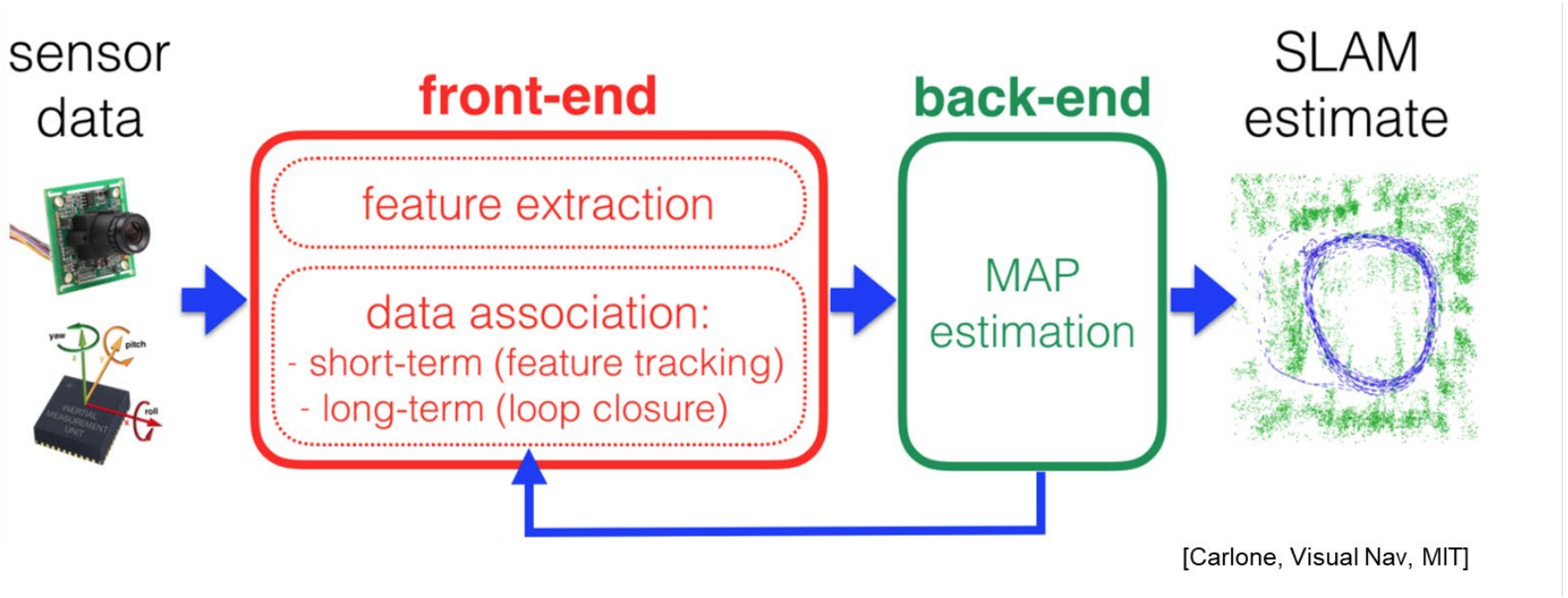
Robotic Mapping & Localization

Kaveh Fathian
Assistant Professor
Computer Science Department
Colorado School of Mines

Lec05: Image Features

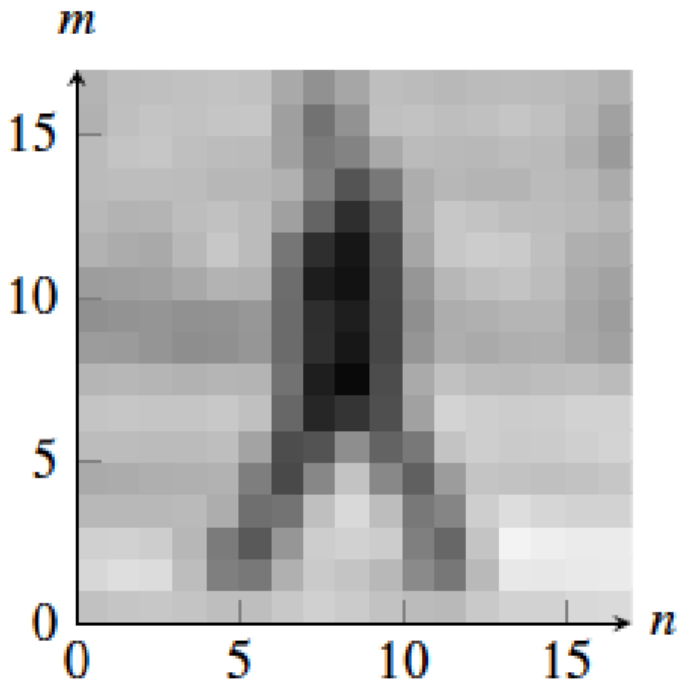
*Courtesy of Thomas Williams, Bill Freeman, Antonio Torralba, Cyrill Stachniss, Sunglok Choi, JunHyeok Choi



[Carlone, Visual Nav, MIT]

- A SLAM pipeline consists of
  - Sensors
  - Frontend algorithms
  - Backend algorithms

