16-720B Computer Vision: Homework 3
Lucas-Kanade Tracking and Correlation Filters

Heethesh Vhavle
Andrew ID: hvhavlen

24 October 2018

1 Lucas-Kanade Tracking

1.1.1

$\frac{\partial \mathcal{W}(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}^T}$ represents the Jacobian of the warped image. For a pure translation the Jacobian is as follows.

$$\frac{\partial \mathcal{W}(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}^T} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{1}$$

1.1.2

We can linearize the objective function locally by first-order Taylor expansion and can rewrite and rearrange our equation to minimize as follows.

$$\arg\min_{\Delta\mathbf{p}} \textstyle\sum_{\mathbf{x}\in\mathbb{N}} \|\mathcal{I}_{t+1}(\mathbf{x}' + \Delta\mathbf{p}) - \mathcal{I}_t(\mathbf{x})\|_2^2$$

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

$$\arg\min_{\Delta\mathbf{p}} \textstyle\sum_{\mathbf{x}\in\mathbb{N}} \left\| \mathcal{I}_{t+1}(\mathbf{x}') + \tfrac{\partial \mathcal{I}_{t+1}(\mathbf{x}')}{\partial \mathbf{x}'^T} \tfrac{\partial \mathcal{W}(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}^T} \Delta\mathbf{p} - \mathcal{I}_t(\mathbf{x}) \right\|_2^2 \tag{2}$$

$$\arg\min_{\Delta\mathbf{p}} \textstyle\sum_{\mathbf{x}\in\mathbb{N}} \left\| \tfrac{\partial \mathcal{I}_{t+1}(\mathbf{x}')}{\partial \mathbf{x}'^T} \tfrac{\partial \mathcal{W}(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}^T} \Delta\mathbf{p} - (\mathcal{I}_t(\mathbf{x}) - \mathcal{I}_{t+1}(\mathbf{x}')) \right\|_2^2$$

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

Comparing the last two forms above we get the following.

$$\begin{gathered} \mathbf{A} = \textstyle\sum_{\mathbf{x}\in\mathbb{N}} \tfrac{\partial \mathcal{I}_{t+1}(\mathbf{x}')}{\partial \mathbf{x}'^T} \tfrac{\partial \mathcal{W}(\mathbf{x};\mathbf{p})}{\partial \mathbf{p}^T} \\ \mathbf{b} = \textstyle\sum_{\mathbf{x}\in\mathbb{N}} \mathcal{I}_t(\mathbf{x}) - \mathcal{I}_{t+1}(\mathbf{x}') \end{gathered} \tag{3}$$

In other words, **A** represents the steepest descent images and **b** represents the error image.

1.1.3

To minimize $\Delta\mathbf{p}$ in equation 2, we take its first derivative and equate it to zero.

$$\begin{aligned} \textstyle\sum_{\mathbf{x}\in\mathbb{N}} 2\mathbf{A}^\top(\mathbf{A}\Delta\mathbf{p} - \mathbf{b}) = 0 \\ \textstyle\sum_{\mathbf{x}\in\mathbb{N}} \mathbf{A}^\top\mathbf{A}\Delta\mathbf{p} = \textstyle\sum_{\mathbf{x}\in\mathbb{N}} \mathbf{A}^\top\mathbf{b} \\ \Delta\mathbf{p} = \textstyle\sum_{\mathbf{x}\in\mathbb{N}} (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{b} \end{aligned} \tag{4}$$

As it can be seen above, in order to solve for $\Delta\mathbf{p}$, $\mathbf{A}^T\mathbf{A}$ must be invertible or non-singular matrix or must have a non-zero determinant to obtain an unique solution for $\Delta\mathbf{p}$.

