16-720A Computer Vision: Assignment 6
Lucas-Kanade Tracking

Instructor: Kris M. Kitani
TAs: Leonid, Mohit, Arjun, Rawal, Aashi, Tanya
Due Date: Tuesday, April 24, 2018 11:59pm

Total Points: 100
Extra Credit Points: 40

Instructions

1. **Integrity and collaboration:** If you work as a group, include the names of your collaborators in your write up. Code should **NOT** be shared or copied. Please **DO NOT** use external code unless permitted. Plagiarism is strongly prohibited and may lead to failure of this course.
2. **Start early!**
3. **Piazza:** If you have any questions, please look at Piazza first. We've created folders for every question, and a sticky thread at the top for each question. Please go through all the previous posts tagged in the appropriate comment before submitting a new question and adding its tag in the sticky.
4. **Write-up:** Please note that we **DO NOT** accept handwritten scans for your write-up in this assignment. Please type your answers to theory questions and discussions for experiments electronically.
5. **Code:** Please stick to the function prototypes mentioned in the handout. This makes verifying code easier for the TAs.
6. **Submission:** Please include all results in your writeup pdf submitted on Gradescope. For code submission, create a zip file, `<andrew-id>.zip`, composed your Matlab files. Please make sure to remove any temporary files you've generated. Your final upload should have the files arranged in this layout.

- `<AndrewId>.zip`
  - `<AndrewId>/`
    - `matlab/`
      - `LucasKanade.m`
      - `LucasKanadeInverseCompositional.m`

- `LucasKanadeAffine.m`
- `LucasKanadeBasis.m`
- `SubtractDominantMotion.m`
- `testCarSequence.m`
- `testUltraSoundSequence.m`
- `testSylvSequence.m`
- `testCarSequenceWithTemplateCorrection.m`
- `testUSSequenceWithTemplateCorrection.m`
- `testAerialSequence.m`
- `testUSSeqAffine.m`
- `carseqrects.mat`
- `carseqrects-wcrt.mat`
- `sylvseqrects.mat`
- `aerialseqrects.mat`
- `InverseCompositionAffine.m`

7. TAs responsible for this assignment: Aashi Manglik (amanglik@andrew.cmu.edu) and Arjun Sharma (arjuns2@andrew.cmu.edu).

* * *

This homework consists of four sections. In the first section you will implement a simple Lucas-Kanade (LK) tracker with one single template; in the second section, the tracker will be generalized to accommodate for large appearance variance. The third section requires you to implement a motion subtraction method for tracking moving pixels in a scene. In the final section you shall study efficient tracking which includes inverse composition and correlation filters. Note the first 3 sections are based on the Lucas-Kanade tracking framework; with the final section also incorporating correlation filters. Other than the course slide decks, the following references may also be helpful:

1. Simon Baker, et al. *Lucas-Kanade 20 Years On: A Unifying Framework: Part 1*, CMU-RI-TR-02-16, Robotics Institute, Carnegie Mellon University, 2002
2. Simon Baker, et al. *Lucas-Kanade 20 Years On: A Unifying Framework: Part 2*, CMU-RI-TR-03-35, Robotics Institute, Carnegie Mellon University, 2003

1 Lucas-Kanade Tracking (30 points)

In this section you will be implementing the Inverse Compositional Lucas & Kanade tracker with one single template. In the scenario of two-dimensional tracking with a pure translation warp function,

$$\mathcal{W}(\mathbf{x};\mathbf{p}) = \mathbf{x} + \mathbf{p} \quad . \tag{1}$$

The problem can be described as follows: starting with a rectangular neighborhood of pixels $\mathbb{N} \in \{\mathbf{x}_d\}_{d=1}^{D}$ on frame $\mathcal{I}_t$, the Lucas-Kanade tracker aims to move it by an offset $\mathbf{p} = [p_x, p_y]^T$ to obtain another rectangle on frame $\mathcal{I}_{t+1}$, so that the pixel squared difference in the two rectangles is minimized:

$$\mathbf{p}^* = \quad \arg\min_{\mathbf{p}} \quad = \sum_{\mathbf{x}\in\mathbb{N}} ||\mathcal{I}_{t+1}(\mathbf{x}+\mathbf{p}) - \mathcal{I}_t(\mathbf{x})||_2^2 \tag{2}$$

$$= \left|\left| \begin{bmatrix} \mathcal{I}_{t+1}(\mathbf{x}_1+\mathbf{p}) \\ \vdots \\ \mathcal{I}_{t+1}(\mathbf{x}_D+\mathbf{p}) \end{bmatrix} - \begin{bmatrix} \mathcal{I}_t(\mathbf{x}_1) \\ \vdots \\ \mathcal{I}_t(\mathbf{x}_D) \end{bmatrix} \right|\right|_2^2 \tag{3}$$

