COL780 Computer Vision
Assignment-2

Somanshu Singla 2018EE10314
Lakshya Kumar Tangri 2018EE10222

October 20, 2021

Template Tracking

Contents

1 Overview of Models Used

We have implemented these two algorithms using OpenCV for our tasks:

1. *Block Matching* : A block matching method divides a video frame into macroblocks and compares each macroblock to a template block using sum of squared distance (SSD) and normalised cross correlation (NCC) to find the macroblock which matches the most with the template[Wika]

   - SSD: Sum of squared differences (SSD) is one of measure of match that based on pixel by pixel intensity differences between the two images . It calculates the summation of squared for the product of pixels subtraction between two images

   $$SSD(x,y) = \sum_{x',y'} |T(\mathbf{x'},\ \mathbf{y'}) - I(\mathbf{x+x'},\ \mathbf{y+y'})|^2 \tag{1}$$

   - NCC:The term in the equation's numerator is nothing more than cross correlation between the template and picture. It should be noted that cross correlation cannot be utilised as a similarity metric since it would yield misleading template matching results. As a result, the term in the denominator is utilised as normalised cross correlation to obtain the proper match.

   $$NCC(x,y) = \frac{\sum_{x',y'}(I(\mathbf{x+x'},\ \mathbf{y+y'}) * T(\mathbf{x'},\ \mathbf{y'}))}{\sqrt{\sum_{x',y'} I(\mathbf{x+x'},\ \mathbf{y+y'})^2}\sqrt{\sum_{x',y'} T(\mathbf{x'},\ \mathbf{y'})^2}} \tag{2}$$

   here, I is the image and T is the template.

2. *Lucas Kanade Template Tracker*:Lucas Kanade algorithm is Gauss-Newton gradient descent non-linear optimization algorithm. We implement the Lucas Kanade with the inverse compositional algorithm for efficient image alignment and extending it to a consistent framework without significantly better efficiency [BGM03] The task is to find the transformation parameters which maximises the similarity between reference and subsequent frames of the tracked object. Hence minimizing the function

   $$\sum_{x} |I(\mathbf{W}(\mathbf{x},\mathbf{p}) - T(\mathbf{x})|^2 \tag{3}$$

   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 p i.e the following expression is (approximately) minimized

   $$\sum_{x} |I(\mathbf{W}(\mathbf{x},\mathbf{p}+\Delta\mathbf{p}) - T(\mathbf{x})|^2 \tag{4}$$

   with respect to $\Delta\mathbf{p}$, and then the parameters are updated

   $$\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p} \tag{5}$$

   These two steps are iterated until the estimates of the parameters $\mathbf{p}$ converge. Typically the test for convergence is whether some norm of the vector $||\Delta\mathbf{p}||$ is below a threshold ϵ

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 1: Algorithm For Lucas Kanade

2 Overview of Pre Processing Techniques Used

We have used the below mentioned techniques in our model:

1. *Filtering*: We have used Bilateral Filtering to reduce the noise in input images.
2. *Image Pyramid*: We have used the concept of Image Pyramid to take our image to a coarser scale so as to ignore noise as well as make motion smaller for Lucas Kanade to work [Gee]

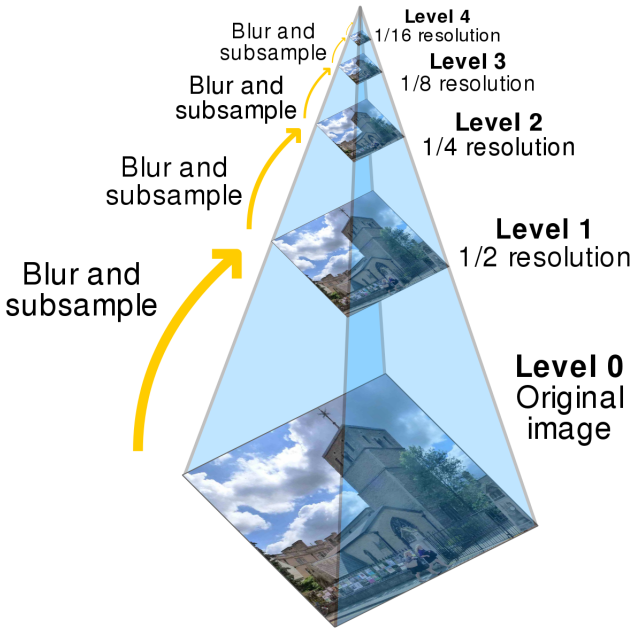


Figure 2: Image Pyramid [Wikb]

