Enhancing Robot Perception using Optical Flow

Student : Pratik Sajnani
Rochester Institute of Technology
ps8369@rit.edu

Advisor : Prof. Zack Butler
Rochester Institute of Technology
zjb@cs.rit.edu

ABSTRACT

Robot perception is a key part of the decision making process of agents that work inside a mechanical system to determine how a robot responds to stimulus from a real-time dynamic environment. This paper discusses the plausibility of sparse and dense optical flow estimation techniques. We employ lucas-kanade algorithm on high-resolution images to establish a common baseline for comparison. The paper also discusses dense optical flow using Farneback's algorithm and the optimization strategy therein.

1. OVERVIEW

Robot perception is a widely researched topic in computer vision and robot navigation. This paper is a survey into two popular ways of using optical flow estimation to perceive changes in the linear subspace generated by image frames. The paper consists of 5 sections. The first section discusses the motivation of the problem. It emphasizes the focus point of the paper and why the problem is worth delving into. The second section is a general introduction to optical flow which includes an illustrative section of the basic idea, the known constraints to achieve a continuous linear vector subspace and the Horn-Schunck method. The third section introduces an optimization called the Lucas-Kanade method which works particularly well for motion estimation. It also brings to fore the problem with using Lucas-Kanade algorithm directly on medium-low resolution images by relating it to the problem of finding the right levels. The fifth section discusses problems with implementation of lucas-kanade method in code and the importance of individual parameters.

2. INTRODUCTION

Robot perception is the way a robot understands its surroundings. It is a key factor responsible for decision making. A mobile robot has the ability to move around and to effectively utilize the ability to get from one point to another it needs to understand its own motion as well as predict, over sufficiently large intervals of time, the motion of objects it may encounter on its path. A robot generally has a good idea of where its headed and hence, it is possible to get a reliable (with minimal error) estimate of the possible trajectory a robot might take if it were unobstructed. In reality, a robot that would be helpful to human beings would need to interact with a possibly obstructive and populated path filled with stationary and moving objects as well as humans. For the safety of the users around the robot and its own, finding the safest path to take and the best velocity to travel at is an important and inseparable problem. A part of the problem of understanding the motion of things around the object is the prediction and tracking of multiple moving objects as observed from a sequence of images off a video feed from the robotâĂŹs camera. The problems of tracking multiple moving objects and that of detecting stationary objects have been solved to a sufficient degree in real-time. This study focuses on utilizing that knowledge and extending it to form an integrated real-time system that makes it viable for the robot to make decisions in complex environments with a sufficient degree of accuracy.

3. TRACKING OBJECTS

Object tracking is defined as the process of locating objects as they move in an environment. In computer vision, the source of information about the environment is image matrices with intensity and color values. The problem of object tracking in a visual system corresponds to identifying the subsequent locations of the same point across a temporal sequence of images. The real world is rife with problems that make it very hard to develop a good general object tracking algorithm. Most algorithms that do good object tracking work for specific contexts and with constraints that are frequently violated in the real world. For example, the technique called optical flow discussed in the subsequent section gives us a general equation but it will fail to differentiate a rotating ball from a stationary ball with lighting changes that resemble rotational motion. A motion tracking algorithm that relies solely on local observations is looking through a very small aperture and because of this limitation, different kinds of motion beyond the aperture appear to be strikingly similar within it. We discuss this limitation in more detail in the section titled aperture problem.

3.1 Optical Flow

Optical flow is a technique employed in computer vision to establish and track objects across temporal frames generated from a video stream. The basic idea is to track instantaneous velocities of all or a statistically important subset of pixels(corner points) across two subsequent images. Hence, it relies on the motion apparent from visual differences across two subsequent images obtained from a camera system. The technique results in the generation of a linear subspace of vectors originating from positions of point in first image and directed towards their estimated motion in the second. The advantage of having a linear subspace is that we can subtract constant motion from it. This is particularly useful in tracking objects from a moving camera where we have some estimate of ego-motion from mechanical systems. Although, this motion might not be precise; it can be corrected by relying on stationary points in the environment. A robot moving down an alley can rely on the stability of ground-plane and surrounding structure to estimate ego-motion to a sufficient degree of accuracy and correct for errors from mechanical systems.

