Kanade-Lucas-Tomasi (KLT) Feature Tracker

Computer Vision Lab.

Jae Kyu Suhr

Introduction

Motivation

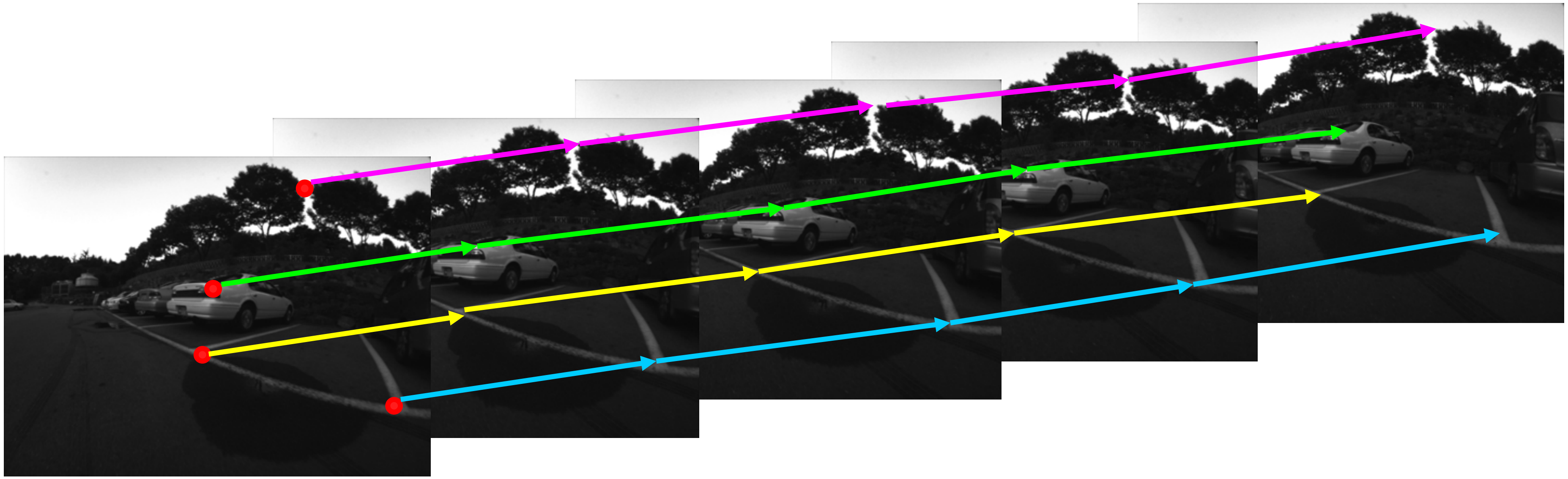
Image sequence

Motivation

Feature point detection

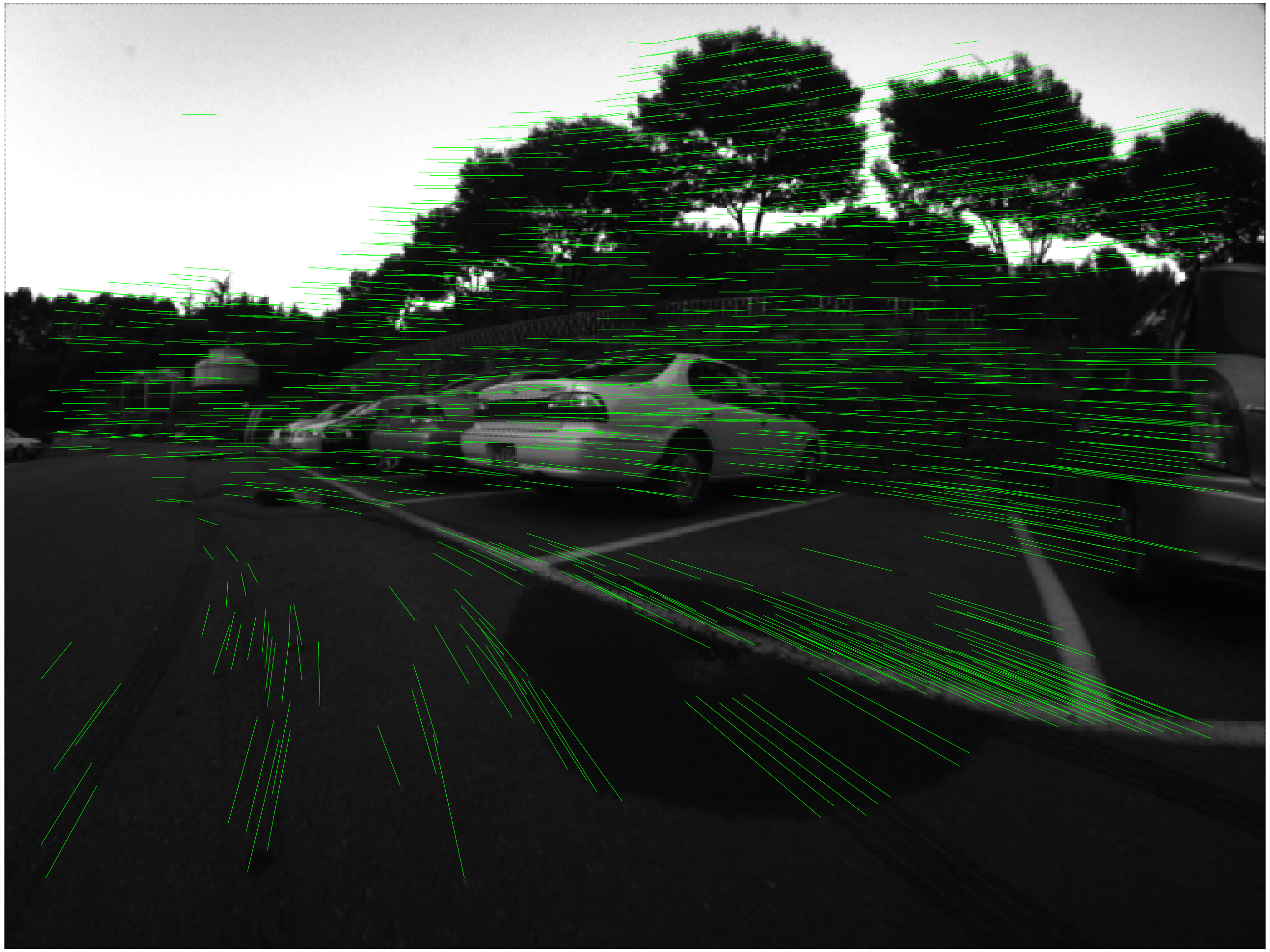
Motivation

Feature point tracking

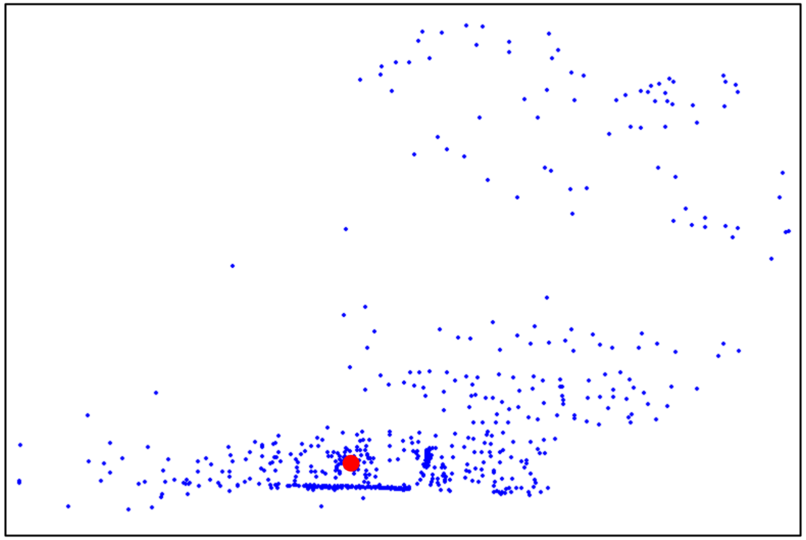


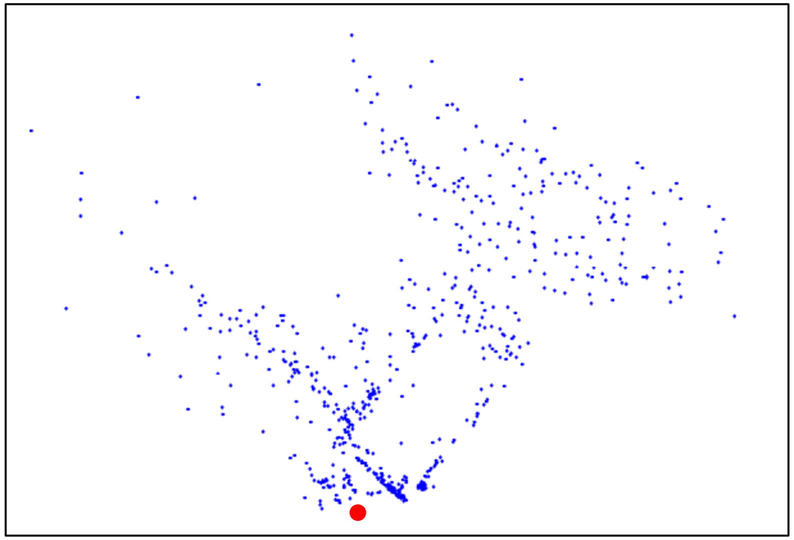
Motivation

Tracking result



3D Reconstruction results





Problem Statement

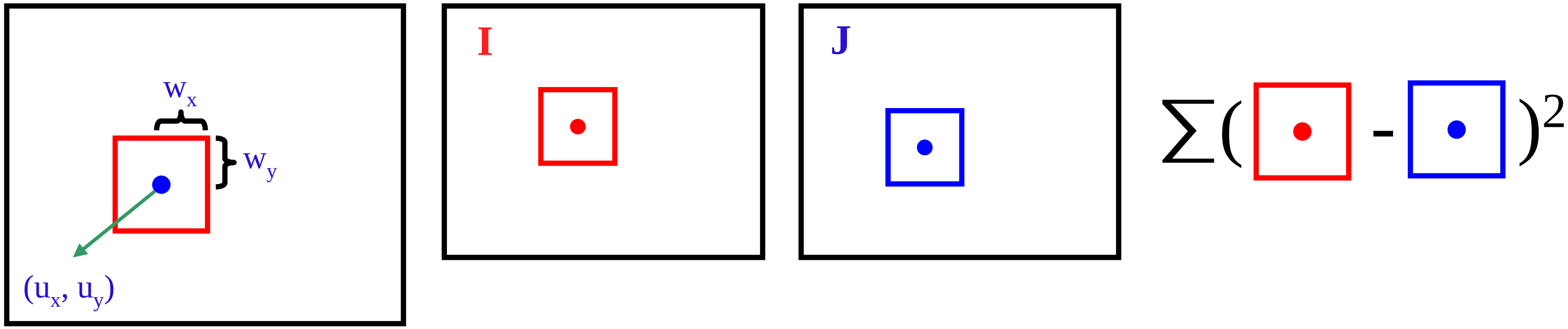
Problem Statement

- I, J : 1^{st} and 2^{nd} grayscaled images
- $I(x,y)$: gray value of I at $[x\ y]^T$
- Let $\mathbf{u}=[u_x\ u_y]^T$ be a point on the 1^{st} image I
- The goal is to find $\mathbf{v}$ on J, where $I(\mathbf{u})$ and $J(\mathbf{v})$ are similar. ($\mathbf{v}= \mathbf{u}+\mathbf{d} = [u_x+d_x\ u_y+d_y]^T$)
- $\mathbf{d}=[d_x\ d_y]^T$ is the image velocity at $\mathbf{u}$ or the optical flow at $\mathbf{u}$.

Problem Statement

- Definition of the residual function ε(**d**).

$$\epsilon(\mathbf{d}) = \epsilon(d_x, d_y) = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} \left(I(x,y) - J(x+d_x, y+d_y)\right)^2 .$$



- ω_x and ω_y integration window size parameter
  integration window size $(2\omega_x+1)\times(2\omega_y+1)$.

Pyramid Implementation

Why pyramid representation?

- Standard KLT algorithm can deal with small pixel displacement.
