24785 Project Report
Template Alignment Optimization

Team 1
Harsh Dhruva, Ashwin Nehete, Bradley Feng, Siqiao Fu, Xiaoyang Zhan

December 10, 2021

Report Summary

1 Abstract

Image alignment is now a widely used technique in Computer Vision, with several applications, from optical flow and tracking to mosaicing. The Lucas-Kanade algorithm is a popular optical flow estimation algorithm, with several extensions used for Image alignment. Image alignment consists of moving and deforming a template to minimize the difference between the template and an image [1]. This sets up an interesting optimization problem, and an opportunity to evaluate the performance of different optimization techniques. We compile and discuss a study of different optimization techniques, to solve the image alignment problem, evaluated on a series of images, which will demonstrate frame by frame tracking of an object over several frames of a video sequence.

Keywords: Image alignment, Lucas-Kanade, Optimization

2 Introduction

In this project, we present a comparative analysis of different optimization techniques used to solve the Lucas-Kanade Image Alignment template tracking problem. The goal of the Lucas-Kanade algorithm is to minimize the sum of squared error between two images, the template and the Image that is warped back to the template [1]. This is a Non-Linear optimization problem that is solved using a Gauss-Newton gradient descent in the Lucas-Kanade framework. We explore the performance of solving this problem with techniques such as Gradient Descent, Newton's Method, Gauss-Newton, BFGS, and Levenberg-Marquardt. We compare the performance of the algorithm using a series of frames, such as a video, in which we will track a particular template object. The performance will be evaluated on criteria such as speed and accuracy of tracking, and the stability or robustness of the algorithm used for tracking.

The problem is non-trivial since minimizing the sum of squared error between two images is a non-linear optimization task. We will be using two different warp functions, a translational warp and an affine warp. Here the warp functions will be linear, although since the pixel values have no direct correlation to the pixel coordinates, the pixel values are non linear over the coordinate domain. To deal with the fractional movement of the template, we will choose an interpolation technique to use on the image. Lucas-Kanade assumes small motion between pixels, and in the optical flow sense, it assumes that the flow is constant in some local neighborhood. Other optical flow assumptions, such as Color Constancy, which allows for pixel to pixel comparison, and Brightness Constraint will also be implied. As multiple images are processed, speed is an important factor in a real world application. We will also explore the trade-off between speed and accuracy for the different optimization techniques.

In the actual use of Lucas-Kanade, the frame rate of the video and the resolution of each frame will affect the final effect of the algorithm. For example, for the same video, if one frame is extracted from each five frames, the difference between the two frames will be greater, the parameters that need to be solved will become larger, and the optimization problem will be more difficult to solve. The same video, the same size template, low resolution contains fewer pixels, provides less information, and the optimization problem will also be more difficult to solve.

3 Problem Statement

Objective Function: $$\min_{\mathbf{x}} \sum_{\mathbf{p}} [I(W(\mathbf{p};\mathbf{x})) - T(\mathbf{p})]^2 \tag{1}$$

where, $I(W(\mathbf{p};\mathbf{x}))$ is the warped image and $T(\mathbf{p})$ is the template image. $\mathbf{p} = \begin{bmatrix} p_i & p_j \end{bmatrix}^T$ represents the pixel coordinate of the image.

With respect to:

Translational warp parameters: $$\mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T \tag{2}$$

Affine warp parameters: $$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{bmatrix}^T \tag{3}$$

where warping is defined as follows for:

Translational Warp: $$\mathbf{W}(\mathbf{p};\mathbf{x}) = \begin{bmatrix} 1+x_1 & 0 & 0 \\ 0 & 1+x_2 & 0 \end{bmatrix} \begin{bmatrix} p_i \\ p_j \\ 1 \end{bmatrix} \tag{4}$$

Affine Warp: $$\mathbf{W}(\mathbf{p};\mathbf{x}) = \begin{bmatrix} 1+x_1 & x_3 & x_5 \\ x_2 & 1+x_4 & x_6 \end{bmatrix} \begin{bmatrix} p_i \\ p_j \\ 1 \end{bmatrix} \tag{5}$$

4 Analysis of Problem Statement

4.1 State variables

The variables in this problem is simply the warping parameters $\mathbf{X}$, which consists of 2 variables (translation warp) and 6 variables (affine warp). Explicitly, they are defined as follows:

Translational warp parameters: $$\mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T \tag{6}$$

Affine warp parameters: $$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{bmatrix}^T \tag{7}$$

In our cases, most of the pixel points, after being warped, will be within the domain of image function $I(x)$, or the interpolated spline of the Image function. For points that are on the edge or outside of the domain, the value of the pixel corresponds to the pixel value at the closest edge. The starting point is chosen within the feasible domain of the Image spline, as:

Translational warp parameters: $$\mathbf{x} = \begin{bmatrix} 0 & 0 \end{bmatrix}^T \tag{8}$$

Affine warp parameters: $$\mathbf{x} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{9}$$

4.2 Implicit functions

In the objective function shown in equation 1, the function $I(\mathbf{x})$ as well as $T(\mathbf{x})$ are cannot be written explicitly. They are pixel values of two images: input image and image template. As the warped pixel values are not guaranteed to warp on the discreet domain of the Image function, we will calculate the warped pixel values by Interpolation. We use the RectBiVariateSpline() function in Python's scipy library, which is a Bivariate spline approximation over a rectangular mesh.

4.3 Modeling assumptions

Small displacement of object between frames.

1. Brightness assumption : We assume that the brightness of original input image ($frame_{i+1}$) is the same as image template ($frame_i$). Mathematically, it could be described as:

$$I_{p_i}u + I_{p_j}v + I_t = \mathbf{0} \tag{10}$$

   where I_{p_i}, I_{p_j}, I_t are the derivative of image function w.r.t x,y and time (in the case where we are extracting several frames from a continuous video to perform alignment). u, v are the optical flow, or velocity of key points, or interesting points in the image

2. Smoothness assumption : We assume that there is no great and sudden change of pixel value in the image in a local region, therefore assuming the flow is locally smooth, which gives:

$$\nabla^2 u = \frac{\partial^2 u}{\partial p_i^2} + \frac{\partial^2 u}{\partial p_j^2} \tag{11}$$

$$\nabla^2 u = \frac{\partial^2 v}{\partial p_i^2} + \frac{\partial^2 v}{\partial p_j^2} \tag{12}$$

   We also assume that the object we are tracking is completely within the frame of the image, as the edge points will violate the smoothness assumption.

3. Small Object Displacement between frames : We assume that there is small motion between pixels in successive frames. As the velocity of the object is increased in the real world, the frame rate at which the video is sampled, will ensure that the displacement between frames is small.

4.4 Problem classification

Our problem is a **NLP** problem because the objective function is non-linear and the decision variables are continuous. Due to the modelling assumptions, our problem is locally convex because as the objective function should behave convex close to the actual object in the frame (as shown in Figure 6). Multiple local minimum will exist if the template is compared through the entire image. However, we also assume that there exists only a single object of which we are tracking, as multiple objects of the same type within the frame will not guarantee local convexity.

4.5 Evaluation metric

We choose Intersection over Union (IoU) as our evaluation metric. Any algorithm that provides predicted bounding boxes as output can be evaluated using IoU. In the numerator we compute the area of overlap between the predicted bounding box and the ground-truth bounding box. The denominator is the area of union, or more simply, the area encompassed by both the predicted bounding box and the ground-truth bounding box.

4.6 Potential difficulties

The image function I and image template function T are all discrete and they do not have mathematical expression. They could even be some random distribution of non-zero values. Therefore, that could bring difficulty on gradients: we cannot solve for gradients, as well as hessian analytically. We use forward difference to calculate the gradients and apply some methods to approximate Hessian matrix.

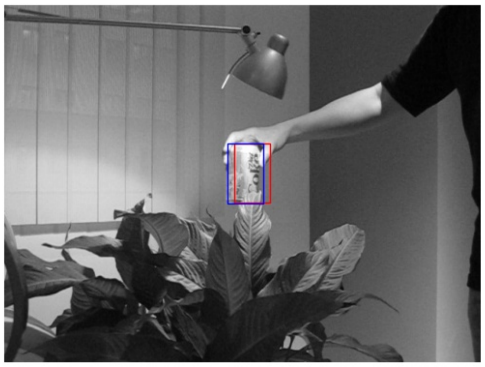
5 Optimization Study

5.1 Gradient Descent

Gradient Descent algorithm just takes a small step in the opposite direction of gradient. We have implemented it in Python by scratch, setting the threshold of the norm of solution to be 0.01 and the maximum number of iterations to be 1000. For Gradient Descent, the updating formula is:

$$\Delta x = -\alpha \nabla f = -2\alpha \nabla I_p \frac{\partial W}{\partial x} \Sigma I(W(x)) - T(p) \tag{13}$$

