1 Implementation of Lucas-Kanade (LK) template tracker

1.1 Introduction

In this project we tracked an object or a human being through out the video using Lucas-Kanade (LK) algorithm.

1.2 Lucas Kanade Algorithm

The goal of the Lucas-Kanade algorithm is to minimize the sum of squared error between two images, the template T and the image I warped back onto the coordinate frame of the template.

$$\sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x}) \right]^2$$

Figure 1: Mean Square error

Warping I back to compute I (W(x; p)) requires interpolating the image I at the sub-pixel locations W(x; p). The minimization in Equation (3) is performed with respect to p and the sum is performed over all of the pixels x in the template image T (x). Minimizing the expression in Equation (1) is a non-linear optimization task even if W(x; p) is linear in p because the pixel values I (x) are, in general, non-linear in x. In fact, the pixel values I (x) are essentially un-related to the pixel coordinates x. To optimize the expression in Equation (3), the Lucas-Kanade algorithm assumes that a current estimate of p is known and then iteratively solves for increments to the parameters Δp; i.e. the following expression is (approximately) minimized:

$$\sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) - T(\mathbf{x}) \right]^2$$

Figure 2: Warpped Mean Square error

with respect to Δp, and then the parameters are update

$$p = p + \Delta p \tag{1}$$

These two steps are iterated until the estimates of the parameters p converge. Typically the test for convergence is whether some norm of the vector Δp is below a threshold ε; i.e. $||\Delta p|| \leq \varepsilon$

1.3 Pipeline followed

1. First step is to crop the template out of the video which we want to track through out the video. The template ($T(x)$) is extracted from the first frame of every video. For this project the following are the templates considered.

Figure 3: Dragon Baby template

Figure 4: Car template

Figure 5: Bolt template

2. The next step is to perfrom affine transform that warps the current frame so that the template in the first frame is aligned with the warped current frame. The affine transform takes care of the change in scale of the template in the current frame. The obtained image is denoted by $I(W(x;p))$

3. The next step is to compute the error between the warped image and the template.

$$T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\,\mathbf{p}))$$

Figure 6: Error

4. The next step is computing the gradient ∇I of the warpped image $I(W(x;p))$. Gradient of the image is computed both w.r.t x and y using the Sobel filter.

$$\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) \tag{2}$$

5. Then the jacobian is computed at (x;p). The jacobian is computed for every point in the warpped image The jacobian is given by :

$$\frac{\partial W}{\partial p} = \begin{pmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{pmatrix} \tag{3}$$

6. The next step is to compute the steepest gradient descent. The steepest gradient is given by $\nabla I * \frac{\partial W}{\partial p}$:

7. Then we need to compute the Hessian matrix. The Hessian matrix is given by

$$H = \sum_x \left[\nabla I * \frac{\partial W}{\partial p}\right]^T * \left[\nabla I * \frac{\partial W}{\partial p}\right] \tag{4}$$

8. Finally computing Δp which is the updated parameters of the affine transformation matrix which shift the bounding box of the template to bound the object in the current frame.

$$\Delta p = H^{-1} \sum_x \left[\nabla I * \frac{\partial W}{\partial p}\right]^T * [T(x) - I(W(x;p))] \tag{5}$$

9. We need to iterate this process till Δp converges to a small threshold value.

The figure below summarizes the algorithm:

The Lucas-Kanade Algorithm

Iterate:

(1) Warp I with $\mathbf{W}(\mathbf{x};\mathbf{p})$ to compute $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$
(2) Compute the error image $T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))$
(3) Warp the gradient ∇I with $\mathbf{W}(\mathbf{x};\mathbf{p})$
(4) Evaluate the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at $(\mathbf{x};\mathbf{p})$
(5) Compute the steepest descent images $\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
(6) Compute the Hessian matrix using Equation (11)
(7) Compute $\sum_{\mathbf{x}} [\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}]^{\mathrm{T}} [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$
(8) Compute $\Delta \mathbf{p}$ using Equation (10)
(9) Update the parameters $\mathbf{p} \leftarrow \mathbf{p} + \Delta \mathbf{p}$

