2018 - 17119 박수현 컴퓨터비전 HW4 Theory

1 Stereo

Question 1

[10 points] Two cameras are said to form a *rectified pair* if their camera coordinate systems differ only by a translation of their origins (*i.e.*, the camera centers) along a direction that is parallel to either the x or y axis of the coordinate systems.

1. Derive an expression for the essential matrix **E** of a rectified pair.

Essential matrix E relates corresponding image points between both cameras, given the rotation and translation, i.e., image point from camera reference frame 2 (X_c') can be expressed in relation to image point from camera reference frame 1 (X_c), 3x3 rotation matrix R, and 3x1 translation vector T : $X_c' = RX_c + T$.

To derive an expression for the essential matrix E in general cases,

$X' = RX + T$

$T \times X' = T \times RX + \underbrace{T \times T}_{=0} = T \times RX$ ↓ T_x on both sides

$X' \cdot (T \times X') = X' \cdot (T \times RX) = X' \cdot ([T_x] RX) = 0$ (∵ X' lies in the epipolar plane normal to $T \times RX$)

$E = [T_x]R$ s.t. $X'^T E X = 0$ ... (A)

Given the problem, there is no rotation between the cameras ($R = I$).

If translation is parallel to the x- axis, then $T = [T_x \; 0 \; 0]^T$

and $E = [T_x]R = [T_x]I = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T_x \\ 0 & T_x & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T_x \\ 0 & T_x & 0 \end{bmatrix}$

If translation is parallel to the y- axis, then $T = [0 \; T_x \; 0]^T$

and $E = [T_x]R = [T_x]I = \begin{bmatrix} 0 & 0 & T_x \\ 0 & 0 & 0 \\ -T_x & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & T_x \\ 0 & 0 & 0 \\ -T_x & 0 & 0 \end{bmatrix}$

2. Prove that the epipolar lines of a rectified pair are parallel to the axis of translation.

Epipolar lines are intersections of the epipolar plane with image planes. Correspondence points lie only on epipolar lines, by epipolar geometry.

Let $p = [x \; y \; f]^T$ be any point in camera reference frame 1

$p' = [x' \; y' \; f]^T$ ~ 2.

Since $p'^T E p = 0$ from equation (A) derived from question 1-1,

If the axis of translation is the x-axis,

$$[x'\ y'\ f]\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & -T_x\\ 0 & T_x & 0\end{bmatrix}\begin{bmatrix}x\\ y\\ f\end{bmatrix} = 0 \iff [x'\ y'\ f]\begin{bmatrix}0\\ -T_x f\\ T_x y\end{bmatrix} = 0 \iff -y' T_x f + f T_x y = 0$$

$$\iff y' = y$$

Hence, every pixel correspondence have the same y-coordinate, and the epipolar line of the rectified pair is parallel to the x-axis.

If the axis of translation is the y-axis,

$$[x'\ y'\ f]\begin{bmatrix}0 & 0 & T_x\\ 0 & 0 & 0\\ -T_x & 0 & 0\end{bmatrix}\begin{bmatrix}x\\ y\\ f\end{bmatrix} = 0 \iff [x'\ y'\ f]\begin{bmatrix}T_x f\\ 0\\ -T_x x\end{bmatrix} = 0 \iff x' T_x f - f T_x x = 0$$

$$\iff x' = x$$

Hence, every pixel correspondence have the same x-coordinate, and the epipolar line of the rectified pair is parallel to the y-axis.

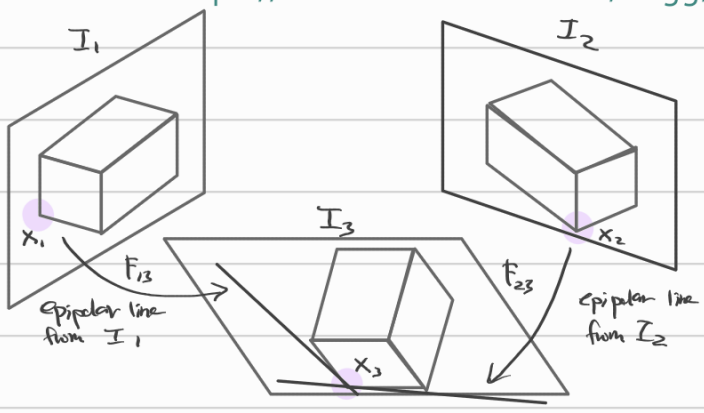
$\therefore$ In general, the epipolar lines of a rectified pair is parallel to the axis of translation.

