ENPM 673 Project 4

Daniel Sahu
Brenda Scheufele

20 April 2020

1 Data Preparation

To initialize the Lucas Kanade (LK) tracker, a template is defined by drawing a bounding box around the object to be tracked in the first frame of the video. We manually selected the points that fit the object without a lot of background included. The image is also converted to grayscale before further processing. The bounding box coordinates (x, y, width, height), where (x,y) is the upper left corner of the bounding box, are:

1. Car = [70,51,107,87]
2. Baby = [160,73,56,71]
3. Bolt = [265,85,40,58]

The templates for the data sets are shown in Figure 1.

Figure 1: Data set Templates: (a) Baby, (b) Car, (c) Bolt

(a) (b) (c)

2 Implementation of the Tracker

After defining an initial bounding box for each data set, the basic Lucas Kanade tracker was implemented. We defined a LucasKanade class which included defining the template, calculating the Jacobian and a method which computed the LK iteration.

The algorithm is similar to that given in the Simon and Matthew paper, e.g.:

1. Form W[x,y] from the affine transformation P
2. Warp I to get I(W([x,y];P)
3. Compute error between the template and image: T(x,y) - I(W([x,y];P))
4. Warp the gradient ∇I with W([x,y];P)
5. Evaluate $\partial W/\partial P$ at W([x,y];P) (Jacobian)
6. Compute steepest descent $\nabla I \partial W/\partial P$
7. Compute the 'Hessian', $\Sigma(\nabla I \partial W/\partial P)^T(\nabla I \partial W/\partial P)$
8. Compute $\Sigma(\nabla I \partial W/\partial P)^T(T(x,y) - I(W([x,y]);P)$
9. Compute ΔP
10. Update $P \leftarrow P + \Delta P$

We take advantage of a number of facts to simplify the pipeline, such as the fact that the Jacobian of an affine transform is constant and the gradient for each frame can be pre-computed. In pseudocode our implementation is:

```
# jacobian, template are computed once per dataset
p = pprev
grad = gradient(frame)
while norm(dP-dP_prev) > threshold:
    # warp frame with current estimate
    I = warp(frame, p)
    # calculate current error (difference from template)
    error = I - template
    # warp gradient
    I_g = warp(grad, p)
    # calculate deepest descent
    descent = I*jacobian
    # calculate hessian
```

```
hessian = transpose(descent)*descent
# calculate change in affine transform
dP = inv(hessian)*(descent*error)
```

The LK tracker pre-computes a perspective transform (really, an affine) for the template image's given bounding box. It also calculates the Jacobian corresponding to the template frame size. Then, each successive call will iteratively compute and return the new warping parameters based on the previous frame's estimate (starting with identity).

This pseudocode picture leaves out a lot of significant details. It is critical to note the following:

1. The computation for each frame is performed after first projecting all relevant variables into the Bounding Box of the initial template. This means all error / descent computations only deal with our region of interest, not the frame as a whole.
2. Most calculations (e.g. Hessian, Descent) are calculated on a per-pixel basis and then summed.
3. This is strictly an implementation detail, but it was important to be cognizant of the differences between 'numpy' and 'cv2' with regards to row vs. column ordering.
4. We implemented several improvements to the base Lucas Kanade tracker during evaluation. These include:
   (a) Illumination Scaling: We scale the illumination of each frame's ROI to match the template illumination.
   (b) Huber Loss: We replaced the default Least Squares estimator with a piecewise Huber Loss function. This scales the residual to reduce the impact of outliers.
   (c) Convergence Criterion: Our convergence criterion consists of checks against the change of dP from iteration to iteration (to make sure we're converging) and a minimum number of iterations. Ideally we would also have a check against the norm of dP, but we have no good heuristic for that, since the range of datasets contain wildly different transform changes from frame to frame.

3 Evaluation of the Tracker

Each data set presented distinct challenges for our tracker, and suggested different potential improvements. Our results can be found at this Google drive.

3.1 Car

We generally performed well with the starting sequence of the Car dataset, which has a high framerate and relatively consistent motion. Things only started to diverge once we experienced significant lighting changes. As per the suggestion in the problem statement we added an averaging function over the intensity to reduce the differences in illumination as the car passes under the bridge. This helped considerably, and the car could be tracked under the bridge.

The latter half of the video requires tracking the car much farther away than in the template. We modified the Lucas Kanade tracker to use a Huber Loss estimator instead of Least Square. With a Huber Loss parameter of $\sigma = 0.8$, the car was tracked throughout the video. However, the performance was very sensitive to the value of σ; for values above 0.8 or below 0.75, track would be lost.

3.2 DragonBaby

For the Baby data set, the illumination looked constant, however, the motion was much faster than in the car video, and our tracker could not keep up. The key challenge for tracking appeared to be the amount of motion from frame to frame and occasional occlusion. The rapid rotation of the subject's face also presented problems, and suggests that we could've achieved better performance with a Perspective Transform instead of an Affine.

3.3 Bolt

The Bolt dataset was difficult because the template features we were tracking are not consistent from frame to frame (i.e. the relative position of the subject's legs and arms are in flux). This results in a significantly more difficult problem from the offset, because we cannot expect a "good" match between frame and template, even in a non-occluded frame. To that end we implemented a moving average filter over several frames to prevent a single bad frame to derail the tracker.

3.4 Conclusion

This project drives home the difficulty of implementing a "one size fits all" tracker for datasets with varying framerates, motion, occlusion, and lighting conditions. We were able to do well when things were relatively slow, like in the car dataset, but the performance of our tracker quickly degraded in the presence of large object changes from frame to frame. This reflects the fundamental assumption behind the Lucas-Kanade tracker: that the frame-to-frame differences in the tracked object are relatively small. As expected, our implementation does better when this assumption hold true.

Our results can be found at this Google drive. Our source code can be found at this GitHub repository.