Image as a 2D discrete signal

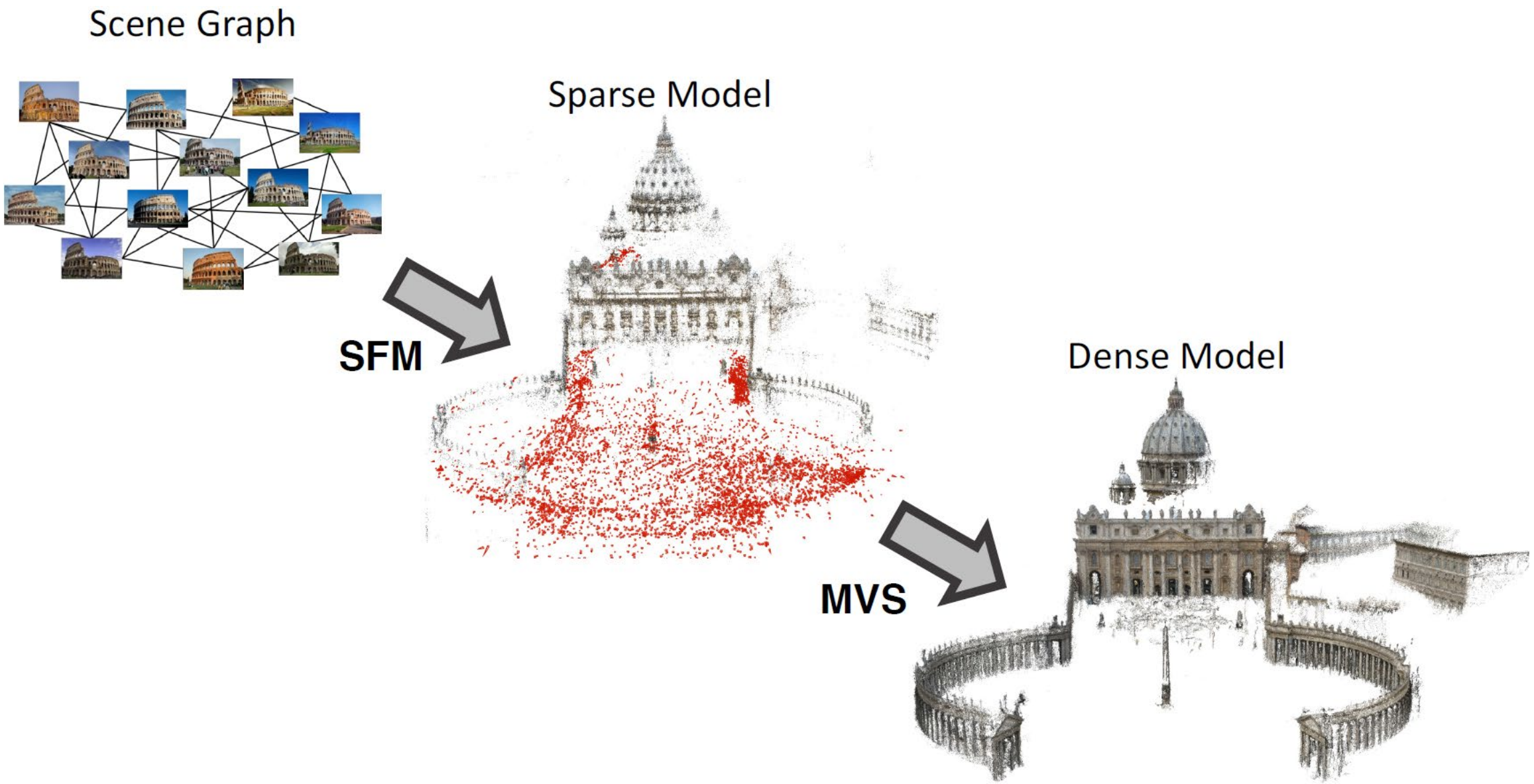


A tiny person of 18 x 18 pixels

Image as a 2D discrete signal

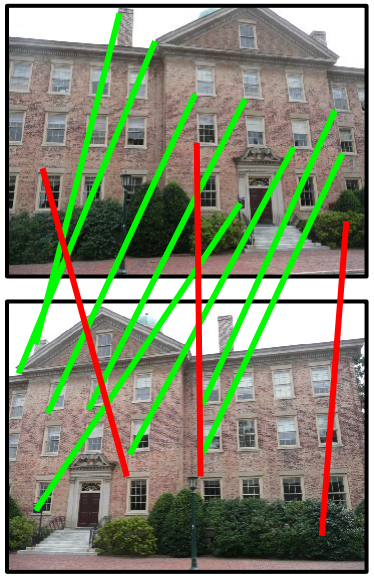
$$\mathbf{I} = \begin{bmatrix}
160 & 175 & 171 & 168 & 168 & 172 & 164 & 158 & 167 & 173 & 167 & 163 & 162 & 164 & 160 & 159 & 163 & 162 \\
149 & 164 & 172 & 175 & 178 & 179 & 176 & 118 & 97 & 168 & 175 & 171 & 169 & 175 & 176 & 177 & 165 & 152 \\
161 & 166 & 182 & 171 & 170 & 177 & 175 & 116 & 109 & 169 & 177 & 173 & 168 & 175 & 175 & 159 & 153 & 123 \\
171 & 174 & 177 & 175 & 167 & 161 & 157 & 138 & 103 & 112 & 157 & 164 & 159 & 160 & 165 & 169 & 148 & 144 \\
163 & 163 & 162 & 165 & 167 & 164 & 178 & 167 & 77 & 55 & 134 & 170 & 167 & 162 & 164 & 175 & 168 & 160 \\
173 & 164 & 158 & 165 & 180 & 180 & 150 & 89 & 61 & 34 & 137 & 186 & 186 & 182 & 175 & 165 & 160 & 164 \\
152 & 155 & 146 & 147 & 169 & 180 & 163 & 51 & 24 & 32 & 119 & 163 & 175 & 182 & 181 & 162 & 148 & 153 \\
134 & 135 & 147 & 149 & 150 & 147 & 148 & 62 & 36 & 46 & 114 & 157 & 163 & 167 & 169 & 163 & 146 & 147 \\
135 & 132 & 131 & 125 & 115 & 129 & 132 & 74 & 54 & 41 & 104 & 156 & 152 & 156 & 164 & 156 & 141 & 144 \\
151 & 155 & 151 & 145 & 144 & 149 & 143 & 71 & 31 & 29 & 129 & 164 & 157 & 155 & 159 & 158 & 156 & 148 \\
172 & 174 & 178 & 177 & 177 & 181 & 174 & 54 & 21 & 29 & 136 & 190 & 180 & 179 & 176 & 184 & 187 & 182 \\
177 & 178 & 176 & 173 & 174 & 180 & 150 & 27 & 101 & 94 & 74 & 189 & 188 & 186 & 183 & 186 & 188 & 187 \\
160 & 160 & 163 & 163 & 161 & 167 & 100 & 45 & 169 & 166 & 59 & 136 & 184 & 176 & 175 & 177 & 185 & 186 \\
147 & 150 & 153 & 155 & 160 & 155 & 56 & 111 & 182 & 180 & 104 & 84 & 168 & 172 & 171 & 164 & 168 & 167 \\
184 & 182 & 178 & 175 & 179 & 133 & 86 & 191 & 201 & 204 & 191 & 79 & 172 & 220 & 217 & 205 & 209 & 200 \\
184 & 187 & 192 & 182 & 124 & 32 & 109 & 168 & 171 & 167 & 163 & 51 & 105 & 203 & 209 & 203 & 210 & 205 \\
191 & 198 & 203 & 197 & 175 & 149 & 169 & 189 & 190 & 173 & 160 & 145 & 156 & 202 & 199 & 201 & 205 & 202 \\
153 & 149 & 153 & 155 & 173 & 182 & 179 & 177 & 182 & 177 & 182 & 185 & 179 & 177 & 167 & 176 & 182 & 180
\end{bmatrix}$$

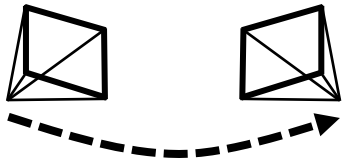
Structure from Motion (SfM)



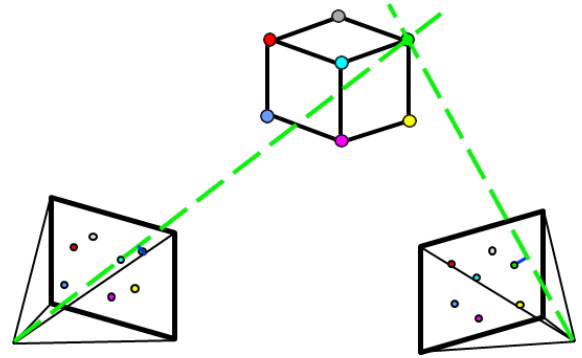
Simple SfM/SLAM Pipeline

1. Feature Extraction → 2. Feature Matching → 3. Geometric Verification → 4. Triangulation





Robust Two-View Geometry Estimation



Lecture Outline

- **Image Features**
  - Features
  - Matching
  - Optical Flow
  - Camera model
  - References

Our Simple SLAM System

robotic-mapping / code / visual-odometry / vo_epipolar.cpp

KavehFathian first commit

Code | Blame | 97 lines (88 loc) · 3.52 KB

```cpp
#include <iostream>
#include <vector>
#include <string>
#include "opencv2/opencv.hpp"

int main()
{
    const char* video_file = "../data/KITTI07/image_0/%06d.png";
    double f = 707.0912;
    cv::Point2d c(601.8873, 183.1104);
    bool use_5pt = true;
    int min_inlier_num = 100;
    double min_inlier_ratio = 0.2;
    const char* traj_file = "vo_epipolar.xyz";
```

Our Simple SLAM System

```cpp
#include <iostream>
#include <vector>
#include <string>
#include "opencv2/opencv.hpp"

int main()
{
    const char* video_file = "../data/KITTI07/image_0/%06d.png";
    double f = 707.0912;
    cv::Point2d c(601.8873, 183.1104);
    bool use_5pt = true;
    int min_inlier_num = 100;
    double min_inlier_ratio = 0.2;
    const char* traj_file = "vo_epipolar.xyz";

    // Open a video and get an initial image
    cv::VideoCapture video;
    if (!video.open(video_file)) return -1;

    cv::Mat gray_prev;
    video >> gray_prev;
    if (gray_prev.empty())
    {
        video.release();
        return -1;
    }
    if (gray_prev.channels() > 1) cv::cvtColor(gray_prev, gray_prev,
    cv::COLOR_RGB2GRAY);

    // Run the monocular visual odometry
    cv::Mat K = (cv::Mat_<double>(3, 3) << f, 0, c.x, 0, f, c.y, 0, 0, 1);
    cv::Mat camera_pose = cv::Mat::eye(4, 4, CV_64F);
    FILE* camera_traj = fopen(traj_file, "wt");
    if (camera_traj == NULL) return -1;
```