Question 2

[15 points] Consider three images I_1, I_2 and I_3 that have been captured by a system of three cameras, and suppose the fundamental matrices $\mathbf{F}_{13}$ and $\mathbf{F}_{23}$ are known. (Notation: the matrix $\mathbf{F}_{ij}$ satisfies the equation $\mathbf{x}_j^\top \mathbf{F}_{ij} \mathbf{x}_i = 0$ for any correspondence $\mathbf{x}_i \leftrightarrow \mathbf{x}_j$ between images I_i and I_j.) In general, given a point $\mathbf{x}_1$ in I_1 and a corresponding point $\mathbf{x}_2$ in I_2, the corresponding point in $\mathbf{x}_3$ in I_3 is uniquely determined by the fundamental matrices $\mathbf{F}_{13}$ and $\mathbf{F}_{23}$.

1. Write an expression for $\mathbf{x}_3$ in terms of $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{F}_{13}$ and $\mathbf{F}_{23}$.

Reference: https://www.robots.ox.ac.uk/~vgg/hzbook/hzbook2/HZtrifocal.pdf



* For question 2, I denote vectors by indicating arrows on top (e.g., $\vec{x_1}$) in order to distinguish them from the cross product operator.

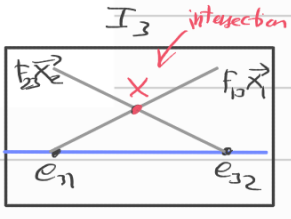
$\vec{x_3}$ must lie on the epipolar line $F_{13}\vec{x_1}$ corresponding to $\vec{x_1}$, and similarly $F_{23}\vec{x_2}$ for $\vec{x_2}$. As $\vec{x_3}$ must lie on both lines, taking the intersection of the epipolar lines gives

$$\vec{x_3} = (F_{13}\vec{x_1}) \times (F_{23}\vec{x_2})$$ ← cross product of the line vectors

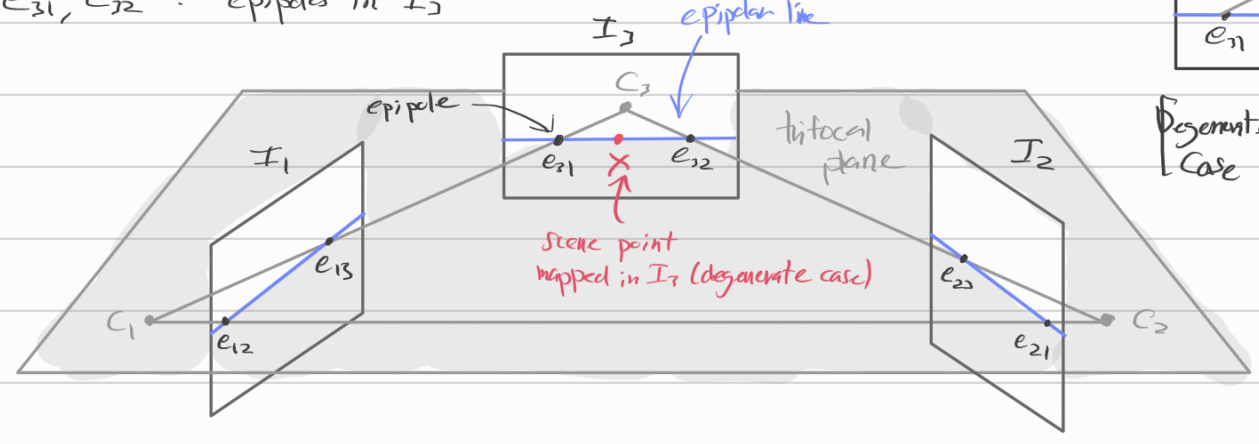
2. Describe a degenerate configuration of three cameras for which the point $\mathbf{x}_3$ cannot be uniquely determined by this expression.