until $\|\Delta \mathbf{p}\| \leq \epsilon$

Figure 7: Lucas Kanade Algorithm

2 Output images by implementing Lucas Kanade

2.1 Tracking Bolt

Figure 8: Tracking Bolt

Figure 9: Tracking Car

Figure 10: Tracking Car

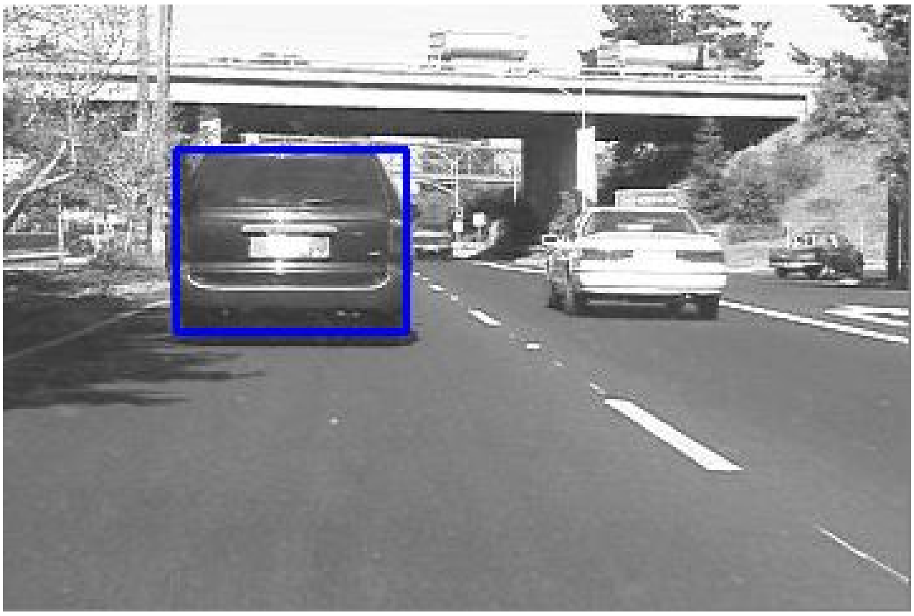

Figure 11: Tracking Car

Figure 12: Tracking Baby

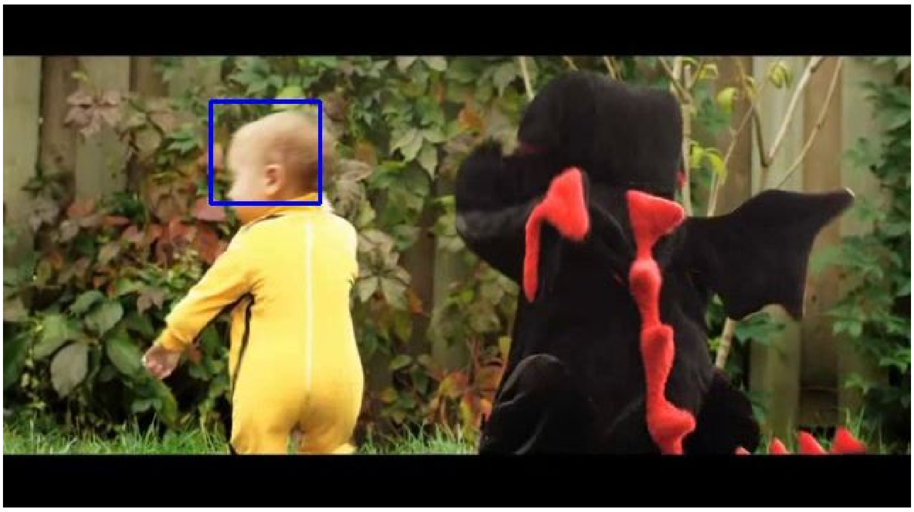

Figure 13: Tracking Baby

3 Evaluation of the tracker

The tracker breaks down in the case of car video because the intensity of the frame in the video changes continuously. In the case of the car video, when the car goes into the shadow, the tracker looses the car because the intensity level changes drastically between frames.