- Solution for this is a pyramidal implementation.
- Let $I^0 = I$ be the 0^{th} level image
- The pyramid representation is built recursively.

$$I^0 \rightarrow I^1 \rightarrow I^2 \rightarrow I^3 \rightarrow I^{Lm} \ldots \ (L_m\text{: } 2\sim4)$$

- I^{L-1} : the image at level L-1

  I^L : the image at level L

Pyramidal feature tracking

- A given point **u** in I, find its corresponding location **v**=**u**+**d**.

- Corresponding point of **u**(**u**^{0}) on the pyramidal image I^L is $\mathbf{u}^L$

$$\mathbf{u}^L = \frac{\mathbf{u}}{2^L}.$$

- Simple overall pyramid tracking algorithm

$\mathbf{d}^{Lm}$ is computed at the pyramid level L_m

↓

$\mathbf{d}^{Lm-1}$ is computed
with an initial guess of $\mathbf{d}^{Lm}$ at L_{m-1}

↓

This continues up to the level 0

Pyramidal feature tracking

- Let $\mathbf{g}^L=[g^L_x \; g^L_y]^T$ be an initial guess at level L
  ($\mathbf{g}^L$ is available from level L_m to level L+1)

- Residual pixel displacement vector $\mathbf{d}^L=[d^L_x \; d^L_y]^T$
  that minimizes the new image matching error function ε^L

$$\epsilon^L(\mathbf{d}^L) = \epsilon^L(d^L_x, d^L_y) = \sum_{x=u^L_x-\omega_x}^{u^L_x+\omega_x} \sum_{y=u^L_y-\omega_y}^{u^L_y+\omega_y} \left(I^L(x,y) - J^L(x+g^L_x+d^L_x, y+g^L_y+d^L_y)\right)^2 .$$

- Window size $(2\omega_x+1)\times(2\omega_y+1)$ is constant for all pyramid