3.2 Derivation of Flow Equation

Let I(first image) and J(second image) be two gray scale intensity images where x' = [x, y] represents the intensity of pixel initially located at (x, y) coordinate. Let 't' be the temporal difference value between the two images, then :

J(x') = I(x', d', t)

where d' = (Δx, Δy) is the displacement vector. We are looking for an estimate of d' that minimizes error in terms of similarity of the intensity values of pixels I(x') and J(x' + d'). Therefore, the error in displacement is defined in terms of the following equation :

error(d') = J(x' + d') - I(x')

Now because d' is assumed to be relatively small, we can use a Taylor series approximation at that point because we know that d' tends to 0. Using the approximation we get the following equation

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0$$

Figure 1: Taylor approximation

The equation has two variables and hence cannot be solved as it is an under determined system. This is a consequence of the aperture problem explained in the next section

3.3 Aperture Problem

When estimating the motion of objects in a visual system, the size aperture of the viewing system determines how one perceives homogeneous motion As stated in [1], this is due to the finite receptive field of motion sensors. This problem was first discussed by P. Stumpf in [5]. The sense of motion obtained from within the aperture is isolated from the larger picture having more detail.The aperture problem can be best understood from the following images:

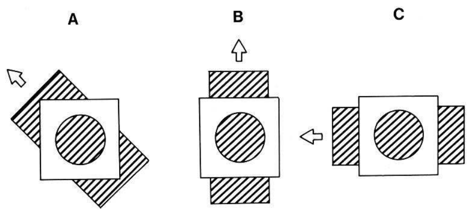


Figure 2: The aperture problem in computer vision

The arrows in figure 2 indicate actual motion. If we rely only on the image data from the aperture, all three directions of actual motion result in the same apparent motion. The popular barber's pole illusion an example of how aperture problems are fairly common and generate interesting patterns even with the context-aware human visual system.

Figure 3: A barber's pole

To overcome the aperture problem, we need to use the notion of a neighborhood window to scale the size of our aperture according to the context of the image. This idea forms the basis for the following error estimate equation as described in [6].

$$\epsilon(\mathbf{d}) = \epsilon(d_x, d_y) = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} (I(x,y) - J(x+d_x, y+d_y))^2$$

Figure 4: Error equation using the notion of a neighborhood

where u_x and u_y are x and y dimension of our neighborhood window.

3.4 Constraints

1. **Brightness Constancy** : Most optical flow estimation techniques assume constancy of brightness plane around a neighborhood window or at every point in a pattern.

2. **Spatial Smoothness** : The small neighborhood works because we assume that the nearby pixels of an image move in a similar manner.

3. **Temporal Smoothness** : The assumption that images were taken one after the other is a fairly applicable assumption. It allows for us to work with smaller neighborhoods.

3.5 Horn-Schunck Method

The idea of smoothness was first proposed by Horn-Schunck. The method assumes a smooth flow field so that a regularization constant can be used to adjust smoothness of the flow field. The double integral over 2-dimensions expresses an energy function from a vector space to scalars thus obtained is expressed by the following equation[7] :

$$E = \iint \left[(I_x u + I_y v + I_t)^2 + \alpha^2 (\|\nabla u\|^2 + \|\nabla v\|^2) \right] \mathrm{d}x\mathrm{d}y$$

Figure 5: Horn-Schunck equation

The equation assumes constant brightness to obtain the same under-determined equation we obtained in section 3.2 using Taylor approximation. The paper also assumes rigid motion which is formally known as spatial coherence. This results in an over-determined due to minimization of the square magnitude of the optical flow velocity.

4. LUCAS-KANADE ALGORITHM

Lucas-Kanade algorithm is a well defined computational process to efficiently derive sparse optical flow. It uses a fixed size neighborhood and assume that the flow remains constant within the domain of this neighborhood. Therefore, In addition to the assumptions made by Horn-Schunck method, it assumes neighborhood flow constancy. The prime reason why lucas-kanade algorithm is widely used is because instead of tracking all points in the image, it uses a feature detection and tracking approach. As stated in [8], the two main components of a feature tracker are accuracy and robustness. The algorithms considers both these criteria.

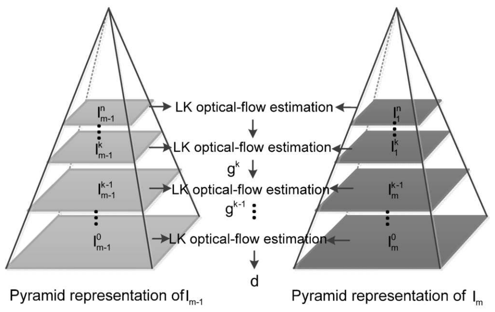
4.1 Handling trade-off : Accuracy vs Robustness

The accuracy perspective of lucas-kanade algorithm is suggests that a small windows should be used. This ensures that the finer image details are not missed while tracking points. It increases precision of the algorithm but on the other, even a small increase in the neighborhood increases computations many times over. This result in a slower process overall and it becomes an infeasible approach for domains like robot navigation. Another problem with choosing a large integration window is the possibility of occlusion.

