Arcot Sridhar Kartik Sai Vishnu Kumar
karcotsr@andrew.cmu.edu

16720-B Computer Vision: Homework 3

Q 1.1

a) $\frac{\partial W(x;\boldsymbol{p})}{\partial(\boldsymbol{p^T})}$ is the Jacobian matrix which arises when we expand $\frac{\partial I(W(x;\boldsymbol{p}))}{\partial(\boldsymbol{p^T})}$ with the chain rule. They are a consequence of performing change of variables in Integral and Differential calculus. In the case of Lucas-Kanade algorithm we wanted to differentiate the source image with $\boldsymbol{p^T}$ and therefore had to use chain-rule, to get change of variables, as there was no direct way to perform that differentiation.

In case of just translation, the $\frac{\partial W(x;\boldsymbol{p})}{\partial(\boldsymbol{p^T})}$ can be calculated as-

$$\frac{\partial W(x;\boldsymbol{p})}{\partial(\boldsymbol{p^T})} = \frac{\partial\left(\begin{bmatrix}1 & 0 & p_x\\ 0 & 1 & p_y\end{bmatrix}\begin{bmatrix}x\\ y\\ 1\end{bmatrix}\right)}{\partial(\boldsymbol{p^T})}$$

Expanding and solving we have,

$$\frac{\partial W(x;\boldsymbol{p})}{\partial(\boldsymbol{p^T})} = \begin{bmatrix}\frac{\partial\left(\begin{bmatrix}1 & 0 & p_x\\ 0 & 1 & p_y\end{bmatrix}\begin{bmatrix}x\\ y\\ 1\end{bmatrix}\right)}{\partial(p_x)} & \frac{\partial\left(\begin{bmatrix}1 & 0 & p_x\\ 0 & 1 & p_y\end{bmatrix}\begin{bmatrix}x\\ y\\ 1\end{bmatrix}\right)}{\partial(p_y)}\end{bmatrix}$$

$$\frac{\partial W(x;\boldsymbol{p})}{\partial(\boldsymbol{p^T})} = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

Therefore we see that $\frac{\partial W(x;\boldsymbol{p})}{\partial(\boldsymbol{p^T})}$ is constant irrespective of x and y for translation.

b) The minimisation equation is given as-

$$p^* = \text{argmin}_{\text{p}}\Sigma_{x\in\mathbb{N}}\|I_{t+1}(x+p) - I_t(x)\|_2^2$$

Assuming we have an initial estimate p and we want to find a correction $\Delta\mathbf{p}$ for p to minimise the L2 norm error definer below we can rephrase the equation as-

$$\boldsymbol{\Delta p}^* = \text{argmin}_{\boldsymbol{\Delta p}}\Sigma_{x\in\mathbb{N}}\|I_{t+1}(x+\boldsymbol{p}+\boldsymbol{\Delta p}) - I_t(x)\|_2^2$$

Rewriting as the above equation with a first order Taylor's series approximation we have,

$$\boldsymbol{\Delta p}^* = \text{argmin}_{\boldsymbol{\Delta p}}\sum_{x\in\mathbb{N}}\left\|I_{t+1}\big(W(x;\boldsymbol{p})\big) + \frac{\partial I\big(W(x;\boldsymbol{p})\big)}{\partial(W(x;\boldsymbol{p})^{\boldsymbol{T}})}\frac{\partial W(x;\boldsymbol{p})}{\partial(\boldsymbol{p^T})}\boldsymbol{\Delta p} - I_t(x)\right\|_2^2$$

Simplifying we have,

$$\boldsymbol{\Delta p}^* = \text{argmin}_{\boldsymbol{\Delta p}}\sum_{x\in\mathbb{N}}\left\|\frac{\partial I\big(W(x;\boldsymbol{p})\big)}{\partial(W(x;\boldsymbol{p})^{\boldsymbol{T}})}\frac{\partial W(x;\boldsymbol{p})}{\partial(\boldsymbol{p^T})}\boldsymbol{\Delta p} - \big(I_t(x) - I_{t+1}\big(W(x;\boldsymbol{p})\big)\big)\right\|_2^2$$

