Perception For Autonomous Robots

Project 4

Lucas-Kanade Template Tracker

Raghav Nandwani (116189941) Sanket Goyal (116155144)

Varsha Eranki (116204569)

Contents

List of Figures

1 Introduction

Optical flow of events inside a frame is found by Lucas Kanade Algorithm. These events in a frame are gathered after considering a pixel and its adjacent pixels. The least square criterion is used to approximate the solution. The change of an object in the image and its pixel are gone through for two frames and at any moment of time, they are nearly the same. For optical flow, this same pixel intensity between two frames is used to find the flow of motion or the vector flow. Following this, the optical flow equations are used in the pixels and the window center.

In this project we implemented the Lucas Kanade tracker on given video sequence i.e. a human walking, a car on the road and a vase on the table. The tracker was initialized by defining a template, i.e. by bounding a rectangle around the object needed to be tracked. For subsequent frames, the tracker updates the parameters of affine transformation and warps it to the current frame.

The drive link to view all the videos is:
https://drive.google.com/drive/folders/1CkKI_U00ALgQnzyZ0iGoDgwbqEnVGcin?usp=sharing

2 Lucas Kanade Algorithm

2.1 Least Square Criterion

'u' and *'v'* are the hypothesized location of template in current frame which is given by:

$$E(u,v) = \sum [I(x+u, y+v) - T(x,y)]^2$$

$$E(u,v) \approx \sum [I(x,y) + uI_x(x,y) + vI_y(x,y) - T(x,y)]^2$$

$$E(u,v) = \sum [uI_x(x,y) + vI_y(x,y) + D(x,y)]^2$$

By taking partial derivatives and equating to zero:

$$\partial E/du = \sum [uI_x(x,y) + vI_y(x,y) + D(x,y)]I_x(x,y) = 0$$

$$\partial E/dv = \sum [uI_x(x,y) + vI_y(x,y) + D(x,y)]I_y(x,y) = 0$$

Form matrix equation which is solved using Least-Squares method:

$$\sum \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\sum \begin{bmatrix} I_xD \\ I_yD \end{bmatrix}$$

2.2 Least Square: Affine Parameters

The Warp $W(x;p)$ takes the values of the pixels x in the coordinate frame and for the template T it maps to the every pixel's area $W(x;p)$ in the coordinate frame I. $W(x;p)$ denotes the set of parameters $p = (p_1, ..., p_n)^T$ is the vector of parameters.

$$W(x;p) = \begin{pmatrix} x + p_1 \\ y + p_2 \end{pmatrix}$$

The optical flow is now the vector of parameters $p = (p_1, p_2)^T$. To calculate Affine warps:

$$W(x;p) = \begin{pmatrix} (1+p_1) \cdot x + p_3 \cdot y + p_5 \\ p_2 \cdot x + (1+p_4) \cdot y + p_6 \end{pmatrix} = \begin{pmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

2.3 Inverse Compositional Image Alignment

Inverse Compositional Image Alignment is used to calculate the reduced the computational cost which is significantly huge in Hessian matrix. The parameters are not calculated in each step but are assumed to be constant for a few iterations and the updated again.

The image and template are then swapped.

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}$$

The inverse compositional algorithm minimizes the following equation:

$$\sum_x \left[T\left(W\left(x;\Delta p\right)\right) - I\left(W\left(x;p\right)\right)\right]^2$$

The roles for I and T are reversed and the updated warp is as follows:

$$W\left(x;p\right) \leftarrow W\left(x;p\right) \circ W\left(x;\Delta p\right)^{-1}$$

The incremental warp is inverted before the composition with the current estimate is carried out. The

parameters for the inverse for the affine warp are:

$$\frac{1}{(1+p_1)\cdot(1+p_4)-p_2\cdot p_3}\begin{pmatrix} -p_1-p_1\cdot p_4+p_2\cdot p_3 \\ -p_2 \\ -p_3 \\ -p_4-p_1\cdot p_4+p_2\cdot p_3 \\ -p_5-p_4\cdot p_5+p_3\cdot p_6 \\ -p_6-p_1\cdot p_6+p_2\cdot p_5 \end{pmatrix}$$

