16-720B Computer Vision: Homework 2
Lucas-Kanade Tracking and Correlation Filters

Daniel Marew

October 25, 2018

1 Lucas-Kanade Tracking

1.1

- What is $\frac{\partial W(x;p)}{p^T}$

$$W(x;p) = \begin{bmatrix} x + p_1 \\ y + p_2 \end{bmatrix}$$

$$W(x;p) = \begin{bmatrix} W_x(x;p) \\ W_y(x;p) \end{bmatrix}$$

$$W_x(x;p) = x + p_1$$

$$W_y(x;p) = y + p_2$$

where $\boldsymbol{x} = [x, y]^T$ and $\boldsymbol{p} = [p_1, p_2]^T$

$$\frac{\partial W(x;p)}{\partial p^T} = \begin{bmatrix} \frac{\partial W_x(x;p)}{\partial p_1} & \frac{\partial W_x(x;p)}{\partial p_2} \\ \frac{\partial W_y(x;p)}{\partial p_1} & \frac{\partial W_y(x;p)}{\partial p_2} \end{bmatrix}$$

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

- What are A and b?

$$\mathcal{I}_{t+1}(x + \boldsymbol{\Delta p}) \approx \mathcal{I}_{t+1}(x\prime) + \frac{\partial \mathcal{I}_{t+1}(x\prime)}{\partial x\prime^T} \frac{\partial W(x;\boldsymbol{p})}{\partial \boldsymbol{p^T}} \boldsymbol{\Delta p} \quad (1)$$

$$\sum_{x \in N} ||\mathcal{I}_{t+1}(x+\Delta p) - \mathcal{I}_t(x)||_2^2 = \sum_{n=1}^{N} ||\mathcal{I}_{t+1}(W(x_n;\boldsymbol{\Delta p})) + \frac{\partial \mathcal{I}(W(x_n;p))}{\partial p^T} \Delta p - \mathcal{I}_t(x)||_2^2 \quad (2)$$

$$= \sum_{n=1}^{N} ||\frac{\partial \mathcal{I}_{t+1}(W(x_n;p))}{\partial p^T} \Delta p + \mathcal{I}_{t+1}(W(x_n;\boldsymbol{\Delta p})) - \mathcal{I}_t(x)||_2^2$$

$$= \sum_{n=1}^{N} ||\frac{\partial \mathcal{I}_{t+1}(W(x_n;p))}{\partial p^T} \Delta p - (\mathcal{I}_t(x) - \mathcal{I}_{t+1}(W(x_n;\boldsymbol{\Delta p})))||_2^2$$

we can rewrite equation 2 in vector form as

$$\Delta p = argmin_{\Delta p} ||\frac{\partial \mathcal{I}_{t+1}\prime}{\partial p^T} \Delta p - (\mathcal{I}_t - \mathcal{I}_{t+1}\prime||_2^2$$

$$\frac{\partial \mathcal{I}_{t+1}(W(x_n;p))}{\partial p^T} = \frac{\partial \mathcal{I}_{t+1}(W(x_n;p))}{\partial W(x_n;p)} \frac{\partial W(x_n;p)}{\partial p^T} = \frac{\partial \mathcal{I}_{t+1}(x_n\prime)}{\partial x_n\prime} \frac{\partial W(x_n;p)}{\partial p^T}$$

Therefore $\boldsymbol{A}$ is the steepest descent image and is given by:

$$\boldsymbol{A = \nabla I_{t+1}\prime \frac{\partial W}{\partial p^T}}$$

Since $\frac{\partial W}{\partial p^T} = I$ for translation

$$\boldsymbol{A = \nabla I_{t+1}\prime}$$

where $\nabla I_{t+1}\prime$ is the warped gradient image.
$\boldsymbol{b}$ is the error image is given by:

$$\boldsymbol{b = \mathcal{I}_t - \mathcal{I}_{t+1}\prime}$$

where $\mathcal{I}_{t+1}\prime$ is warped $\mathcal{I}_{t+1}$

- What conditions must $A^T A$ meet so that a unique solution to Δp can be found?