1.3 Car Sequence Tracking

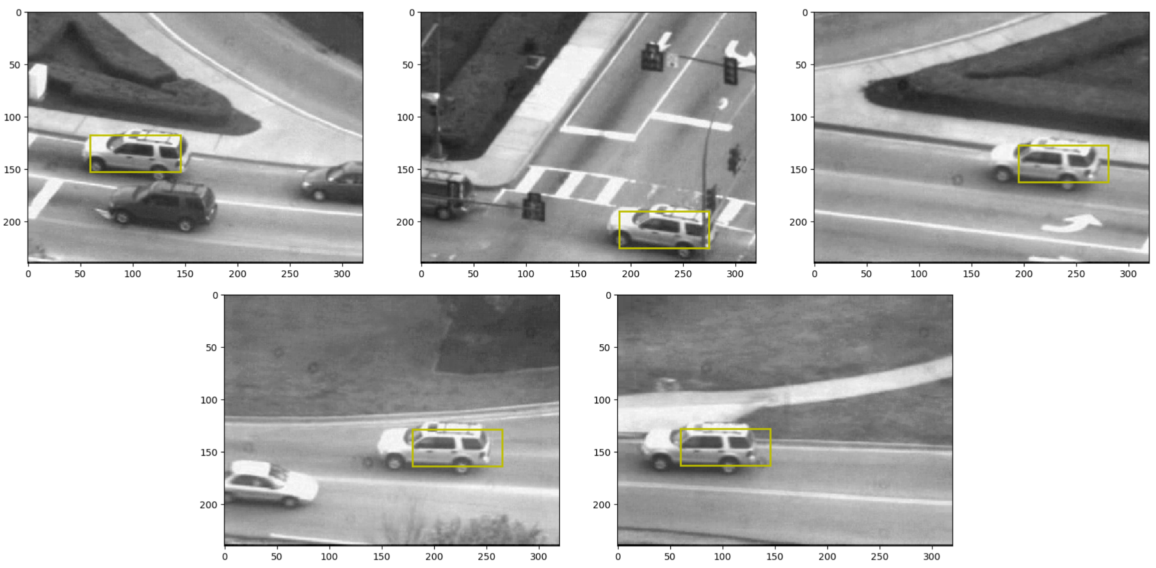

Figure 1: Lucas-Kanade Tracking with One Single Template at Frames 1, 100, 200, 300, 400

1.4 Car Sequence Tracking with Template Drift Correction

We are using Strategy 3 for template drift correction here. Since the template is updated every frame and registered to the initial template (drift correction), the object is tracked correctly till the end and improves the tracking robustness compared to the original Lucas-Kanade. If we implement this using Lucas-Kanade with Affine Transform, we can achieve even better resuts compared to the current pure translation case.

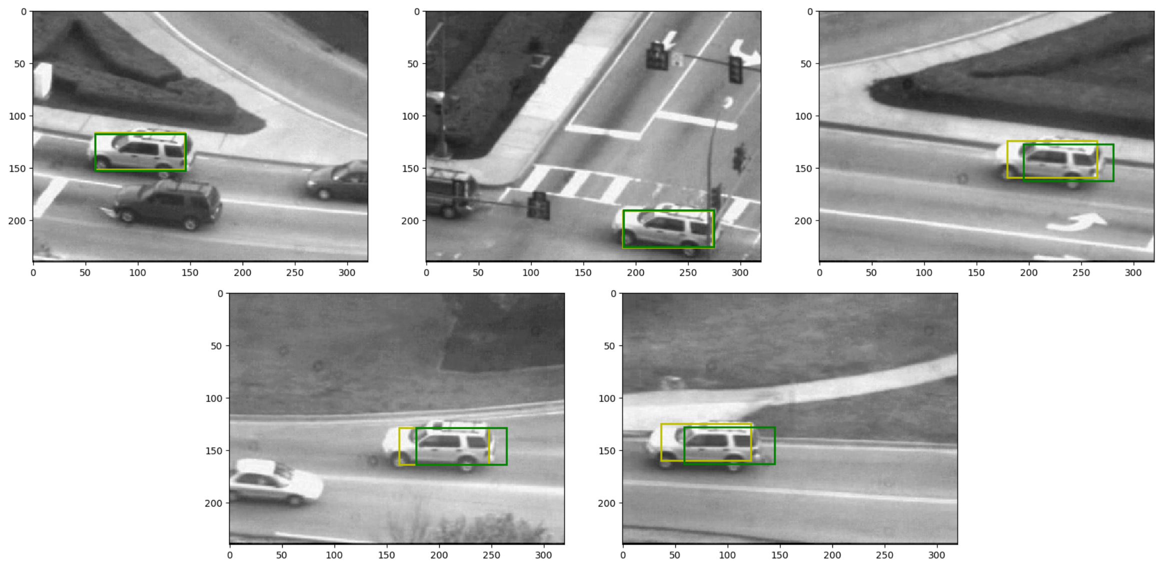

Figure 2: Lucas-Kanade Tracking with Template Drift Correction at Frames 1, 100, 200, 300, 400

