16720 Computer Vision: Homework 4 (Tracking)

Instructor: Martial Hebert
TAs: Xinlei Chen, Arne Suppe, Jacob Walker and Feng Zhou
Due Date: November 2nd, 2013, 11:59 PM
Version: 1.2, Last Update: October 21, 2013

Instructions

This homework consists of three parts. In the first part, you will implement a simple Lucas-Kanade (LK) tracker [1] with a fixed template. In the second part, you will generalize the LK tracker by taking into account large variation in appearance. In the last part, you will implement a motion subtraction method for tracking moving pixels in the scene.

Notice that all the three parts built upon the Lucas-Kanade tracking framework. Please refer to the course slides (`motionIntro.pdf`, `motionIntro+LKT1.pdf`, and `flow+motionSegmentation.pdf`) and the two references [1, 2] for more detailed background.

- A complete homework submission for this homework consists of three parts. A pdf file with answers for the conceptual questions, and the write-up, and three directories of Matlab files for the implementation portions of the homework, which should be named q1, q2 and q3.
- Please pack your system and write-up into a single file named “AndrewId”.zip.
- All questions marked with a Q require a submission.
- For the implementation part, please stick to the headers, variable names, and file conventions provided.
- In your submission file, please **DO NOT** include the data files: `carSequence.mat`, `bookSequence.mat`, `Sequence1.tar.gz`.
- To speed up the running time, try to optimize your implementation using vector-based operation and avoid to loop for every pixel.
- If you have any questions, please contact: **Feng Zhou, zhfe99@gmail.com**

1 Lucas-Kanade Tracking

In this section you will track a specific object in an image sequence (`carSequence.mat`) by template tracking.

The first groundbreaking work on template tracking was the Lucas-Kanade tracker (see Fig.2 in [1] for a review of the algorithm). It basically assumes that the template undergoes constant motion in a small region. The Lucas-Kanade Tracker works on two frames at a time, and does not assume any statistical motion model throughout the sequence. The algorithm estimates the deformations between two image frames under the assumption that the intensity of the objects has not changed significantly between the two frames.

Starting with a rectangle R_t on frame I_t, the Lucas-Kanade Tracker aims to move it by an offset (u, v) to obtain another rectangle R_{t+1} on frame I_{t+1}, so that the pixel squared difference in the two rectangles is minimized:

$$\min_{u,v} J(u,v) = \sum_{(x,y)\in R_t} (I_{t+1}(x+u, y+v) - I_t(x,y))^2 \tag{1}$$

Q1.1. (5 points) Starting with an initial guess of (u, v) (usually $(0, 0)$), we can compute the optimal $(u^*,\ v^*)$ iteratively. In each iteration, the objective function is locally linearized by first-order Taylor expansion and optimized by solving a linear system that has the form $A\Delta p = b$, where $\Delta p = (u, v)^T$ is the template offset.

- What is A^TA?
- What conditions must A^TA meet so that the template offset can be calculated reliably?

Q1.2. (15 points) Upload the following function in your submission directory:

```
[u,v] = LucasKanade(It,It1,rect)
```

that computes the optimal local motion from frame I_t to frame I_{t+1} that minimizes Eq. 1. Here `It` is the image frame I_t , `It1` is the image frame I_{t+1} , and `rect` is the 4-by-1 vector representing a rectangle on the image frame I_t . The four components of the rectangle are $[x1, y1, x2, y2]$, where $(x1, y1)$ is the top-left corner and $(x2, y2)$ is the bottom-right corner. The rectangle is inclusive, i.e., it includes all the four corners. To deal with fractional movement of the template, you will need to interpolate the image using the MATLAB command `interp2`. You will also need to iterate the estimation until the change in (u, v) is below a threshold.

Q1.3. (10 points) Write a script `testCarSequence.m` that loads the video sequence (`carSequence.mat`)[1] and run the Lucas-Kanade Tracker to track the speeding car. The rectangle in the first frame is $[x1, y1, x2, y2]$ = [328, 213, 419, 265]. In other words, the

[1]The variable `sequence` is of the dimension, $h \times w \times m \times n$, where h and w are the image height and width, m is number of color channel, n is number of frames.

