Identification of Moving Obstacles with Pyramidal Lucas Kanade Optical Flow and k means Clustering

W.S.P. Fernando*, Lanka Udawatta^{+} , Pubudu Pathirana^{++}
*Faculty of Engineering, University of Moratuwa, Sri Lanka.
Email: shehan117@gmail.com
$^{+}$Faculty of Engineering, University of Moratuwa, Sri Lanka
Email: lanka@elect.mrt.ac.lk
$^{++}$Deakin University, Australia.
Email: pubudu.pathirana@deakin.edu.au

Abstract — **This paper describes the methodology for identifying moving obstacles by obtaining a reliable and a sparse optical flow from image sequences. Given a sequence of images, basically we can detect two-types of on road vehicles, vehicles traveling in the opposite direction and vehicles traveling in the same direction. For both types, distinct feature points can be detected by Shi and Tomasi corner detector algorithm. Then Pyramidal Lucas Kanade method for optical flow calculation is used to match the sparse feature set of one frame on the consecutive frame. By applying k means clustering on four component feature vector, which are to be the coordinates of the feature point and the two components of the optical flow, we can easily calculate the centroids of the clusters and the objects can be easily tracked. The vehicles traveling in the opposite direction produce a diverging vector field, while vehicles traveling in the same direction produce a converging vector field.**

Keywords: **optical flow, feature point, k means clustering, centroid.**

I. Introduction

Motion is one of the most important research topics in computer vision. It is the base for many other problems such as visual tracking, structure from motion, etc. Visual tracking by analyzing motion and shape in video gains more attractions in many computer vision applications including video surveillance, motion analysis and extraction for computer animation. In computer vision based robots and autonomous vehicles, their sensory input is vision and they extract relevant information from image sequences acquired by a CCD camera mounted on a robot or autonomous vehicle. Vehicles have already been designed to navigate in out door scenes using computer vision such as VAMORS, VITA II, CMU-NAVLAB. In the recent years, these technologies have received increasing publicity in both the civilian and military domains [1],[2]. In our paper we are going to analyze the optical flow pattern which is the brightness movement of the pixels in image plane, correspond to the motion in the 3D environment [3]. Optical flow plays a significant role in motion estimation and motion explanation. Which means the accuracy of the objective is mainly depends on the accurate computation of optical flow and is also the foundation of the research. Some of the methods of calculating optical flow are non-robust. Which means they will fail to correctly compute optical flow when the two assumptions: data conservation and spatial coherence are violated. Some of the methods calculates a precise optical flow, but unfortunately they have high computation cost. For that reason in this research we use pyramidal implementation of the Lucas Kanade method which have proved as a very fast and a reliable optical flow estimator algorithm. In this research work our main objective is to investigate the problem of identify moving objects and tracking by calculating centroids of moving objects by analyzing optical flow, and applying k means clustering criteria in image sequences acquired by a CCD camera in roads of suburban area. The optical flow field is detected in image sequences will not performed in each and every pixel in an image [4],[6]. This is performed only for selected feature points or corner points, that can be extracted by Shi and Tomasi feature tracking algorithm [8]. Thus given two consecutive images, we can track the corner points in the image. By this method we can have a sparse flow field [7],[9],[11].

For the feature points or corners, we define a feature vector which contains four features (the location of the flow and direction of the flow). By analyzing Euclidean distance between two flow vectors we can easily cluster the feature points by a method known as k means clustering algorithm. And also, we could easily calculate the centroids of each cluster. By analyzing centroids we can easily track the moving objects. As the camera translates, due to the relative motion between the camera and the moving object creates two types of flow vectors. If the motion of vehicle approaching at an opposite speed will produce a diverging flow on the image plane. On the contrary, if the motion of a vehicle departing or overtaking, will produce a converging flow. These observations can be used to detect and distinguish the relative motions.

The rest of the paper is organized as follows: First the theoretical Background of the different approaches for calculating optical flow (section II), and a mathematical illustration of the pyramidal optical flow calculation (section III). Section IV describes the iterative Lucas-Kanade Method. Then the calculation of feature points or corners of an image in section V. And in section VI, describes the clustering of feature points. Results are shown in section VII. In section VIII gives a conclusion. And finally with Appendix A, which describes the step by step derivation of Lucas Kanade method. Appendix B gives the pseudo code for complete pyramidal Lucas Kanade method.

