Perception for Autonomous Robots-ENPM673

Report

By:

1. Sanket Acharya

2. Mithun Bharadwaj

3. Abhiram Dapke

INDEX

1. AIM OF THE PROJECT:

The aim of this project is to track the subject of the given databases using Lukas-Kanade tracker. This algorithm is usually used for optical flow but in this context can be used to track a template image by minimizing a loss function.

2. LUCAS-KANADE METHOD

The Lucas Kanade method, also known as sparse optical flow, calculates the displacement vectors of individual features rather than tracking all the pixels within a frame and rendering a full motion vector field. This method uses gradient information obtained from consecutive frames to search along the optimal path for a region that best matches the desired feature.

The Lucas-Kanade algorithm makes a “best guess” of the displacement of a neighborhood by looking at changes in pixel intensity which can be explained from the known intensity gradients of the image in that neighborhood. For a simple pixel, we have two unknowns (u and v) and one equation (that is, the system is underdetermined). We need a neighborhood to get more equations. Doing so makes the system overdetermined and we must find a least squares solution. The LSQ solution averages the optical flow guesses over a neighborhood. The Lucas-Kanade algorithm is an efficient method for obtaining optical flow information at interesting points in an image (i.e. those exhibiting enough intensity gradient information). It works for moderate object speeds.

3. DERIVATION OF LUCAS-KANADE ALGORITHM:

The Lucas-Kanade algorithm (which is a Gauss-Newton gradient descent non-linear optimization algorithm) is then derived as follows. The non-linear expression in Equation (4) is approximated by performing a first order Taylor expansion on I(W(x; p+p)) to give:

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

In the expression of delta I, delta I is computed in the coordinate frame of I and then warped back onto the coordinate frame of T using the current estimate of the warp.

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \begin{pmatrix} \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_n} \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_n} \end{pmatrix}.$$

We follow the notational convention that the partial derivatives with respect to a column vector are laid out as a row vector. This convention has the advantage that the chain rule results in a matrix multiplication. For example, the affine warp in Equation (2) has the Jacobian:

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

Equation (1) is a least squares problem and has a closed from solution which can be derived as follows. The partial derivative of the expression in Equation (1) with respect to p is:

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

Setting this expression to equal zero and solving gives the closed form solution of Equation (1) as:

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

where is the (Gauss-Newton approximation to the) *Hessian* matrix:

$$H = \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right].$$

4. IMAGE ALIGNMENT:

Image alignment (also known as image registration) is the technique of warping one image (or sometimes both images) so that the features in the two images line up perfectly.

In many applications, we have two images of the same scene or the same document, but they are not aligned. In other words, if you pick a feature (say a corner) on one image, the coordinates of the same corner in the other image is very different.

Applications of Image Alignment:

Image alignment has numerous applications.

In many document processing applications, the first step is to align the scanned or photographed document to a template. For example, if you want to write an automatic form reader, it is a good idea to first align the form to its template and then read the fields based on a fixed location in the template.

In some medical applications, multiple scans of a tissue may be taken at slightly different times and the two images are registered using a combination of techniques.

The most interesting application of image alignment is perhaps creating panoramas. In this case the two images are not that of a plane but that of a 3D scene. In general, 3D alignment requires depth information. However, when the two images are taken by rotating the camera about its optical axis (as in the case of panoramas), we can use the technique described in this tutorial to align two images of a panorama.

5. IMAGE WARPING:

Image warping is the process of digitally manipulating an image such that any shapes portrayed in the image have been significantly distorted. Warping may be used for correcting image distortion as well as for creative purposes.

6. TRACKING VS. DETECTION:

Usually tracking algorithms are faster than detection algorithms. The reason is simple. When you are tracking an object that was detected in the previous frame, you know a lot about the appearance of the object. You also know the location in the previous frame and the direction and speed of its motion. So in the next frame, you can use all this information to predict the location of the object in the next frame and do a small search around the expected location of the object to accurately locate the object. A good tracking algorithm will use all information it has about the object up to that point while a detection algorithm always starts from scratch.

Therefore, while designing an efficient system usually an object detection is run on every nth frame while the tracking algorithm is employed in the n-1 frames in between. Why don't we simply detect the object in the first frame and track subsequently? It is true that tracking benefits from the extra information it has, but you can also lose track of an object when they go behind an obstacle for an extended period of time or if they move so fast that the tracking algorithm cannot catch up. It is also common for tracking algorithms to accumulate errors and the bounding box tracking the object slowly drifts away from the object it is tracking. To fix these problems with tracking algorithms, a detection algorithm is run every so often. Detection algorithms are trained on a large number of examples of the object. They, therefore, have more knowledge about the general class of the object. On the other hand, tracking algorithms know more about the specific instance of the class they are tracking.