Q1.1 (5 points)

Starting with an initial guess of $\mathbf{p}$ (for instance, $\mathbf{p} = [0,0]^T$), we can compute the optimal $\mathbf{p}^*$ iteratively. In each iteration, the objective function is locally linearized by first-order Taylor expansion,

$$\mathcal{I}_{t+1}(\mathbf{x}' + \Delta\mathbf{p}) \approx \mathcal{I}_{t+1}(\mathbf{x}') + \frac{\partial \mathcal{I}_{t+1}(\mathbf{x}')}{\partial \mathbf{x}'^T} \frac{\partial \mathcal{W}(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}^T} \Delta\mathbf{p} \tag{4}$$

where $\Delta\mathbf{p} = [\Delta p_x, \Delta p_y]^T$, is the template offset. Further, $\mathbf{x}' = \mathcal{W}(\mathbf{x};\mathbf{p}) = \mathbf{x} + \mathbf{p}$ and $\frac{\partial \mathcal{I}(\mathbf{x}')}{\partial \mathbf{x}'^T}$ is a vector of the $x-$ and $y-$ image gradients at pixel coordinate $\mathbf{x}'$. In a similar manner to Equation 3 one can incorporate these linearized approximations into a vectorized form such that,

$$\arg\min_{\Delta\mathbf{p}} ||\mathbf{A}\Delta\mathbf{p} - \mathbf{b}||_2^2 \tag{5}$$

such that $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$ at each iteration.

- What is $\frac{\partial \mathcal{W}(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}^T}$?
- What is $\mathbf{A}$ and $\mathbf{b}$?
- What conditions must $\mathbf{A}^T\mathbf{A}$ meet so that a unique solution to $\Delta\mathbf{p}$ can be found?

Q1.2 (15 points)