I. THEORETICAL BACKGROUND

A. Introduction to Motion field and Optical Flow

The motion field in an image is the real motion of the object in the 3-D scene, projected onto the image plane. The optical flow is defined as the "flow" of gray values at the image plane in time varying images. This is what we observe. Optical flow or image flow can be defined as the apparent motion at the image plane based on visual perception. and has the dimensions of velocity. We denote the optical flow with $\vec{v} = (u, v)$. Where u and v denote the x and y components of the optical flow vector at a point respectively. If the optical flow is determined from two consecutive images, it appears as a displacement vector $\vec{d}$, from the features in the first to those in the second image. Let, $\vec{d} = (d_x, d_y)$. Where d_x and d_y denote the x and y components of the displacement vector at a point respectively [4].

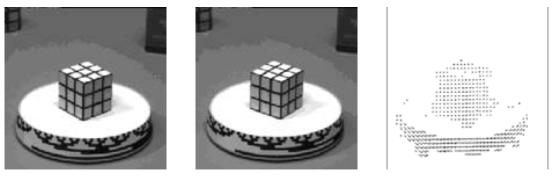

(a) first frame (b) second frame (c) optical flow

Fig 1. A Rubik's cube on a rotating turntable.

B. Different Approaches of Optical Flow Calculation

1) Gradient Based Methods (or differential methods)

This method is based on simple image intensity conservation equation. Let image brightness at the point (x, y) in the image plane at time t is denoted by $I(x, y, t)$. When the pattern moves we assume that the brightness of a particular point in the pattern is constant so that,

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \qquad (1)$$

Where $\delta x, \delta y$ and δt are the inter frame displacement and time interval respectively. The first order differential equation of (1) can be written as:

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \qquad (2)$$

Where $\frac{dx}{dt} = u \quad \frac{dy}{dt} = v$

can be used to approximate the velocity component in the direction of a spatial gradient of the image intensity.
To find the other component as well, we impose another constraint which measures the departure from smoothness in the velocity flow. This new measurement can be written as:

$$\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \qquad (3)$$

By combining equations (2) and (3) with a suitable weighting factor, and by using calculus of variation to minimize the total error functional, we can formulate an iterative solution for optical flow [4],[5].

2) Feature Based Approach

In this type of method, tries to establish correspondences of invariant features or more precisely corners, between time-varying images. Points of high intensity variation, gray level corners can be matched for determining displacement fields. This paper discuss, the tracking of feature points, in a image sequence by using pyramidal Lucas Kanade method [6].

C. Pyramidal Implementation of the Lucas – Kanade Feature Tracker

Let I, J be two, two dimensional grayscale images described as below. And Let $\vec{x} = (x, y)$ be a pixel location in image pane. Then ,

$$I(\vec{x}) = I(x, y) \qquad (4)$$

$$J(\vec{x}) = J(x, y) \qquad (5)$$

The image $I(\vec{x})$ will be referred to as the first image and $J(\vec{x})$ as the second image. Let upper left corner pixel coordinate vector is $(0,0)$. The n_x and n_y be the width and height of the images respectively. The lower right pixel coordinate vector is $(n_x - 1, n_y - 1)$. Consider a specific image point $\vec{u} = (u_x, u_y)$ on the first image I. The goal of feature tracking is to find the location $\vec{v} = \vec{u} + \vec{d}$, on the second image J, such that $I(\vec{u})$ *and* $J(\vec{v})$ are almost equal. And also it is essential to define the similarity between image brightness of $I(\vec{u})$ and $J(\vec{v})$ in a two dimensional neighborhood sense due to the aperture problem. This two dimensional neighborhood is also referred to as integration window.

Let ω_x and ω_y be two integers. We define the image displacement $\vec{d}$, as being the vector minimizes the residual function $\varepsilon(\vec{d})$, [7], [9] defined as follows:

$$\varepsilon(\vec{d})=\varepsilon(d_x,d_y)=\sum_{x=u_x-\omega_x}^{u_x+\omega_x}\sum_{y=u_y-\omega_y}^{u_y+\omega_y}[I(x,y)-J(x+d_x,y+d_y)]^2 \qquad (6)$$

The similarity is measured on a image neighborhood of size $(2\omega_x+1)\times(2\omega_y+1)$

Typical values for ω_x and ω_y are 1, 2, 3,…..pixels.