Hint: Consider the epipolar geometry of the situation. Draw a picture!

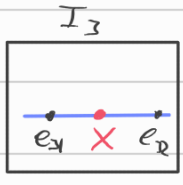
[Normal case]



e_{31}, e_{32} := epipoles in I_3



[Degenerate Case]



Point $\vec{x_3}$ cannot be uniquely determined when epipolar lines $F_{13}\vec{x_1}$ and $F_{23}\vec{x_2}$ coincide and they have an infinitude of points in common. The condition in which this happens is when the 3D scene point X lies on the plane through the three camera centers (:= trifocal plane), as illustrated above. ($\vec{x_3}$, e_{31}, e_{32} are collinear).

To a more extreme case, if the three camera centers are collinear, $e_{31} = e_{32}$ and point $\vec{x_3}$ cannot be determined by any means.

Question 3

[15 points] Suppose two cameras fixate on a point P (Figure 1) in space such that their optical axes intersect at that point. Show that if the image coordinates are normalized so that the coordinate system origin $(0, 0)$ coincides with the principal point, the $\mathbf{F}_{33}$ element of the fundamental matrix $\mathbf{F}$ is zero.

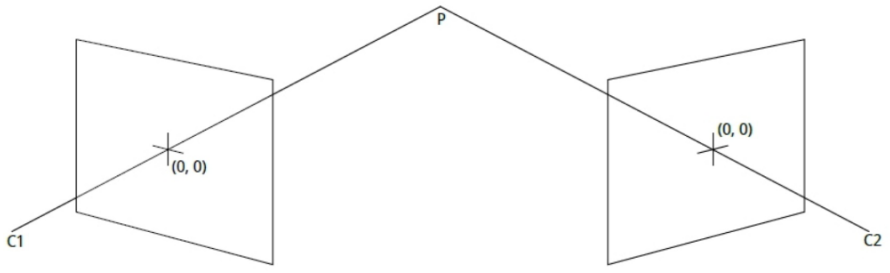


Figure 1: $C1$ and $C2$ are the optical centers. The principal axes intersect at point P.

Let coresponding image coordinates p and p', and $F = \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix}$

The epipolar constraint sets that $p'^T F p = 0$.

As the image coordinates are normalized, correspondences $p = p' = [0\ 0\ 1]^T$.

$$p'^T F p = [0\ 0\ 1]\, F \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = [F_{31}\ F_{32}\ F_{33}] \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = F_{33} = 0$$

$\therefore F_{33} = 0$

2 Optical Flow

Question 4

[5 points] As we discussed in class, the "aperture problem" shows our inability to perceive the motion correctly. Discuss what does the aperture problem imply in terms of spatial information. (1) Why do we perceive the motion ambiguously? (2) Why is the *aperture* necessary in the Aperture problem?

Hint: Consider why the aperture is needed in the first place.

(2)

The aperture problem stems from our need to utilize small-sized apertures to capture motion, the size of which is meant to satisfy three optical flow assumptions: brightness consistency, spatial coherence, and temporal persistence. In other words, optical flow holds for a sufficiently small space-time step. As a result, we tend to use a small aperture, which is expected to capture small areas of an image that has similar intensity, surface, and small changes.