Rephrasing the above equation we have,

$$\boldsymbol{\Delta p}^* = \text{argmin}_{\boldsymbol{\Delta p}}\|\boldsymbol{A\Delta p} - \boldsymbol{b}\|_2^2$$

Where,

A can be written as,

Arcot Sridhar Kartik Sai Vishnu Kumar
karcotsr@andrew.cmu.edu

$$\begin{bmatrix} \frac{\partial I\left(W(x_1;\boldsymbol{p})\right)}{\partial (W(x_1;\boldsymbol{p})^{\boldsymbol{T}})} & \cdots & \boldsymbol{0^T} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0^T} & \cdots & \frac{\partial I\left(W(x_N;\boldsymbol{p})\right)}{\partial (W(x_N;\boldsymbol{p})^{\boldsymbol{T}})} \end{bmatrix} \begin{bmatrix} \frac{\partial W(x_1;\boldsymbol{p})}{\partial (\boldsymbol{p^T})} \\ \vdots \\ \frac{\partial W(x_N;\boldsymbol{p})}{\partial (\boldsymbol{p^T})} \end{bmatrix}$$

b can be written as,

$$\begin{bmatrix} (I_t(x_1) - I_{t+1}\left(W(x_1;\boldsymbol{p})\right)) \\ \vdots \\ (I_t(x_N) - I_{t+1}\left(W(x_N;\boldsymbol{p})\right)) \end{bmatrix}$$

c)

$$\boldsymbol{\Delta p}^* = \text{argmin}_{\boldsymbol{\Delta p}} \|\boldsymbol{A\Delta p} - \boldsymbol{b}\|_2^2$$

We solve the above minimisation equation by calculating $\boldsymbol{\Delta p}$ –

$$\boldsymbol{\Delta p} = (\boldsymbol{A^T A})^{-1} \boldsymbol{A^T b}$$

We would've had a unique solution if $\boldsymbol{A^T A}$ was invertible. That is, all columns in A are linearly independent.

Arcot Sridhar Kartik Sai Vishnu Kumar
karcotsr@andrew.cmu.edu

Q1.3

Uncomment visualisation code segment in testCarSequence.py (line 34) for visualising output.

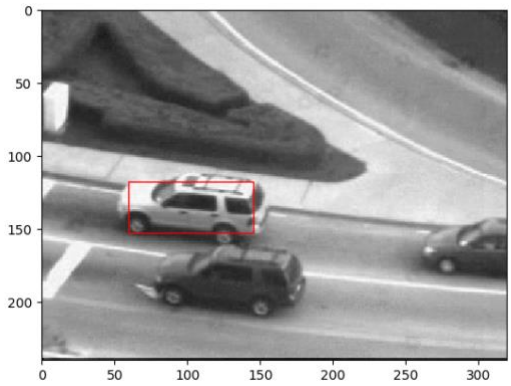

Figure 1: Frame-1

Figure 2: Frame-100

Figure 3: Frame-200

Figure 4: Frame-300

Figure 5: Frame-400

Arcot Sridhar Kartik Sai Vishnu Kumar
karcotsr@andrew.cmu.edu

Q1.4

Uncomment visualisation code segment in testCarSequenceWithTemplateCorrection.py (line 88) for visualising output.

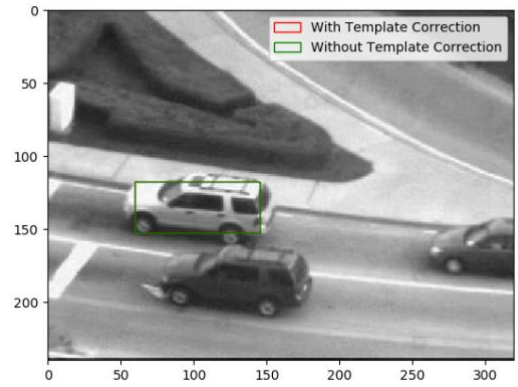

Figure 6: Frame-1

Figure 7: Frame-100

Figure 8: Frame-200

Figure 9: Frame-300

Figure 10: Frame-400

Arcot Sridhar Kartik Sai Vishnu Kumar
karcotsr@andrew.cmu.edu

Q2.1

We have,

$$I_{t+1}(\boldsymbol{x}) = I_t(\boldsymbol{x}) + \sum_{k}^{K} w_k \mathcal{B}_k(\boldsymbol{x})$$

Where, $\{\mathcal{B}_k\}_{k=1}^{K}$is the set of basis images of the template and $w_k$is the weight associated with each basis.