Implement a function with the following signature

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

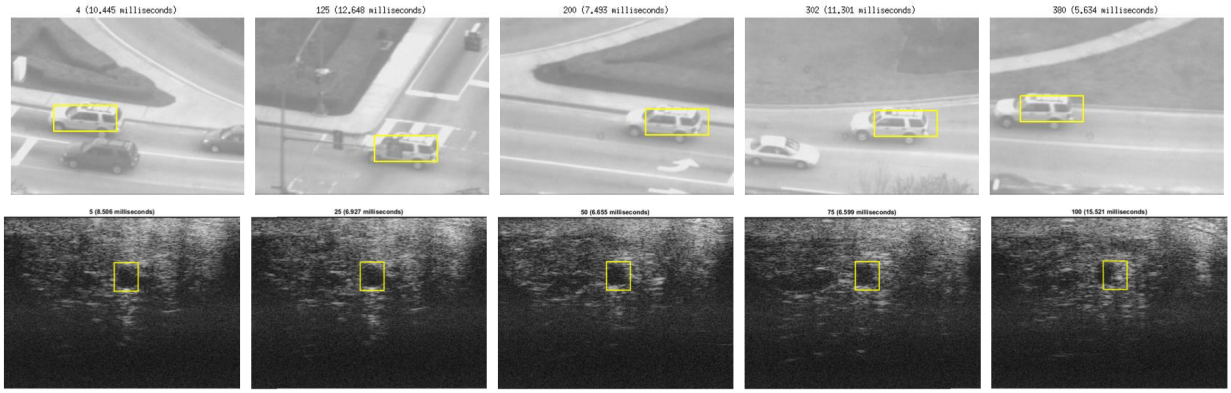

Figure 1: Lucas-Kanade Tracking with One Single Template

that computes the optimal local motion from frame $\mathcal{I}_t$ to frame $\mathcal{I}_{t+1}$ that minimizes Equation 3. Here `It` is the image frame $\mathcal{I}_t$, `It1` is the image frame $\mathcal{I}_{t+1}$, and `rect` is the 4-by-1 vector that represents a rectangle describing all the pixel coordinates within $\mathbb{N}$ within the image frame $\mathcal{I}_t$. The four components of the rectangle are $[x_1, y_1, x_2, y_2]^T$, where $[x_1, y_1]^T$ is the top-left corner and $[x_2, y_2]^T$ is the bottom-right corner. The rectangle is inclusive, i.e., in includes all the four corners. $[u, v]$ is vector $p + \Delta p$. To deal with fractional movement of the template, you will need to interpolate the image using the MATLAB function `interp2`. You will also need to iterate the estimation until the change in $||\Delta \mathbf{p}||_2^2$ is below a threshold. It is recommended to implement the inverse compositional version of the Lucas-Kanade tracker (Section 2.2 in [2]). You are encouraged (but not required) to implement the original Lucas-Kanade algorithm in Section 2.1 as well. Implementing the original L-K algorithm will help you appreciate the performance improvement of the inverse compositional algorithm.

Q1.3 (10 points)

Write a script `testCarSequence.m` that loads the video frames from `carseq.mat`, and runs the Lucas-Kanade tracker that you have implemented in the previous task to track the car. `carseq.mat` can be located in the `data` directory and it contains one single three-dimensional matrix: the first two dimensions correspond to the *height* and *width* of the frames respectively, and the third dimension contain the indices of the frames (that is, the first frame can be visualized with `imshow(frames(:, :, 1))`). The rectangle in the first frame is $[x_1, y_1, x_2, y_2]^T = [60, 117, 146, 152]^T$. Report your tracking performance (image + bounding rectangle) at frames 1, 100, 200, 300 and 400 in a format similar to Figure 1. Also, create a file called `carseqrects.mat`, which contains one single $n \times 4$ matrix `rects`, where each row stores the `rect` that you have obtained for each frame, and n is the total number of frames.

Now, lets apply the your algorithm on a challenging problem: tracking a beating vessel in an ultrasound volume. Unlike regular scenes in the world, medical images acquired by an ultrasound transducer undergo significant non-rigid motion.

Write a new script `testUltrasoundSequence.m` that loads the video frames from `usseq.mat`, and tracks the beating vessel using the Lucas-Kanade tracker that you have implemented in the previous question. The rectangle in the first frame is $[x1, y1, x2, y2] = [255, 105, 310, 170]$. Report your tracking performance (image + bounding rectangle) at frames 5, 25, 50, 75 and 100 in a format similar to Figure 1. Also, create a file called `usseqrects.mat`, which contains one single $n \times 4$ matrix `rects`, where each row stores the `rect` that you have obtained for each frame, and n is the total number of frames.

Q1.4 Extra Credit (20 points)

As you might have noticed, the image content we are tracking in the first frame differs from the one in the last frame. This is understandable since we are updating the template after processing each frame and the error can be accumulating. This problem is known as *template drifting*. There are several ways to mitigate this problem. Iain Matthews et al. (2003, https://www.ri.cmu.edu/publication_view.html?pub_id=4433) suggested one possible approach. Write two scripts `testCarSequenceWithTemplateCorrection.m` and `testUSSequenceWithTemplateCorrection.m` with a similar functionality to **Q1.3**, but with a template correction routine incorporated. Save the resulting `rects` as `carseqrects-wcrt.mat` and `usseqrects-wcrt.mat`, and also report the performance at those frames. An example is given in Figure 2.

Here the green rectangles are created with the baseline tracker in **Q1.3**, the yellow ones with the tracker in **Q1.4**. The tracking performance has been improved non-trivially. Note that you do not necessarily have to draw two rectangles in each frame, but make sure that the performance improvement can be easily visually inspected.