- $\mathbf{g}^L$ is used to pre-translate the image patch in 2nd image J
  $\rightarrow$ $\mathbf{d}^L$ is small and therefore easy to compute using KLT algorithm

Pyramidal feature tracking

- Assume that $\mathbf{d}^L$ is computed, new initial guess $\mathbf{g}^{L-1}$ at level L-1

$$\mathbf{g}^{L-1} = 2\left(\mathbf{g}^{\mathbf{L}} + \mathbf{d}^{L}\right).$$

- Initial guess for the deepest level L_m

$$\mathbf{g}^{L_m} = [0 \quad 0]^T.$$

- The final optical flow solution **d**

$$\mathbf{d} = \sum_{L=0}^{L_m} 2^L\, \mathbf{d}^L.$$

Standard KLT algorithm

Standard KLT optical flow computation

- Goal is to find $\mathbf{d}^L$ that minimizes the matching function ε^L.

  Same operation is performed for all levels L, drop the superscript L and define the new images A and B

$$\forall (x, y) \in [p_x - \omega_x, p_x + \omega_x] \times [p_y - \omega_y, p_y + \omega_y],$$

$$A(x, y) \doteq I^L(x, y),$$

$$B(x, y) \doteq J^L(x + g_x^L, y + g_y^L).$$

- Let us change the displacement vector and the point vector

$$\overline{\nu} = [\nu_x \quad \nu_y]^T = \mathbf{d}^L$$

$$\mathbf{p} = [p_x \quad p_y]^T = \mathbf{u}^L$$

Standard KLT optical flow computation

- Goal is to find $\overline{\nu}$ that minimizes the matching function

$$\varepsilon(\overline{\nu}) = \varepsilon(\nu_x, \nu_y) = \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \left(A(x,y) - B(x+\nu_x, y+\nu_y)\right)^2 .$$