Let, $\boldsymbol{B}(\boldsymbol{x})$ be a row vector with values of the bases images at $\boldsymbol{x}$. Then-

$$\boldsymbol{B}(\boldsymbol{x}) = [\mathcal{B}_1(\boldsymbol{x}) \quad \cdots \quad \mathcal{B}_{k-1}(\boldsymbol{x}) \quad \mathcal{B}_k(\boldsymbol{x})]$$

Let $\boldsymbol{w}$ be a column vector with the weights associated with each bases. Then-

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{bmatrix}$$

Then the equation for I_{t+1} can be given as –

$$I_{t+1}(\boldsymbol{x}) = I_t(\boldsymbol{x}) + \boldsymbol{B}(\boldsymbol{x})\boldsymbol{w}$$

We can flatten the image as,

$$I_{t+1} = \begin{bmatrix} I_{t+1}(\boldsymbol{x_1}) \\ I_{t+1}(\boldsymbol{x_2}) \\ \vdots \\ I_{t+1}(\boldsymbol{x_n}) \end{bmatrix}$$

We can flatten the basis as, of dimension $n \times 1$,

$$\mathcal{B}_k = \begin{bmatrix} \mathcal{B}_k(\boldsymbol{x_1}) \\ \mathcal{B}_k(\boldsymbol{x_2}) \\ \vdots \\ \mathcal{B}_k(\boldsymbol{x_n}) \end{bmatrix}$$

We can stack all the basis as, of dimension $n \times k$,

$$\boldsymbol{B} = \begin{bmatrix} \mathcal{B}_1(\boldsymbol{x_1}) & \dots & \mathcal{B}_k(\boldsymbol{x_1}) \\ \mathcal{B}_1(\boldsymbol{x_2}) & \dots & \mathcal{B}_k(\boldsymbol{x_2}) \\ \vdots & \vdots & \vdots \\ \mathcal{B}_1(\boldsymbol{x_n}) & \dots & \mathcal{B}_k(\boldsymbol{x_n}) \end{bmatrix}$$

We can write the equation for I_{t+1} as-

$$I_{t+1} = I_t + \boldsymbol{B}\boldsymbol{w}$$

Which can be simplified as-

$$\boldsymbol{B^T B w} = \boldsymbol{B^T}[I_{t+1} - I_t]$$

because **B** is a set of orthogonal bases. Therefore we have,

$$\begin{bmatrix} \|\mathcal{B}_1\|_2^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \|\mathcal{B}_k\|_2^2 \end{bmatrix} \boldsymbol{w} = \boldsymbol{B^T}[I_{t+1} - I_t]$$

Simplifying the weight vector can be given as-

$$\boldsymbol{w} = \begin{bmatrix} \frac{1}{\|\mathcal{B}_1\|_2^2} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \frac{1}{\|\mathcal{B}_k\|_2^2} \end{bmatrix} \boldsymbol{B^T}[I_{t+1} - I_t]$$

Arcot Sridhar Kartik Sai Vishnu Kumar
karcotsr@andrew.cmu.edu

Q2.3

Uncomment visualisation code segment in testSylvSequence.py (line 42) for visualising output.