D. Choice of the integration window

Selecting a small integration window : is preferable not to "smooth out " the details contained in the images. And also minimizes the errors in occlusion problems. (select small values for ω_x and ω_y) [7],[8].

Selecting a large integration window: In order to handle large motions, it is preferable to pick large integration windows. But may have the disadvantage of incorrect feature tracking due to multiple motions. In our method, we select $d_x \le \omega_x$ and $d_y \le \omega_y$, which implies a small window, gives better results for the displacement $\vec{d}$ [7], [8].

According to above emphasis of the integration window, it is clear that there is a trade off between the sizes we select. To provide a solution to that problem, the Pyramidal Implementation of the classical Lucas – Kanade algorithm can be used. And also an iterative implementation of the Lucas – Kanade optical flow calculation provides sufficient estimation of the local tracking.

E. Resolution pyramids

The central motivation behind pyramidal representation in applying for a feature tracker is that to be able to handle large pixel motions. (example: larger than the ω_x and ω_y)

Let us define, the pyramid representation of a generic image I of size $n_x \times n_y$. Let $I_0 = I$, be the 0 th level image. This image is essentially be the highest resolution image or raw image. The image width and height at that level are defined as $n_x^0 = n_x$ and $n_y^0 = n_y$.

The Pyramid Representation is then, built in a recursive pattern as compute I^1 from I_0. Then compute I^2 from I^1, and so on…… Let L (L = 0,1,2,….n) where n is the maximum level number, be a generic pyramid level and let I^{L-1} be the image at level L-1. We denote the width and height of I^{L-1} as n_x^{L-1} and n_y^{L-1} respectively.

Gaussian Pyramid: The Gaussian pyramid is an image pyramid structure consists of low-passed filtered, sub sampling images by a factor of 2, of the preceding level of the pyramid. The base level of the pyramid is the original image [14].

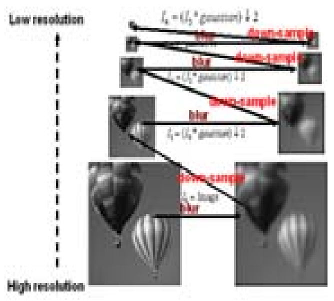


Fig 2: creating an image pyramid

The low pass filter is used for image anti aliasing before image sub sampling. Anti aliasing is a technique of minimizing the distortion artifacts known as aliasing, when representing a high-resolution signal at a lower resolution. Gaussian filter can be used as a low pass filter. After that, sub sampling was accomplished by a rate of 2. If we repeat the smoothing and sub sampling operations iteratively, we obtain series of images, called the Gaussian pyramid. So we have $\{I^L\}L=0,1,2,....,L_m$ and $\{J^L\}L=0,1,2,....,L_m$. The value L_m, is the height of the pyramid (picked heuristically). **Ex:** for an image I_0, of size 640 × 480, the images : I_1 = 320 × 240, I_2 = 160 ×120, I_3 = 80 × 60, I_4 = 40 × 30.

Applying pyramidal construction in feature tracking.

Goal: For a given point $\vec{u}$ in image I, find it's corresponding location $\vec{v}=\vec{u}+\vec{d}$, in image J, or, find it's pixel displacement vector $\vec{d}$ as eq. 6. For $L(L=0,1,2,...,L_m)$ define $\vec{u}^L=(u_x^L,u_y^L)$ the corresponding coordinates of the point $\vec{u}$, on the pyramidal images I^L, Then u^L, are computed as follows :

$$\vec{u}^L=\frac{\vec{u}}{2^L} \qquad (7)$$

The division operation is applied to both coordinates independently. The overall pyramidal tracking algorithm can present as follows. First, the optical flow is computed at the deepest pyramid level L_m. Then, the result of that computation is propagated, to the upper level L_m-1, in a form of an initial guess. By taking the initial guess, the refined optical flow is computed at level L_m-1, and that result is propagated to the level L_m-2 and so on, up to the level 0.