7.TYPES OF ALGORITHMS:

1. Steepest descent algorithm:

In mathematics, the method of steepest descent or stationary-phase method or saddle-point method is an extension of Laplace's method for approximating an integral, where one deforms a contour integral in the complex plane to pass near a stationary point (saddle point), in roughly the direction of steepest descent or stationary phase. The saddle-point approximation is used with integrals in the complex plane, whereas Laplace's method is used with real integrals.

2. Gauss-Newton algorithm:

The method of steepest descent or stationary-phase method or saddle-point method is an extension of Laplace's method for approximating an integral, where one deforms a contour integral in the complex plane to pass near a stationary point (saddle point), in roughly the direction of steepest descent or stationary phase. The saddle-point approximation is used with integrals in the complex plane, whereas Laplace's method is used with real integrals.

3. Levenberg-Marquardt algorithm:

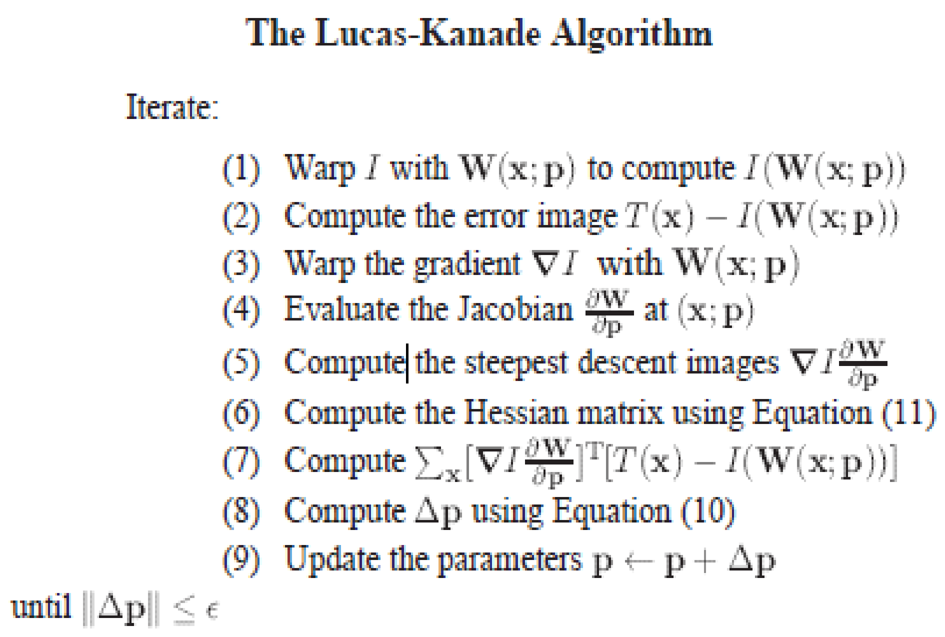
The Levenberg–Marquardt algorithm (LMA or just LM), also known as the damped least-squares (DLS) method, is used to solve non-linear least squares problems. These minimization problems arise especially in least squares curve fitting.

The LMA is used in many software applications for solving generic curve-fitting problems. However, as with many fitting algorithms, the LMA finds only a local minimum, which is not necessarily the global minimum. The LMA interpolates between the Gauss–Newton algorithm (GNA) and the method of gradient descent. The LMA is more robust than the GNA, which means that in many cases it finds a solution even if it starts very far off the final minimum. For well-behaved functions and reasonable starting parameters, the LMA tends to be a bit slower than the GNA. LMA can also be viewed as Gauss–Newton using a trust region approach.

8. PROCEDURE:

- Start off with an estimate for the parameters for affine transformation. Affine transformation is chosen to track objects where rotation is involved. If the object just moves in the frame without rotation, translation parameters are sufficient to track the object. It significantly increases the speed of computation as we need to compute only 2 parameters instead of 6.
- Identify the template image from the initial frame. This is necessary to specify the object to be tracked.
- Taking the template frame and the next frame, we need to minimize a least square loss function but since this function is non-linear because of the pixel intensities, it is not an ideal optimization problem.
- We employ an iterative method as shown in the figure below to estimate the parameters of the affine transformation.
- The convergence or stopping condition of this algorithm is decided by a user defined threshold. We stop the iterations when the norm of Δp is less than the defined threshold.