OpenCV library

Camera parameters

SLAM system parameters

Convert RGB images to gray

Camera calibration matrix

Pose (position & orientation) estimates

Our Simple SLAM System

```
34      while (true)
35      {
36          // Grab an image from the video
37          cv::Mat img, gray;
38          video >> img;
39          if (img.empty()) break;
40          if (img.channels() > 1) cv::cvtColor(img, gray, cv::COLOR_RGB2GRAY);
41          else                    gray = img.clone();
42
43          // Extract optical flow
44          std::vector<cv::Point2f> pts_prev, pts;
45          cv::goodFeaturesToTrack(gray_prev, pts_prev, 2000, 0.01, 10);
46          std::vector<uchar> status;
47          cv::Mat err;
48          cv::calcOpticalFlowPyrLK(gray_prev, gray, pts_prev, pts, status, err);
49          gray_prev = gray;
50
51          // Calculate relative pose
52          cv::Mat E, inlier_mask;
53          if (use_5pt)
54          {
55              E = cv::findEssentialMat(pts_prev, pts, f, c, cv::RANSAC, 0.999, 1,
                inlier_mask);
56          }
57          else
58          {
59              cv::Mat F = cv::findFundamentalMat(pts_prev, pts, cv::FM_RANSAC, 1, 0.
                99, inlier_mask);
60              E = K.t() * F * K;
61          }
62          cv::Mat R, t;
63          int inlier_num = cv::recoverPose(E, pts_prev, pts, R, t, f, c, inlier_mask);
64          double inlier_ratio = static_cast<double>(inlier_num) / static_cast<double>
            (pts.size());
65
66          // Accumulate relative pose if result is reliable
67          cv::Vec3b info_color(0, 255, 0);
68          if ((inlier_num > min_inlier_num) && (inlier_ratio > min_inlier_ratio))
69          {
70              cv::Mat T = cv::Mat::eye(4, 4, R.type());
71              T(cv::Rect(0, 0, 3, 3)) = R * 1.0;
72              T.col(3).rowRange(0, 3) = t * 1.0;
73              camera_pose = camera_pose * T.inv();
74              info_color = cv::Vec3b(0, 0, 255);
75          }
```

Iterate over images

Feature detection & tracking using optical flow

Estimate camera pose (in the form of essential matrix) from consecutive images

RANSAC outlier rejection

Extract camera pose from essential matrix

Camera pose (and trajectory) over time

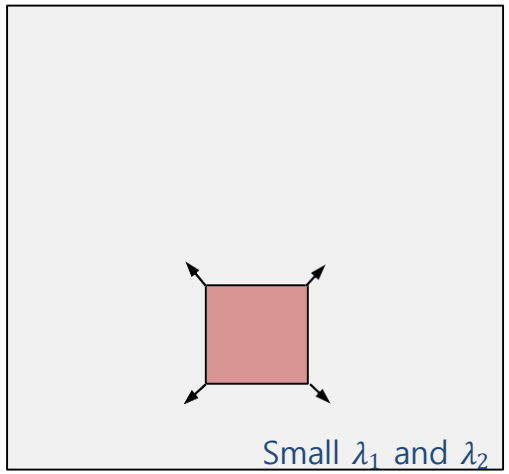
Image Features

Features as Image Corners

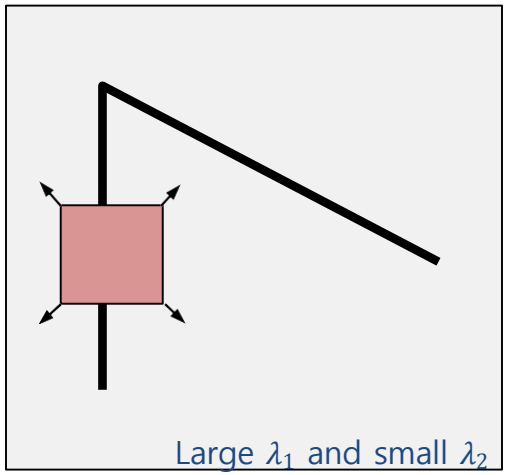
- **What is a good feature? (Requirements)**
  - **Repeatability** (to be invariance/robustness to transformation and noise)
  - **Distinctiveness** (to be easy to distinguish or match)
  - **Locality** (due to occlusion)
  - Quantity (sufficient number), accuracy (localization), efficiency (computing time), …

Harris Corner (1988)

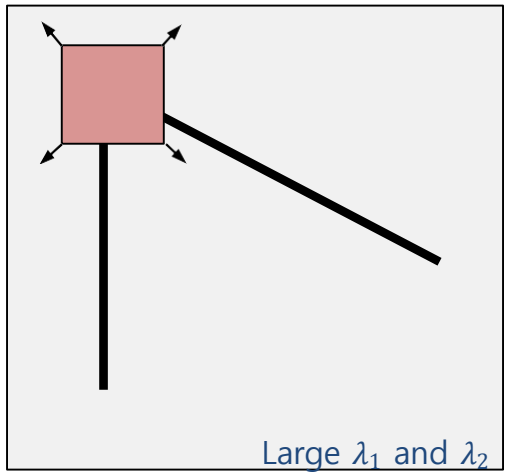
- Key idea: **Sliding window**



"flat" region:
no change
in all directions



"edge":
no change
along the edge direction



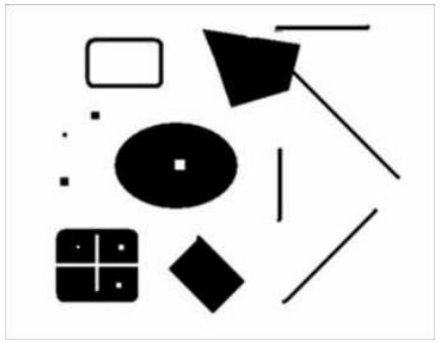
"corner":
significant change
in all directions

- Formulation
  - $I(x+\Delta_x, y+\Delta_y) \approx I(x,y) + [I_x(x,y) \quad I_y(x,y)] \begin{bmatrix}\Delta_x \\ \Delta_y\end{bmatrix}$ where $I_x = \frac{\partial I}{\partial x}$ and $I_y = \frac{\partial I}{\partial y}$
  - $D(\Delta_x, \Delta_y) = \sigma_{(x,y)\in W} \left(I(x+\Delta_x, y+\Delta_y) - I(x,y)\right)^2 \approx [\Delta_x \quad \Delta_y] \begin{bmatrix}\sigma_W I_x^2 & \sigma_W I_x I_y \\ \sigma_W I_x I_y & \sigma_W I_y^2\end{bmatrix} \begin{bmatrix}\Delta_x \\ \Delta_y\end{bmatrix}$

M

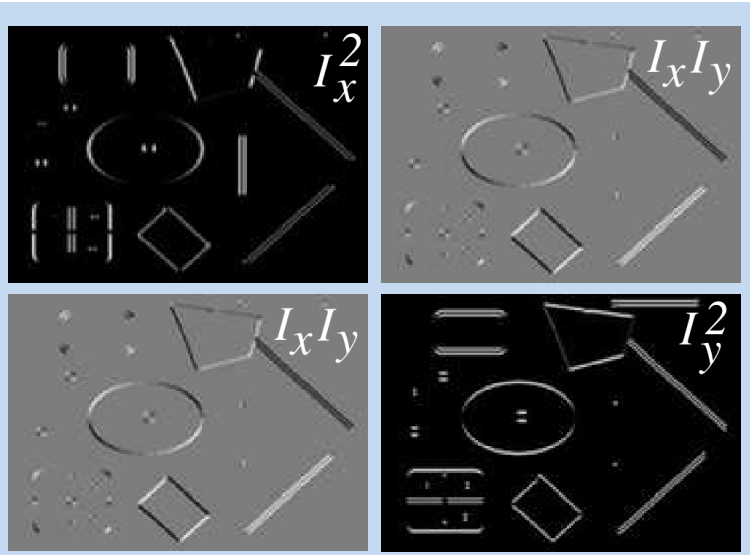
Harris Corner (1988)

$I(x, y)$



- Key idea: **Sliding window**
  - Formulation
    - $I(x + \Delta_x, y + \Delta_y) \approx I(x, y) + [I_x(x, y) \quad I_y(x, y)] \begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix}$ where $I_x = \frac{\partial I}{\partial x}$ and $I_y = \frac{\partial I}{\partial y}$
    - $D(\Delta_x, \Delta_y) = \sigma_{(x,y) \in W} \left(I(x + \Delta_x, y + \Delta_y) - I(x, y) \right)^2 \approx [\Delta_x \quad \Delta_y] \begin{bmatrix} \sigma_W I_x^2 & \sigma_W I_x I_y \\ \sigma_W I_x I_y & \sigma_W I_y^2 \end{bmatrix} \begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix}$