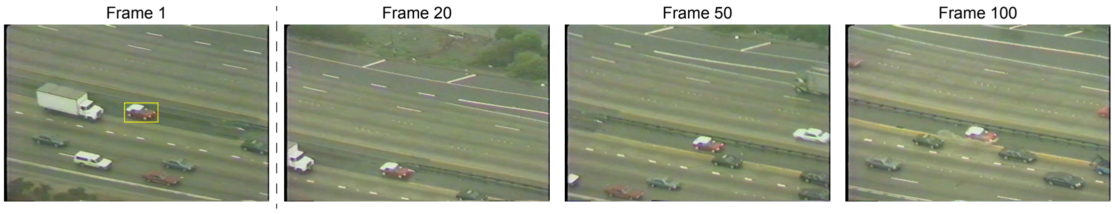

Figure 1: Lucas-Kanade tracking. The bounding box of the car in the first frame is given, and you need to implement the Lucas-Kanade tracker to locate the car's position in the rest of the frames.

rectangle starts from (328, 213) (row 213 and column 328 in the image) and ends at (419, 265). In your answer sheet, print the tracking result (image + rectangle) at frames 20, frame 50 and frame 100.

Make a `carPosition.mat` file which has top-left and bottom-right corners of your tracked object for us to evaluate your tracking result. The `carPosition.mat` file should contain a variable called `box` which is a $n \times 4$ matrix; n is number of frames and each row in the matrix contains 4 numbers: $[x1, y1, x2, y2]$, where indicates the coordinates of top-left and bottom-right corners to the object you tracked.

Q1.4. (5 points) In your answer sheet, give a short analysis on the possible reasons the tracker could fail; give a possible solution to fix some of these problems.

2 Lucas-Kanade Tracking with Appearance Basis

In class, we have learned how to handle geometrical change in the Lucas-Kanade tracker. However, real data is often corrupted by unknown image noise or under varying illumination conditions. In this section, you will implement a variant of the Lucas-Kanade Tracker (see Section 3.4 in [2]) to model linear appearance variation in the tracking. You will be using the image in the `bookSequence.mat` as shown in Fig. 2b.

2.1 Appearance Basis

One way to address issue of appearance variation is to use eigen-space approach (a.k.a., principal component analysis). Suppose that we are given a set of k image bases, $\{B_c\}_{c=1}^{k}$, where each basis B_c is of the size $n \times m$. We can approximate the appearance variation of the new frame $I_{t+1} \in \mathbb{R}^{n \times m}$ at time $t+1$ as a linear combination (Fig. 2a) of the previous frame $I_t \in \mathbb{R}^{n \times m}$ and the bases weighted by a vector $\mathbf{w} = [w_1, \cdots, w_k]^T \in \mathbb{R}^k$:

$$I_{t+1} = I_t + \sum_{c=1}^{k} w_c B_c. \tag{2}$$

Q2.1. (5 points) Write a formula in your answer sheet to optimize Eq. 2 over $\mathbf{w}$ in terms of I_{t+1}, I_t and $\{B_c\}_{c=1}^{k}$.

2.2 Tracking

Given k bases ($\{B_c\}_{c=1}^{k}$), our goal is then to simultaneously find the translation (u, v) and the weight of basis $\mathbf{w} \in \mathbb{R}^k$ that minimize the following objective function,

$$\min_{u,v,\mathbf{w}} J(u, v, \mathbf{w}) = \sum_{(x,y)\in R_t} (I_{t+1}(x+u, y+v) - I_t(x, y) - [\sum_c w_c B_c](x, y))^2 \tag{3}$$

where we define $[\sum_c w_c B_c](x, y)$ to be the value of image $[\sum_c w_c B_c]$ at the position associated with pixel location (x, y).

Generally speaking, there are two ways to optimize Eq. 3. In principle, we can simultaneously optimize the variable $(u, v, w_1, \cdots, w_k)$ by gradient descent from an initial value. See section 3.4.1 in [2] for more details about the optimization. In addition, we can more simply take a coordinate-descent strategy to approximate Eq. 3, ie, alternating between the following two steps until the error $J(u, v, \mathbf{w})$ converges.

- fixing $\mathbf{w}$ and optimize (u, v).
- fixing (u, v) and optimize $\mathbf{w}$ (similar to the optimization of Eq. 2).