- To find the optimum $\overline{\nu}$

$$\left.\frac{\partial \varepsilon(\overline{\nu})}{\partial \overline{\nu}}\right|_{\overline{\nu}=\overline{\nu}_{\mathrm{opt}}} = [0 \;\; 0].$$

$$\frac{\partial \varepsilon(\overline{\nu})}{\partial \overline{\nu}} = -2 \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \left(A(x,y) - B(x+\nu_x, y+\nu_y)\right) \cdot \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix} .$$

- Let us substitute $B(x+\nu_x, y+\nu_y)$ by its 1st order Taylor expansion about the point $\overline{\nu} = [0 \;\; 0]^T$

$$\frac{\partial \varepsilon(\overline{\nu})}{\partial \overline{\nu}} \approx -2 \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \left(\underbrace{A(x,y) - B(x,y)}_{\text{frame difference}} - \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix} \overline{\nu} \right) \cdot \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix} .$$

Image gradient

Standard KLT optical flow computation

$$\frac{\partial \varepsilon(\overline{\nu})}{\partial \overline{\nu}} \approx -2 \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \Big(\underbrace{A(x,y) - B(x,y)}_{\delta I(x,y)} - \underbrace{\begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix}}_{\nabla I} \overline{\nu} \Big) \cdot \underbrace{\begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix}}_{\nabla I}.$$

$$\delta I(x,y) \doteq A(x,y) - B(x,y). \qquad \nabla I = \begin{bmatrix} I_x \\ I_y \end{bmatrix} \doteq \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix}^T.$$

$$\begin{aligned} I_x(x,y) &= \frac{\partial A(x,y)}{\partial x} = \frac{A(x+1,y) - A(x-1,y)}{2}, \\ I_y(x,y) &= \frac{\partial A(x,y)}{\partial y} = \frac{A(x,y+1) - A(x,y-1)}{2}. \end{aligned}$$

$$\Downarrow$$

$$\frac{1}{2} \frac{\partial \varepsilon(\overline{\nu})}{\partial \overline{\nu}} \approx \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \left(\nabla I^T \overline{\nu} - \delta I \right) \nabla I^T,$$

$$\Downarrow$$

$$\frac{1}{2} \left[\frac{\partial \varepsilon(\overline{\nu})}{\partial \overline{\nu}} \right]^T \approx \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \left(\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \overline{\nu} - \begin{bmatrix} \delta I\, I_x \\ \delta I\, I_y \end{bmatrix} \right).$$

Standard KLT optical flow computation

$$\frac{1}{2}\left[\frac{\partial\varepsilon(\overline{\nu})}{\partial\overline{\nu}}\right]^T \approx \sum_{x=p_x-\omega_x}^{p_x+\omega_x}\sum_{y=p_y-\omega_y}^{p_y+\omega_y}\left(\boxed{\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}}\overline{\nu} - \boxed{\begin{bmatrix} \delta I\, I_x \\ \delta I\, I_y \end{bmatrix}}\right).$$

$$G \doteq \sum_{x=p_x-\omega_x}^{p_x+\omega_x}\sum_{y=p_y-\omega_y}^{p_y+\omega_y}\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad \text{and} \quad \overline{b} \doteq \sum_{x=p_x-\omega_x}^{p_x+\omega_x}\sum_{y=p_y-\omega_y}^{p_y+\omega_y}\begin{bmatrix} \delta I\, I_x \\ \delta I\, I_y \end{bmatrix}.$$

$$\frac{1}{2}\left[\frac{\partial\varepsilon(\overline{\nu})}{\partial\overline{\nu}}\right]^T \approx G\overline{\nu} - \overline{b}.$$

$$\overline{\nu}_{\text{opt}} = G^{-1}\,\overline{b}.$$ (G must be invertible).

❖ Standard LK is valid only the pixel displacement is small.

(Because of the first order Taylor approximation)

Therefore, iterative version of LK is necessary.

Iterative KLT algorithm

Iterative KLT optical flow computation

- k : iteration index

- $\overline{\nu}^{k-1} = [\nu_x^{k-1} \quad \nu_x^{k-1}]^T$

  : initial guess from the previous iteration 1,2,…,k-1

- B_k : new translated image according to $\overline{\nu}^{k-1}$

$$\forall (x, y) \in [p_x - \omega_x, p_x + \omega_x] \times [p_y - \omega_y, p_y + \omega_y],$$
$$B_k(x, y) = B(x + \nu_x^{k-1}, y + \nu_y^{k-1}).$$

- The goal is to compute $\overline{\eta}^k = [\eta_x^k \quad \eta_y^k]$ that minimize

$$\varepsilon^k(\overline{\eta}^k) = \varepsilon(\eta_x^k, \eta_y^k) = \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \left(A(x, y) - B_k(x + \eta_x^k, y + \eta_y^k)\right)^2 .$$

Iterative KLT optical flow computation

- One step LK optical flow computation is

$$\overline{\eta}^k = G^{-1}\,\overline{b}_k,$$

$$\overline{b}_k = \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} \delta I_k(x,y)\, I_x(x,y) \\ \delta I_k(x,y)\, I_y(x,y) \end{bmatrix}, \qquad \delta I_k(x,y) = A(x,y) - B_k(x,y).$$

- I_x, I_y are computed only once at the beginning of the iteration.
- G (2×2matrix) remains constant throughout the iteration loop.
- Only δI_k needs to be recomputed at each iteration
- Once $\overline{\eta}^k$ is computed, a new pixel displacement guess $\overline{\nu}^{k-1}$

$$\overline{\nu}^k = \overline{\nu}^{k-1} + \overline{\eta}^k.$$

- The iteration goes on until $\overline{\eta}^k$ is smaller than a threshold
  or reached at the maximum number of iteration.
  (5 are enough to reach convergence)

Iterative KLT optical flow computation

- At 1st iteration, the initial guess $\overline{\nu}^0 = [0 \;\; 0]^T$.