M



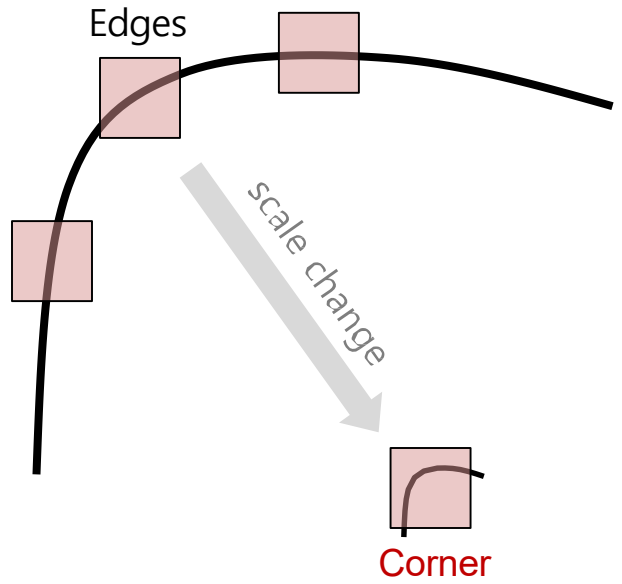
M

  - **Harris corner response**
    - $\text{cornerness} = \det(\text{M}) - k \, \text{trace}(\text{M})^2$
    - Note: $\det(\text{M}) = \lambda_1 \lambda_2$, $\text{trace}(\text{M}) = \lambda_1 + \lambda_2$, and $k \in [0.04, 0.06]$
  - Note: **Good-Feature-to-Track** (a.k.a. GFTT or Shi-Tomasi corner; 1994)
    - $\text{cornerness} = \min(\lambda_1, \lambda_2)$

$\det(\text{M}) - k \, \text{trace}(\text{M})^2$

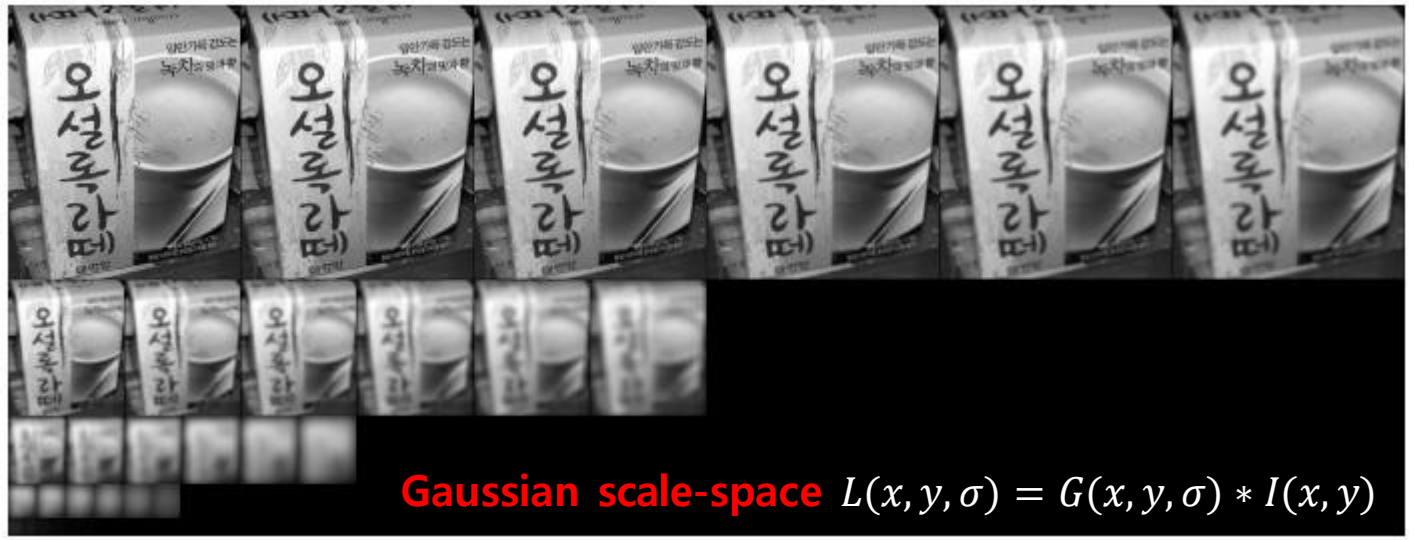
Harris Corner (1988)

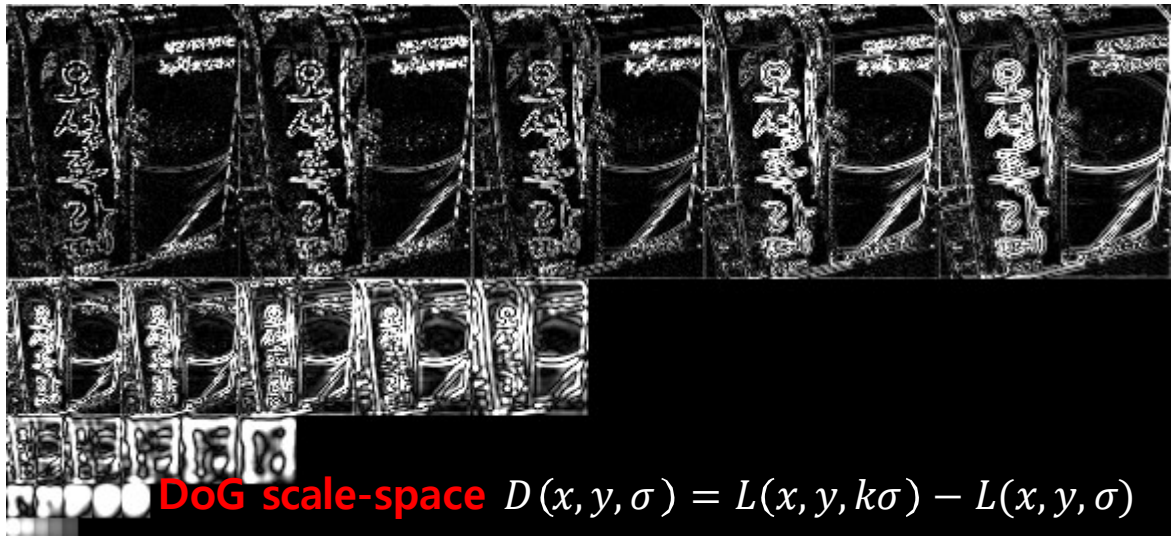
- Properties
  - **Invariant** to translation, rotation, and intensity shift ($I \rightarrow I + b$) ~~intensity scaling ($I \rightarrow aI$)~~
  - Variant to **image scaling**



SIFT (Scale-Invariant Feature Transform; 1999)

- Key idea: **Scale-space** (~ image pyramid) **and DoG** (difference of Gaussian)



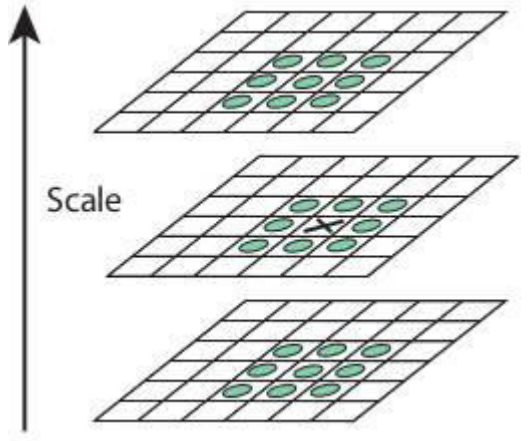


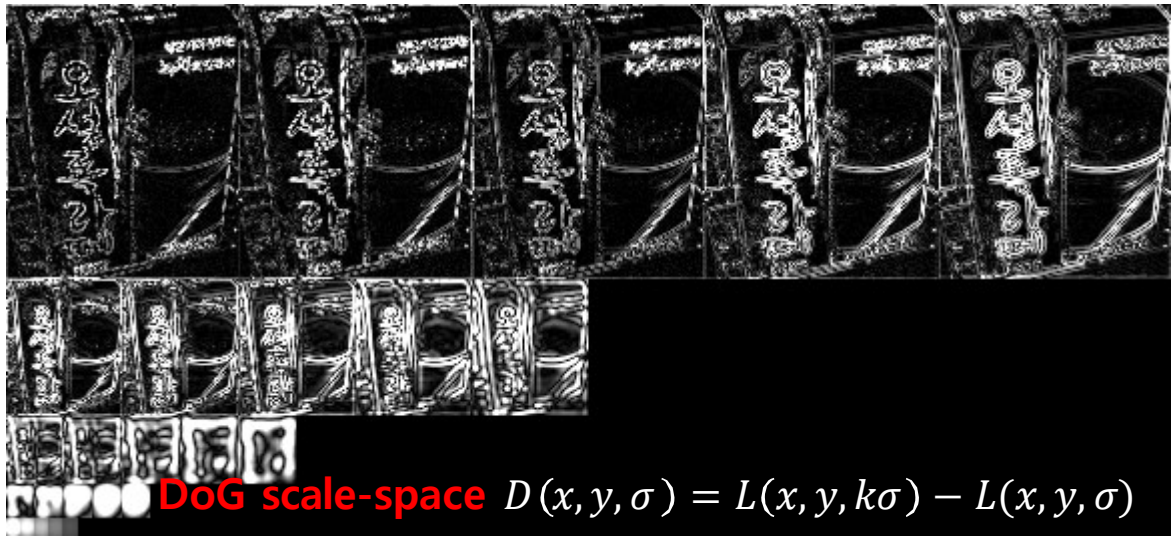
Note: The patent for SIFT feature and descriptors was expired at March 6, 2020!