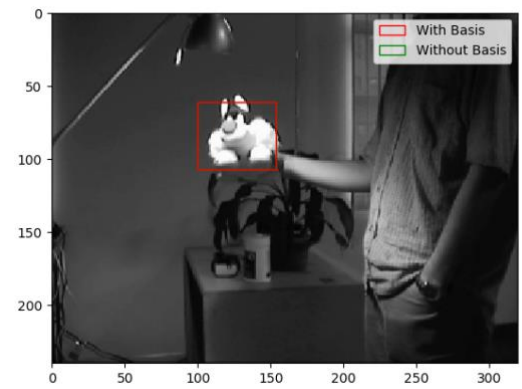

Figure 11: Frame-1

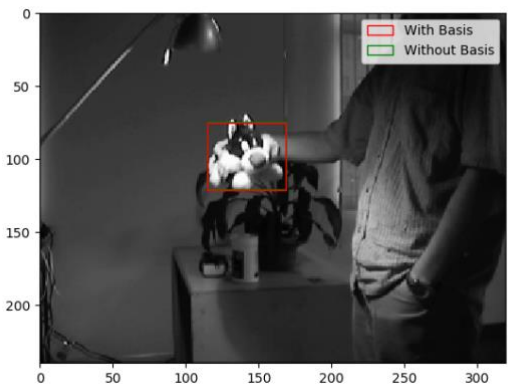

Figure 12: Frame-200

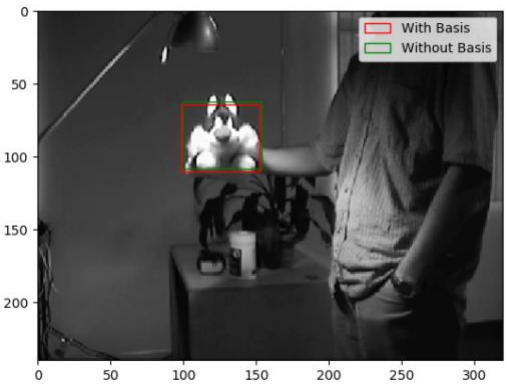

Figure 13: Frame-300

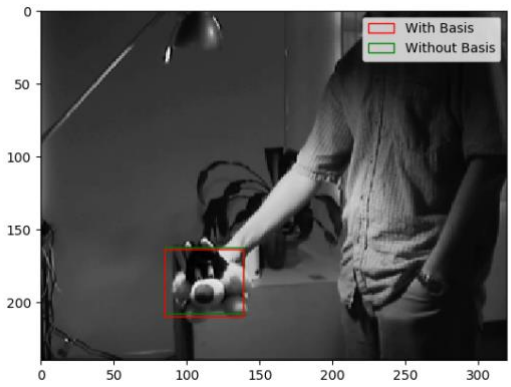

Figure 14: Frame-350

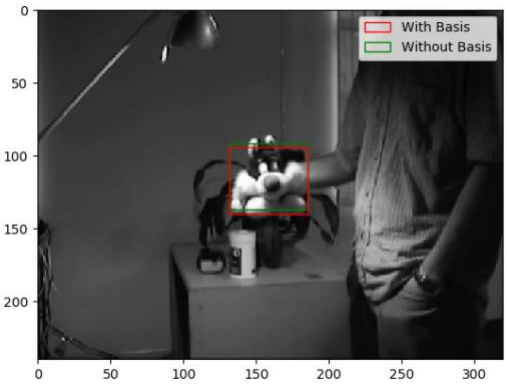

Figure 15: Frame-400

Arcot Sridhar Kartik Sai Vishnu Kumar
karcotsr@andrew.cmu.edu

Q3.3

Uncomment visualisation code segment in testAerialSequence.py (line 25) for visualising output.