Q2.2. (15 points + 10 extra points) Upload the following function in your submission directory to optimize Eq. 3:

```
[u,v] = LucasKanadeBasis(It,It1,rect,basis)
```

where `basis` (contained in the `bookSequence.mat`) is a cell array of length `k` and `k` is the number of bases. You will be awarded extra points if your implementation optimizes over the weight and the translation simultaneously.

Q2.3. (15 points) Write a script `testBookSequence.m` that loads the video sequence (`bookSequence.mat`) and run the new Lucas-Kanade Tracker to track the books. The rectangle in the first frame is $[x1, y1, x2, y2] = [247, 102, 285, 161]$. In your answer sheet, please compare the result between `LucasKanade` and `LucasKanadeBasis` and plot the result (image + bounding box) on frame 30, 150, 248.

Make a `bookPosition.mat` file which has top-left and bottom-right corners of your tracked object for us to evaluate your tracking result. The `bookPosition.mat` file should contain a variable called `box` which is a $n \times 4$ matrix; n is number of frames and each row in the matrix contains 4 numbers: $[x1, y1, x2, y2]$, where indicates the coordinates of top-left and bottom-right corners to the object you tracked.

3 Affine Motion Subtraction

In this section, you will implement a tracker for estimating dominant affine motion in a sequence of images and subsequently identify pixels corresponding to moving objects in the scene. You will be using the images in the file `Sequence1.tar.gz` consisting of aerial views of moving vehicles from a non-stationary camera.

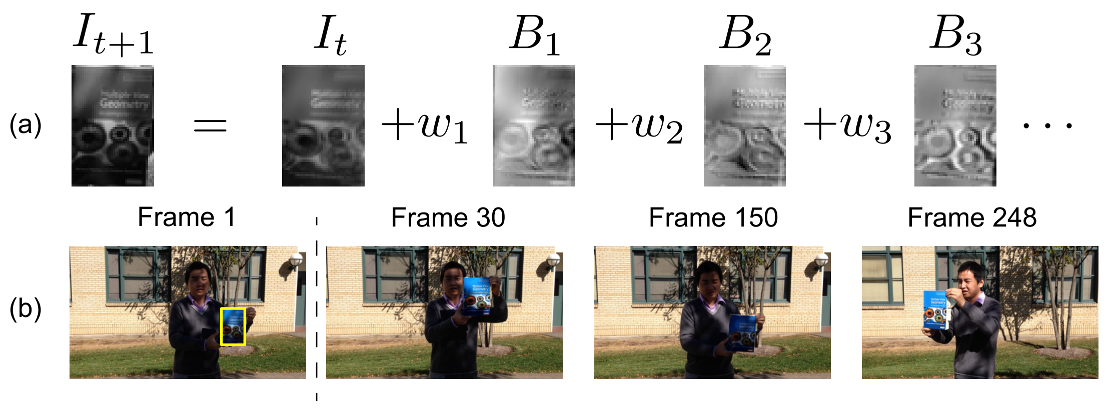

Figure 2: Lucas-Kanade tracking with basis. (a) The appearance variation is modeled as a linear combination of the bases. (b) Example frames of the sequence (`bookSequence.mat`). The bounding box of the book in the first frame is given, and you need to locate the book's position in the rest of the frames.

3.1 Dominant Motion Estimation

You will start by implementing a tracker for affine motion using the equations for affine flow described in class and in the Motion slides from class. As in the previous section, the tracker will only work on two frames at a time.

To estimate dominant motion, the entire image I_t will serve as the "template" to be tracked in image I_{t+1}, i.e. I_{t+1} is assumed to be approximately an affine warped version of I_t. This approach is reasonable under the assumption that a majority of the pixels correspond to stationary objects in the scene whose depth variation is small relative to their distance from the camera.

Using the equations for the affine model of flow, you can recover the vector $\Delta p = [a, b, c, d, e, f]^T \in \mathbb{R}^6$ of affine flow parameters. They will be related to the equivalent affine transformation matrix as:

$$M = \begin{bmatrix} 1+a & b & c \\ d & 1+e & f \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

relating the homogenous image coordinates of I_t to I_{t+1} according to $\mathbf{x}_{t+1} = M\mathbf{x}_t$ (where $\mathbf{x} = [x, y, 1]^T$). For the next pair of consecutive images in the sequence, image I_{t+1} will serve as the template to be tracked in image I_{t+2}, and so on through the rest of the sequence. Note that M will differ between successive image pairs. As before, each update of the affine parameter vector, Δp is computed via a least squares method using the pseudo-inverse as described in class.