2 Lucas-Kanade Tracking with Appearance Basis

2.1 Appearance Basis

Given a set of k image basis $\{\mathcal{B}_k\}_{k=1}^{K}$ weighted by $\mathbf{w} = [w_1, \ldots, w_K]^T$, we have the following relation.

$$\mathcal{I}_{t+1}(\mathbf{x}) = \mathcal{I}_t(\mathbf{x}) + \sum_{k=1}^{K} w_k \mathcal{B}_k(\mathbf{x}) \tag{5}$$

Reorganizing this and expanding the summation.

$$\mathcal{I}_{t+1}(\mathbf{x}) - \mathcal{I}_t(\mathbf{x}) = w_1 \mathcal{B}_1(\mathbf{x}) + \cdots + w_k \mathcal{B}_k(\mathbf{x}) + \cdots + w_K \mathcal{B}_K(\mathbf{x}) \tag{6}$$

Multiplying both sides by $\mathcal{B}_k(\mathbf{x})$ and using the orthogonal property of basis we get the following.

$$\mathcal{B}_k(\mathbf{x})(\mathcal{I}_{t+1}(\mathbf{x}) - \mathcal{I}_t(\mathbf{x})) = 0 + \cdots + w_k \mathcal{B}_k(\mathbf{x}) \mathcal{B}_k(\mathbf{x}) + \cdots + 0 \tag{7}$$

Reorganizing the above equation we can express any $\{w_k\}_{k=1}^{K}$ as follows.

$$w_k = \frac{\mathcal{B}_k(\mathbf{x})(\mathcal{I}_{t+1}(\mathbf{x}) - \mathcal{I}_t(\mathbf{x}))}{\|\mathcal{B}_k(\mathbf{x})\|^2} \tag{8}$$

$$w_k = \mathcal{B}_k(\mathbf{x})(\mathcal{I}_{t+1}(\mathbf{x}) - \mathcal{I}_t(\mathbf{x})) \tag{9}$$

2.3 Tracking Results

In this case, the results Lucas-Kanade Tracking and Lucas-Kanade Tracking with Appearance Basis seems to be very similar. But if we consider another scenario with greater variation in appearance or illumination of the object to be tracked, the Appearance Basis method will definitely give better results.

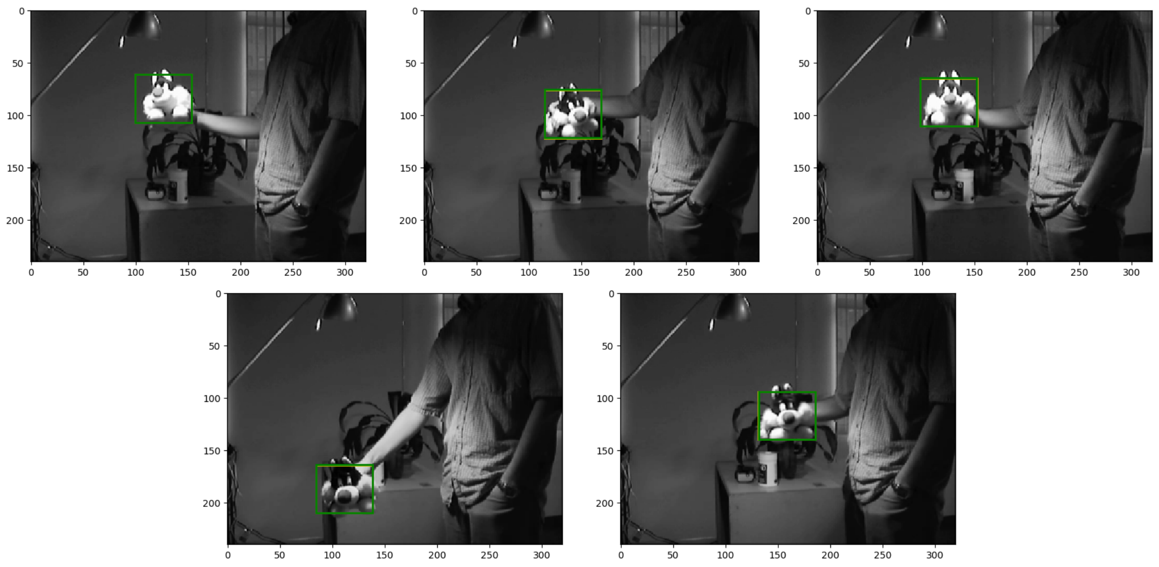

Figure 3: Lucas-Kanade Tracking with Appearance Basis at Frames 1, 200, 300, 350, 400