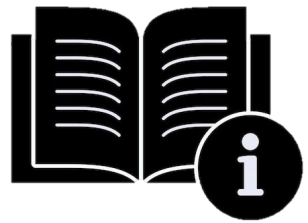
SIFT (Scale-Invariant Feature Transform; 1999)

- Key idea: **Scale-space** (~ image pyramid) **and DoG** (difference of Gaussian)
- Part #1) **Feature point detection**
  1. Find **local extrema** (minima and maxima) in DoG scale-space
  2. Localize their position accurately (sub-pixel level) using 3D quadratic function
  3. Eliminate **low contrast candidates**, $|D(\mathbf{x})| < \tau$
  4. Eliminate **candidates on edges**,

$$\frac{\text{trace}(H)^2}{\det(H)} < \frac{(r+1)^2}{r} \quad \text{where} \quad H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$



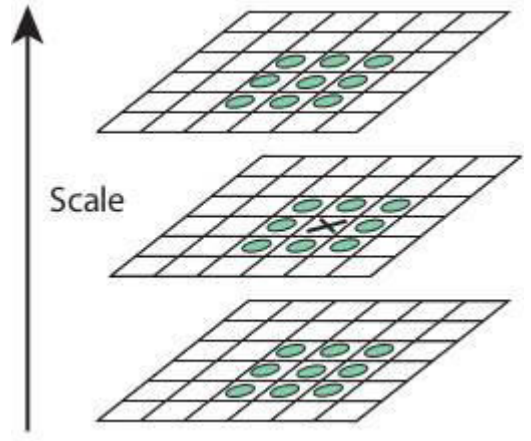


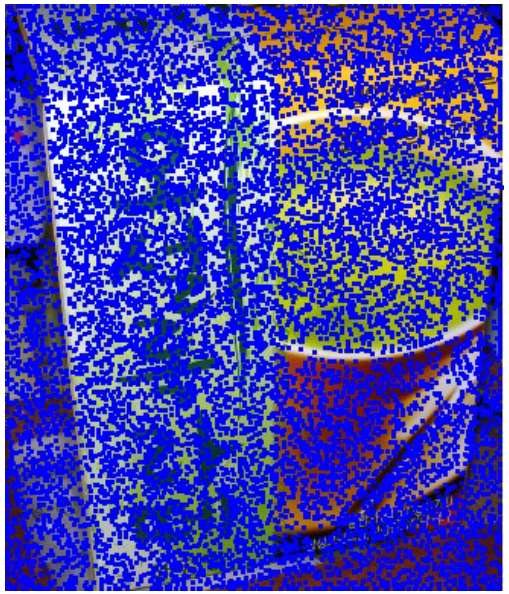


SIFT (Scale-Invariant Feature Transform; 1999)

- Key idea: **Scale-space** (~ image pyramid) **and DoG** (difference of Gaussian)
- Part #1) **Feature point detection**
  1. Find **local extrema** (minima and maxima) in DoG scale-space
  2. Localize their position accurately (sub-pixel level) using 3D quadratic function
  3. Eliminate **low contrast candidates**, $|D(\mathbf{x})| < \tau$
  4. Eliminate **candidates on edges**,

$$\frac{\text{trace}(H)^2}{\det(H)} < \frac{(r+1)^2}{r} \quad \text{where} \quad H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$





Local extrema (N: 11479)

Feature points (N: 971)

SIFT (Scale-Invariant Feature Transform; 1999)

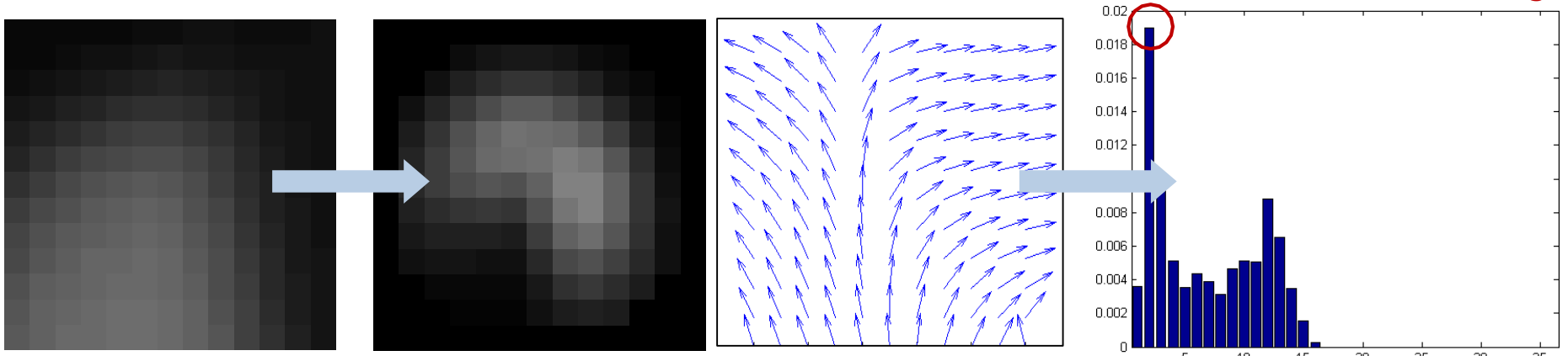
- Part #2) **Orientation assignment**
  1. Derive magnitude and orientation of gradient of each patch

$$m(x,y) = \sqrt{\big(L(x+1,y) - L(x-1,y)\big)^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}$$

  2. Find the strongest orientation
     - Histogram voting (36 bins) with Gaussian-weighted magnitude

Feature scales and orientations

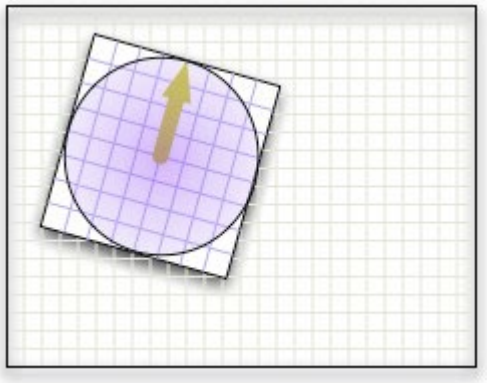


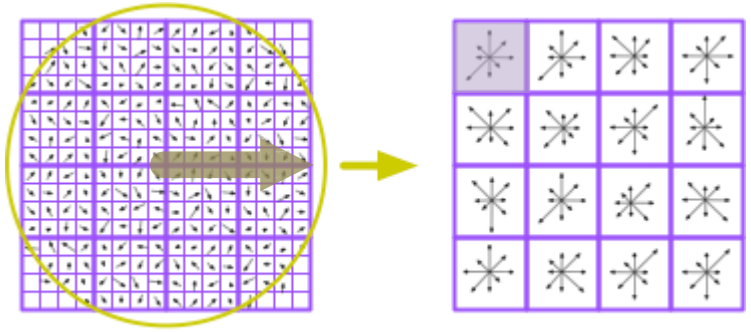
feature patch | gradient magnitude $m(x,y)$ | gradient orientation $\theta(x,y)$ | histogram voting (36 bins)

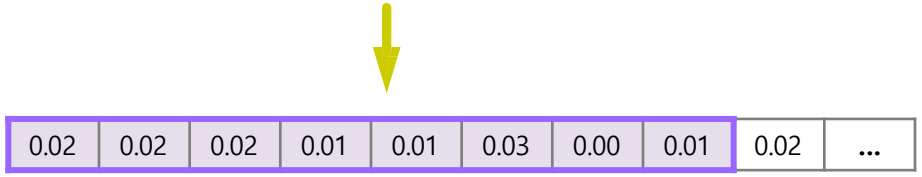
SIFT (Scale-Invariant Feature Transform; 1999)