II. Detail Mathematics of the recursive operation, between the generic level L+1 and L

Notation 1. Let $\vec{g}^{L} (L = 0,1,2,..., L_m)$ be the initial guess for the optical flow at level L, such that $\vec{g}^{L} = (g_x^L, g_y^L)$

Notation 2. Let $\vec{d}^{L} (L = 0,1,2,..., L_m)$ be the residual pixel displacement at level L, such that $\vec{d}^{L} = (d_x^L, d_y^L)$

A. First the algorithm, is initialized by setting the initial guess for optical flow at the highest level L_m as $\vec{0}$.

$$\vec{g}^{Lm} = (0,0) \tag{8}$$

B. From notation 1, initial guess for optical flow at level L, which is derived from computations from level L_m to L + 1 is: $\vec{g}^{L}$

C. In order to compute the optical flow at level L, find $\vec{d}^{L}$ that minimizes the residual function ε^{L} as follows:

$$\varepsilon^L(\vec{d}^L) = \varepsilon(d_x^L, d_y^L) = \sum_{x=u_x^L-\omega_x^L}^{u_x^L+\omega_x^L} \sum_{y=u_y^L-\omega_y^L}^{u_y^L+\omega_y^L} [I^L(x,y) - J^L(x+g_x+d_x, y+g_y+d_y)]^2 \tag{9}$$

For all the values of L, the size of the integration window is fixed at $(2\omega_x+1)\times(2\omega_y+1)$

D. The, above residual function can be easily solved through the standard Lucas – Kanade method.

E. For now we suppose that this residual displacement $\vec{d}^{L}$ is computed. Then, the total displacement vector is propagated to the next level L – 1, by passing as the new initial guess : $\vec{g}^{L-1}$.

$$\vec{g}^{L-1} = 2(\vec{g}^{L} + \vec{d}^{L}) \tag{10}$$

F. The, above residual function can be easily solved through the standard Lucas – Kanade method.

In level L – 1, the optical flow residual vector $\vec{d}^{L-1}$, is computed through the same procedure. And goes on, until the finest image resolution is reached. (L=0)

G. The, above residual function can be easily solved through the standard Lucas – Kanade method.

Finally the finest optical flow d would be:

$$\vec{d} = \vec{g}^{0} + \vec{d}^{0} \tag{11}$$

As we have seen in the algorithm, this method enables large pixel motions, while keeping the size of the integration window relatively small [7].

III. Iterative Optical Flow Computation (Iterative Lucas – Kanade method)

Notation 3. The same type of operation is performed at each of the level L, for simplicity we drop the superscripts L, and define $A(x, y)$ and $B(x, y)$ such that: $A(x, y) = I^L(x, y)$ and $B(x, y) = J^L(x + g_x, y + g_y)$.

Notation 4. The residual displacement $\vec{d}^{L}$ can be defined as $\vec{v} = (v_x, v_y)$

Notation 5. The position vector $\vec{u}^{L}$ can be defined as $\vec{p} = (p_x, p_y)$

At every level L of the pyramid, the goal is to find the residual displacement vector $\vec{v}$, that minimize the residual function ε^{L}. The initial guess of the flow vector $\vec{g}^{L}$, is used to pre-translate the image patch in the second image J, so that the residual vector $\vec{v}$ can be kept small, for classical Lucas – Kanade algorithm [7],[11],[12]. The solution is given by:

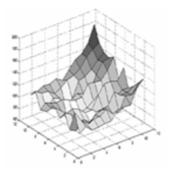
$$\vec{v} = -G^{-1}.b \tag{12}$$

For derivation refer Appendix A. This expression is valid only if the matrix G is invertible. Then this is the standard Lucas – Kanade optical flow equation, which is valid only if the pixel displacement is small (in order to validate Taylor expansion). In practice, to get an accurate solution it is necessary to iterate multiple times on this scheme.

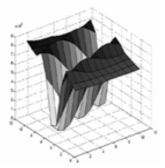
IV. Finding feature Point (Corner point) of an image

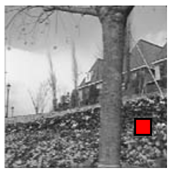
In the computation of optical flow, consider the matrix G. To have a solution for optical flow this needs to be invertible. In other words, the minimum eigen value of G must be larger than a threshold λ_T. Suppose λ_1, λ_2 be the two eigen values calculated from the matrix G for a point in image plane.. If min(λ_1, λ_2) > λ_T then that particular point is considered as a feature point or a corner. And these points are, where the Lucas-Kanade algorithm really works. For these points the aperture Problem disappears. For an edge point we can have a large λ_1 and small λ_2. For a low textured windowed point we can expect small λ_1 and λ_2. For high textured window point we have large λ_1 and λ_2. Finally, points with large eigen values are the points that are good to track and are fed to the pyramidal Lucas-Kanade tracker [7], [8].