3 Affine Motion Subtraction

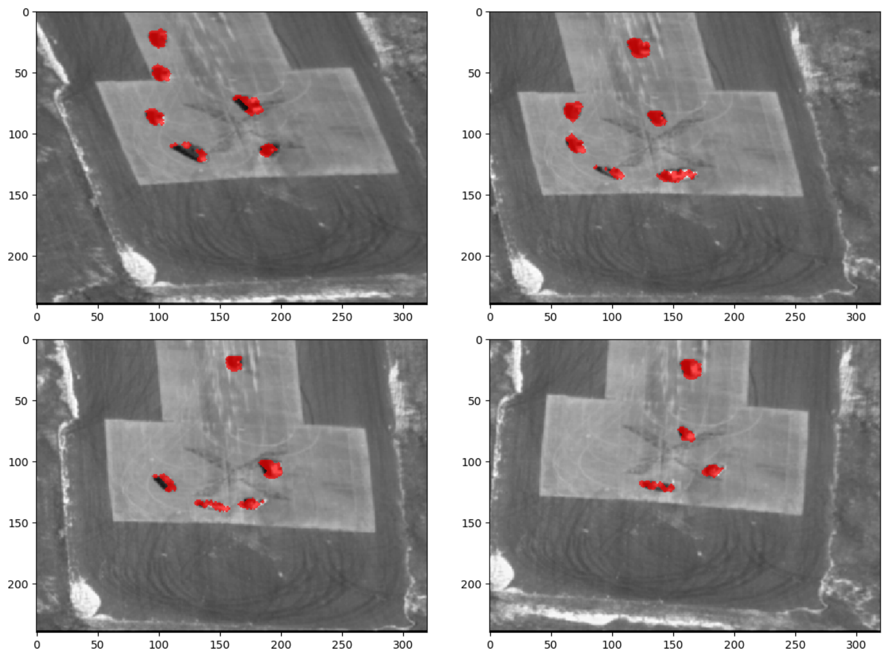

Figure 4: Result of Dominant Motion Subtraction after Lucas-Kanade Registration with Affine Transform at Frames 30, 60, 90, 120

4 Efficient Tracking

4.1 Inverse Composition

In Lucas-Kanade Inverse Composition, we are swapping the template and the current image and the image registration is done the other way around. Thus, there are certain steps of original Lucas-Kanade (steps 3-6) that can be computed only once without re-computing them in every iteration. The steps that we move out of the iteration also involves computationally costly Hessian calculation, which we are keeping as a constant and computing only once now. This along with its operations such as calculating gradients, multiplying with the Jacobian to get the steepest descent images and so on are consequently moved out of the loop. Thus, the Inverse Composition method is more efficient than the original method.

4.2 Correlation Filters

We need to minimize the following equation where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ is a set of sub-images from Lena image and $\mathbf{x}_n \in \mathbb{R}^D$ $(D = 1305)$. $\mathbf{y} = [y_1, \ldots, y_N]$ are the output labels, where y_n lies between zero and one. We need to find the resultant linear discriminant weight vector $\mathbf{g}$ for the penalty values $\lambda = 0$ and $\lambda = 1$. We can expand the terms of the equation as follows.

$$\frac{1}{2}\|\mathbf{y} - \boldsymbol{X}^\top \mathbf{g}\|_2^2 + \frac{1}{2}\lambda\|\mathbf{g}\|_2^2 \tag{10}$$

$$\begin{aligned} &= (\mathbf{y} - \boldsymbol{X}^\top \mathbf{g})^\top (\mathbf{y} - \boldsymbol{X}^\top \mathbf{g}) + \lambda \mathbf{g}^\top \mathbf{g} \\ &= \mathbf{y}^\top \mathbf{y} - (\boldsymbol{X}\mathbf{y})^\top \mathbf{g} - \mathbf{g}^\top \boldsymbol{X}\mathbf{y} + \mathbf{g}^\top \boldsymbol{X}\boldsymbol{X}^\top \mathbf{g} + \lambda \mathbf{g}^\top \mathbf{g} \end{aligned} \tag{11}$$