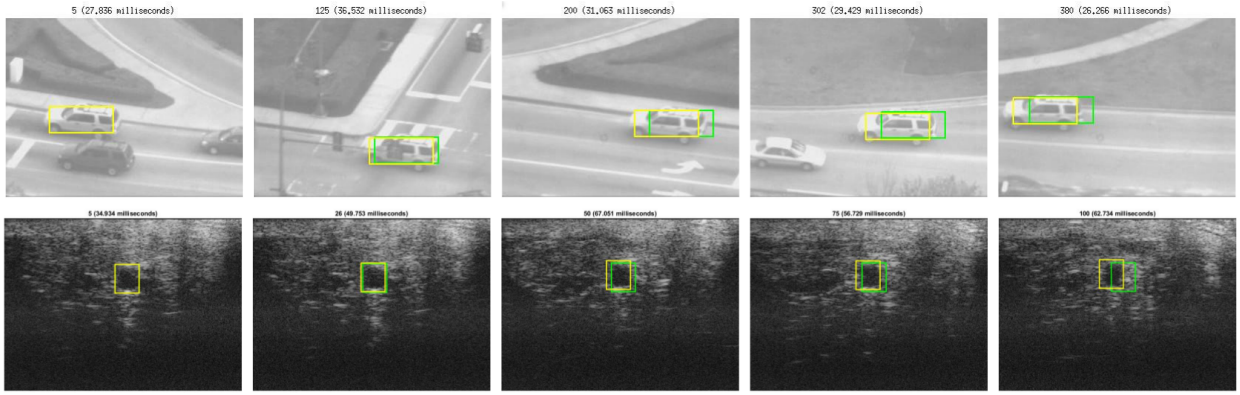

Figure 2: Lucas-Kanade Tracking with Template Correction

2 Lucas-Kanade Tracking with Appearance Basis (35 points)

The tracker we have implemented in the first secion, with or without template drifting correction, may suffice if the object being tracked is not subject to drastic appearance variance. However, in real life, this can hardly be the case. We have prepared another sequence `sylvseq.mat` (the initial rectangle is $[102, 62, 156, 108]$), with exactly the same format as `carseq.mat`, on which you can test the baseline implementation and see what would happen. 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.

2.1 Appearance Basis

One way to address this issue is to use eigen-space approach (aka, principal component analysis, or PCA). The idea is to analyze the historic data we have collected on the object, and produce a few bases, whose linear combination would most likely to constitute the appearance of the object in the new frame. This is actually similar to the idea of having a lot of templates, but looking for too many templates may be expensive, so we only worry about the *principal* templates.

Mathematically, suppose we are given a set of k image bases $\{\mathcal{B}_k\}_{k=1}^K$ of the same size. We can approximate the appearance variation of the new frame $\mathcal{I}_{t+1}$ as a linear combination of the previous frame $\mathcal{I}_t$ and the bases weighted by $\mathbf{w} = [w_1, \ldots, w_K]^T$, such that

$$\mathcal{I}_{t+1}(\mathbf{x}) = \mathcal{I}_t(\mathbf{x}) + \sum_{k=1}^{K} w_k \mathcal{B}_k(\mathbf{x}) \tag{6}$$

Q2.1 (5 pts)

Express $\mathbf{w}$ as a function of $\mathcal{I}_{t+1}$, $\mathcal{I}_t$, and $\{\mathcal{B}_k\}_{k=1}^K$, given Equation 6. Note that since the $\mathcal{B}_k$'s are orthobases, thay are orthogonal to each other.

2.2 Tracking

Given K bases, $\{\mathcal{B}_k\}_{k=1}^K$, our goal is then to simultaneously find the translation $\mathbf{p} = [p_x, p_y]^T$ and the weights $\mathbf{w} = [w_1, \ldots, w_K]^T$ that minimizes the following objective function:

$$\min_{\mathbf{p},\mathbf{w}} = \sum_{\mathbf{x}\in\mathbb{N}} ||\mathcal{I}_{t+1}(\mathbf{x}+\mathbf{p}) - \mathcal{I}_t(\mathbf{x}) - \sum_{k=1}^{K} w_k \mathcal{B}_k(\mathbf{x})||_2^2 \ . \tag{7}$$

Again, starting with an initial guess of $\mathbf{p}$ (for instance, $\mathbf{p} = [0, 0]^T$), one can linearize $\mathcal{I}_{t+1}(\mathbf{x}+\mathbf{p}+\Delta\mathbf{p})$ with respect to $\Delta\mathbf{p}$. In a similar manner to Equation 5 one can incorporate these linearized approximations into a vectorized form such that,

$$\arg\min_{\Delta\mathbf{p},\mathbf{w}} ||\mathbf{A}\Delta\mathbf{p} - \mathbf{b} - \mathbf{B}\mathbf{w}||_2^2 \ . \tag{8}$$

As discussed in Section 3.4 of [2] (ignore the inverse compositional discussion) this can be simplified down to

$$\arg\min_{\Delta \mathbf{p}} ||\mathbf{B}^{\perp}(\mathbf{A}\Delta\mathbf{p} - \mathbf{b})||_2^2 \tag{9}$$

where $\mathbf{B}^{\perp}$ spans the null space of $\mathbf{B}$. Note that $||\mathbf{B}^{\perp}\mathbf{z}||_2^2 = ||\mathbf{z} - \mathbf{B}\mathbf{B}^T\mathbf{z}||_2^2$ when $\mathbf{B}$ is an orthobasis.