(a) low textured region


(b) Intensity Profile

(c) Edge


(d) Intensity Profile


(e) high textured region

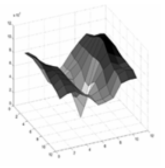
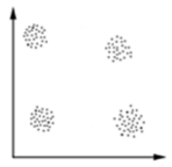

(f) Intensity Profile

Fig 3. Representation of regions

For Summary of the whole algorithm, refer Appendix B

V. Cluster Analysis

Clustering is the classification of objects into different groups, or more precisely, the partitioning of a dataset into subsets (clusters), so that the data in each subset (ideally) share some common trait - often proximity according to some defined distance measure.
We can show this with a simple graphical example:


(a) sample data set

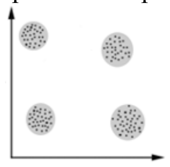

(b) After Clustering to 4 subsets

Fig 4 : Example of Clustering a dataset

In this case we easily identify the 4 clusters into which the data can be divided.

k means Clustering

k-means algorithm executes a sharp classification, in which each object is either assigned to a class or not. The k means algorithm assigns each point to the cluster whose center (also called centroid) is nearest. The center of a cluster is the average of all the points in the cluster — that is, its coordinates are the arithmetic mean for each dimension separately over all the points in the cluster. The algorithm steps are :

- Choose the number of clusters, k.
- Randomly generate k clusters and determine the cluster centers, or directly generate k random points as cluster centers.
- Assign each point to the nearest cluster center.
- Recompute the new cluster centers.
- Repeat the two previous steps until some convergence criterion is met (usually that the assignment hasn't changed).

The advantage of this algorithm are its simplicity and speed which allows it to run on large datasets. The disadvantage is that it does not yield the same result with each run, since the resulting clusters depend on the initial random assignments.

VI. Results

Here we present the results of the Pyramidal Implementation of the Lucas – Kanade Feature Tracker method and the clustering of feature points of the image with k means clustering. All the test frames were 320 × 240 pixel images. The image sequences were taken as three sets, taken in a suburban area. These sets were: while the vehicle moving along a straight road, The vehicle moving straight forward and another vehicle is approaching on the other lane and the vehicle moving straight forward and another vehicle is overtaking. From the example flows, we can see that the objects that move opposite to the motion of the camera produces diverging vector field, while the objects traveling in the same direction (overtaking) produces converging vector field.

A. Estimation of feature tracking / Point Correspondence

We initialized the maximum number of feature points to 2000. The size of the integration window to 3 by 3. In fig 8. the green unfilled circles indicate the centroids of each cluster.

B. Examples of Calculated Flows

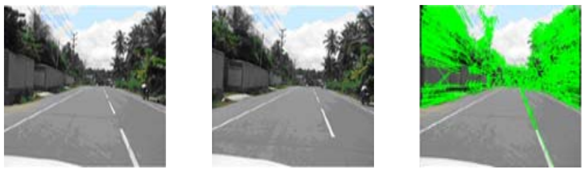

(a) first frame (b) Second frame (c) optical flow

Fig 5. optical flow: vehicle move along a straight road.

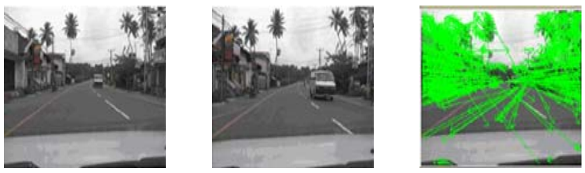

(a) first frame (b) Second frame (c) optical flow

Fig 6. optical flow when a vehicle traveling in the opposite direction.

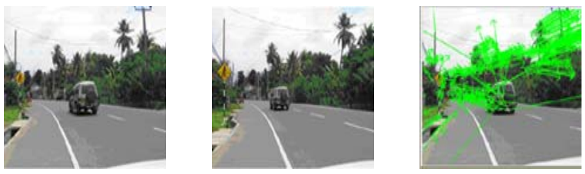
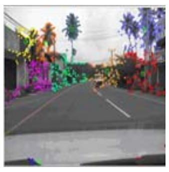

(a) first frame (b) Second frame (c) optical flow

Fig 7. optical flow when a vehicle traveling in the same direction

C. Examples of Clustering

(a) clustering while the vehicle move along straight road


(b) clustering of vehicle, traveling in opposite direction.

