{x,y} (I_x u + I_y v + I_t)^2 + \lambda(u_x^2 + u_y^2 + v_x^2 + v_z^2)\, dxdy \tag{2}$$

We have used I in place of f to denote that we are working with images To minimize Equation 2 we directly use Calculus of Variation, and obtain the corresponding Euler-Lagrange equation.

$$u_{x,y}^{k+1} = \bar{u}_{x,y}^{k} - \frac{I_x[I_x\bar{u}_{x,y}^{k} + I_y\bar{v}_{x,y}^{k} + I_t]}{\lambda^2 + I_x^2 + I_y^2} \tag{3}$$

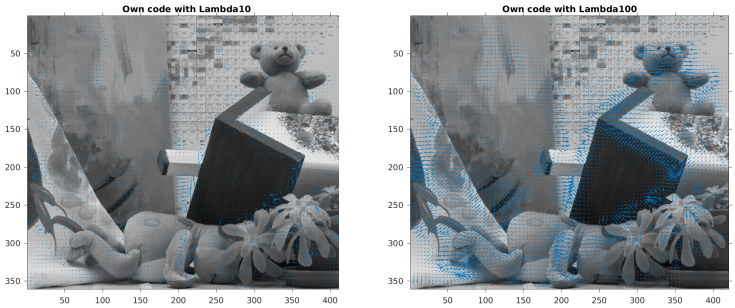
$$v_{x,y}^{k+1} = \bar{v}_{x,y}^{k} - \frac{I_y[I_x\bar{u}_{x,y}^{k} + I_y\bar{v}_{x,y}^{k} + I_t]}{\lambda^2 + I_x^2 + I_y^2} \tag{4}$$

where $\bar{u}_{x,y}^{k}$ denotes the average value of u (averaged over 4 points up,down,left,right) at the point (x, y) and at the k^{th} iteration.

We choose the boundary conditions as zero everywhere.

Varying the Parameter λ

The parameter λ is a regularizing parameter. It basically weighs the effect of the smoothness function and the constraint equation. Optical Flow with varying lambda have been saved in the images folder.



Robust Smoothness Function

It is noted that the smoothness cost function

$$(u_x^2 + u_y^2 + v_x^2 + v_y^2)$$

makes the image velocity too smooth and as a result gives bad performance in the presence of outliers. Hence we use a more robust function. We choose the function