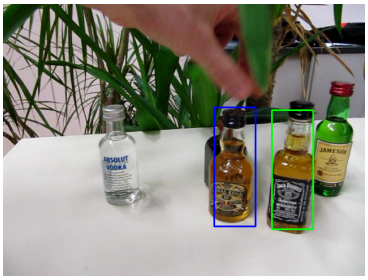
3 Block Matching

3.1 Implementation Details

We have used the inbuilt *matchTemplate* function in OpenCV varying its parameters related to criterion. Building upon this as base , we have added features like multiscale implementation and adaptive switch using which we can track features at multiple scales and also update our template frames after a certain threshold to take care of changing sizes/shapes of obejcts being tracked.

3.2 Problems with Block Matching

- Confusion between similar looking objects[1]: As the model is entirely based on appearance it doesn't really utilize the temporal consistency that the object can't move by a very large distance in a single frame.



(a) Liquor Data Set

(b) Bolt Data Set

Figure 3: Example of confusion between similar looking objects

- Difficulties with size variations: In this model we slide a fixed sized window over the image, so model can't cope up with the change in size of objects to be tracked. Even with the pyramidal approach model can't generalize to any random size.

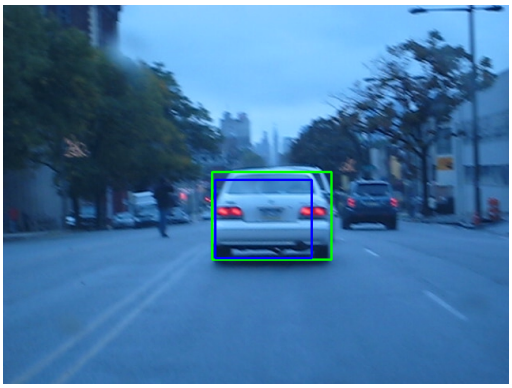
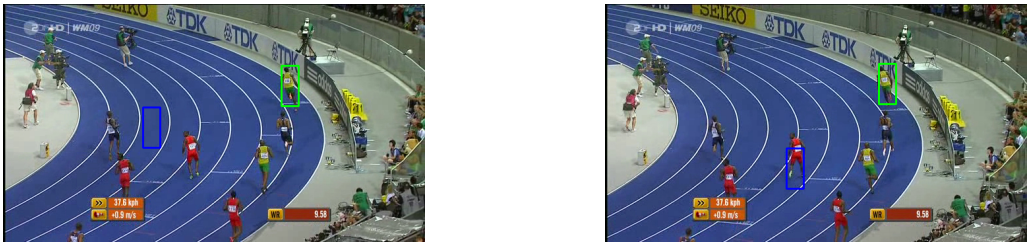


Figure 4: Example of failure to predict the correct size of template

[1]In all the examples blue bounding box depicts the prediction by the model and green bounding box depicts the ground truth for the given input image

- Appearance change of the object to track: In case there is an sudden appearance change of object to be tracked model can perform very poorly. In the Bolt dataset after sometime Bolt's back is visible instead of his face and the model totally fails there.



(a) Bolt Data Set (b) Bolt Data Set

Figure 5: Example of model performance on appearance change

3.3 Results

- mIOU Bolt: 0.0993 (NCC)
- mIOU BlurCar2: 0.6577 (NCC)
- mIOU Liquor: 0.6754 (NCC)

4 Lucas Kanade

4.1 Implementation Details

There are basic nine steps of applying the Lucas Kanade Algorithm as it can be seen in Figure 1.For each type of geometrical transformation the only step which changed was evaluating the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ as it changes for affine, projective and translation After computing warp W we compute its inverse on the four corners of bounding box on warped image to get the corresponding predicted positions on the image.We can also update our template frame in case of shape changes.