  The optimal value for Δp that minimizes equation 2 is given by

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

  for Δp to be unique $A^T A$ has to be **invertible**.

1.2

1.3

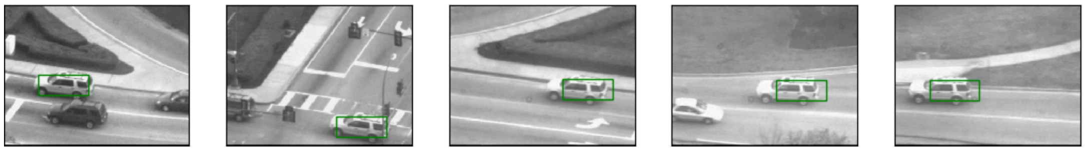


Figure 1: Lucas-Kanade Tracking with One Single Template

1.4

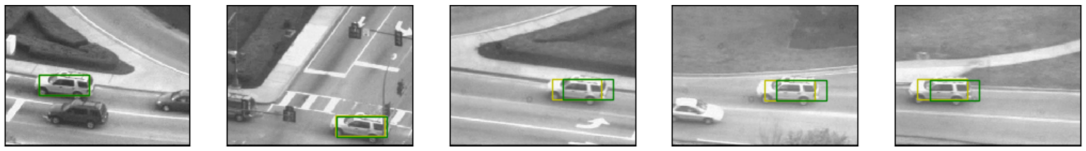


Figure 2: Lucas-Kanade Tracking with Template Correction

2 Lucas-Kanade Tracking with Appearance Basis

2.1 Appearance Basis

$$\boldsymbol{\mathcal{I}_{t+1}(x) = \mathcal{I}_t(x) + \sum_{k=1}^{K}} w_k \boldsymbol{\mathcal{B}_k} \tag{3}$$

Equation 4 can be expressed in vector form as:

$$\boldsymbol{I_{t+1} = I_t + Bw}$$

$$\boldsymbol{I_{t+1} - I_t = Bw}$$

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

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

Since the columns of $\boldsymbol{B}$ are orthobases $\boldsymbol{B^{-1} = B^T}$

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

2.2

2.3

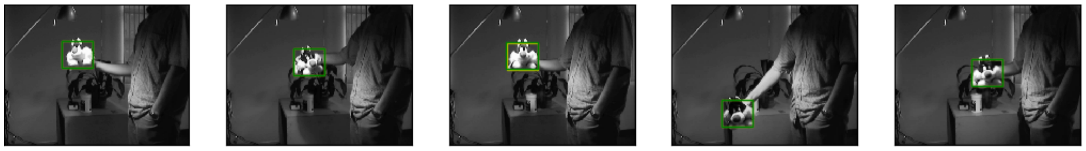


Figure 3: Lucas-Kanade Tracking with Appearance Basis

Both tracker perform well for this task that is why the bounding boxes overlap.

3 Affine Motion Subtraction

3.1

3.2

3.3

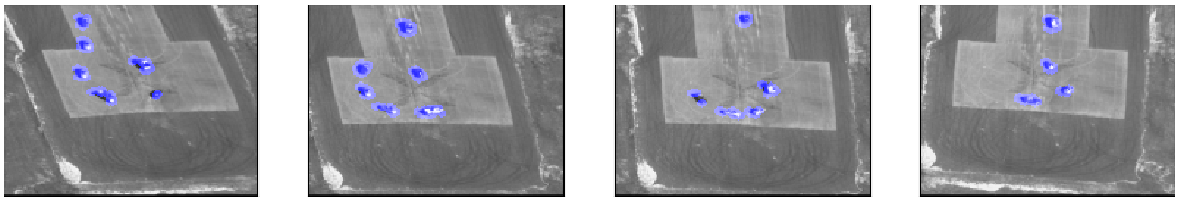


Figure 4: Lucas-Kanade Affine Motion Subtraction

4 Efficient Tracking

4.1 Inverse Composition

Describe why the inverse compositional approach is more computationally efficient than the classical approach?