Q2.2 (15 pts)

Implement a function with the following signature

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

where `bases` is a three-dimensional matrix that contains the bases. It has the same format as `frames` as is described earlier and can be found in `sylvbases.mat`.

Q2.3 (15 pts)

Write a script `testSylvSequence.m` that loads the video frames from `sylvseq.mat` and runs the new Lucas-Kanade tracker to track the sylv (the toy). The bases are available in `sylvbases.mat` in the `data` directory. The rectangle in the first frame is $[x_1, y_1, x_2, y_2]^T = [102, 62, 156, 108]^T$. Please report the performance of this tracker at frames 1, 200, 300, 350 and 400 (the frame + bounding box), in comparison to that of the tracker in the first section. That is, there should be two rectangles for each frame, as exemplified in Figure 3. Also, create a `sylvseqrects.mat` for all the `rects` you have obtained for each frame. It should contain one single $N \times 4$ matrix named `rects`, where N is the number of frames, and each row contains $[x_1, y_1, x_2, y_2]^T$, where $[x1, y1]^T$ is the coordinate of the top-left corner of the tracking box, and $[x_2, y_2]^T$ the bottom-right corner.

Figure 3: Lucas-Kanade Tracking with Appearance Basis

3 Affine Motion Subtraction (35 points)

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 `aerialseq.mat`, which consists aerial views of moving vehicles from a non-stationary camera.

3.1 Dominant Motion Estimation

In the first section of this homework we assumed the the motion is limited to pure translation. In this section you shall implement a tracker for affine motion using a planar affine warp function. To estimate dominant motion, the entire image $\mathcal{I}_t$ will serve as the template to be tracked in image $\mathcal{I}_{t+1}$, that is, $\mathcal{I}_{t+1}$ is assumed to be approximately an affine warped version of $\mathcal{I}_t$. This approach is reasonable under the assumption that a majority of the pixels correspond to the stationary objects in the scene whose depth variation is small relative to their distance from the camera.

Using a planar affine warp function you can recover the vector $\Delta\mathbf{p} = [p_1, \ldots, p_6]^T$,

$$\mathbf{x}' = \mathcal{W}(\mathbf{x}; \mathbf{p}) = \begin{bmatrix} 1+p_1 & p_2 \\ p_4 & 1+p_5 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} p_3 \\ p_6 \end{bmatrix} \quad . \tag{10}$$

One can represent this affine warp in homogeneous coordinates as,

$$\tilde{\mathbf{x}}' = \mathbf{M}\tilde{\mathbf{x}} \tag{11}$$

where,

$$\mathbf{M} = \begin{bmatrix} 1+p_1 & p_2 & p_3 \\ p_4 & 1+p_5 & p_6 \\ 0 & 0 & 1 \end{bmatrix} \quad . \tag{12}$$

Note that $\mathbf{M}$ will differ between successive image pairs. Starting with an initial guess of $\mathbf{p} = \mathbf{0}$ (i.e. $\mathbf{M} = \mathbf{I}$) you will need to solve a sequence of least-squares problem to determine $\Delta\mathbf{p}$ such that $\mathbf{p} \rightarrow \mathbf{p} + \Delta\mathbf{p}$ at each iteration. Note that unlike previous examples where the template to be tracked is usually small in comparison with the size of the image, image $\mathcal{I}_t$ will almost always not be contained fully in the warped version $\mathcal{I}_{t+1}$. Hence, one must only consider pixels lying in the region common to I_t and the warped version of I_{t+1} when forming the linear system at each iteration.

3.1 (15 points)