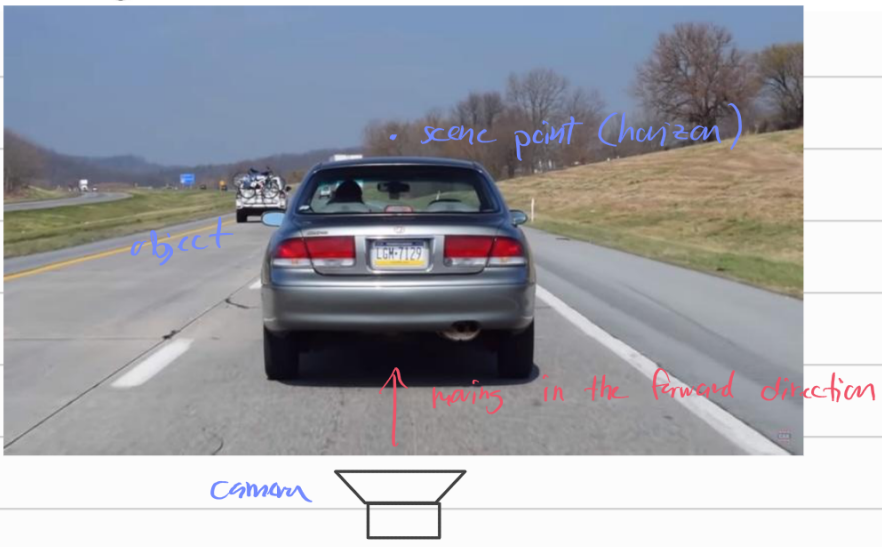
(1) However, such characteristic of the aperture creates the aperture problem by making ends of one-dimensional spatial structures (e.g., line, bar, edge) not visible. The motion direction in a small aperture becomes ambiguous as the motion component parallel to the one-dimensional spatial structure cannot be inferred based on the visual input. This means that a variety of contours of different orientations moving at different speeds can be perceived identically to our eyes. Mathematically, using the constraint equation $I_x u + I_y v + I_t = 0$, there can be several (u, v) components of optical flow.

Ironically, for the input cases being one-dimensional spatial structure, the use of small apertures can violate the three aforementioned assumptions for optical flow and produce ambiguities in motion direction.

Question 5

[5 points] Consider a situation where the motion of the camera is in the forward direction. Does the optical flow equation still hold? Find a simple way to figure out which scene point the object is moving towards.

picturing the situation...



The optical flow equation holds when the following two assumptions hold:

(A) Brightness consistency: brightness of patch remains same in both images

$\Leftrightarrow I(x+u\delta t,\ y+v\delta t,\ t+\delta t) = I(x,y,t)$

(B) Small motion: displacement $(\delta x, \delta y) = (u\delta t,\ v\delta t)$ is small (← spatial coherence)

time step δt is small

$\Leftrightarrow I(x,y,t) + \delta x \frac{\partial I}{\partial x} + \delta y \frac{\partial I}{\partial y} + \delta t \frac{\partial I}{\partial t} = I(x,y,t)$

As seen from the expressions of the assumptions, these assumptions suggest that pixels of images captured by the camera be inspected, and don't impose any constraint on the stillness of the camera. Therefore, whether the motion of the camera is in forward direction does not affect the validity of optical flow equation; the equation holds as long as assumptions (A) & (B) hold.

To figure out which scene point the object is moving towards, we must acknowledge the fact that both the camera and the object are moving, and thus optical flow is determined upon the relative motion between the camera and the scene.

1. Find the velocity components of the camera (u_1, v_1)
2. Find the components of the image velocity in the direction of the image intensity gradient at the image of a scene point. (u_2, v_2)
3. Subtract (1) from (2), i.e., $(u_2-u_1,\ v_2-v_1)$ to determine the real velocity components of the optical flow.

Question 6

[10 points] As we discussed in class, it is hard to estimate the optical flow with Lucas-kanade flow method in some situations. Suppose you are trying to estimate the flow inside the aperture in Figure 2, discuss whether this case will be difficult or not.

Figure 2: Q

Justify your answer in terms of the following matrix's characteristic:

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_y I_x & \sum I_y^2 \end{bmatrix}$$

Optical flow estimation is difficult for flat regions where gradients have small magnitude and no observable gradient change in any direction, and for edge regions where no gradient change is observed along the edge direction (giving rise to the aperture problem). In contrast, corner regions in an image allows us to observe a significant gradient change in all direction, making optical flow computation reliable. Figure 2 corresponds to the corner situation. Hence, it would not be difficult to estimate the flow inside the aperture.