In the classical approach, the most computationally intensive operation is computing the Hessian (i.e $A^T A$) which has to be done in every iteration. Inverse compositional algorithm cleaverly avoids re-computing this computationally intensive operation by re-stating the optimization objective such that the Hessian remains constant in every iteration and therefore can be pre-computed. for this reason, inverse compositional approach is more computationally efficient than the classical Lucas-Kande algorithm.

4.2 Correlation Filters

$$\boldsymbol{\mathcal{L}(X, y; g) = \frac{1}{2}||y - X^T g||_2^2 + \frac{\lambda}{2}||g||_2^2} \tag{4}$$

$$\boldsymbol{\mathcal{L}(X, y; g) = \frac{1}{2}(y - X^T g)^T (y - X^T g) + \frac{\lambda}{2} g^T g}$$

$$\boldsymbol{\mathcal{L}(X, y; g) = \frac{1}{2}(y^T - g^T X)(y - X^T g) + \frac{\lambda}{2} g^T g}$$

$$\boldsymbol{\mathcal{L}(X, y; g) = \frac{1}{2}(y^T y - y^T X^T g - g^T X y + g^T X X^T g) + \frac{\lambda}{2} g^T g}$$

$$\boldsymbol{\frac{\partial \mathcal{L}(X, y; g)}{\partial g} = \frac{1}{2}(-Xy - Xy + 2XX^T g) + \lambda g}$$

$$\boldsymbol{\frac{\partial \mathcal{L}(X, y; g)}{\partial g} = -Xy + XX^T g + \lambda g}$$

Setting $\boldsymbol{\frac{\partial \mathcal{L}(X,y;g)}{\partial g} = 0}$

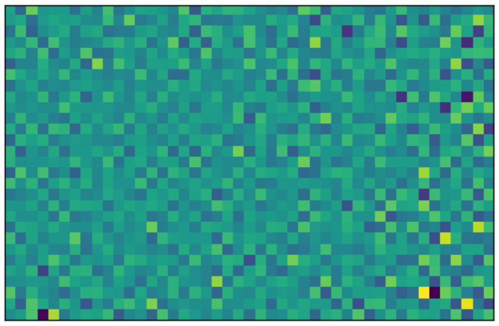
$$\boldsymbol{-Xy + XX^T g + \lambda g = 0}$$

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

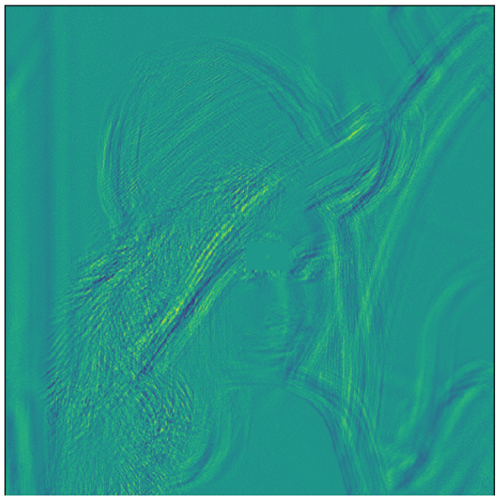
$$\boldsymbol{g = (XX^T + \lambda I)^{-1} Xy}$$

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

Ans. $\boldsymbol{g = (S + \lambda I)^{-1} Xy}$

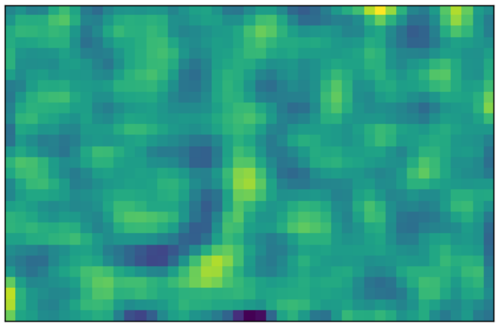


(a) Visualization of g with $\lambda = 0$.