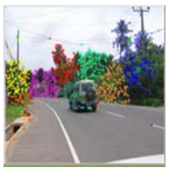

(c) clustering when vehicle traveling in the same direction

Fig 8 : Clustering different types of motion.

VII. Conclusion

By the pyramidal Lucas Kanade method we were able to calculate a sparse and reliable optical flow from image sequences on feature points. These distinct feature points or corners can be detected by Shi and Tomasi corner detector algorithm. As shown in fig 7,8 and 9 the algorithm was sensitive to detect and track points reliably. The image sequence was 320 × 240 pixel images and the window size was selected as 3 × 3. By applying k means clustering on four component feature vector, which are to be the coordinates of the feature point and the two components of the optical flow, we can easily calculate the centroids of the clusters so that the objects can be easily tracked. The vehicles traveling in the opposite direction produce a diverging vector field, while vehicles traveling in the same direction produce a converging vector field.

Appendix A

Optical flow computation with Lucas Kanade Method. For a generic level of the Gaussian pyramid, the goal is to find the vector $\vec{v}$ .For the images A and B, the matching function that minimizes the residual $\varepsilon(\vec{v})$ would be:

$$\varepsilon(\vec{v}) = \varepsilon(v_x, v_y) = \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} [A(x,y) - B(x+v_x, y+v_y)]^2$$

The vector $\vec{v}$ is assumed to be a very small displacement vector. So the Taylor expansion of $B(x+v_x, y+v_y)$ is,

$$B(x+v_x, y+v_y) = B(x,y) + v_x \frac{\partial B}{\partial x} + v_y \frac{\partial B}{\partial y}$$

By substituting for $B(x+v_x, y+v_y)$ we have,

$$\varepsilon(\vec{v}) = \sum\sum[A(x,y) - B(x,y) - v_x \frac{\partial B}{\partial x} - v_y \frac{\partial B}{\partial y}]^2$$

To minimize the error function, we differentiate $\varepsilon(\vec{v})$ with respect to v_x and v_y and set their values to 0.

$$\frac{\partial \varepsilon(\vec{v})}{\partial v_x} = 0 \quad \text{and} \quad \frac{\partial \varepsilon(\vec{v})}{\partial v_y} = 0$$

$$\frac{\partial \varepsilon(v)}{\partial v_x} = -2\sum\sum[A(x,y) - B(x,y) - v_x \frac{\partial B}{\partial x} - v_y \frac{\partial B}{\partial y}](\frac{\partial B}{\partial x})$$

$$\frac{\partial \varepsilon(v)}{\partial v_y} = -2\sum\sum[A(x,y) - B(x,y) - v_x \frac{\partial B}{\partial x} - v_y \frac{\partial B}{\partial y}](\frac{\partial B}{\partial y})$$

The quantity $B(x,y) - A(x,y)$ can be interpreted as temporal derivative of the point $\vec{p}$ as δI_t .

The quantities $\frac{\partial B}{\partial x}$ and $\frac{\partial B}{\partial y}$ are merely equal to $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$

Refer **Notation 3**. For a point $\vec{p}$, I_x and I_y can be computed in the neighborhood $(2\omega_x+1)\times(2\omega_y+1)$ by central difference operator.

$$\frac{\partial I}{\partial x} = \frac{A(x+1,y) - A(x-1,y)}{2}$$

$$\frac{\partial I}{\partial y} = \frac{A(x,y+1) - A(x,y-1)}{2}$$

Then, $\sum\sum[\delta I_t I_x + v_x I_x^2 + v_y I_x I_y] = 0$

$$\sum\sum[\delta I_t I_y + v_x I_x I_y + v_y I_y^2] = 0$$

Denote

$$G = \begin{bmatrix} \sum\sum I_x^2 & \sum\sum I_x I_y \\ \sum\sum I_x I_y & \sum\sum I_y^2 \end{bmatrix}$$

and

$$b = \begin{bmatrix} \sum\sum I_x \delta I_t \\ \sum\sum I_y \delta I_t \end{bmatrix}$$

Then, $\vec{v} = -G^{-1} b$

This is the standard Lucas Kanade optical Flow equation, which is valid only if the displacement vector is small. Practically, to get a more accurate result, iterate this multiple times.