- Part #3) **Feature descriptor extraction**
  1. Build a 4x4 gradient histogram (8 bins) from each patch (16x16 pixels)
     - Use Gaussian-weighted magnitude again
     - Use relative angles w.r.t. the assigned feature orientation
  2. Encode the histogram into a 128-dimensional vector
     - Apply normalization to be an unit vector

Feature scales and orientations



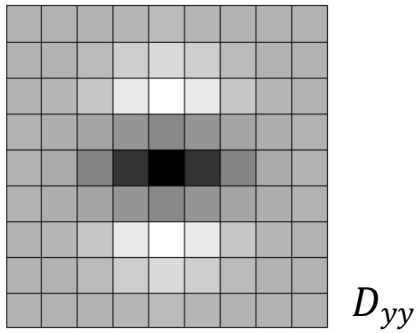




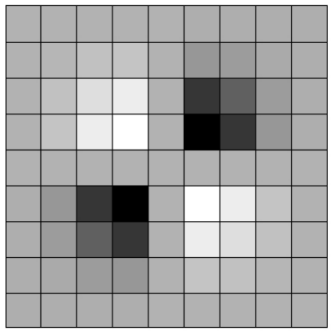
Keypoint descriptor (dim: 128)

SURF (Speeded Up Robust Features; 2006)

- Key idea: **Approximation of SIFT**
  - e.g. DoG approximation Haar-like features and **integral image**

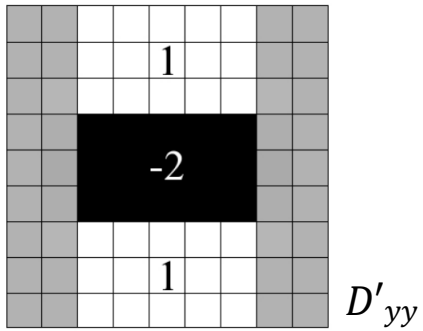


D_{yy}

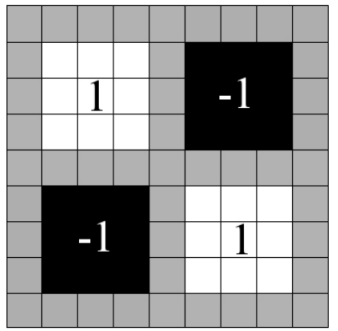


D_{xy}

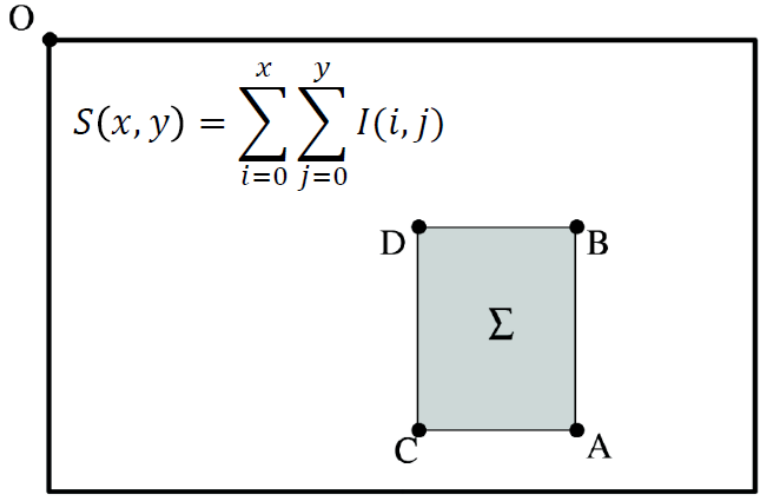
DoG approximation



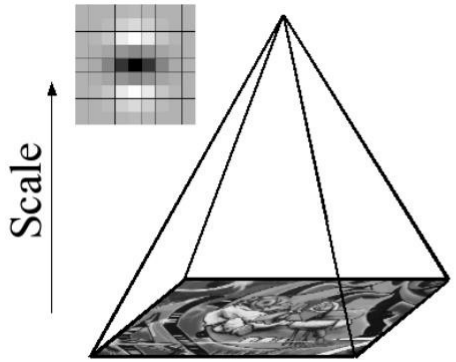
D'_{yy}

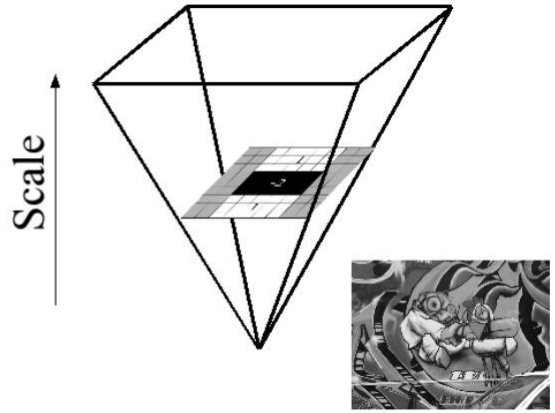


D'_{xy}



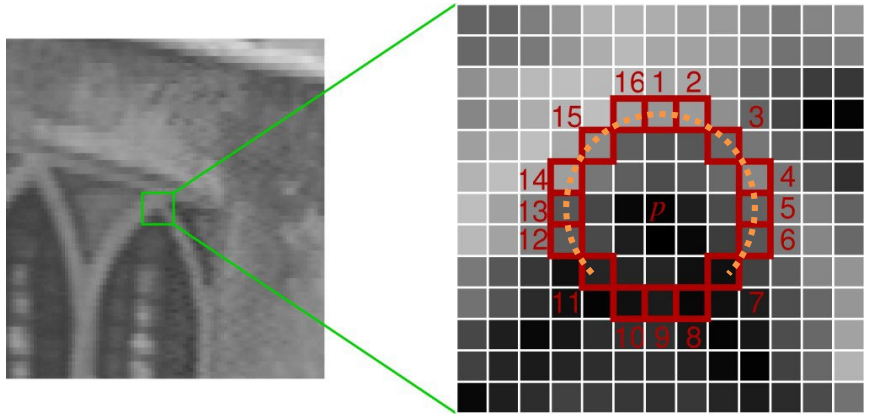
Σ=A-B-C+D



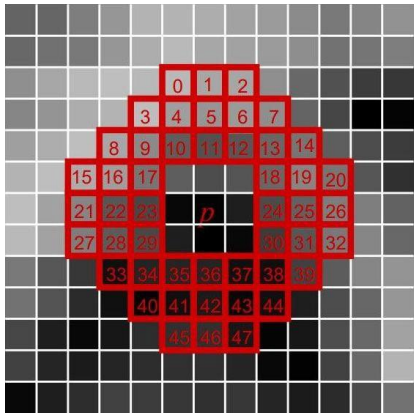


FAST (Features from Accelerated Segment Test; 2006)

- Key idea: **Intensity check of continuous arc of n pixels**
  - Is this point p a corner? (I_p: intensity at p, t: the intensity threshold)
    - Is a segment of n continuous pixels brighter than $I_p + t$? (OR) Is the segment darker than $I_p - t$?
    - Note: High-speed non-corner rejection: Checking intensity values at 1, 9, 5, and 13 (n: 12)

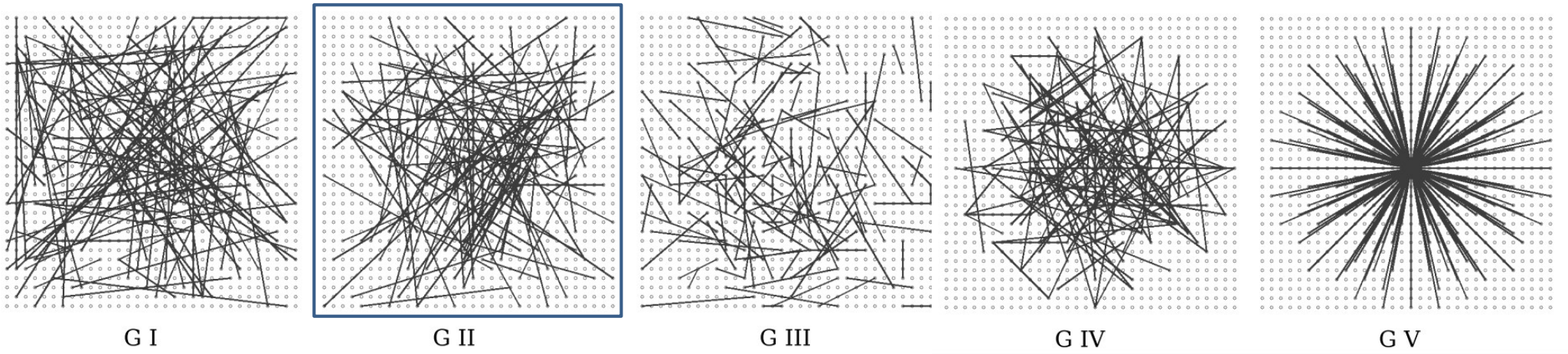


  - Too many corners! → Non-maximum suppression
- Versions
  - FAST-9 (n: 9; `cv.FastFeatureDetector_TYPE_9_16`), FAST-12 (n: 12), …
  - FAST-**ER**: Training a decision tree to **e**nhance **r**epeatability with more pixels