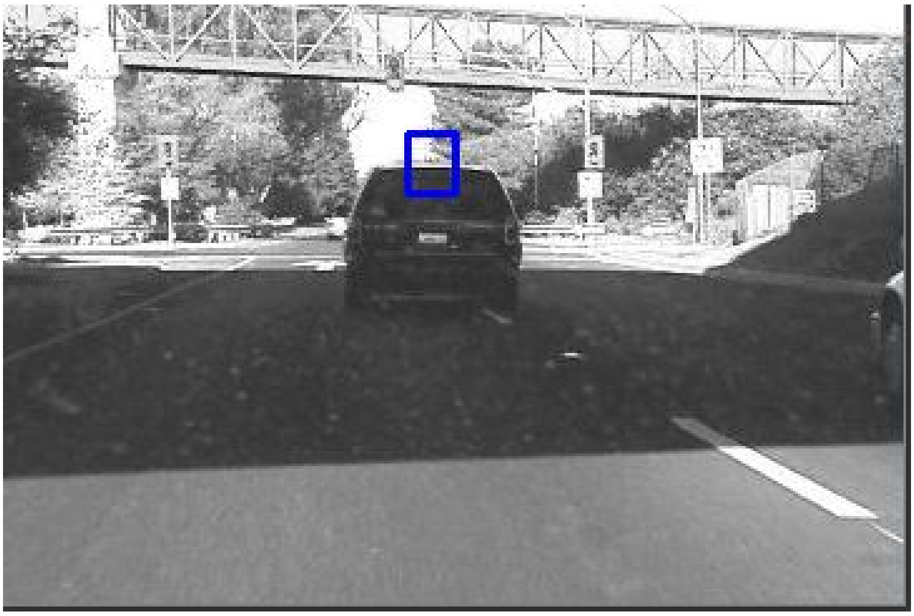

Figure 14: Tracking error

4 Robustness to Illumination

In order to increase the robustness of the tracker, we need to scale the brightness of pixels in each frame so that the average brightness of pixels in the tracked region stays the same as the average brightness of pixels in the template. This can be done by gamma correction.

After increasing the intensity of the frames in the video the following results we re obtained:

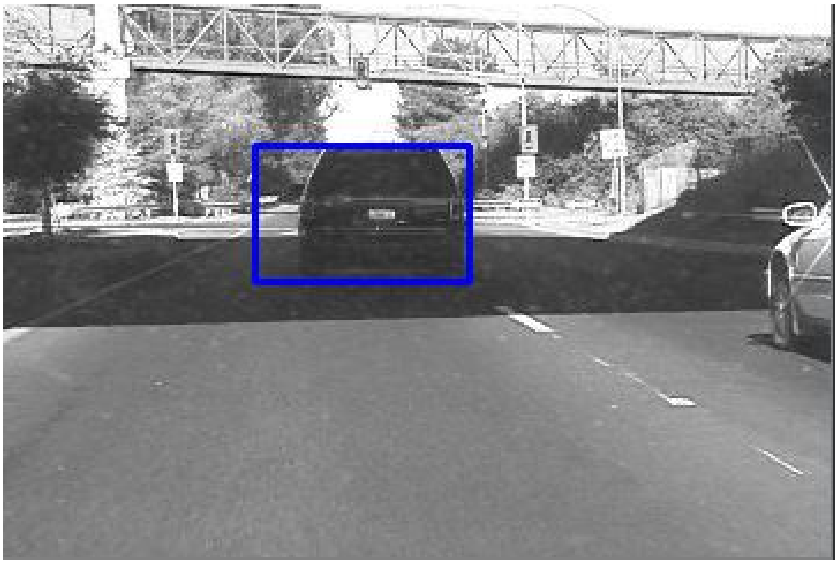

Figure 15: Tracking error

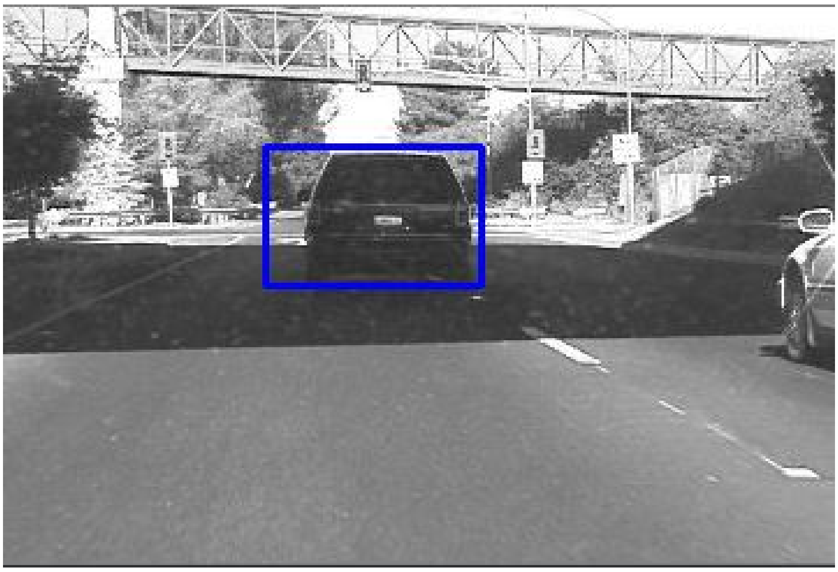

Figure 16: Tracking error

5 Challenges Faced:

Of course, Tweaking.

6 Team Members:

1. Eashwar Sathyamurthy 2. Akwasi A Obeng 3. Achal P Vyas

7 Output Videos:

To access output videos please use this link.