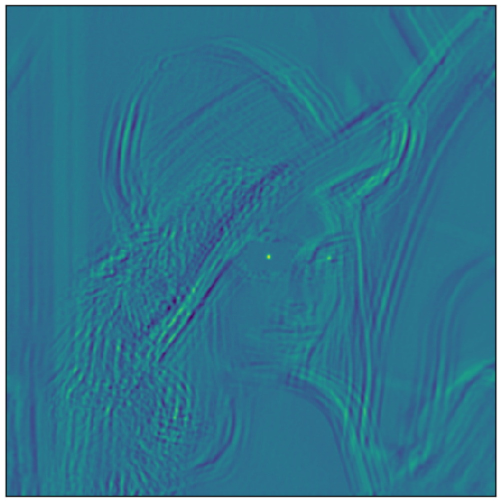


(b) response of g on Lena image using correlation $\lambda = 0$

Figure 5: Visualization of g and its response on Lena image using correlation with $\lambda = 0$



(a) Visualization of g with $\lambda = 1$.

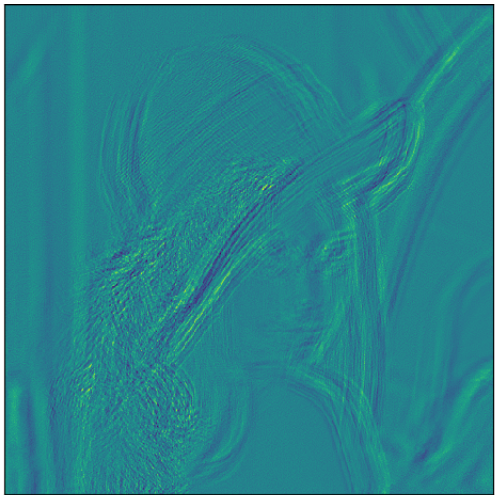


(b) response of g on Lena image using correlation $\lambda = 1$.

Figure 6: Visualization of g and its response on Lena image using correlation with $\lambda = 1$.

Can you comment on which value of λ performs best and why?

As we can see on figure 6 the optimal $\boldsymbol{g}$ with $\lambda = 1$ performs best since we get maximum response at the location of the template image (Lena's left Eye) and attenuated response elsewhere. When we set $\lambda = 1$. it adds a constraint on the optimization objective (trust region) in such away that forces the optimizer to pick small values for $\boldsymbol{g}$. This is important because, it stops g from arbitrarily amplifying noise. That is why we get a better response with $\lambda = 1$ (with regularization) than $\lambda = 0$ (with out regularization).


(a) response of g on Lena image using convolution with $\lambda = 0$.

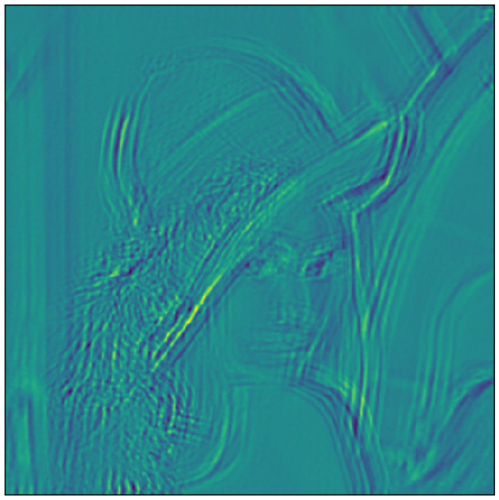

(b) response of g on Lena image using convolution with $\lambda = 1$.

Figure 7: response on Lena image using convolution with $\lambda = 0$. and $\lambda = 1$.

Why does this get a different response to the one obtained using correlate and How could you use the numpy indexing operations to get a response more similar to the one obtained using correlate?

Convolution and correlation are related but are not the same operations. That is why we get a different response with convolution from the one we got with correlation. The response we get after Performing correlation on an image with a filter is the same reponse as the respose we get if we rotatated the filter by 180 degrees and perform convolution on the image. In order to get the same response as correlation, we need to rotate the filter g by 180 degrees. We can do that in numpy in the following way

$g\prime$ = np.flipud(np.fliplr(g))
response = convolve(image, $g\prime$)