$$\Psi(x) = \Psi_{max}\frac{x^2}{100 + x^2}$$

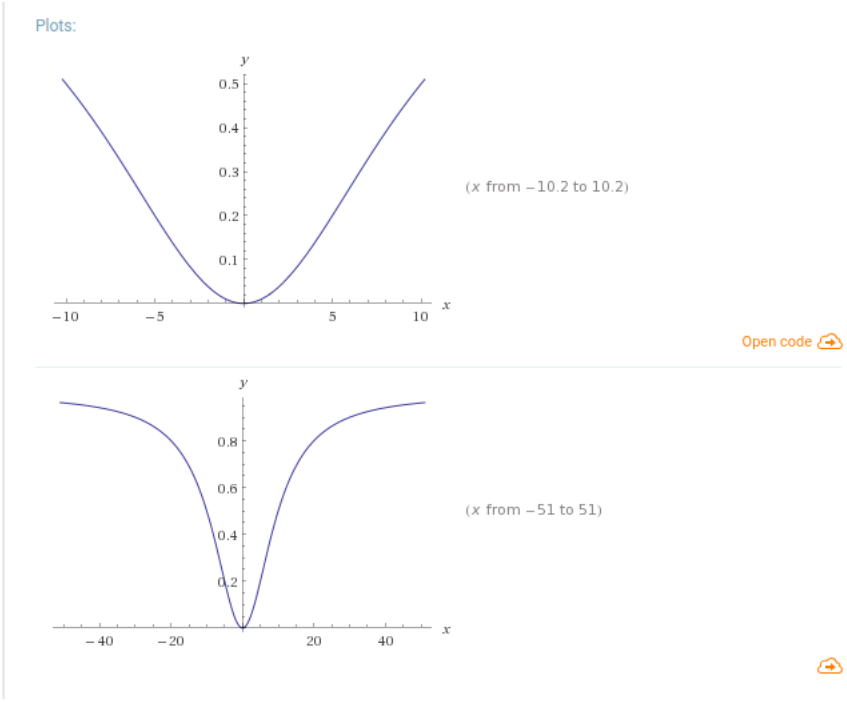


Figure 2: Plot of $\Psi(x)$

Now we derive Euler Lagrange Equations for the same using Calculus of Variation. We get

$$\Psi^{''}(x) = -200\frac{3x^2 - 100}{(x^2 + 100)^3}$$

The update equations we get are as follows

$$u_{i,j}^{k+1} = \frac{\bar{u_x}_{i,j}^{k}\alpha_{u_{i,j}} + \bar{u_y}_{i,j}^{k}\beta_{u_{i,j}} - \Lambda I_x(I_x u_{i,j} + I_y v_{i,j} + I_t)}{\alpha_{u_{i,j}} + \beta_{u_{i,j}}} \tag{5}$$

$$v_{i,j}^{k+1} = \frac{\bar{v_x}_{i,j}^{k}\alpha_{v_{i,j}} + \bar{v_y}_{i,j}^{k}\beta_{v_{i,j}} - \Lambda I_y(I_x u_{i,j} + I_y v_{i,j} + I_t)}{\alpha_{v_{i,j}} + \beta_{v_{i,j}}} \tag{6}$$

Here $\bar{u_x}_{i,j}^{k}$ denotes the average u along the x-direction at point (i,j) at the k^{th} iteration. $\alpha_{u_{i,j}}$ corresponds to the (i,j) element of $\Psi^{''}(u_x)$ and $\beta_{u_{i,j}}$ corresponds to the (i,j) element of $\Psi^{''}(u_y)$. Similarly for $\alpha_{v_{i,j}}$ and $\beta_{v_{i,j}}$.

We use these update equations to compute (u, v) which is the optical flow.

Ego motion from Optical Flow

Suppose it is known that the scene is fixed and there occurs only ego motion (motion of the camera). It turns out that it is possible to recover the motion parameters of the camera. Assume that the camera motion parameters are (t, ω), then we can say the scene moves with a velocity $V = -t - \omega \times r$.

It turns out that it is easier to compute the motion parameters if it is pure translation and pure rotation.

Pure Translation

For the case of pure translation, it turns out that any factor of t will result in the same optical flow, and hence we restrict t to be unit norm. To get t for the case of pure translation we construct

a G matrix (details given in Chapter 17 of Horn's book), and the cost function minimizes when t is the eigen vector corresponding to the smallest eigen value. Pathological cases have not been considered here.

Pure Rotation

For the case of pure rotation, it turns out that we can find a matrix M and a vector n such that $M\omega = n$, where both M and n can be computed from the optical flow data. This lets us find $\omega = M^{-1}n$.

Structure Recovery

In the case of pure translation, when implementing the condition for minimum error we directly get the formula for depth.