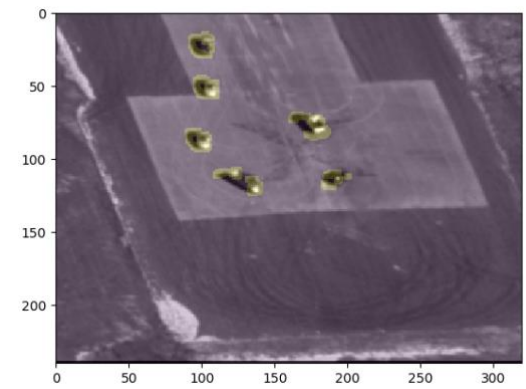

Figure 16: Frame-30

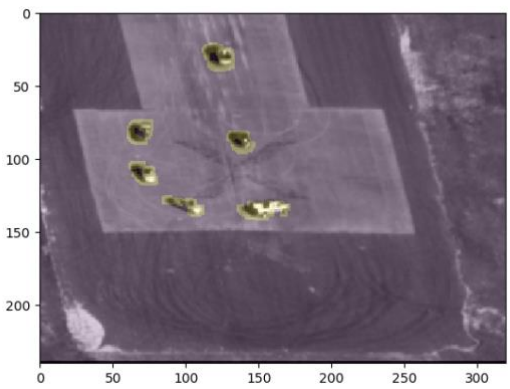

Figure 17: Frame-60

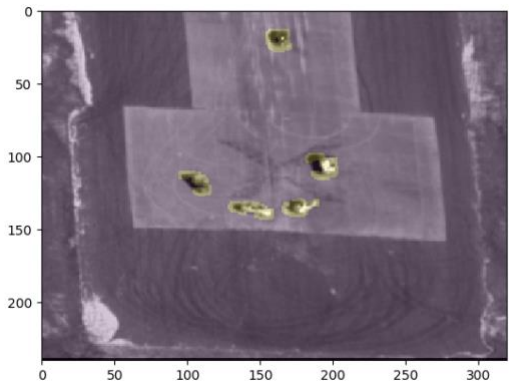

Figure 18: Frame-90

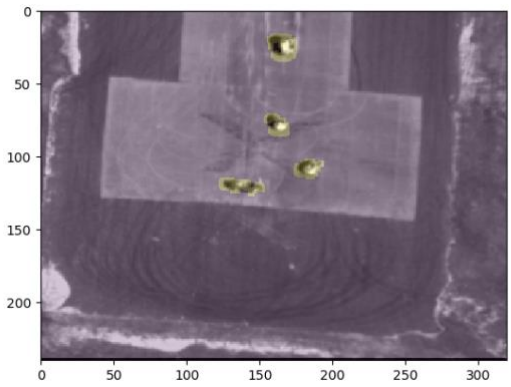

Figure 19: Frame-120

Arcot Sridhar Kartik Sai Vishnu Kumar
karcotsr@andrew.cmu.edu

Q4.1

In the usual Forward Additive method at each iteration of the Lucas Kanade, to compute the least squares solution we recompute the Hessian (Warped gradient and the Warp Jacobian) because in each update the parameters $\boldsymbol{p}$ keeps changing.

However, in Inverse Compositional method takes advantage of the fact that most warps are closed under composition [1]. Therefore instead of computing steepest descent and finding $\boldsymbol{\Delta p}$ on warped source image, we can instead do it on the template and compose our current estimate of $\boldsymbol{p}$ with $\boldsymbol{\Delta^{-1}p}$**.** What this means is that instead of updating the parameters by solving the equation derived for forward additive method, we can instead solve this-

$$\boldsymbol{\Delta p}^{*} = \text{argmin}_{\boldsymbol{\Delta p}} \sum_{x \in \mathbb{N}} \left\| \frac{\partial T\big(W(x;\mathbf{0})\big)}{\partial (W(x;\mathbf{0})^{T})} \frac{\partial W(x;\mathbf{0})}{\partial (\boldsymbol{p}^{T})} \boldsymbol{\Delta p} - \big(I_{t+1}\big(W(x;\boldsymbol{p})\big) - (I_t(x)) \right\|_2^2$$

We can see that all the terms in the above equation, except for $I_{t+1}\big(W(x;\boldsymbol{p})\big)$, can be precomputed. The only computationally intensive step would be warping the source image with the current estimate of p and solving for the least squares by simply multiplying the precomputed Hessian with the error image, $\big(I_{t+1}\big(W(x;\boldsymbol{p})\big) - (I_t(x))$. Therefore we can clearly see that Inverse Compositional update is going to computationally much more efficient.

[1] - Baker, Simon, and Iain Matthews. "Equivalence and efficiency of image alignment algorithms." *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 1. IEEE Computer Society; 1999, 2001.