Since we were doing optimization for every 2 consecutive frame 289 (the last frame) frame here:


(a) Frame 35

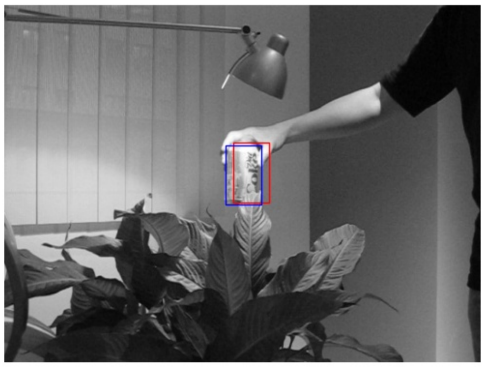

(b) Frame 40

Figure 1: Fast motion between frames where Gradient Descent successfully continues tracking

frame	x_0	x_1	obj
0	-0.0286	3.2631	123773
100	-1.1126	0.4043	611647
200	-14.1938	-3.4351	1510184
289	-0.0068	0.0561	77073
IOU	0.5023		

Table 1: Gradient Descent: Optimization results for frames 0, 100, 200, 289

5.2 Newton's Method

The Newton's method will iteratively find the minimizer of the second order approximation. We also implemented it in Python by scratch.

$$\Delta x = -\alpha \nabla^2 f^{-1} \nabla f^T \tag{14}$$

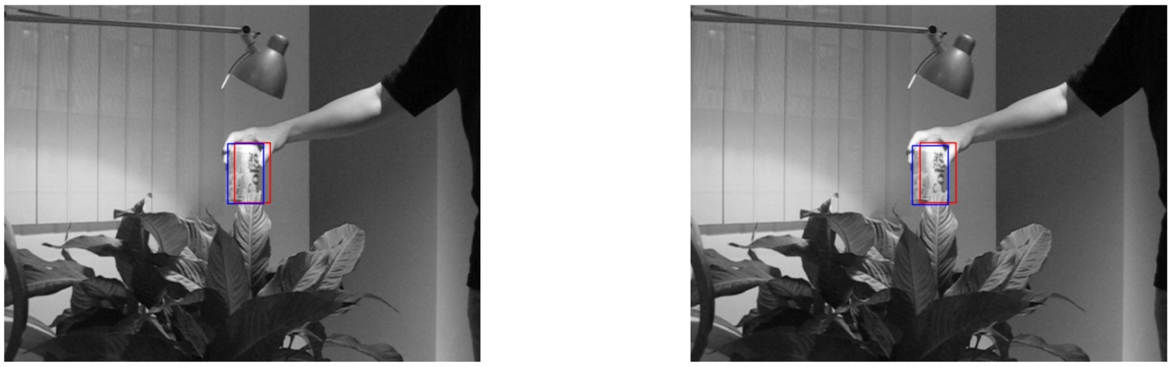
where:

$$\nabla f = -2\nabla I_p \frac{\partial W}{\partial x} \Sigma I(W(x)) - T(p) \tag{15}$$

and,

$$\nabla^2 f = (\nabla_p^2 I \frac{\partial W}{\partial x})^T \frac{\partial W}{\partial x} \tag{16}$$

The summarized solution is shown here:



(a) Frame 35

(b) Frame 40

Figure 2: Fast motion between frames where Newton's Method successfully continues tracking

frame	x_0	x_1	obj
0	-0.0260	3.2614	123827
100	-1.1421	0.4342	595471
200	-14.0713	-3.4099	1386422
289	6685	-9634	1587600
IOU	0.4411		

Table 2: Newton's Method: Optimization results for frames 0, 100, 200, 289

5.3 Gauss-Newton

Lucas Kanade method assumes a small and constant velocity between two nearby frames. We want to minimize the sum of squared error between template T and the second frame I warped back onto the template. Gauss-Newton is generally used to solve non-linear least squares problem. It is a modification of the Newton's method. This algorithm although has a slight advantage in the sense that Gauss-Newton does not require to compute the second order derivative ($\nabla^2 I$) which can be challenging to compute; also a computationally expensive task for the tracking as we run the optimization for each pair of frames in the video sequence.