A small window is preferred in any case, even if we are to miss out some details, we would rather use an algorithm that gives us good results. For a robot perception application it is important to get feedback as close to real-time as possible. The following section explains how lucas-kanade handles this trade-off and achieves both these criteria with significant merit.

4.2 Pyramidal Scheme

Instead of compromising on accuracy, the lucas-kanade algorithm suggests another approach which involves using down-sampled images. The down-sampled images are used to make feature detection relatively fast. For example, a 4-level down-sampling algorithm reduces a 640x480 image to 80x60 image as input for initial estimate of feature tracker. This is relatively fast and the loss of detail is not significant. Lucas-Kanade algorithm uses the feature detector to generate an initial guess at the most down-sampled image and then reduce errors in higher level images using information from lower levels. The following diagram explains how the guesses are transferred and used in an iterative-refinement process.



I^n_{m-1}
I^k_{m-1}
I^{k-1}_{m-1}
I^0_{m-1}
LK optical-flow estimation
LK optical-flow estimation
g^k
LK optical-flow estimation
g^{k-1}
LK optical-flow estimation
d
I^n_m
I^k_m
I^{k-1}_m
I^0_m
Pyramid representation of I_{m-1}
Pyramid representation of I_m

Figure 6: Error equation using the notion of a neighborhood

5. IMPLEMENTATION DETAILS

The first assumption of optical flow equation is that the two images I and J are subsequent images taken from a video stream and that because they are so closely spaced temporally, the amount of motion that will be captured is small. The benefit of this assumption is that instead of trying to look for the corresponding point in the entire J image, we focus our attention on a neighborhood centered around the location of the point in image I. In my problem, I used phone which has a camera that operates at 30Hz which means it samples the continuous real world motion into bins of size 1/30th of a second. For our problem, we are looking for obstacles that are usually human beings(and CS students) or other autonomous machines. The phone camera lens used has a 1920-by-1080-pixel resolution.

5.1 Effect of Camera Ego-Motion and Bright Points

The following is an image frame from the original video used for testing the lucas-kanade algorithm.

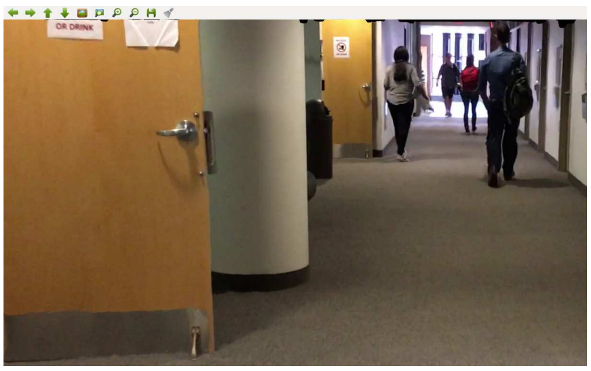


Figure 7: Image from original video file

The following is the optical flow obtained using lucas-kanade tracking algorithm. Notice how the bright points towards the end of the hallway are selected even when they are stationary. This makes it difficult to use lucas-kanade algorithm directly. The bright points in the hallway stand out more than important points

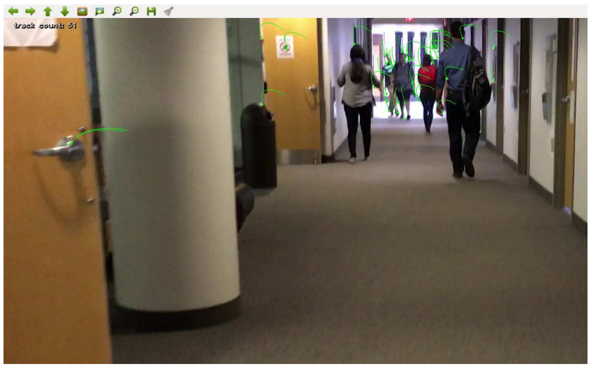


Figure 8: Bright points problem

Another consequence of the bright points problem is that the lucas-kanade algorithm suffers from is the problem of tracking only points that a corner detection algorithm decides to be important. Therefore if the image has more texture in the background that the foreground, it will track the background instead. This is explained by making a fast lateral camera movement. Notice how the points tracked are the ones belonging to the window at the end of the hallway.