BRIEF (Binary Robust Independent Elementary Features; 2010)

- Key idea: **Intensity comparison of a sequence of random pairs (binary test)**
  - Path size: 31 x 31 pixels (Note: Applying smoothing for stability and repeatability)

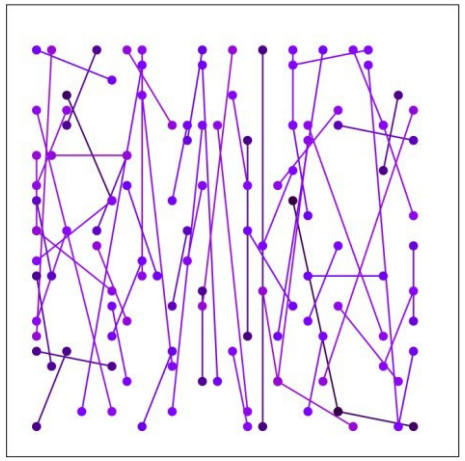


  - Descriptor size: 128 tests (128 bits) → 16 bytes
    - Note) SIFT: 128-dimensional vector → 512 bytes
- Versions: The number of tests
  - BRIEF-32, BRIEF-64, BRIEF-128 (16 bytes), BRIEF-256 (32 bytes), BRIEF-512 (64 bytes), …
- Combination examples
  - SIFT feature points + BRIEF descriptors
  - FAST feature points + BRIEF descriptors

ORB (Oriented FAST and rotated BRIEF, 2011)

- Key idea: **Adding rotation invariance to BRIEF**
  - **Oriented FAST**
    - Generate scale pyramid for scale invariance
    - Detect *FAST-9* points (filtering with Harris corner response)
    - Calculate feature orientation by *intensity centroid* $C = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}})$

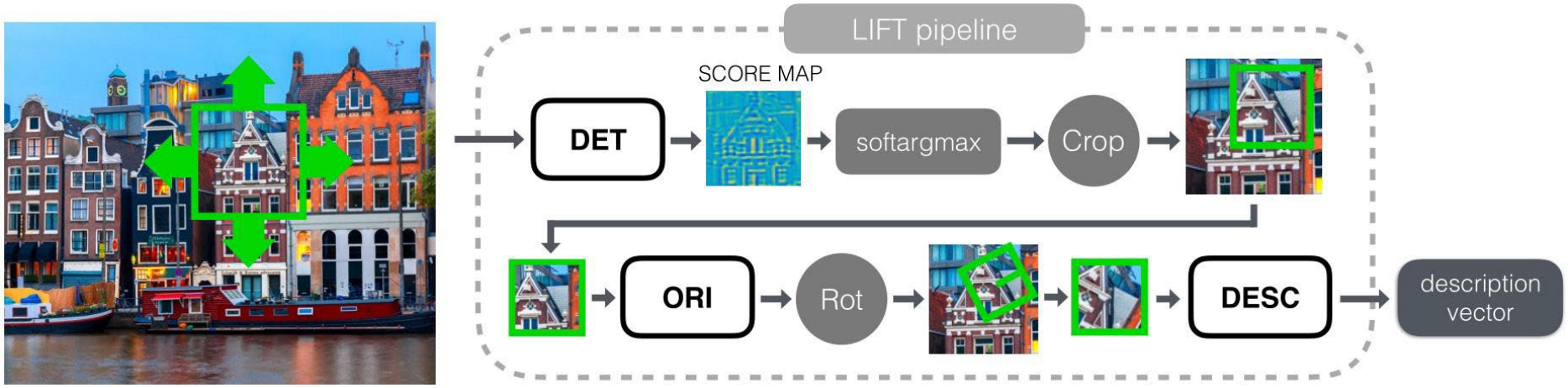
      $\theta = \tan^{-1} \frac{m_{01}}{m_{10}}$ where $m_{pq} = \sigma_{x,y} x^p y^q I(x, y)$
  - **Rotation-aware BRIEF**
    - Extract BRIEF descriptors w.r.t. the known orientation
    - Use better comparison pairs trained by greedy search

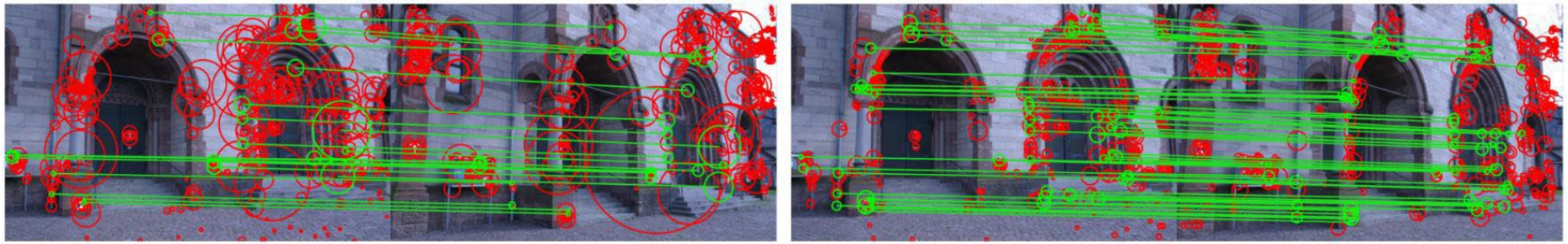


- Combination (default): **ORB**
  - FAST-9 detector (with orientation) + BRIEF-256 descriptor (with trained pairs)
- Computing time
  - ORB: **15.3 [msec]** / SURF: 217.3 [msec] / SIFT: 5228.7 [msec] @ 24 images (640x480) in Pascal dataset

LIFT (Learned Invariant Feature Transform; 2016)

- Key idea: **Deep neural network**
  - CNN network: **DET** (feature detector) + **ORI** (orientation estimator) + **DESC** (feature descriptor)
  - Training data: Photo Tourism dataset with VisualSFM (SIFT)