Q4.2

$$\underset{\boldsymbol{g}}{\mathbf{argmin}} \left(\frac{\mathbf{1}}{\mathbf{2}} \|\boldsymbol{y} - \boldsymbol{X}^{T}\boldsymbol{g}\|_2^2 + \frac{\boldsymbol{\lambda}}{\mathbf{2}} \|\boldsymbol{g}\|_2^2 \right)$$

Rephrasing the norm terms we have,

$$\underset{\boldsymbol{g}}{\mathbf{argmin}} \left(\frac{\mathbf{1}}{\mathbf{2}} (\boldsymbol{y} - \boldsymbol{X}^{T}\boldsymbol{g})^{\boldsymbol{T}} (\boldsymbol{y} - \boldsymbol{X}^{T}\boldsymbol{g}) + \frac{\boldsymbol{\lambda}}{\mathbf{2}} \boldsymbol{g}^{T}\boldsymbol{g} \right)$$

To find minimum we differentiate with respect to g,

$$\frac{\boldsymbol{\partial}\left(\frac{\mathbf{1}}{\mathbf{2}} (\boldsymbol{y} - \boldsymbol{X}^{T}\boldsymbol{g})^{\boldsymbol{T}} (\boldsymbol{y} - \boldsymbol{X}^{T}\boldsymbol{g})\right)}{\boldsymbol{\partial g}} + \frac{\frac{\boldsymbol{\lambda}}{\mathbf{2}} \boldsymbol{\partial}(\boldsymbol{g}^{T}\boldsymbol{g})}{\boldsymbol{\partial g}} = \mathbf{0}$$

$$\frac{\boldsymbol{\partial}\left(\frac{\mathbf{1}}{\mathbf{2}} (\boldsymbol{y}^{T}\boldsymbol{y} - \boldsymbol{y}^{T}\boldsymbol{X}^{T}\boldsymbol{g} - \boldsymbol{g}^{T}\boldsymbol{X}\boldsymbol{y} + \boldsymbol{g}^{T}\boldsymbol{X}\boldsymbol{X}^{T}\boldsymbol{g})\right)}{\boldsymbol{\partial g}} + \frac{\frac{\boldsymbol{\lambda}}{\mathbf{2}} \boldsymbol{\partial}(\boldsymbol{g}^{T}\boldsymbol{g})}{\boldsymbol{\partial g}} = \mathbf{0}$$

$$\frac{\mathbf{1}}{\mathbf{2}} ((-\boldsymbol{y}^{T}\boldsymbol{X}^{T})^{T} - (\boldsymbol{X}\boldsymbol{y}) + (\boldsymbol{X}\boldsymbol{X}^{T} + \boldsymbol{X}\boldsymbol{X}^{T})\boldsymbol{g}) + \boldsymbol{\lambda}\boldsymbol{g} = \mathbf{0}$$

$$\frac{\mathbf{1}}{\mathbf{2}} (-\mathbf{2}\boldsymbol{X}\boldsymbol{y} + \mathbf{2}(\boldsymbol{X}\boldsymbol{X}^{T})\boldsymbol{g}) + \boldsymbol{\lambda}\boldsymbol{g} = \mathbf{0}$$

$$(\boldsymbol{X}\boldsymbol{X}^{T} + \boldsymbol{\lambda}\boldsymbol{I})\boldsymbol{g} = \boldsymbol{X}\boldsymbol{y}$$

Simplifying we have the solution for $\boldsymbol{g}$,

$$\boldsymbol{g} = (\boldsymbol{X}\boldsymbol{X}^{T} + \boldsymbol{\lambda}\boldsymbol{I})^{-1}\boldsymbol{X}\boldsymbol{y}$$

Simplifying with $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^{T}$ we get,