4.2 Results

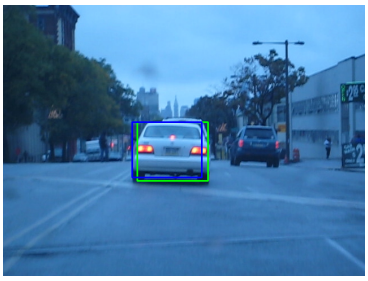
- mIOU Bolt: 0.1638 [Projective]
- mIOU BlurCar2: 0.3580 [Affine]
- mIOU Liquor: 0.5489 [Translation]

4.3 Comparison with Block Matching

We can easliy observe using the mIOU scores that the LK method has superior performance for Bolt dataset and worse performance in the BlurCar2 and Liquor dataset. The worse performance can be due to:

- In Bolt dataset although it improves upon the block matching the absolute performance is poor because human body can't be modelled by a geometric transformation
- In BlurCar2 dataset there are very sudden changes between even consecutive frames and we know LK algorithm can't handle very fast object dynamics

- In Liquor dataset similar objects between which model can confuse are lesser thus it doesn't really test the metrics where block matching can fail but it does have large movements which degrade the LK algorithm's performance.



(a) Liquor Data Set

(b) Bolt Data Set

Figure 6: Example of Large movement of object between two close frames that leads to failure in LK

Some Improvements:

1. In the bolt data set LK handles the confusion between similar looking objects better than block matching:

Figure 7: LK not confusing between similar objects

2. LK can conform to the bounding box size requirements better than block matching:

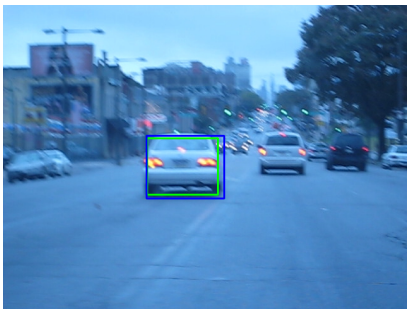


Figure 8: LK adjusting to size requirements better

Here, LK algorithm has increased the size in response to the object better than block matching which as we can see in 4 is not able to adjust to the bounding box size requirements.

5 Pyramid based Lucas Kanade

5.1 Implementation Details

Apart from the previous part we added list of scaled images and scaled templates for tracking and start from the coarsest resolution in hopes of reducing large pixel motion between frames to few pixels. The Warp parameters we get from this is then fed into Lucas Kanade algorithm as initialization for next finer resolution. This process is repeated till we get to the original size after which we get final parameters for geometric transformation. Finally we compute inverse of warp W on the four corners of bounding box to get the corresponding predicted positions on the image.

5.2 Results

- mIOU Bolt: 0.1638 [Projective]
- mIOU BlurCar2: 0.3718 [Affine]
- mIOU Liquor: 0.5490 [Translation]

5.3 Comparison with non pyramid LK

We can observe that mIOU scores have not changed by a lot, some reasons are:

- For Bolt Dataset, a human body movement can't be handled by a geometric transformation so increase pyramid levels didn't help much
- In BlurCar2 we can see there is a noticeable change in mIOU of about 0.02 because as we noted earlier there are large object movements so adding some pyramid levels can help in convergence.

6 Live Demo

6.1 Implementation Details

We have prepared live demo using both Lucas Kanade affine tracker and block matching as bl

References

[BGM03] Simon Baker, Ralph Gross, and Iain Matthews. "Lucas-Kanade 20 Years On: A Unifying Framework: Part 3". In: *Int. J. Comput. Vis* 56 (Dec. 2003).

[Gee] Geeks For Geeks. *Image Pyramid using OpenCV — Python*. Available at https://www.geeksforgeeks.org/image-pyramid-using-opencv-python/.

[Wika] Wikipedia. *Block Matching*. Available at https://en.wikipedia.org/wiki/Block-matching_algorithm.

[Wikb] Wikipedia. *Pyramid (image processing)*. Available at https://en.wikipedia.org/wiki/Pyramid_(image_processing).