Let the optical flow (u, v). Using the Lucas-Kanade method,

$$\underbrace{\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_y I_x & \sum I_y^2 \end{bmatrix}}_{A^T A} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}}_{A^T B}$$

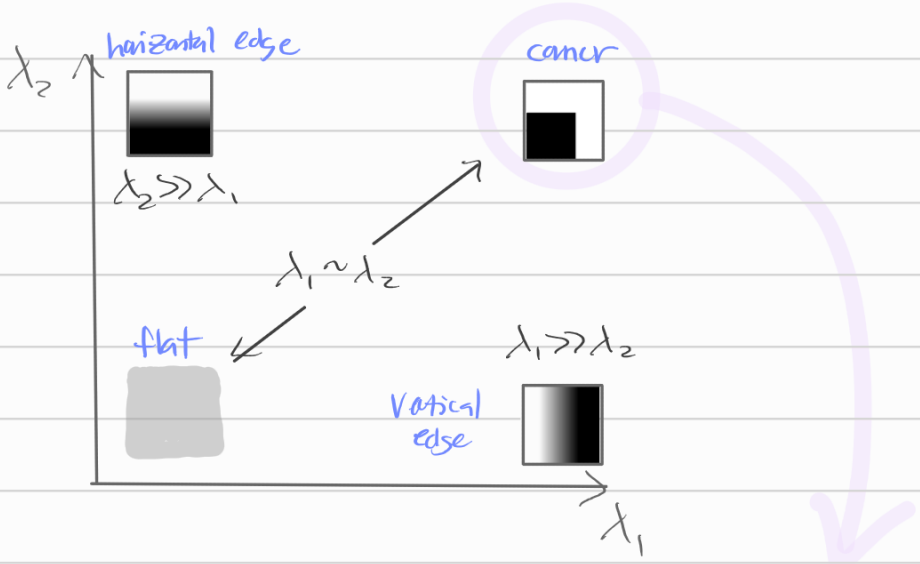
, which is of the form $A^T A x = A^T B$ and $x = (A^T A)^{-1} A^T B$

In order for the equation to be solvable, $A^T A$ must be invertible, i.e., $\det(A^T A) = \lambda_1 \lambda_2 \neq 0$ (λ_1 and λ_2 are eigenvalues of $A^T A$).

Further, $A^T A$ must be well-conditioned, i.e., there must be a significant amount of change in the output.

$\Rightarrow$ The condition are $\lambda_1 > \varepsilon$ and $\lambda_2 > \varepsilon$, and $\lambda_1 \geq \lambda_2$ but not $\lambda_1 \gg \lambda_2$.

The order relation of eigenvalues differ by the kind of image patch (flat region, edge region, corner region), as illustrated below.



The condition applies when the aperture captures the corner region. Therefore, since Figure 2 indicates the corner region, it satisfies the conditions where optical flow estimation works reliably.

Reference
https://www.youtube.com/watch?v=6wMoHgpVUn8
https://medium.com/data-breach/introduction-to-harris-corner-detector-32a88850b3f6
https://www.cs.ubc.ca/~lsigal/425_2018W1/Lecture12.pdf

2018-17119 박수현

Explanation of HW4 code

Part 1 Dominant Motion Estimation: Lucas-Kanade Method

`dp = lucas_kanade_affine(img1, img2, p, Gx, Gy)`

Assumptions

- Assumptions for the optical flow constraint equation to hold:
  - Color constancy: an image point in $I(t)$ looks the same in $I(t+1)$.
  - Small motion: image points do not move very far
- Assumption to make affine transformation in optical flow reasonable:
  - A majority of the pixels correspond to stationary objects in the scene whose depth variation is small relative to their distance from the camera.