The following identities of Matrix Calculus can be used. $\mathbf{A}$ and $\mathbf{b}$ are not functions of $\mathbf{x}$.

$$\begin{aligned} \tfrac{\partial \mathbf{b}^\top \mathbf{A}\mathbf{x}}{\partial \mathbf{x}} &= \mathbf{A}^\top \mathbf{b} \\ \tfrac{\partial \mathbf{x}^\top \mathbf{A}\mathbf{x}}{\partial \mathbf{x}} &= \left(\mathbf{A} + \mathbf{A}^\top\right)\mathbf{x} \\ \tfrac{\partial \mathbf{x}^\top \mathbf{a}}{\partial \mathbf{x}} &= \mathbf{a} \\ \tfrac{\partial \mathbf{x}^\top \mathbf{x}}{\partial \mathbf{x}} &= \mathbf{2x} \end{aligned} \tag{12}$$

In order to do find the value of $\mathbf{g}$ to minimize the equation 10, we can find its derivative and equate it to zero. Also setting $\mathbf{S} = \mathbf{X}\mathbf{X}^\top$ and simplifying the terms. $\mathbf{S}$ here is a symmetric matrix.

$$\begin{aligned} &\tfrac{\partial}{\partial \mathbf{g}} \left(\mathbf{y}^\top \mathbf{y} - (\boldsymbol{X}\mathbf{y})^\top \mathbf{g} - \mathbf{g}^\top \boldsymbol{X}\mathbf{y} + \mathbf{g}^\top \boldsymbol{S}\mathbf{g} + \lambda \mathbf{g}^\top \mathbf{g}\right) \\ &= -2(\boldsymbol{X}\mathbf{y}) + \left(\boldsymbol{S} + (\boldsymbol{S})^\top\right)\mathbf{g} + 2\lambda\mathbf{g} \\ &= -2\boldsymbol{X}\mathbf{y} + 2\boldsymbol{S}\mathbf{g} + 2\lambda\mathbf{g} \end{aligned} \tag{13}$$

$$\begin{aligned} -2\boldsymbol{X}\mathbf{y} + 2\boldsymbol{S}\mathbf{g} + 2\lambda\mathbf{g} &= 0 \\ \boldsymbol{S}\mathbf{g} + \lambda\mathbf{g} &= \boldsymbol{X}\mathbf{y} \\ (\boldsymbol{S} + \lambda\boldsymbol{I})\,\mathbf{g} &= \boldsymbol{X}\mathbf{y} \end{aligned} \tag{14}$$

If $(\boldsymbol{S} + \lambda\boldsymbol{I})$ is invertible with non-zero determinant, we can obtain the required form of equation as follows.

$$\mathbf{g} = (\boldsymbol{S} + \lambda\boldsymbol{I})^{-1} \boldsymbol{X}\mathbf{y} \tag{15}$$