$$Z = \frac{\alpha^2 + \beta^2}{u\alpha + v\beta}$$

Results

Original Pair of Images

Figure 3: Images used for Optical Flow

Matlab Optical Flow

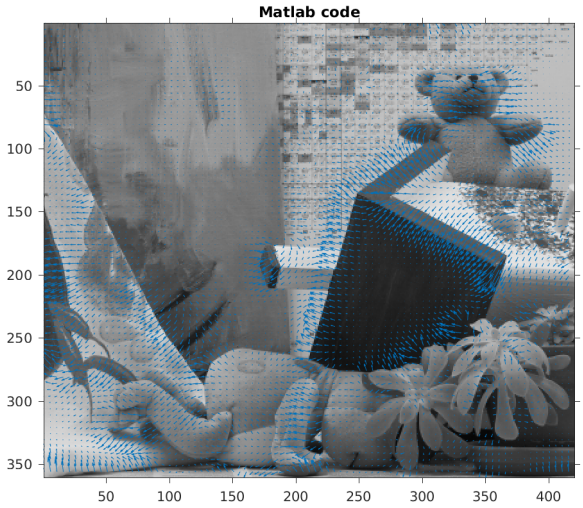


Figure 4: Matlab direct implementation of Optical Flow

Normal Optical Flow

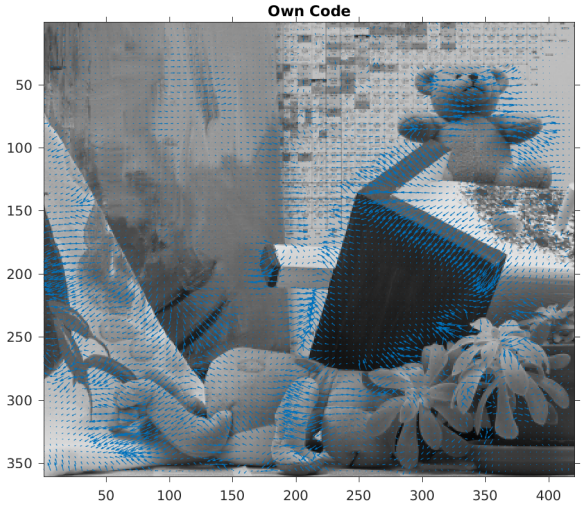


Figure 5: Optical Flow using HS method for $\lambda = 100$

Robust Optical Flow

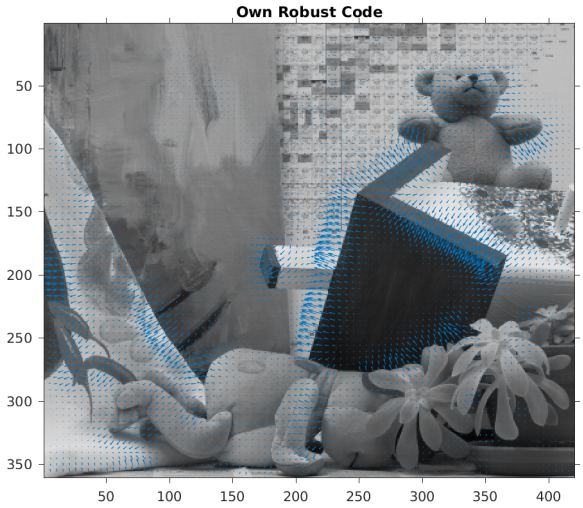


Figure 6: Robust Optical Flow for $\frac{\lambda}{\Psi_{max}} = 100$

Structure from Optical Flow

Figure 7: Aloe Figure

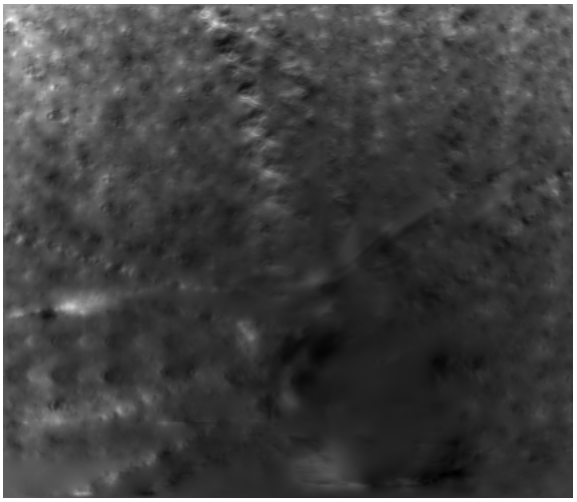


Figure 8: Structure Recovery from Pure Translation