Algorithm

1. For all image points x in template $T(x) = I(t)$, warp the template via affine transformation: $W(x;p)$. Using the affine transformation matrix $M = \begin{pmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{pmatrix}$, each point within the template is mapped to its corresponding location in $I(t+1)$. This additional image alignment step helps minimize SSD(Sum of Squared Distances) of brightness of pixels.
   - Here, I intend to compute the mesh grid of warped coordinates to ease the mapping process. To do so, I generate homogeneous coordinates of the beginning pixel and ending pixel in $T(x)$ and apply affine transformation via matrix multiplication with W only on those beginning and ending coordinates. Then, using `np.linspace`, I generate 1-dimensional vectors of warped x- coordinates and y- coordinates by generating evenly spaced coordinate values of the specified intervals. Using `np.meshgrid`, the meshgrid of warped coordinates is computed.
2. Locate the pixels in $I(t+1)$ that correspond to the warped coordinates, hence computing $I(W(x;p))$.
   - `RectBivariateSpline` returns a new class, representing an input image as a spline. Then we can call the method `RectBivariateSpline.ev(xi, yi, dx=0, dy=0)` to evaluate input coordinates and return corresponding interpolated values in the image. So, I create a spline from $I(t+1)$, and pass the meshgrid of warped coordinates to the spline to retrieve matching interpolated values in $I(t+1)$.
3. Compute the error image $T(x) - I(W(x;p))$.
   - The error image is reshaped so that it has only one column.
4. Warp image gradients I_x, I_y to compute ∇I.
   - Just as done to $T(x) = I(t)$, I derive image gradients of $I(t+1)$ that correspond to the aforementioned warped coordinates. Similar to step 2, splines of I_x, I_y are computed, and interpolated values corresponding to the warp coordinates are evaluated.
   - ∇I consists of all the warped image gradients in the dimension of `(img2_width * img2_height, 2).` That is, I registered all gradients in x- direction to the first column, and gradients in y- direction to the second column. In effect, each row of ∇I contains gradients in x- and y- direction for each pixel location.
   - Since gradient values are in big numbers, it is crucial to scale the gradients to lie in between -1 and 1. Hence, I normalize ∇I by dividing the elements by the max value.
5. Evaluate Jacobian $\frac{\partial W}{\partial p} = \begin{pmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{pmatrix}$ and compute $\nabla I \frac{\partial W}{\partial p}$.

- The form of Jacobian of affine transformation is determined as stated above.
- For each coordinate [x, y], I insert x and y into the Jacobian matrix, and compute $\nabla I \frac{\partial W}{\partial p}$ via matrix multiplication of the corresponding row in ∇I with the Jacobian.
- The entire array of $\nabla I \frac{\partial W}{\partial p}$ is multiplied by the max value of warped gradients, i.e., to undo normalization.

6. Compute Hessian $H = \sum_x [\nabla I \frac{\partial W}{\partial p}]^T [\nabla I \frac{\partial W}{\partial p}]$
7. Compute $\Delta p = H^{-1} \sum_x [\nabla I \frac{\partial W}{\partial p}]^T [T(x) - I(W(x;p))]$
   - Δp should have the dimension of (6, 1) through computation. The dimensions of the three factors are as follows: `(6, 6) @ (6, img1_width * img1_height) @ (img1_width * img1_height, 1).`
   - To keep it consistent with the parent function, I flatten Δp.

Part 2 Moving Object Detection: Affine Motion Subtraction

`moving_image = subtract_dominant_motion(img1, img2)`

Assumptions

- Same as presented in Part 1

Algorithm

1. Using zeros as an initial estimate of p, update p through additive iterative update ($p \leftarrow p + \nabla p$) to derive the optimal parameters that derive minimum SSD.
   - The point at which it can be assumed that the image windows are very close in similarity, i.e., the condition for terminating the iteration, is defined by the parameter `epsilon`. When the search window moves by less than `epsilon`, I stop updating p.[1] The norm of ∇p denotes the size of difference. By doing this to each successive tracking window, the point can be tracked throughout several images in a sequence, until it is either obscured or goes out of frame.[2]
2. Warp $I(t)$ using M so that is registered to $I(t+1)$.
3. Compute $T(x) - I(W(x;p))$, which indicates the movement between images.
4. Using hysteresis thresholding, identify locations where the absolute difference of each exceeds the lower threshold and is below the upper threshold.

Parameters

- Minimum window size `epsilon` = 0.018
  - `epsilon` is critical in determining the execution time. When `epsilon` is equal to 0.01, the program took about 30 minutes to render the images. Experimenting values between 0.01 and 0.02, I concluded that

---

1 https://docs.opencv.org/2.4/modules/video/doc/motion_analysis_and_object_tracking.html#:~:text=more%20than%20maxLevel.-,criteria,-%E2%80%93%20parameter%2C%20specifying%20the

2 https://en.wikipedia.org/wiki/Lucas%E2%80%93Kanade_method

0.018 was the ideal value that doesn’t downgrade the performance much but provides a tight bound for checking small differences.

- Upper threshold value `th_hi` = 0.2 * 256
- Lower threshold value `th_lo` = 0.15 * 256
  - Since moving objects are cleanly distinguishable from other objects in the resulting image frames, I do not adjust threshold values.