The first order term for Taylor series expansion of objective function is calculated as:

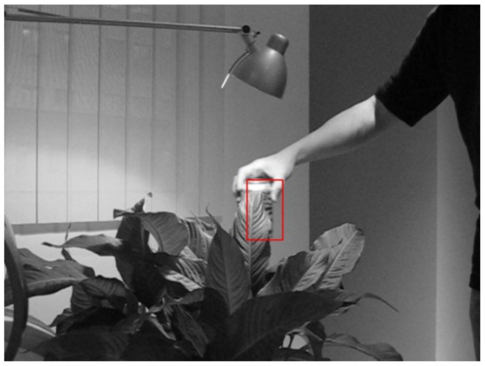
$$\sum_{p_i,p_j}\left[I(W(p_i,p_j;x)) + \nabla I\frac{\partial W}{\partial x}\Delta x - T(p_i,p_j)\right] \tag{17}$$

To demonstrate Lucas-Kanade using Gauss-Newton update step Δx as:

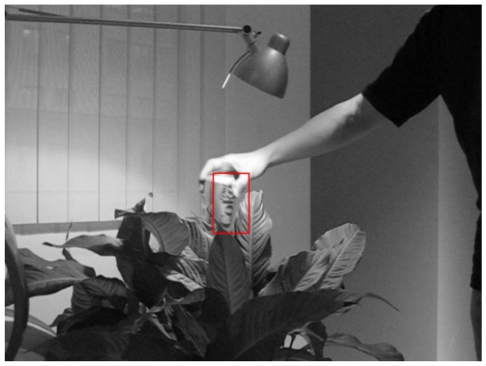
$$\Delta x = H^{-1}\sum_{p_i,p_j}\left[\nabla I\frac{\partial W}{\partial x}\right]^T[T(p_i,p_j) - I(W(p_i,p_j;x))] \tag{18}$$

where H is the approximation of Hessian matrix from the Gauss-Newton method:

$$H = \sum_{p_i,p_j}\left[\nabla I\frac{\partial W}{\partial x}\right]^T\left[\nabla I\frac{\partial W}{\partial x}\right] \tag{19}$$



(a) Frame 35



(b) Frame 40

Figure 3: Fast motion between frames where Gauss-Newton successfully continues tracking

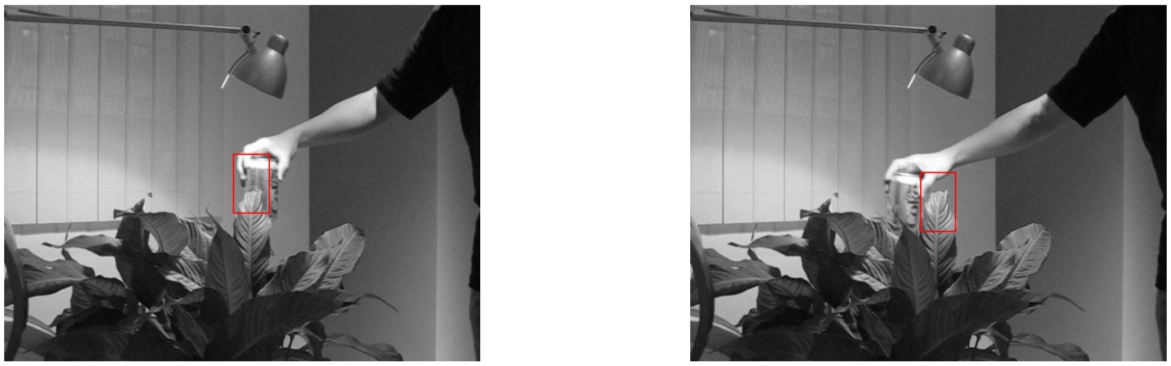
5.4 Quasi-Newton Methods (BFGS)