4.3 Filter Responses using Correlation

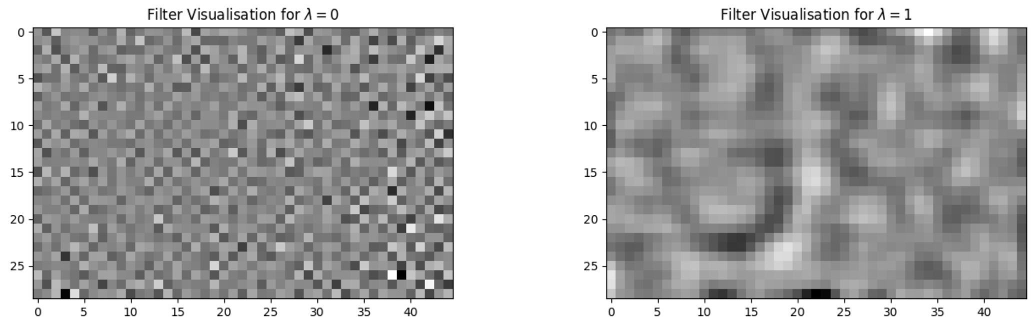

Figure 5: Resultant Linear Discriminant Weight Vector for $\lambda = 0, 1$

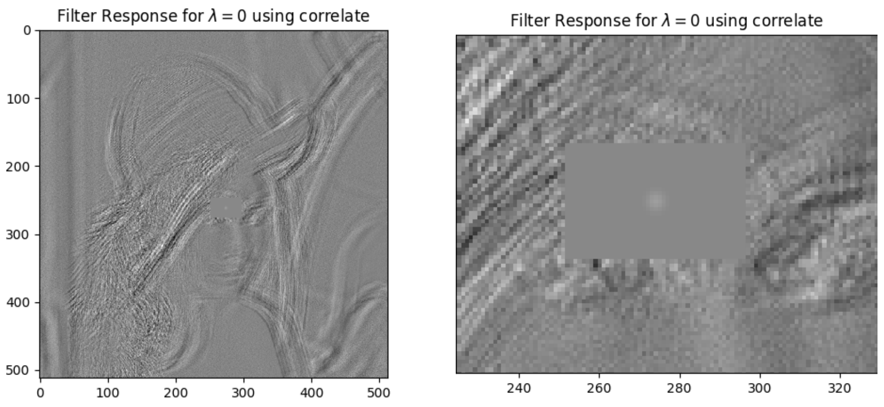

Figure 6: Filter Response on Lena Image for $\lambda = 0$, using Correlation

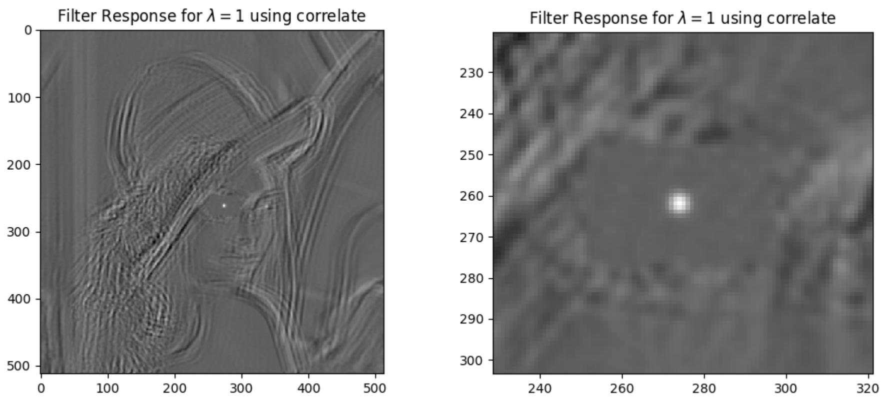

Figure 7: Filter Response on Lena Image for $\lambda = 1$, using Correlation

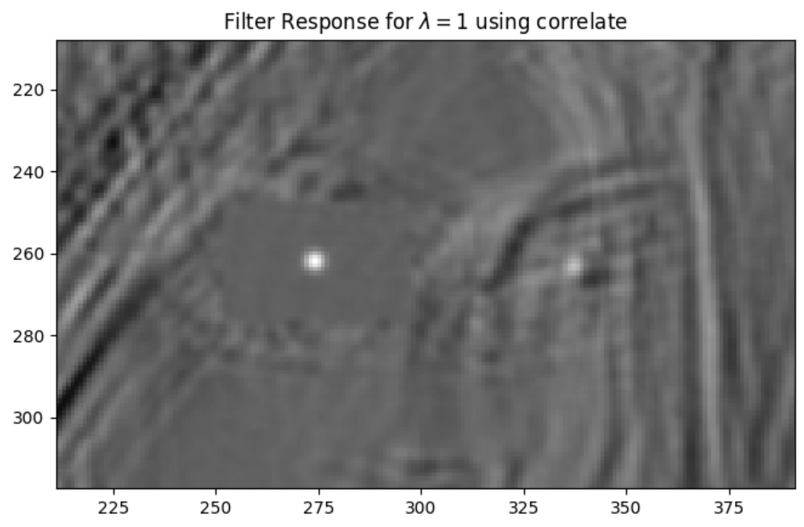

Figure 8: Filter Response on Lena Image for $\lambda = 1$, showing responses on both the eyes

The filter with $\lambda = 1$ performs better because we are performing L2 regularization. In other words we can consider the second term as the radius of our trust region and thus would perform better by decreasing the overlap with the trust region. The regularization term helps us in creating an unique filter from the set of sub-images and thus we have a sharper response at the eye.