- It’s then possible to estimate the new points of the bounding box once the transformation parameters have been found.



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 $\nabla I$ 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 $\|\Delta \mathbf{p}\| \leq \epsilon$
```

Iterative procedure for Lucas-Kenade tracker

9. SCREENSHOTS OF OUTPUTS:

1. Car Tracking

FIGURE: Input image of the car

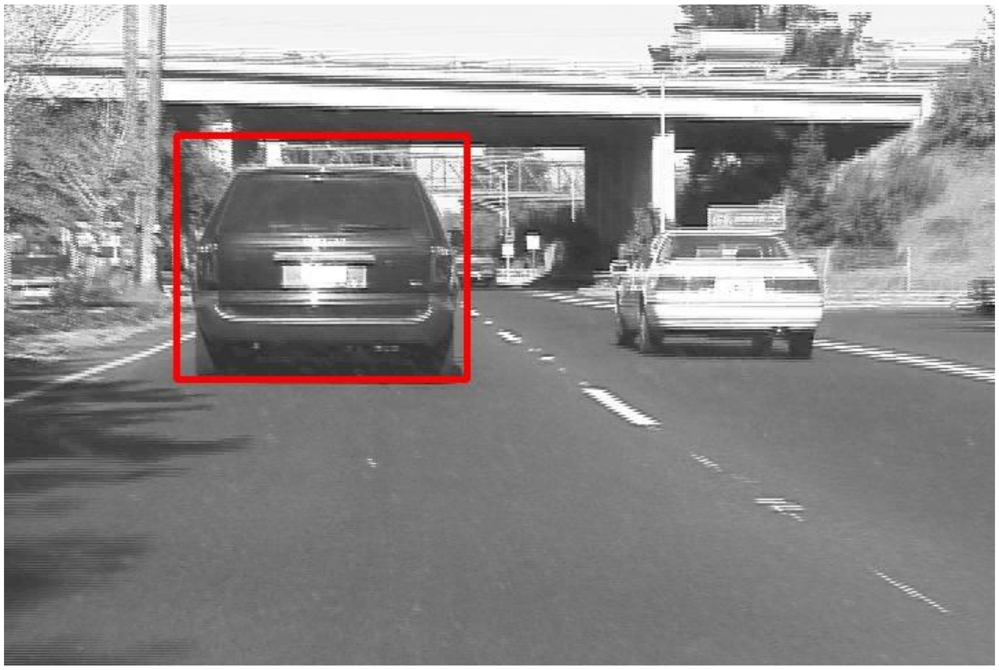


FIGURE: Output image of the car

FIGURE: Input image of the car

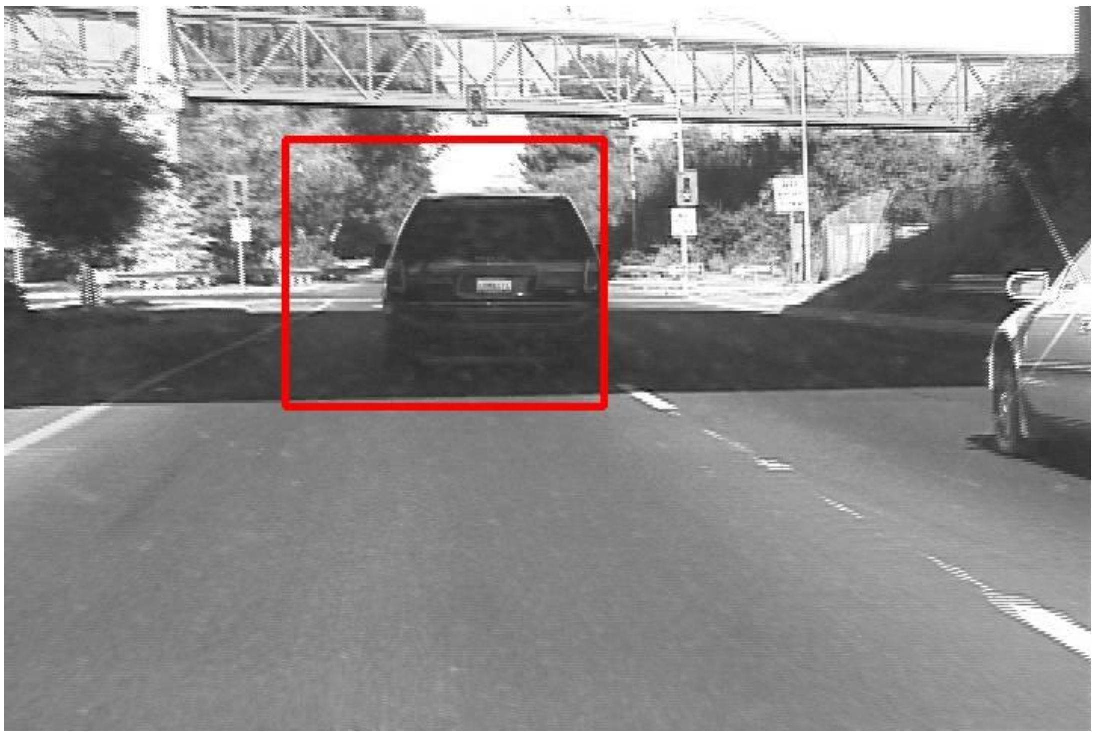


FIGURE: Output image of the car

2. Human Tracking

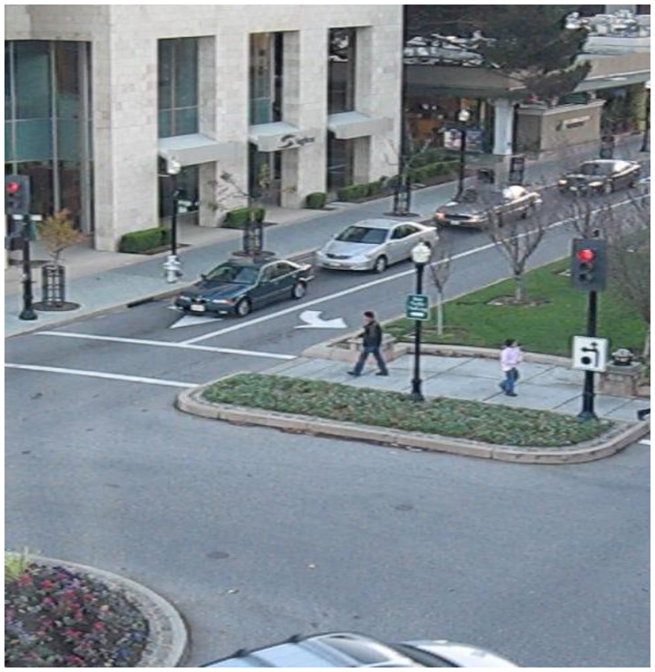


FIGURE: Input image of the human

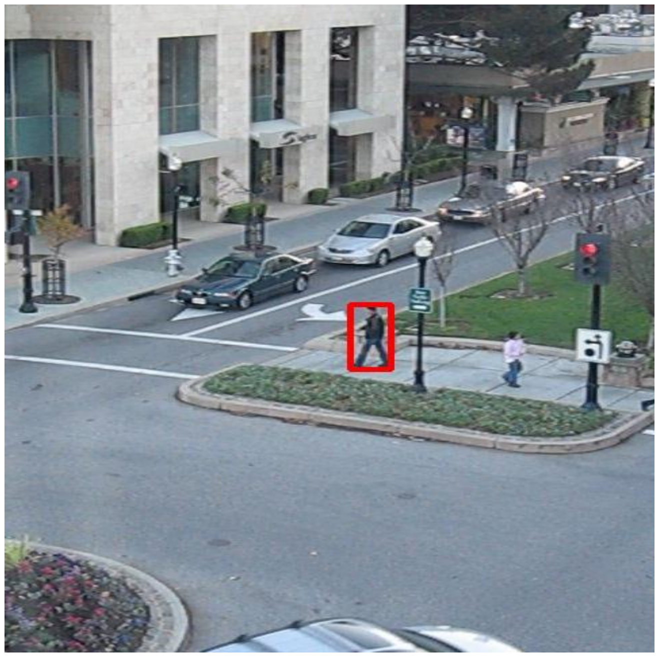


FIGURE: Output image of the human