Note: Unlike previous examples where the template to be tracked is usually small in comparison with the size of the image, image I_t will almost always not be contained fully in the warped version of I_{t+1}. Hence the matrix of image derivatives, A, and the temporal derivatives, I_t , must be computed only on the pixels lying in the region common to I_t and the warped version of I_{t+1} to be meaningful.

Q3.1. (15 point) Write a function

```
M = LucasKanadeAffine(It, It1)
```

where `M` is the affine transformation matrix, and `It` and `It1` are I_t and I_{t+1} respectively. `LucasKanadeAffine` should be relatively similar to `LucasKanade` from the first question.

3.2 Moving Object Detection

Once you are able to compute the transformation matrix M relating an image pair I_t and I_{t+1}, a naive way of determining pixels lying on moving objects is as follows. Warp the image I_t using M so that it is registered to I_{t+1}, and subtract it from I_{t+1}. The locations where the absolute difference exceeds a threshold can then be declared as corresponding to locations of moving objects. For better results, hysteresis thresholding may be used. We have provided MATLAB code for hysteresis thresholding (`hysthresh.m`).

Q3.2. (15 points) Using the above method for dominant motion estimation, write a MATLAB function

```
moving_image = SubtractDominantMotion(image1, image2)
```

where `image1` and `image2` (in `uint8` format) form the input image pair and `moving_image` is a binary image of the same size as the input images, in which the non-zero pixels correspond to locations of moving objects. The script `test_motion.m` that we will use for grading this section has been provided to you for testing your function. This script simply makes repeated calls to `SubtractDominantMotion` on every consecutive image pair in the file `Sequence1.tar.gz`, makes an AVI movie out of the `moving_image` returned for every image pair processed, and saves it as a file *motion.avi* for your offline viewing pleasure. An example of such a movie is also provided consisting of results using the methods described above. You will notice that the estimates of moving pixels are a bit noisy[2].

Q3.3. (10 extra points) We do not expect perfect results – this is, after all, real world data! You are free to implement any algorithm of your choice for estimating the `moving_image` following the affine registration step[3]. We will be awarding extra credit to the best looking results as judged by the instructors.

4 Change Log

4.1 Version 1.1

Add the description of the dimension of the variable “sequence” in the attached matlab mat file.

[2]For de-noising your output, you may also want to look at the imerode and imdilate morphological operators in MATLAB.

[3]A basic version of this would involve computing the difference between the warped version of the first image and the second image and thresholding.

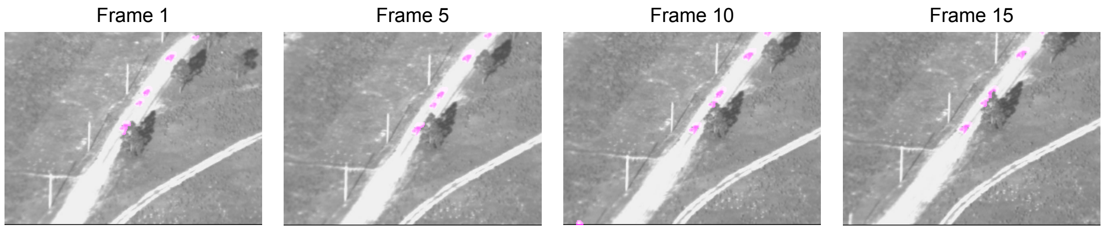

Figure 3: Affine Motion Subtraction. The example frames of `Sequence1.tar.gz`.

4.2 Version 1.2

Add the notion in the introduction section about using "vector-based implementation in Matlab to speed up the computation".

References

[1] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework: Part 1. Technical Report RI-TR-02-16, CMU, 2002.

[2] S. Baker, T. Ishikawa R. Gross, and I. Matthews. Lucas-Kanade 20 years on: A unifying framework: Part 2. Technical Report RI-TR-03-01, CMU, 2003.