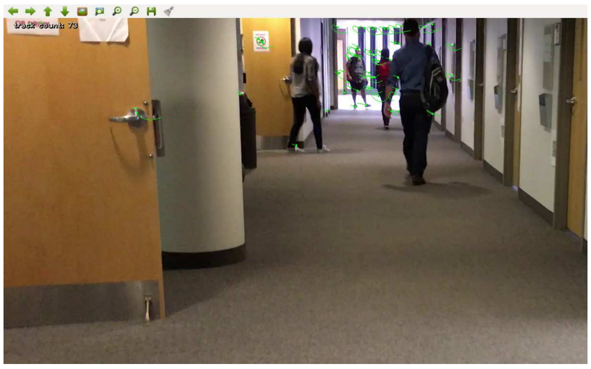


Figure 9: Spurious camera motion effect

6. DENSE OPTICAL FLOW ESTIMATION

Dense optical flow estimation involves tracking flow per pixel. An advantage of doing per pixel computations is that instead of tracking a few corner points, we track moving boundaries of images. Therefore, we don't see the bright points problem happening.

6.1 Farneback's Algorithm

Farneback's algorithm uses a two-frame equation where the pixel neighborhood is estimated using coefficients of quadratic polynomials and how they undergo translation from the first frame to the second[9]. Like Lucas-Kanade, it uses a least squared approach. Farneback was trying to work with high-noise images taken from a helicopter for the WITAS project[9].

The following is the local neighborhood model used in

$$f_1(\mathbf{x}) = \mathbf{x}^T \mathbf{A}_1 \mathbf{x} + \mathbf{b}_1^T \mathbf{x} + c_1$$

Figure 10: Neighborhood model

$$\begin{aligned} f_2(\mathbf{x}) = f_1(\mathbf{x}-\mathbf{d}) &= (\mathbf{x}-\mathbf{d})^T \mathbf{A}_1 (\mathbf{x}-\mathbf{d}) + \mathbf{b}_1^T (\mathbf{x}-\mathbf{d}) + c_1 \\ &= \mathbf{x}^T \mathbf{A}_1 \mathbf{x} + (\mathbf{b}_1 - 2\mathbf{A}_1\mathbf{d})^T \mathbf{x} + \mathbf{d}^T \mathbf{A}_1 \mathbf{d} - \mathbf{b}_1^T \mathbf{d} + c_1 \\ &= \mathbf{x}^T \mathbf{A}_2 \mathbf{x} + \mathbf{b}_2^T \mathbf{x} + c_2. \end{aligned}$$

Figure 11: Using displacement vector

The following is a snapshot of the flow field obtained from the dense optical flow algorithm.

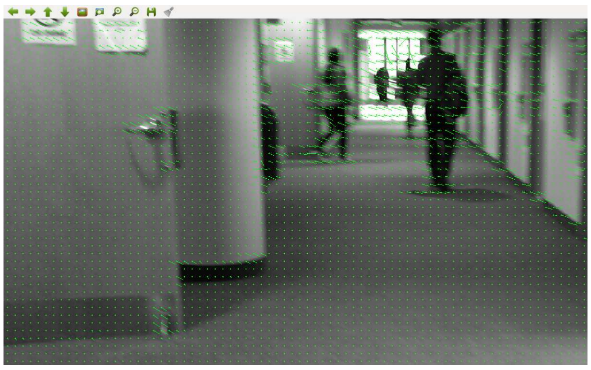


Figure 12: Dense Flow

6.2 Optimization Strategy

The following HSV color space visualization of dense optical flow indicates how the number of computations involved in dense flow estimation is relatively high.

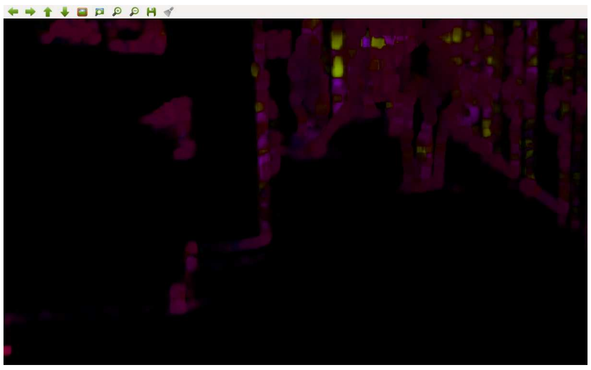


Figure 13: HSV visualization of dense flow

An optimization approach is to use only the border points of previously identified moving contours regions. This might significantly reduce computations and make it feasible to compute dense optical flow. The apparent motion of pixels obtained from dense optical flow estimation can be fed to a Lucas-Kanade algorithm that keeps track of how the contour lines move in a neighborhood window.