- Assuming that K iterations are necessary to reach convergence, the final solution for the optical flow vector $\overline{\nu} = \mathbf{d}^L$

$$\overline{\nu} = \mathbf{d}^L = \overline{\nu}^K = \sum_{k=1}^{K} \overline{\eta}^k.$$

- This overall procedure is repeated at all levels L-1, L-2, … , 0

Goal: Let $\mathbf{u}$ be a point on image I. Find its corresponding location $\mathbf{v}$ on image J

Build pyramid representations of I and J: $\{I^L\}_{L=0,\ldots,L_m}$ and $\{J^L\}_{L=0,\ldots,L_m}$

Initialization of pyramidal guess: $\mathbf{g}^{L_m} = [g_x^{L_m} \;\; g_x^{L_m}]^T = [0 \;\; 0]^T$

for $L = L_m$ **down to** 0 **with step of** -1

Location of point $\mathbf{u}$ *on image* I^L: $\mathbf{u}^L = [p_x \;\; p_y]^T = \mathbf{u}/2^L$

Derivative of I^L with respect to x: $I_x(x,y) = \dfrac{I^L(x+1,y) - I^L(x-1,y)}{2}$

Derivative of I^L with respect to y: $I_y(x,y) = \dfrac{I^L(x,y+1) - I^L(x,y-1)}{2}$

Spatial gradient matrix:

$$G = \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} I_x^2(x,y) & I_x(x,y)\, I_y(x,y) \\ I_x(x,y)\, I_y(x,y) & I_y^2(x,y) \end{bmatrix}$$

Initialization of iterative L-K: $\overline{\nu}^0 = [0 \;\; 0]^T$

for $k = 1$ **to** K **with step of** 1 (or until $\|\overline{\eta}^k\| <$ accuracy threshold)

Image difference: $\delta I_k(x,y) = I^L(x,y) - J^L(x + g_x^L + \nu_x^{k-1}, y + g_y^L + \nu_y^{k-1})$

Image mismatch vector:

$$\overline{b}_k = \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} \delta I_k(x,y)\, I_x(x,y) \\ \delta I_k(x,y)\, I_y(x,y) \end{bmatrix}$$

Optical flow (Lucas-Kanade): $\overline{\eta}^k = G^{-1}\, \overline{b}_k$

Guess for next iteration: $\overline{\nu}^k = \overline{\nu}^{k-1} + \overline{\eta}^k$

end of for-loop on k

Final optical flow at level L: $\mathbf{d}^L = \overline{\nu}^K$

Guess for next level $L-1$: $\mathbf{g}^{L-1} = [g_x^{L-1} \;\; g_y^{L-1}]^T = 2\,(\mathbf{g}^{\mathrm{L}} + \mathbf{d}^L)$

end of for-loop on L

Final optical flow vector: $\mathbf{d} = \mathbf{g}^0 + \mathbf{d}^0$

Location of point on J: $\mathbf{v} = \mathbf{u} + \mathbf{d}$

Solution: The corresponding point is at location $\mathbf{v}$ on image J

Feature selection

- Goal is to find the location of **v** on J corresponding to **u** on I.

  How to detect **u** on I

$$\overline{\nu}_{\text{opt}} = G^{-1}\,\overline{b}.$$

$$G \doteq \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} I_x^2 & I_x\,I_y \\ I_x\,I_y & I_y^2 \end{bmatrix} \quad \text{and} \quad \overline{b} \doteq \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} \delta I\, I_x \\ \delta I\, I_y \end{bmatrix}.$$