Appendix B

```
Input :
    Initialization of pyramidal guess :
        $\vec{g}^{Lm} = (0,0)$
Output :
    Find the corresponding location $\vec{v}$ on image $J$ for a
    feature point $\vec{u}$ on image $I$ .
begin
    Build pyramid representations of $I$ and $J$ :
    $\{I^L\} L = 0,1,2,....,L_m \quad \{J^L\} L = 0,1,2,....,L_m$
```

```
for L = L_m down to 0 with step of -1
begin
    $\vec{u}^L = \frac{u}{2^L}$
    $I_x(x,y) = \frac{I^L(x+1,y) - I^L(x-1,y)}{2}$
    $I_y(x,y) = \frac{I^L(x,y+1) - I^L(x,y-1)}{2}$
    $G = \begin{bmatrix} \sum\sum I_x^2 & \sum\sum I_x I_y \\ \sum\sum I_x I_y & \sum\sum I_y^2 \end{bmatrix}$

    Initialize $\vec{v}^0 = (0,0)$
    for k = 1 to K with step of 1
    begin
        $\delta I_k(x,y) = I^L(x,y) - J^L(x + g_x + v_x, y + g_y + v_y)$
        $b_k = \begin{bmatrix} \sum\sum I_x \delta I_k \\ \sum\sum I_y \delta I_k \end{bmatrix}$
        $\vec{\eta}^k = -G^{-1} b_k$
        $\vec{v}^k = \vec{v}^{k-1} + \vec{\eta}^k$
    end

    final optical flow at level L : $\vec{d}^L = \vec{v}^K$
    guess for next level L – 1 :
    $\vec{g}^{L-1} = (g_x^{L-1}, g_y^{L-1}) = 2(\vec{g}^L + \vec{d}^L)$
end
find the total optical flow vector : $\vec{d} = \vec{g}^0 + \vec{d}^0$
location of point on $J$ : $\vec{v} = \vec{u} + \vec{d}$
end.
```

REFERENCES

[1] A. Giachetti, M. Campani, and V. Torre, "The Use of Optical Flow for Road Navigation," *IEEE Transactions. on Robotics and Automation,,* vol. 14, Issue.1, pp. 34-48, February 1998.

[2] D. Lieb, A. Lookingbill, and S. Thrun, "Adaptive Road Following using Self-Supervised Learning and Reverse Optical Flow," *Stanford Artificial Intelligence Laboratory, Stanford University,* 2005.

[3] R. Siegwart, I. R. Nourbakhsh, "Introduction to Autonomous Mobile Robots," *Prentice Hall of India (pvt.) Ltd.*, 2005.

[4] Emanuele Trucco, Alessandro Verri, "Introductory Techniques for 3-D Computer Vision," *Prentice Hall, Inc.* 1998.

[5] B. K. P. Horn, B. G. Schunck, "Determining Optical Flow," *Artificial Intelligence Laboratory, MIT, Cambridge, MA*, pp. 185-203, 1981.

[6] Q. X. Wu, "A Correlation-Relaxation-Labeling Framework for Computing Optical Flow-Template Matching from a New Perspective," *IEEE Transactions. on Pattern Analysis and Machine Intelligence,* vol. 17, No.8 , pp. 843-853, September 1995.

[7] J. Y. Bouguet, "Pyramidal implementation of the Lucas Kanade Feature Tracker Description of the algorithm," *Intel Corporation Microprocessor Research Labs*, 2003.

[8] J. Shi, C. Tomasi, "Good Features to Track,"IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94), *Seattle*, pp. 593-600, June 1994.

[9] H. Eltoukhy, K. Salama, "Multiple Camera Tracking," *Stanford image sensors group Electrical Engineering Department, Stanford University*, 2005.

[10] C. Tomasi, T. Kande, "Detection and Tracking of Point Features," *Technical Report CMU-CS-91-132* , April 1991.

[11] B. D. Lucas, T. Kanade "An Iterative Image Registration Technique with an Application to Stereo Vision (DARPA)," *Proceedings of Imaging Understanding Workshop,* pp. 121-130*,* April 1981.

[12] S. Birchfield, "Derivation of Kanade - Tomasi Tracking Equation," January 1997.

[13] D. H. Ballard, C. M. Brown, "Computer Vision," *Prentice-Hall Inc., Englewood Cliffs, New Jersey*, 1982.

[14] "Gaussian Pyramid," *The MathWorks, Video and Image Processing Blockset,* http://www.mathworks.fr