The filter with $\lambda = 0$ increases the area of the overlap with the trust region and disables the L2 regularization and hence it gives a blurry response around the eye, which means that the filter correlates with multiple sub-images with neither of them having a strong peak or a response.

4.4 Filter Responses using Convolution

Convolution is a mathematical sliding and window operation that generates some output based on the filter, which is flipped. Correlation is another sliding window dot product operation that measures the similarity between two images. This is the reason why we get the following responses with convolution.

Since in, convolution the only difference is that the filter is flipped, we can do this using Numpy indexing as follows.

```
convolve(image, filter[::-1, ::-1])
```

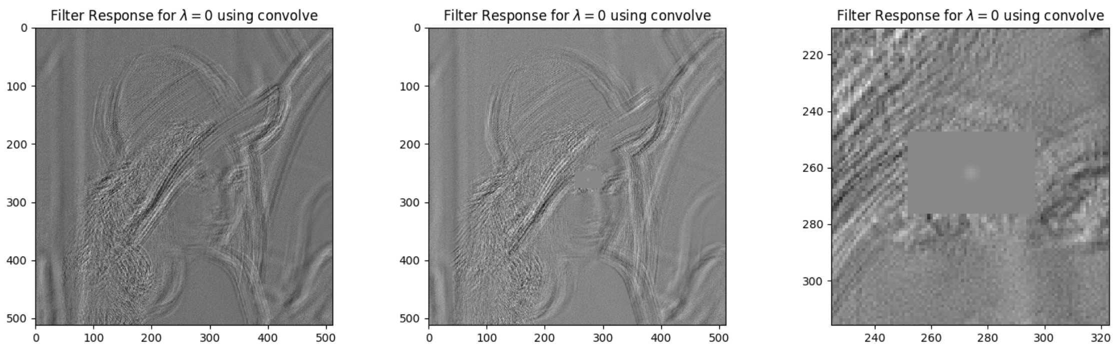
Filter Response for $\lambda = 0$ using convolve
Filter Response for $\lambda = 0$ using convolve
Filter Response for $\lambda = 0$ using convolve

Figure 9: Filter Response on Lena Image for $\lambda = 0$, using Convolution and Flipped Filter Convolution

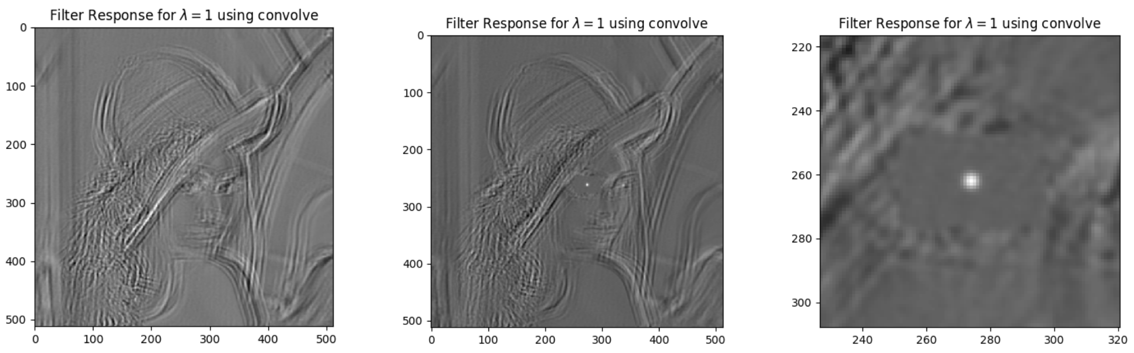
Filter Response for $\lambda = 1$ using convolve
Filter Response for $\lambda = 1$ using convolve
Filter Response for $\lambda = 1$ using convolve

Figure 10: Filter Response on Lena Image for $\lambda = 1$, using Convolution and Flipped Filter Convolution