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

Arcot Sridhar Kartik Sai Vishnu Kumar
karcotsr@andrew.cmu.edu

Q4.3

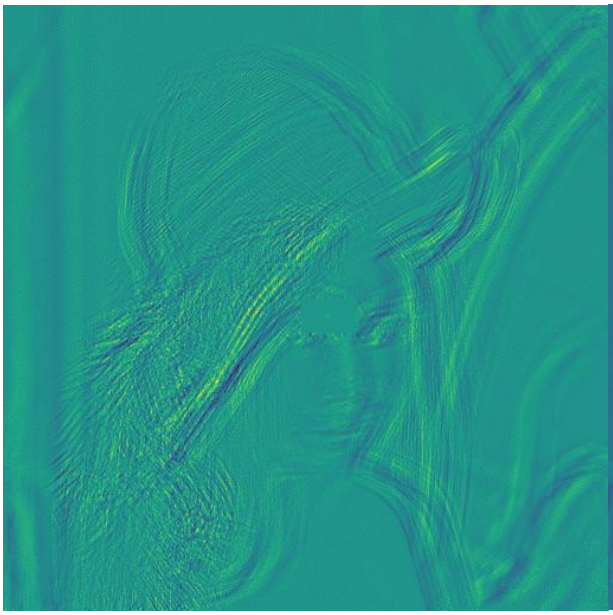

Figure 20: Lambda=0

Figure 21: Lambda=1

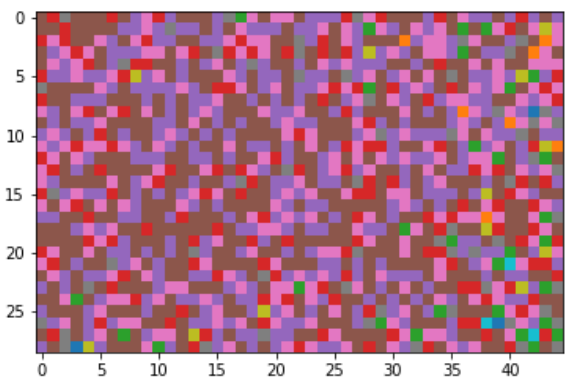

Figure 22: Correlation filter from λ =0, cmap=Tab10

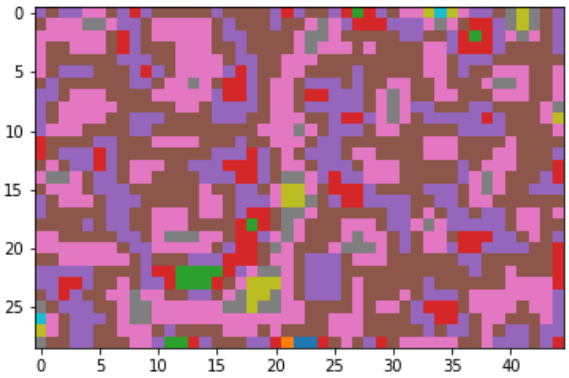

Figure 23: Correlation filter from λ =1, cmap=Tab10

We can see for λ=0 only one response is generated on of the eyes. Whereas with λ =1 the response is generated on both the eyes. Additionally, when we look at the correlation filters generated with different lambdas, below, we can see that for λ=0 it is overfitted. Therefore non-zero λ gives better results and more robust to variation.

Explanation:

Providing a non-zero lambda and solving for the filter g, from the minimisation equation derived in Q4.2, is equivalent to adding a regularisation term in our regression equation. We perform regularisation to avoid our learnt filter from overfitting with the noise present in our data.

Furthermore another mathematical argument is, for λ=0, the equation-

$$\boldsymbol{g = (XX^T + \lambda I)^{-1} Xy}$$

will have an unique solution if and only if $\boldsymbol{XX^T}$ (XX^T is positive semidefinite) is invertible. In cases of matrices of images, there is no guarantee that they will be full rank and invertible. Hence, by adding $\boldsymbol{\lambda I}$ we ensure that we make a positive definite matrix which will always be invertible and thereby get a unique solution. Although we are inducing bias into the solution we will ensure that our solution does not overfit and is more robust.

Arcot Sridhar Kartik Sai Vishnu Kumar
karcotsr@andrew.cmu.edu

Q4.4

Below are the response you get when we convolve the correlation filter with the image.