By performing the first order Taylor expansion

$$\sum_x \left[T\left(W\left(x;0\right)\right)+\bigtriangledown T\frac{\partial W}{\partial p}\Delta p-I\left(W\left(x;p\right)\right)\right]^2$$

Assuming W is the identity warp, the least squares question is estimated as:

$$\Delta p = H^{-1}\sum_x \left[\bigtriangledown T\frac{\partial W}{\partial p}\right]^T \left[I(W(x;p))-T(x)\right]$$

The Hessian marix where I is replaced with T

$$H=\sum_x \left[\bigtriangledown T\frac{\partial W}{\partial p}\right]^T \left[\bigtriangledown T\frac{\partial W}{\partial p}\right]$$

2.4 The Inverse Compositional Algorithm

Pre-compute:

1. Evaluate the gradient $\bigtriangledown T$ of the template $T(x)$
2. Evaluate the Jacobian $\frac{\partial W}{\partial p}$ at $(x;0)$
3. Compute the steepest descent images $\bigtriangledown T\frac{\partial W}{\partial p}$
4. Compute the Hessian matrix

Iterate

5. Warp I with $W(x;p)$ to compute $I(W(x;p))$
6. Compute the error in the image $I(W(x;p))-T(x)$

7. Evaluate $\sum_x \left[\bigtriangledown T \frac{\partial W}{\partial p}\right]^T \left[I\left(W\left(x;p\right)\right) - T\left(x\right)\right]$

8. Compute Δp

9. Update the warp $W\left(x;p\right) \leftarrow W\left(x;p\right) \circ W\left(x;\Delta p\right)^{-1}$

untill $\|\Delta p\| \leq \epsilon$

3 Giving in the Features

The features are lost as the frame progresses and the original bounding box has to be reiterated to get the new features. This is overcome with the help of feeding the points again after a few frames so that the tracker and the bounding box is perfect.

4 Histogram Equalization for under the Bridge

1. Histogram Equalization was applied for tracking the car under the bridge.

2. This method helps in equalizing the darker intensities thereby being able to detect the car.

3. We observed that in two frames the bounding rectangle tracks slightly away from car motion, but gradually the change in parameters converges giving a lesser error value. Gamma correction, a method to minimize the darkness of an image is tried but the output is not as required.

5 Frame moving rapidly: Pyramid

1. The Pyramid concept is implemented when the video frames are under continuous swift motion (input video of vase in our case). The pyramid was implemented by decreasing the frame resolution by four times, and this is implemented in total three layers, the first layer is the normal layer, and the second layer and third layer have two times and four times lower resolution respectively.

2. These parameters are calculated from bottom to top layer and at every iteration, the bottom layer is warped to the layer above it with the revised parameters. Therefore, the parameters are upgraded to the first layer by taking in to consideration the rapid shifts in the lower resolution layers.

6 Pipeline

1. The algorithm is initialized by defining a template around the object to be tracked.

2. Parameters are updated in the current frame $(p + \Delta p)$

   - For the human this worked perfectly.
   - For the car, due to the sudden change in intensity, Histogram equalization is used. In addition to this, features are extracted at every 100^{th} frame again that are to be tracked in subsequent frames.

- For vase, to counteract swift movement between frames, we implemented Pyramid algorithm on every frame and then implemented the Lucas Kanade function

3. These parameters help draw the bounding box.

7 Analysis

1. 0.001, 0.001 and 0.0001 is used for Δp in human, car and vase as threshold.
2. Robust results are calculated from Lucas-Kanade Inverse Compositional Algorithm variation is used as on implementation.
3. For robustness, the features are fed in again after a couple of frames.
4. Three layer pyramid algorithm was applied for vase.
5. Histogram Equalization is used to apply the Lucas Kanade algorithm for darker images.