We implemented BFGS hessian approximation method used for solving unconstrained problems. It gradually improves the hessian approximation of the objective function to determine the descent direction. In our implementation, we used the scipy library function *scipy.optimize.minimize()* function and specified the *method='BFGS'*. In the python code, we define the objective function

frame	x_0	x_1	obj
0	-0.0272	3.2630	123779
100	-1.1807	0.4338	603369
200	-14.2366	-3.4434	1444108
289	-0.0153	0.0596	77455
IOU	0.5067		

Table 3: Gauss Newton: Optimization results for frames 0, 100, 200, 289

that takes in x, template, warped image as the input. The library returns the optimal values of the state variable x with the default $tol = 1e - 6$.
Interestingly, as shown in Figure 4 it was noted that BFGS fails to converge to minima and track the template when optimizing two consecutive frames that have fast motion of the object being tracked. In contrast to Gauss-Newton (shown in Figure 5) is able to successfully track the object.



(a) Frame 35

(b) Frame 40

Figure 4: Fast motion between frames where BFGS fails to track the object

We speculate that BFGS is unable to generate a good enough approximation of the hessian which does not allow it to achieve the minima, resulting in failure of tracking the object in subsequent frames.

frame	x_0	x_1	obj
0	-0.0276	3.2632	123770
100	0.0098	0.0296	48967
200	-0.0424	-0.0074	2268661
289	-0.0251	0.0033	55032
IOU	0.1301		

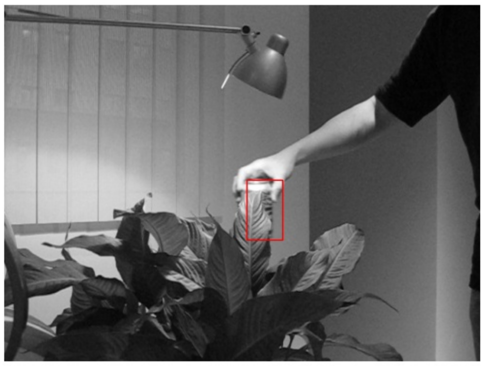
Table 4: BFGS: Optimization results for frames 0, 100, 200, 289

5.5 Levenberg-Marquardt

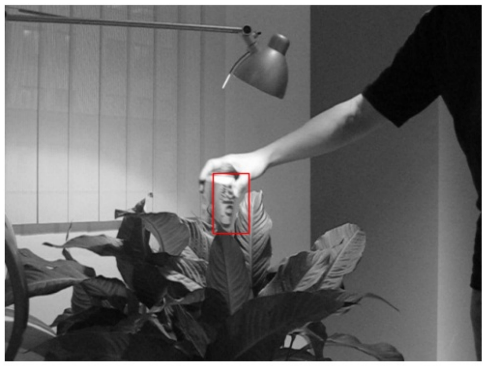
$$H_{LM} = \Sigma_{\mathbf{p}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{x}}\right]^T \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{x}}\right] + \delta \Sigma_{\mathbf{p}} \begin{pmatrix} \left(\nabla I \frac{\partial \mathbf{W}}{\partial x_1}\right)^2 & 0 & \cdots & 0 \\ 0 & \left(\nabla I \frac{\partial \mathbf{W}}{\partial x_2}\right)^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \left(\nabla I \frac{\partial \mathbf{W}}{\partial x_n}\right)^2 \end{pmatrix} \tag{20}$$

Of the various approximations, generally the steepest descent work better further away from the local minima, and the Newton and Gauss-Newton approximations work better close to the local minima where the quadratic approximation is good. Another aspect that none of the algorithms that we have discussed so far take into account is whether the error gets better or worse after each iteration. If the error gets worse, we need to try a smaller step in the same direction rather than making the error worse.

One algorithm that tries to combine the diagonal approximation with the full Hessian to get the best of both worlds is the well known Levenberg-Marquardt algorithm. For very small $\delta \ll 1$, the Levenberg-Marquardt Hessian H_{LM} is approximately the Gauss-Newton Hessian. For large $\delta \gg 1$, the Hessian is approximately the Gauss-Newton diagonal approximation to the Hessian, but with a reduced step size of $1/\delta$. It starts with a small initial value of δ. After each iteration, the parameters are provisionally updated and the new error evaluated. If the error has decreased, the value of delta is reduced, $\delta \to \delta/10$. If the error has increased, the provisional update to the parameters is reversed and δ increased, $\delta \to \delta \times 10$.