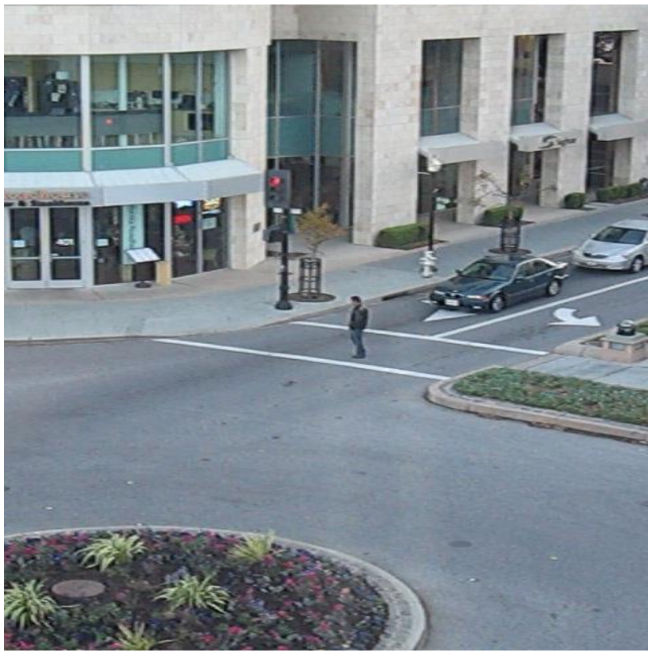


FIGURE: Input image of the human

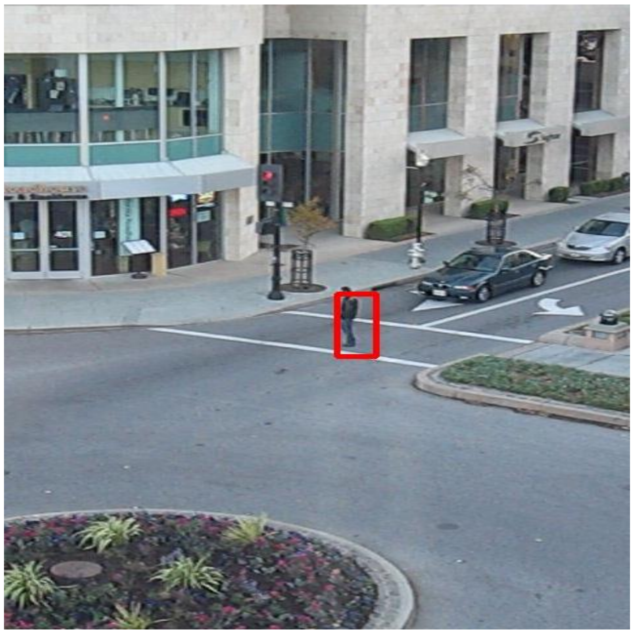


FIGURE: Output image of the human

3. Vase tracking

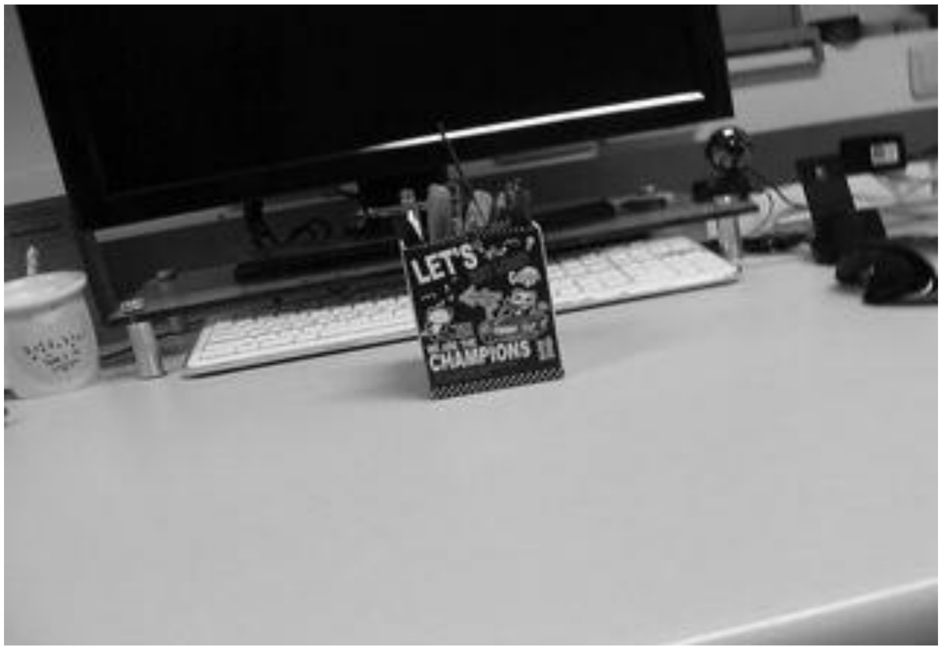


FIGURE: Input image of a vase

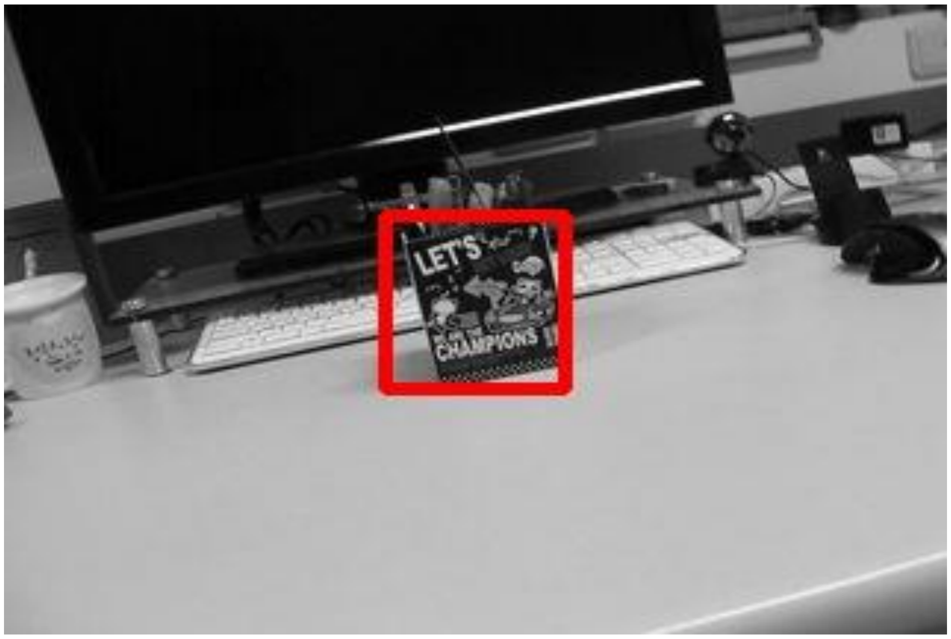


FIGURE: Output image of a vase

FIGURE: Input image of a vase

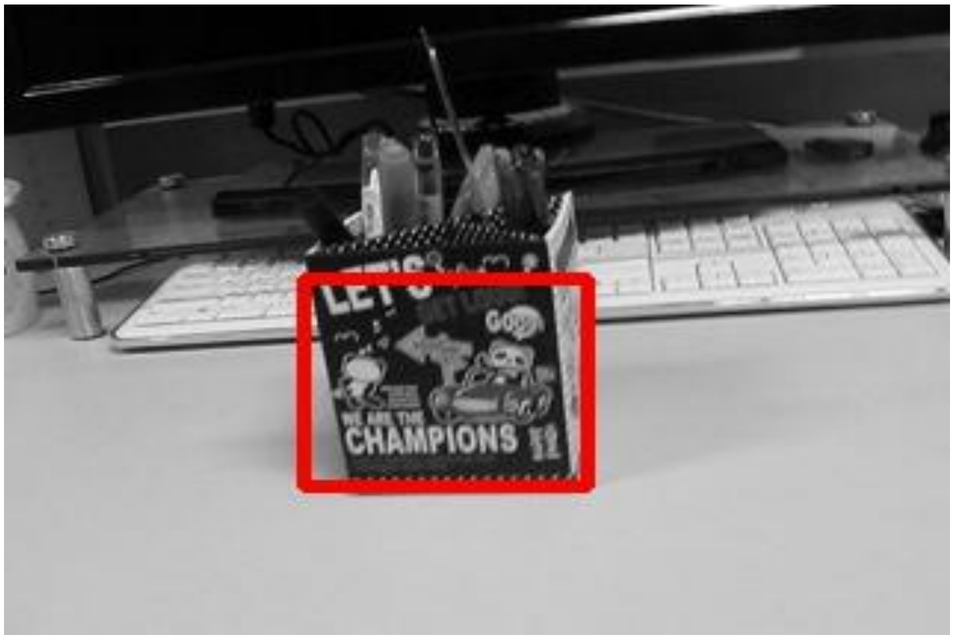


FIGURE: Output image of a vase

10. LIMITATIONS OF LUCAS-KANADE ALGORITHM:

1. The motion is large (larger than a pixel) – Taylor doesn’t hold

2. Non-linear: Iterative refinement

3. Local minima: coarse-to-fine estimation

11. REFERENCES:

1. Research paper – “Lucas-Kanade 20Years on: A Unifying Framework”

2. Computer Vision: Algorithms and Applications. (Richard Szeliski)