8 Results

8.1 Human

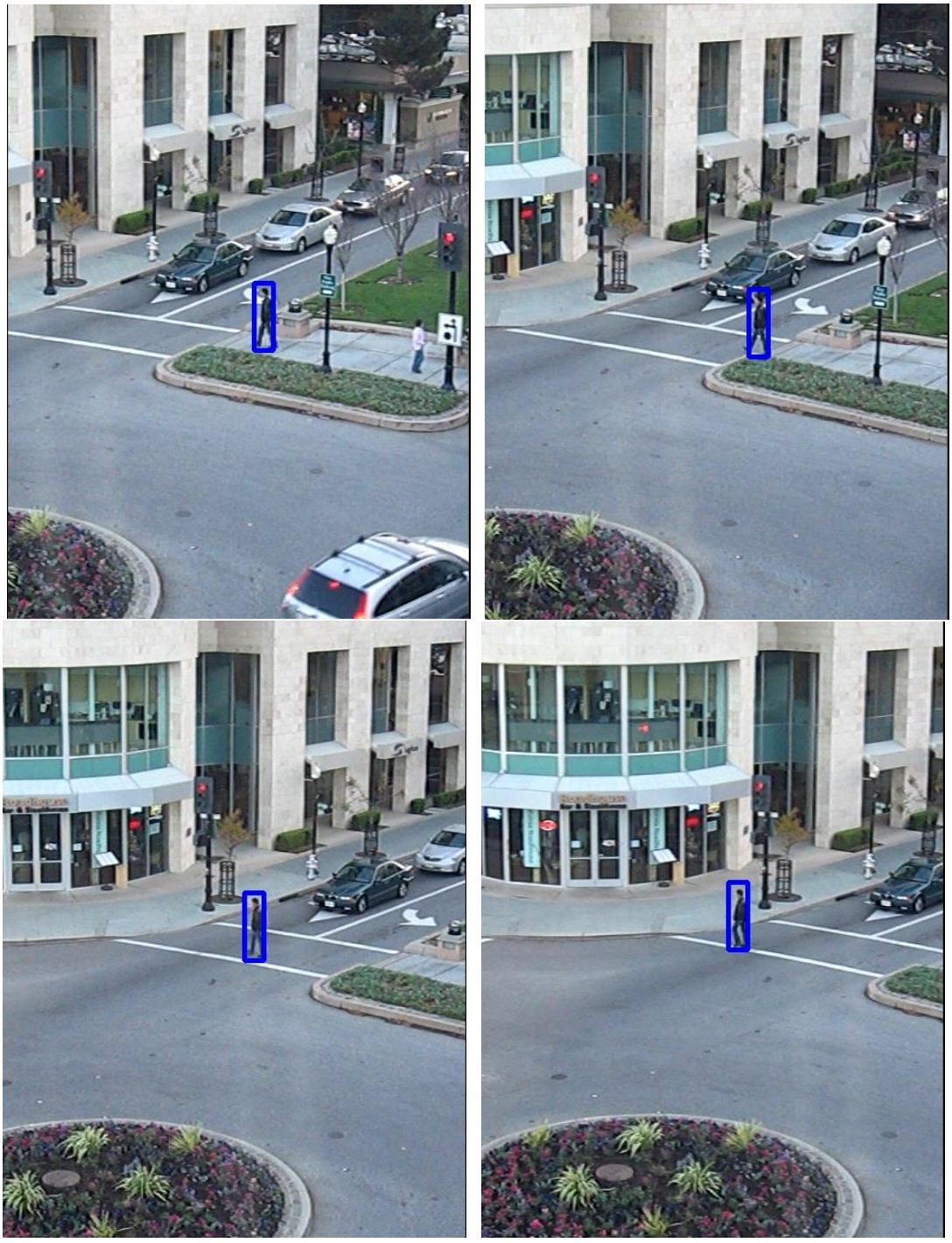

Figure 1: Human

8.2 Car

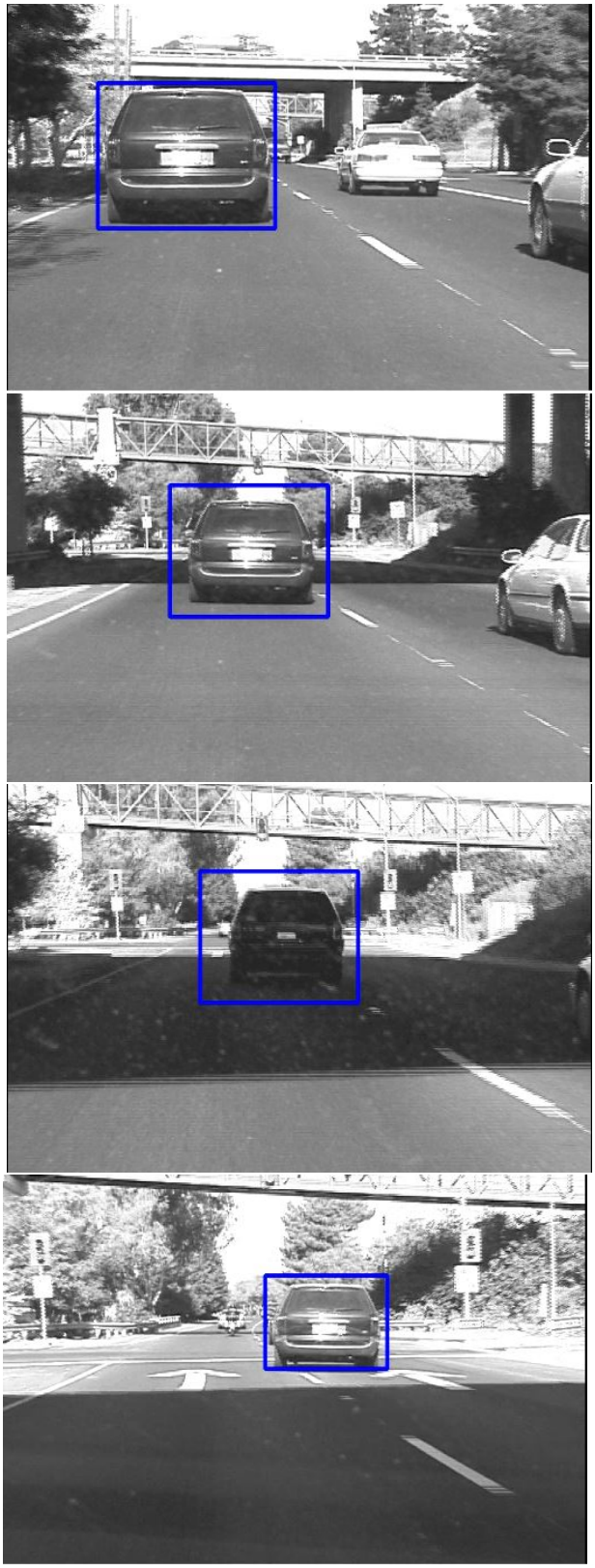

Figure 2: Car

8.3 Vase

Figure 3: Vase

9 References

1. Lucas-Kanade 20 Years On - Simon Baker and Iain Matthews - https://www.ri.cmu.edu/pub_files/pub3/ /baker_simon_2002_3.pdf

2. *lecture-tracking.pdf* from ENPM673 Coursework material

3. Lucas-Kanade Tracker (KLT) - Dr. Mubarak Shah - https://www.youtube.com/watch?v=tzO245uWQxA

4. OpenCV : Documentation - https://docs.opencv.org/3.0-rc1/d7/d8b/tutorial_py_lucas_kanade.html

5. Lucas Kanade Tracker - Jay Rambhia - https://jayrambhia.com/blog/lucas-kanade-tracker