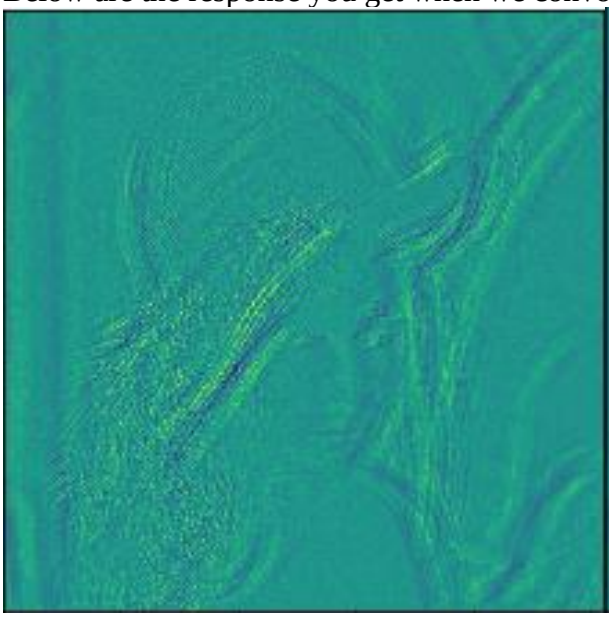

Figure 24: Lambda=1

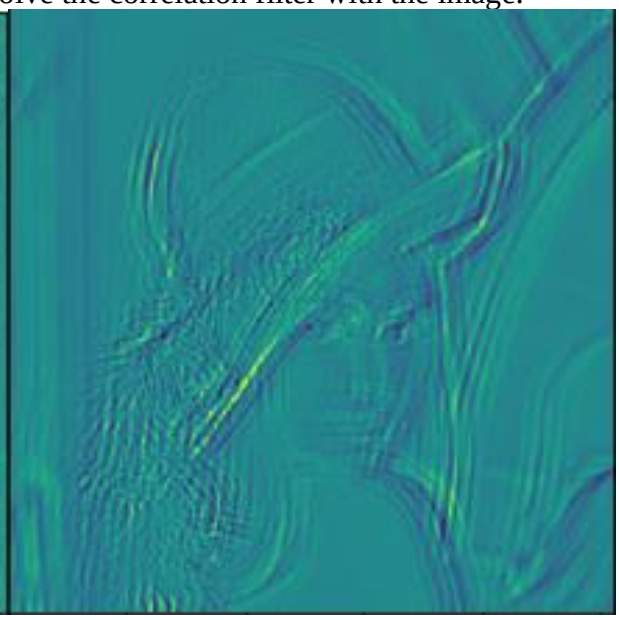

Figure 25: Lambda=0

We see different response with convolution operation because, in the equation -

$$\underset{g}{\mathbf{argmin}} \left(\frac{1}{2} \|y - X^T g\|_2^2 + \frac{\lambda}{2} \|g\|_2^2 \right)$$

the term $X^T g$, is circulant correlation represented as a matrix multiplication. (X is a circulant Toeplitz form of an image). Therefore, the filter g derived from this equation will give the desired impulse like response with a correlation operation. Whereas if we want a convolution filter we should solve this equation instead-

$$\underset{h}{\mathbf{argmin}} \left(\frac{1}{2} \|y - Xh\|_2^2 + \frac{\lambda}{2} \|h\|_2^2 \right)$$

We can correct this by flipping our correlation filter over both the axes and then convolve. This can be done using numpy.flip() function. i.e. h = numpy.flip(g) will give us the convolution filter to use from the correlation filter. When we use the flipped correlation filter and perform convolution we get-

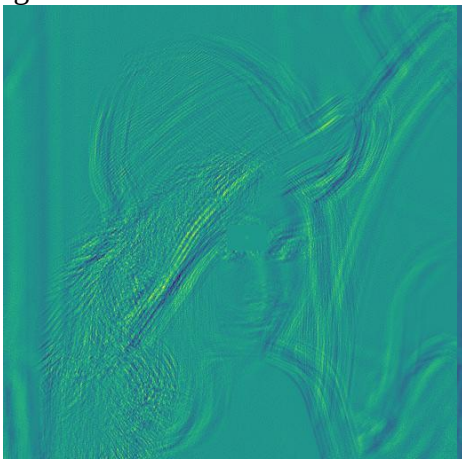

Figure 26: Lambda=1

Figure 27: Lambda=0

We can see that the above results are exactly like what we got by applying correlation filter on the image with the correlate function as observed in **Q4.3**.