Running the HSV visualization through Canny edge detection shows how much information is being calculated per frame.

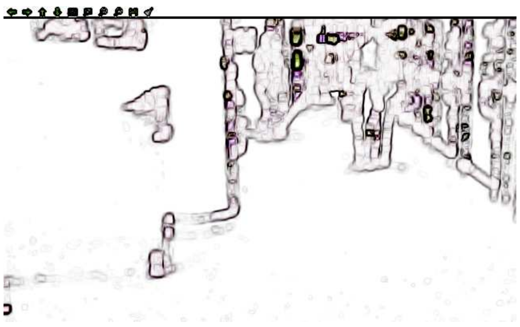


Figure 14: Edge image of dense flow

The amount of computations increases with the perceived number of moving entities in the background and the foreground. Therefore, we need to optimize the algorithm to be able to use it for a close to real-time estimate of moving contours.

7. CONCLUSION

Lucas-Kanade algorithm is an excellent feature tracker but despite its ability to make tracking an efficient process, it heavily relies on the ability of the underlying feature detector which might be sensitive to motion of bright background points.

Instead, a better algorithm is Farneback's algorithm which no only provides information about the contours of bodies but makes sure that the high-texture background does not overshadow the foreground points. An optimization on the tracking of images similar to lucas-kanade for dense flow estimation would make it feasible to use it for robot navigation.

8. FUTURE WORK

Using a background subtraction algorithm based on known ground-plane and wall locations or understanding of the depth on the robot device might be a huge factor in upscaling the ability of sparse flow estimation techniques.
The optimization strategy suggested for dense optical flow might result in lesser computations along the estimated contour. This will also help the robot perceive objects and people with their contour lines. This might help further with identification of subjects in the environment.

9. REFERENCES

[1] : https://www.liverpool.ac.uk/ marcob/Trieste/aperture.html

[2] : http://people.csail.mit.edu/lpk/mars/temizer$_2$001/$Optical_Flo$

[3] : https://en.wikipedia.org/wiki/Optical$_{flow}$

[4] : https://stoomey.wordpress.com/2008/04/18/20/

[5] : Stumpf, P. (1911) anticipation of the aperture problem,Reichardt detectors, and perceived motion loss at equiluminance.Perception, 25, 1235-1242.

[6] : Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm - Jean-Yves Bouguet Intel Corporation Microprocessor Research Labs

[7] : Determining Optical Flow Berthold K.P. Horn and Brian G. Rhunck Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cam bridge, MA 02139, U.S.A.

[8] : yves Bouguet, Jean. ”Pyramidal implementation of the lucas kanade feature tracker.” Intel Corporation, Microprocessor Research Labs (2000). [9] : FarnebÃd'ck, Gunnar. ”Two-frame motion estimation based on polynomial expansion.” Scandinavian conference on Image analysis. Springer Berlin Heidelberg, 2003.

10. APPENDIX : BREAKDOWN OF WORK

1. Week 1: Investigation of problem and general discussions around approach to be taken (8 hours)
2. Week 2: Simple Morphological processing - erosion, dilation, gradient analysis, histogram equalization (6 hours)
3. Week 3: Testing edge detection to guess feature points (7 hours)
4. Week 4: Writing an image down-sampling utility (8 hours)
5. Week 5: Color space conversions over images from video to notice how brightness constancy is violated (6 hours)
6. Week 6: Visualizing corner points in images using harris corner detector and SIFT point detection (6 hours)
7. Week 7: Reading paper related to lucas-kanade implementation and learning derivations of the equation (10 hours)
8. Week 8: Implementing partial steps for lucas-kanade in code (10 hours)
9. Week 9: Reading about Farneback's algorithm (6 hours)
10. Week 10: Visualizing sparse and dense optical flow space (8 hours)
11. Week 11: Conversions to other color spaces to see optical flow clearly (6 hours)
12. Week 12: Formulating the optimization strategy for dense optical flow (8 hours)
13. Week 13: Exploring feasibility of using output from dense flow in Lucas-Kanade directly (8 hours)
14. Week 14: Reading about optical flow constraints and Farneback's algorithm. Understanding ROSS and integration (10 hours)

15. Week 15: Attempting integration with ROSS to get 640x480 images from robot (8 hours)
16. Week 16: Using canny edge detection over HSV image and writing report (12 hours)