Write a function with the following signature

```
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 section.

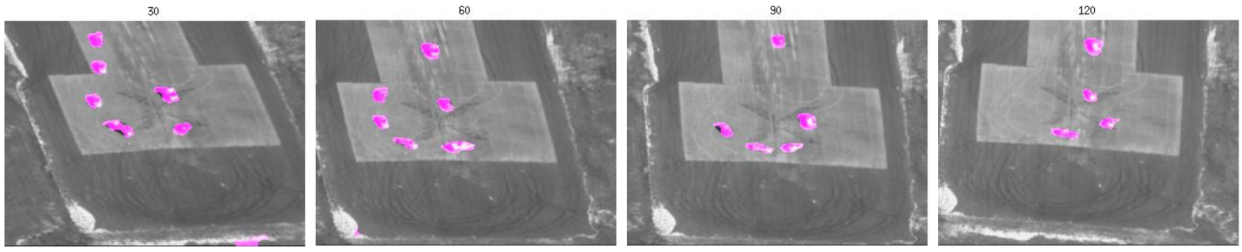

Figure 4: Lucas-Kanade Tracking with Appearance Basis

3.2 Moving Object Detection

Once you are able to compute the transformation matrix $\mathbf{M}$ relating an image pair $\mathcal{I}_t$ and $\mathcal{I}_{t+1}$, a naive way for determining pixels lying on moving objects is as follows: warp the image $\mathcal{I}_t$ using $\mathbf{M}$ so that it is registered to $\mathcal{I}_{t+1}$ and subtract it from $\mathcal{I}_{t+1}$; the locations where the absolute difference exceeds a threshold can then be declared as corresponding to locations of moving objects. To obtain better results, you can check out the following MATLAB functions: `bwselect`, `bwareaopen`, `imdilate`, and `imerode`.

3.2 (10 points)

Using the function you have developed for dominant motion estimation, write a function with the following signature

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

where `image1` and `image2` form the input image pair, and `mask` is a binary image of the same size that dictates which pixels are considered to be corresponding to moving objects. You should invoke `LucasKanadeAffine` in this function to derive the transformation matrix $\mathbf{M}$, and produce the aforementioned binary mask accordingly.

3.3 (10 points)

For this question, write a script `testAerialSequence.m` that loads the image sequence from `aerialseq.mat` and run the motion detection routine you have developed to detect the moving objects. Report the performance at frames 30, 60, 90 and 120 with the corresponding binary masks superimposed, as exemplified in Figure 4. Feel free to visualize the motion detection performance in a way that you would prefer, but please make sure it can be visually inspected without undue effort. The MATLAB function `imfuse` may be useful.

Now, generate another script `testUSSeqAffine.m` that loads `usseq.mat` and `usrects.mat` (or `usseqrects-wcrt.mat` if you implemented Q1.4), and run the global motion detection algorithm you wrote. Since the images undergo non-rigid motion, we are still going to ap- proximate the motion using the same affine-based motion model. However, we only want to visualize the motion of the vessel that is undergoing motion from frame to frame. So, generate a mask based on the

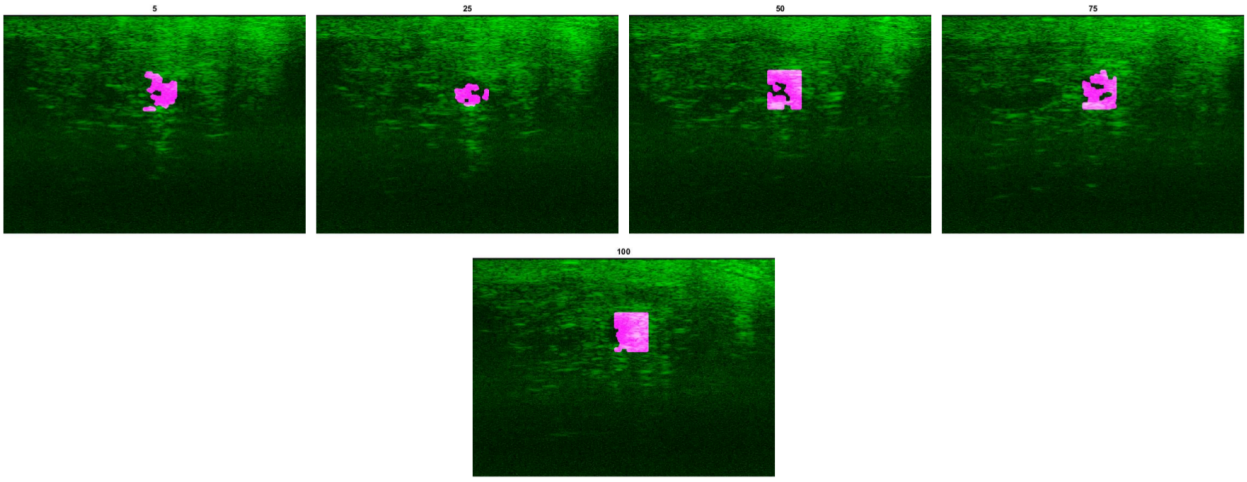

Figure 5: Lucas-Kanade Tracking of Affine Motion

rectangle coordinates that you have previously saved, and display only the moving pixels in the rectangle for each frame. You can still continue to use `imfuse` for this purpose. Include the results of your algorithm at frames 5, 25, 50, 75, and 100 in your writeup. Save the set of images as a single file and include it as a JPEG or PNG with your submission.

4 Extra Credit

Q4.1 Classical Lucas-Kanade Algorithm (10 points)

Reimplement the function `LucasKanadeInverseCompositional(It,It1)` as `LucasKanade(It,It1)` using the original Lucas Kanade method (Section 2.1 in [2]). In your own words please describe why the inverse compositional approach is more computationally efficient than the classical approach?

Q4.2 Deep Learning for Tracking (4 points)