(a) Frame 35



(b) Frame 40

Figure 5: Fast motion between frames where Levenberg-Marquardt successfully continues tracking

The summarized solution is shown here:

frame	x_0	x_1	obj
0	-0.0272	3.2630	123780
100	-1.1610	0.4319	572155
200	-14.22	-3.4451	1467580
289	-0.0073	0.0589	75941
IOU	0.5038		

Table 5: Levenberg-Marquardt: Optimization results for frames 0, 100, 200, 289

5.6 Analysis of Solution

Most of our solutions are valid for both brightness and smoothness assumptions, except for BFGS solutions, and those failure cases at the final stage, because of the fully obstacle in front of our tracking object, which causes the objective function fails to describe the right object we want to track and does not converge to the template region.
Our solutions, except for the failure cases, absolutely makes sense. Since we assume small displacement between every 2 frames for objects, the optimized bounding box should have small warping parameters, which is exactly what's happening here: Although the objective function, mathematically speaking, is a non-convex function (as an example, Figure 6, 7 shows multiple local minima regions), the only optimal solution we care about is the one that generates new bounding box near the original bounding box. The other local minima do not correspond to the tracking object. Intuitively, the reasons that there are other local minimum being far away from original bounding box are:

1. The true solution, has some error since the bounding box does not exactly capture the object, and therefore due to the existing noise in each image frame the objective function value will not be zero.

2. There are some regions (image patches) in the image, whose pixel value distribution has low average error between our object image patch, although it does not look similar. In another words, in the "eyes" of objective function, there are multiple patches in the image that looks similar with our objective patch.

We were unable to compile metrics for comparing the speed of the algorithms, as we ran the code on multiple systems, each with different specifications. Intuitively, we observed the Gauss-Newton framework to be the fastest, and BFGS to be the slowest performing algorithm. Gradient descent and Levenberg Marquardt perform well in terms of IOU but the Gauss-Newton is the best performing algorithm in terms of the IOU and speed.