- Find the point **u** whose G matrix is non-singular

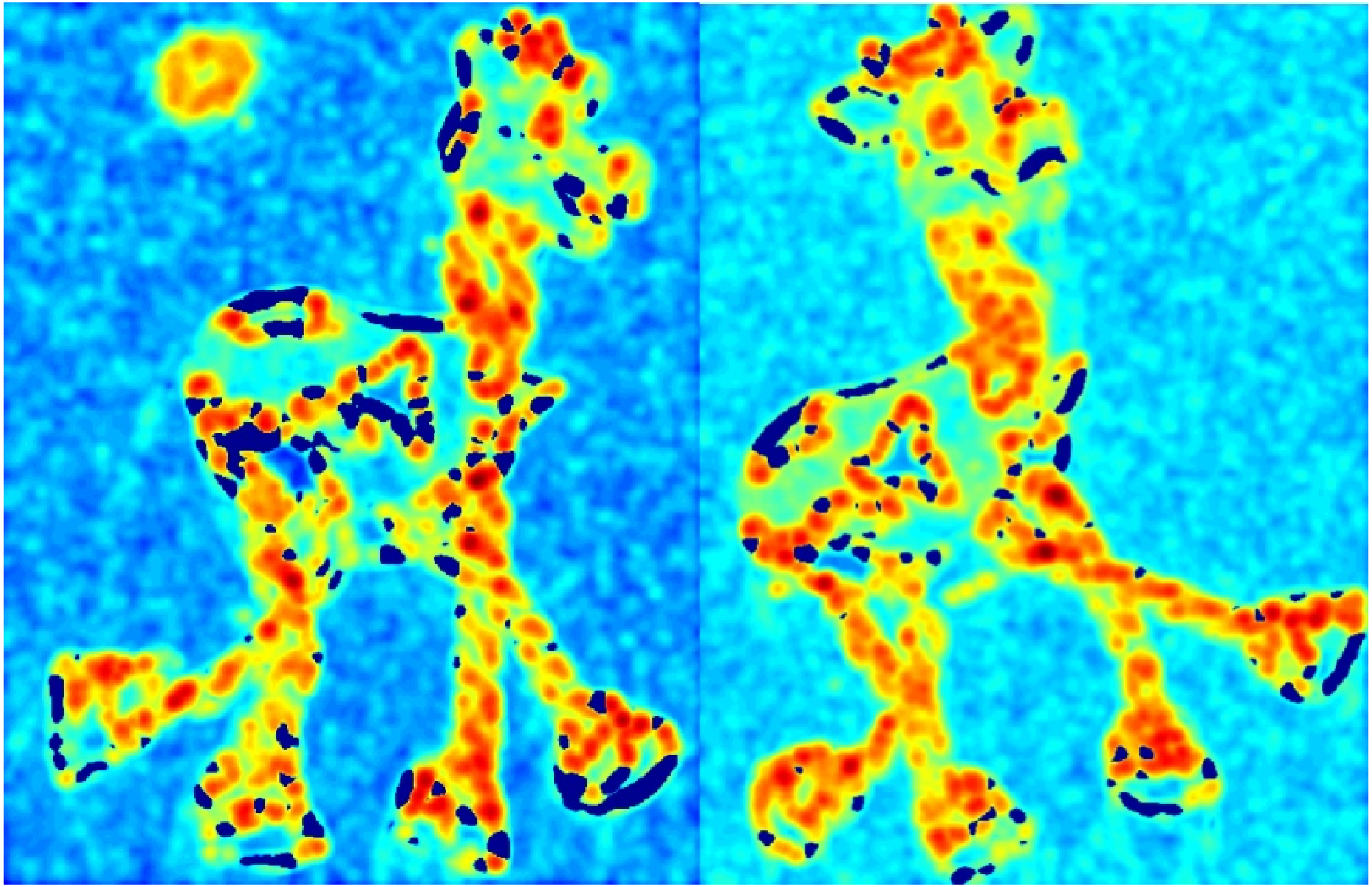
  The minimum eigenvalue of G must larger than a threshold.

Feature selection

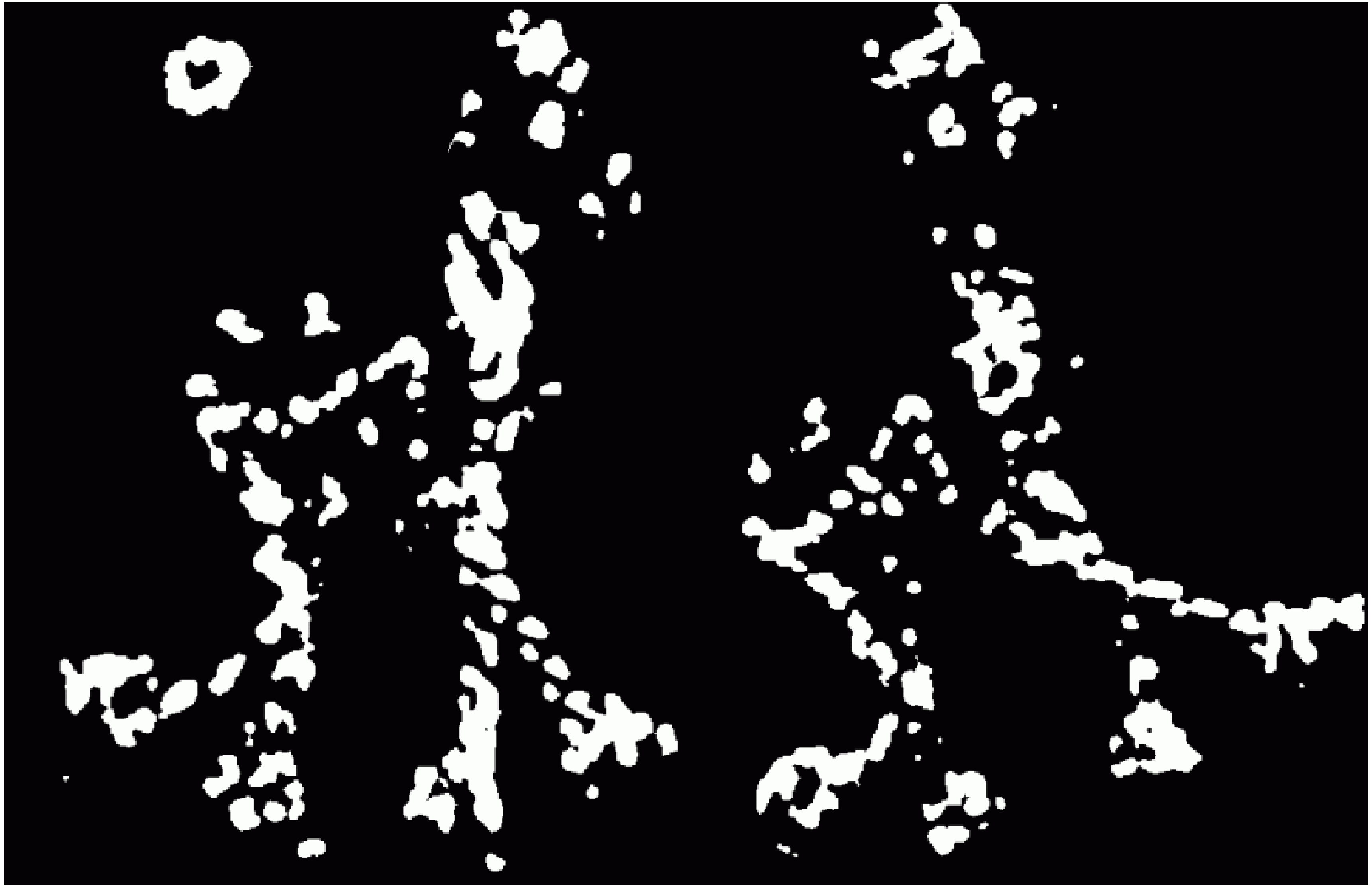
1. Compute the G matrix and its minimum eigenvalue λ_m at every pixel in the image I.
2. Call λ_{max} the maximum value of λ_m over the whole image.
3. Retain the image pixels that have a λ_m value larger than a threshold.
4. Retain the local maximum pixels (a pixel is kept if its λ_m value is larger than that of any other pixel in its 3×3 neighborhood).
5. Keep the subset of those pixels so that the minimum distance between any pair of pixels is larger than a given threshold distance.