The Lucas-Kanade algorithm you implemented in this assignment was published in the 1980s. More recently, several Deep Learning based approaches have improved the state-of-the-art significantly. Once such paper that utilizes features from a neural network in the Inverse Compositional Lucas-Kanade algorithm is Chaoyang Wang, et al. *Deep-LK for Efficient Adaptive Object Tracking*. Read this paper and describe in your own words, how this paper uses a neural network to implement the Inverse Compositional Lucas-Kanade algorithm.

Q4.3 Mean-Shift Tracking (3 + 3 points)

In this question you will derive another tracking algorithm called Mean-shift tracking. Mean-shift tracking iteratively computes a mean-shift vector to track a moving target in video frames. It uses color or grayscale intensity distribution of the target template to track it.

(a) Probability density function around the neighborhood of a point x could be represented as:

$$p(x) = \frac{1}{n} c \sum_{i=1}^{n} k(\|x - x_i\|^2) \tag{13}$$

Here, c is some constant and $k(\|x - x_i\|^2)$ is a kernel function, e.g., RBF kernel. Mean-shift vector $m(x)$ is computed as follows.

$$m(x) = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} - x \tag{14}$$

where w_i is a weight for point x_i. Intuitively, $m(x)$ gives the direction from the location of the old centre to the new centre, where the new centre is computed by weighing the points according to some weights w_i. Iteratively doing mean-shift can be used, for example, to estimate the modes of a distribution using points sampled from it by selecting a weighting function which gives higher weights to points in denser regions.

Prove that the mean-shift vector is proportional to the gradient of probability distribution $p(x)$, with $w_i = \nabla k(\|x - x_i\|^2)$

(b) A density over intensities can be computed in a image patch by,

$$p(u) = \frac{1}{n_x} \sum_{i=1}^{n_x} \delta[S(x_i) - u] \tag{15}$$

where u is a intensity value (e.g. 0-255 in a grayscale image), $S(x_i)$ is the intensity of the pixel x_i and δ is the Dirac-delta function. Let us call the density of a patch centered at y in the candidate image as $p_u(y)$ and that of the target patch as q_u. The goal in tracking is then to find a candidate patch centre y such that the target density q_u is very similar to the estimated candidate density $p_u(y)$. One notion of similarity is given by $d(y) = \sqrt{1 - \rho(y)}$ where $\rho(y)$ is a Bhattacharya coefficient given by :

$$\rho(y) = \sum_{u=1}^{m} \sqrt{p_u(y) q_u} \tag{16}$$

Observe that the Bhattacharya coefficient is simply the bin-wise multiplication of two normalized histograms. After doing the first-order Taylor series expansion of $\rho(y)$ around $p_u(y_0)$, one can get the following:

$$\rho[p_u(y), q_u] = \text{constant} + \frac{1}{2} \sum_{u=1}^{m} p_u(y) \sqrt{\frac{q_u}{p_u(y_0)}} \tag{17}$$

From some initial $p_u(y_0)$, the first derivative in Taylor series expansion can be used to take a step along the gradient where $\rho(y)$ is increasing. Maximizing $\rho(y)$ is same as maximizing the second term in Equation 17. The second term will be higher for those values of y whose $p_u(y)$ is similar to the intensity distribution of

target template q_u. Higher the similarity, greater will be its value (note that it is always positive).

Using the idea of mean-shift vector introduced in part (a) and similarity distribution defined here develop a algorithm that could track a target object in a video. Provide a pseudo code or flow chart to present your algorithm in the write-up.
Hint: Substitute the expression for $p_u(y)$. Do you see something you can use as weights for x_i?