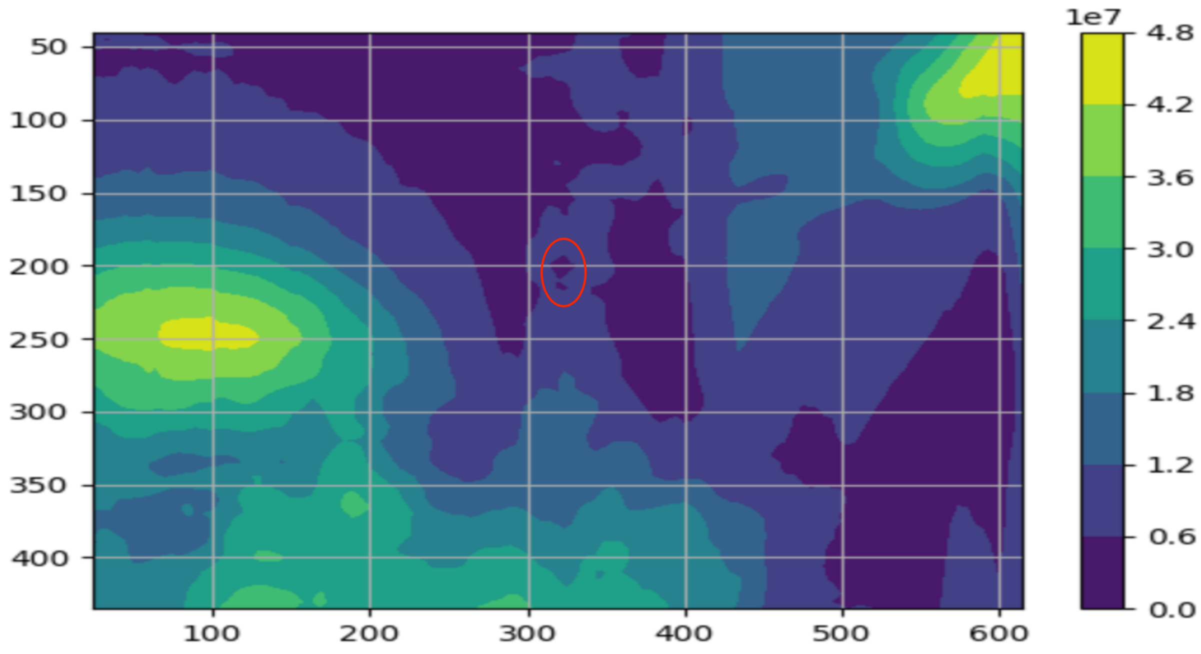


Figure 6: Objective Function for frame 0 & 1

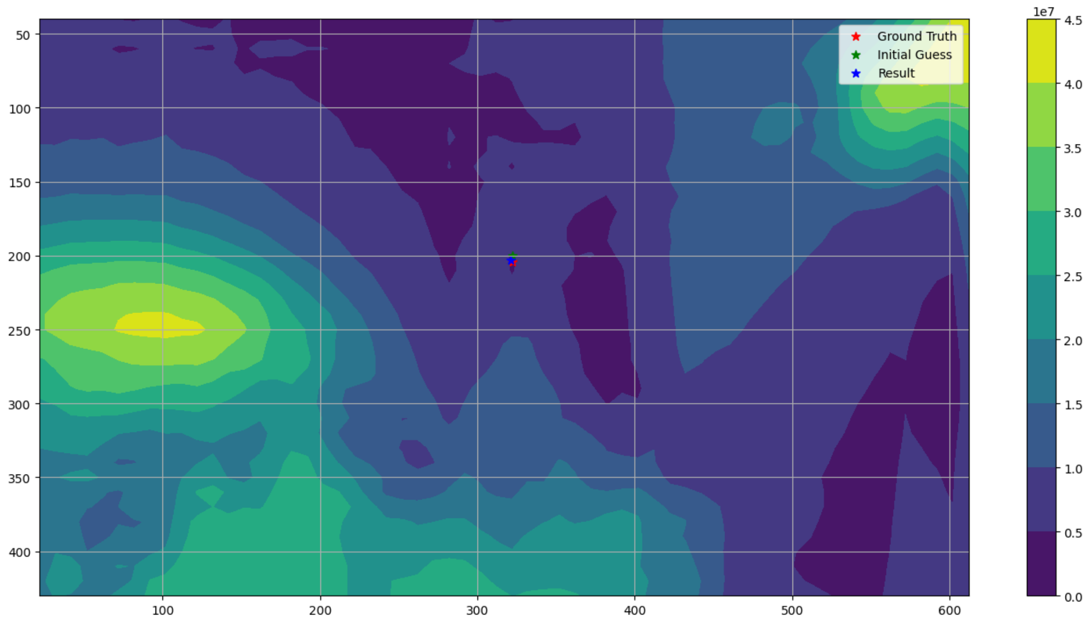


Figure 7: Objective Function for frame 0 & 1 marked with the ground truth, inital guess, final point

Analysis of Solution	
Local Optimality	We have got local optimality. See in figure 6/7, the problem is locally convex around the starting point (original bounding box)
Global Optimality	We can not guarantee to find the global minimum, but we do not need the global See in figure 6/7, the only solution that makes sense is the local optimum.
Uniqueness	There are other local minimums. They represent the positions where the pixel distribution looks very similar with the content we want to track. However, that is not meaningful solution. We only focus on neighbor regions of initial points, according to our assumptions

Table 6: Analysis of Solution

6 Conclusion

In this project, we compile and discuss a study of different optimization techniques, to solve the image alignment problem, evaluated on a series of images, which was demonstrated frame by frame tracking of an object over several frames of a video sequence. We were able to achieve good performance on the translation warp, however, in the affine case, our results were not up to par, and more work will need to be done in terms of template updates and potentially looking into adding bounding box constraints. Gauss-Newton is used in the original Lucas-Kanade template tracking because it is the best in terms of both speed and the accuracy based on our experiments and results.

7 Appendix

Link to the github repository: **24785-ProjectCV**
Youtube: Link to Project **Demo Videos**

References

[1] Lucas, Bruce D., and Takeo Kanade. ”An iterative image registration technique with an application to stereo vision.” 1981.

[2] Baker, Simon, and Iain Matthews. ”Lucas-kanade 20 years on: A unifying framework.” International journal of computer vision 56.3 (2004): 221-255.