Feature selection

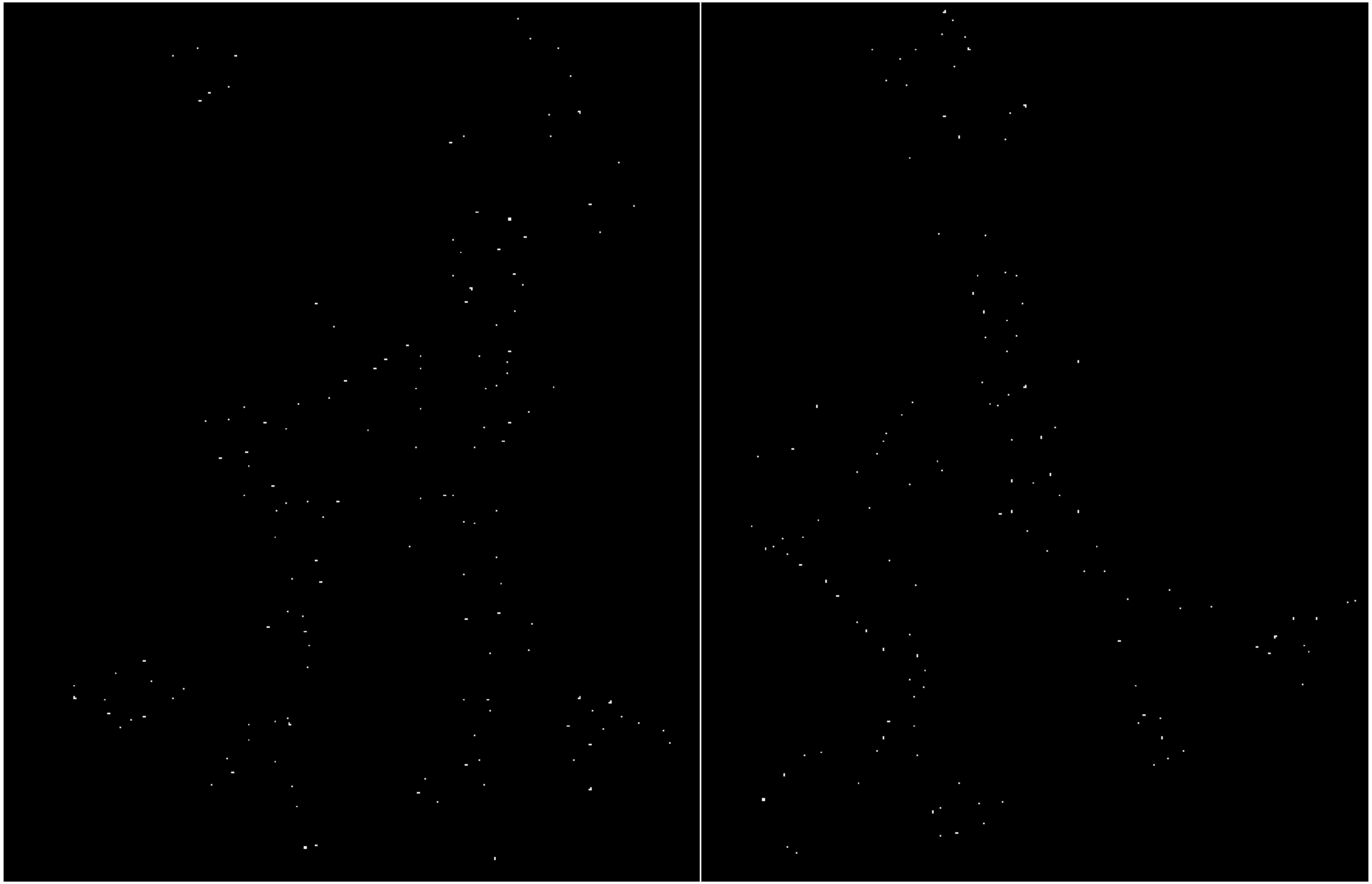
Feature selection



Feature selection



Feature selection



Feature selection

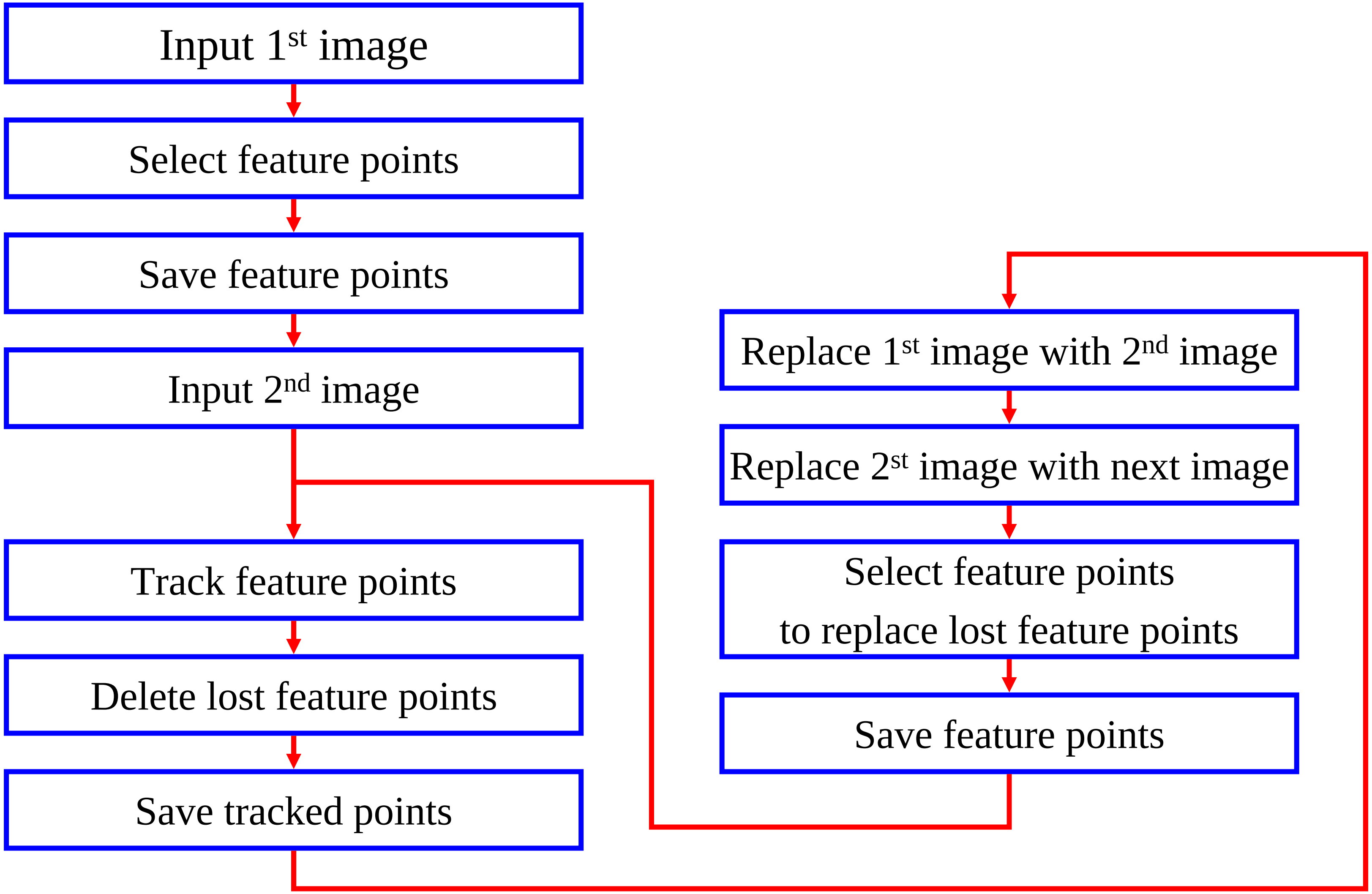
Declaring a feature "Lost"

- A feature point falls outside of the image.

- An image patch around the tracked point varies too much between image I and image J.

  (A cost function (SSD) is larger than a threshold).

Flow chart



Input 1st image
Select feature points
Save feature points
Input 2nd image
Track feature points
Delete lost feature points
Save tracked points
Replace 1st image with 2nd image
Replace 2st image with next image
Select feature points
to replace lost feature points
Save feature points

References

- Bruce D. Lucas and Takeo Kanade. “An Iterative Image Registration Technique with an Application to Stereo Vision”. IJCAI, pages 674-679, 1981.

- J. Shi and C. Tomasi, "Good Features to Track," CVPR'94

- Jean-Yves Bouguet, “Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm”, Intel Corporation Microprocessor Research Labs.

- http://www.ces.clemson.edu/~stb/klt/ C++ code
  KLT: An Implementation of the Kanade-Lucas-Tomasi Feature Tracker

- OpenCV: CalcOpticalFlowLK, CalcOpticalFlowPyrLK

- http://vision.ucla.edu//MASKS/labs.html Matlab code
  An Invitation to 3D Vision

Thank you

Q & A