CHUN Hiu Sang `hschun`
TSE Ho Nam `hntse` **Homework 7 Write-up** COMP 4901L
December 4, 2019

1 Theory Questions

Q1.1 Calculating the Jacobian

$$L = \sum_x [T(x) - I(W(x;p))]^2$$

$$\begin{aligned}\mathbf{J} = \frac{\partial L}{\partial \mathbf{p}} &= \frac{\partial L}{\partial I(W(x;p))}\frac{\partial I(W(x;p))}{\partial W(x;p)}\frac{\partial W(x;p)}{\partial \mathbf{p}} \\ &= (\sum_x 2[I(W(x;p)) - T(x)]) \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{bmatrix}\end{aligned}$$

Q1.2 Computational complexity

In Inverse Compositional method:

- Computational complexity for precomputations is
  $2O(n)$ (x and y gradients of template) + $O(n \cdot 2p)$ (jacobian) + $O(n \cdot 2p)$ (for $\nabla T \frac{\partial W}{\partial p}$)
  + $O(n \cdot p^2)$ (hessian) + $O(p^3)$ (inverse of hessian)
  $= O(n \cdot p^2)$ (assuming $p < n$)
- Computational complexity of each runtime iteration is
  $O(3^2 \cdot n + n)$ (warping image and sampling) + $O(n)$ (error image)+ $O(n \cdot p)$
  (for $\sum_x [\nabla T \frac{\partial W}{\partial p}]^\top [I(W(x;p)) - T(x)]$) + $O(p^2)$ (for Δp) + $O(3^3 + 3^3)$ (inverse of warp and update of warp)
  $= O(n \cdot p)$ (assuming $p < n$)

In Lucas-Kanade method:

- Computational complexity of each runtime iteration is
  $O(n + n)$ (warping image and sampling) + $O(n)$ (error image) + $O(2n)$ (warping gradient of image) + 0 (skipped as jacobian is reduced to identity transform) + 0 (for $\nabla I \frac{\partial W}{\partial p}$, skipped too) + $O(n \cdot p^2)$ (hessian) + $O(p^3)$ (inverse of hessian) + $O(n \cdot p)$ (for $\sum_x [\nabla I \frac{\partial W}{\partial p}]^\top [T(x) - I(W(x;p))]$) + $O(p^2)$ (for Δp) + $O(p)$ (updating parameters)
  $= O(n \cdot p^2)$ (assuming $p < n$)

Comparing the two methods, the Inverse Compositional method runs faster because the computational cost per iteration is lower, and the pre-computation cost is charged once on initialization only, making it more efficient.

2 Lucas-Kanade Tracker

Q2.4 Tracking a Car

Note that the demo scripts are `matlab/lk_demo.m` and `matlab/mb_demo.m` respectively.

Initialization coordinates of LK tracker for car: `tracker = [168 145 295-168 255-145];`
Initialization coordinates of MB tracker for car: `tracker = [124 143 337-124 275-143];`

For the LK tracker, we can see it does pretty well if the initial bounding box is set properly. It is set such that the size matches the last frame as the car gets smaller. If we set the bounding box as large as the first frame, both trackers will lose track quite soon as the bounding box will soon contain information about the irrelevant backgrounds. Though as the car goes through the shadow, the tracker slides behind a bit that at the end the bounding box does not include the whole care. This is because the standard LK tracker is not illumination invariant.

For the MB tracker, templates that just cover the back of the car with less of the background tend to work better. The tracker could break down starting from the point when the car goes under the bridge - if not tuned right. The reason is similar to the LK tracker: though being more computationally efficient, the MB tracker is not particularly robust.

The best video `results/car.mp4` is generated using Lucas-Kanade method.

Q2.5 Tracking Runway Markings

Note that the demo scripts are `matlab/lk_demo_landing.m` and `matlab/mb_demo_landing.m` respectively.

Initialization coordinates of LK tracker for landing: `tracker = [443 90 560-443 131-90];`
Initialization coordinates of MB tracker for landing: `tracker = [440 90 565-440 135-90];`

The best video `results/landing.mp4` is generated using Matthews-Baker method.

3 Extra Credit

Q3.1x Adding Illumination Robustness

Note that the demo script is `matlab/lk_demo.m`. The car video sequence is used for testing.

We have shown the results of both standard LK tracker and robust LK tracker in the video `results/lk_car_robust.mp4`. The robust LK tracker works better than the standard one, as expected. When the car is in shadow, the standard LK tracker slides behind the car that in the last frame, the bounding box does not contain the whole car anymore, where the robust one still does.

Q3.2x LK Tracking on an Image Pyramid

Note that the demo script is `matlab/lk_demo_landing.m`. The landing video sequence is used for testing.

We have shown the results of both standard LK tracker and pyramid LK tracker in the video `results/lk_landing_pyramid.mp4`. From the video, we can see the standard LK tracker loses the numbers quite quickly as the camera shakes quite vigorously. However, the pyramid LK tracker still manages to track the middle of the numbers. Notice though since we are using a translational tracker, the size of the tracker does not change so eventually the tracker cannot contain all the images.

The pyramid LK tracker runs more quickly, as expected. It uses about 8 seconds while the standard one uses about 13 seconds.