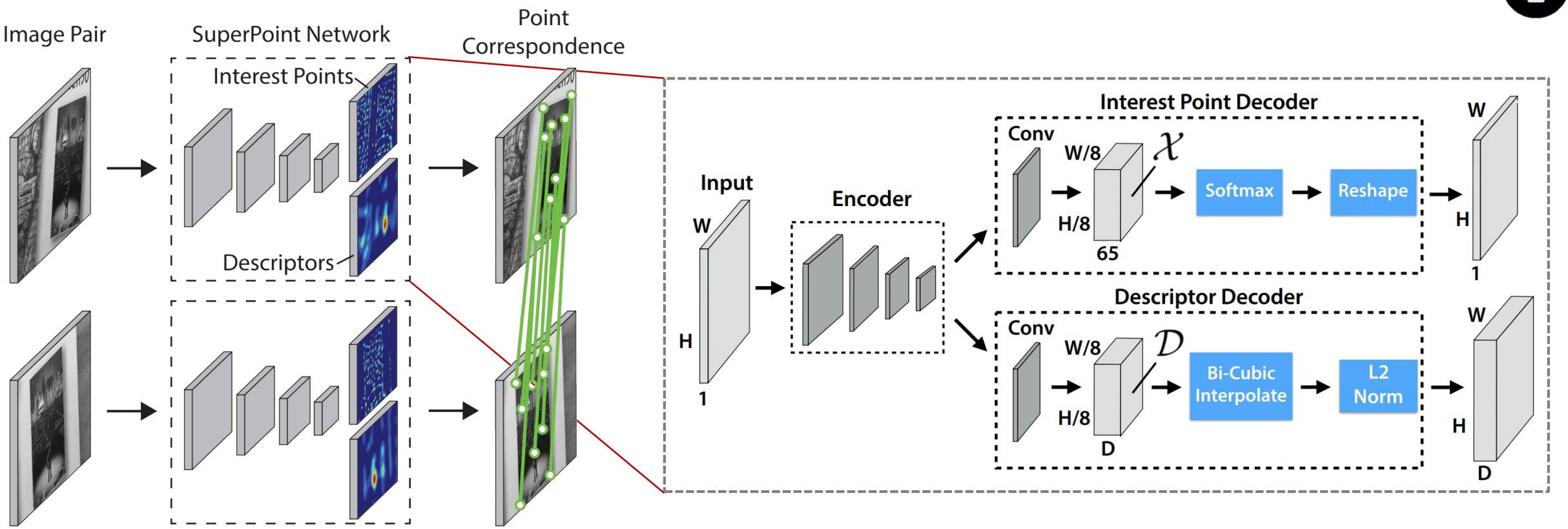




SIFT LIFT

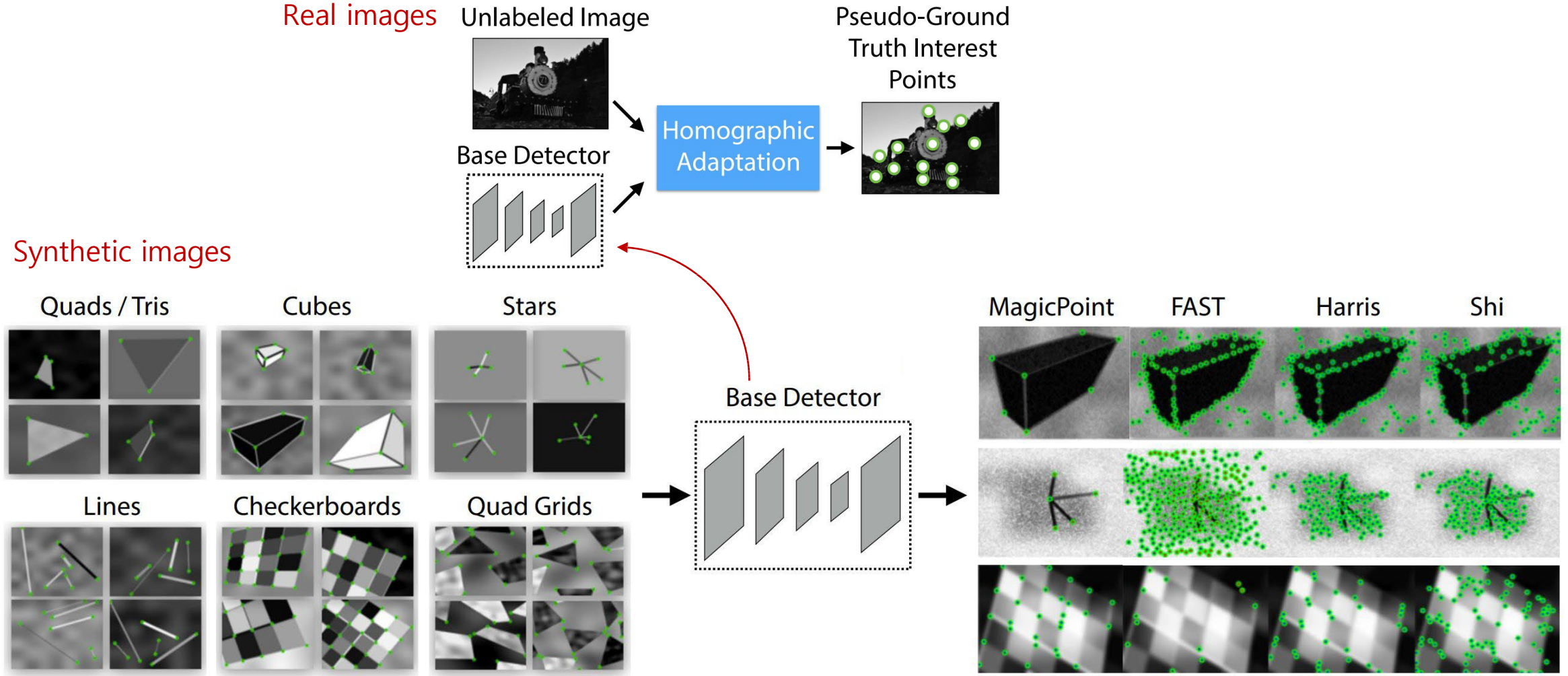
SuperPoint (2017)

- Key idea: **Self-supervised training with homography transformation**
  - CNN network: Encoder (~ VGG) + Decoders (for point and descriptor)



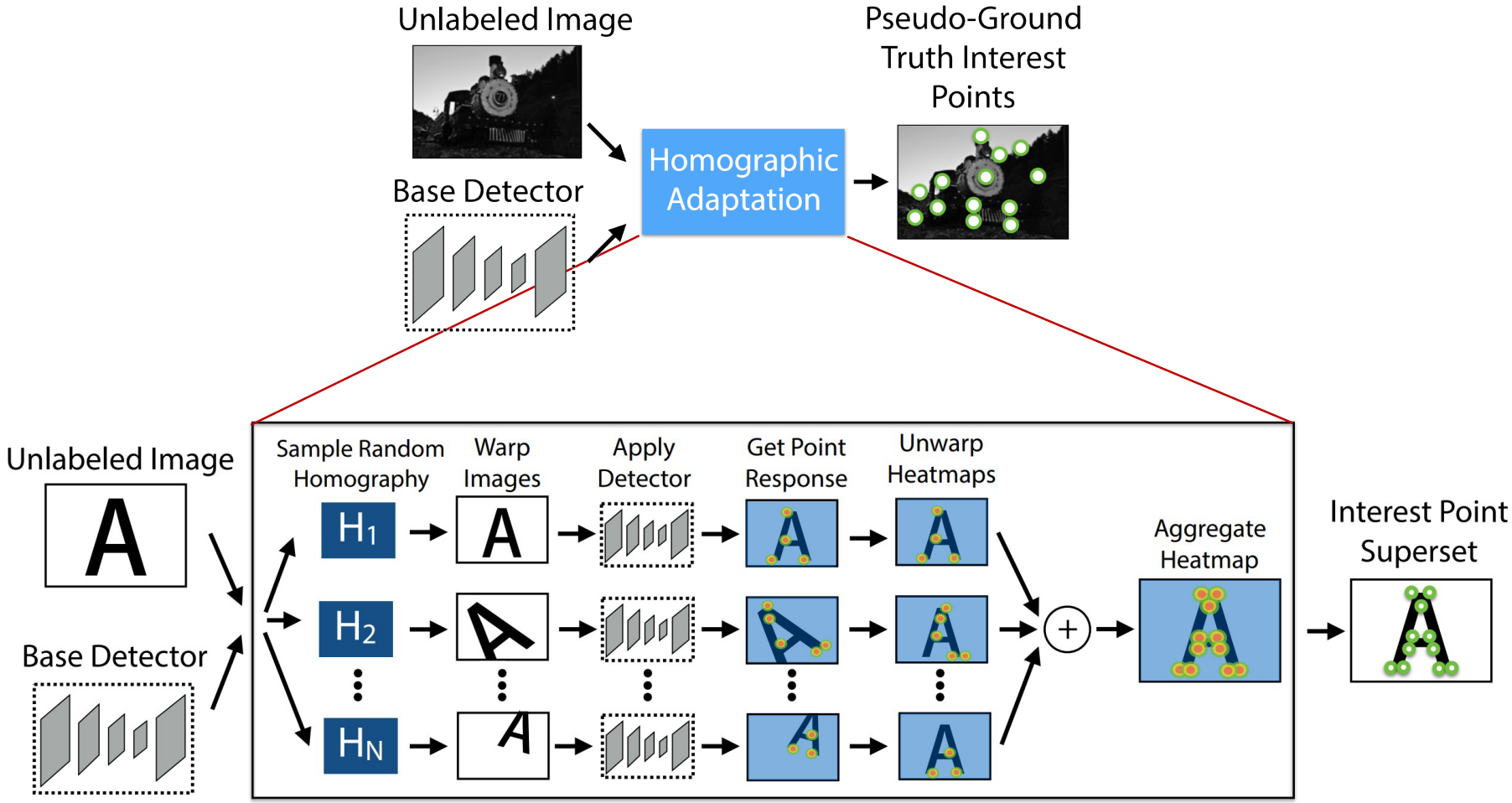
SuperPoint (2017)

- Key idea: **Self-supervised training with homography transformation**
  - Training data generation: The base detector, *MagicPoint* → Ground truth interest points



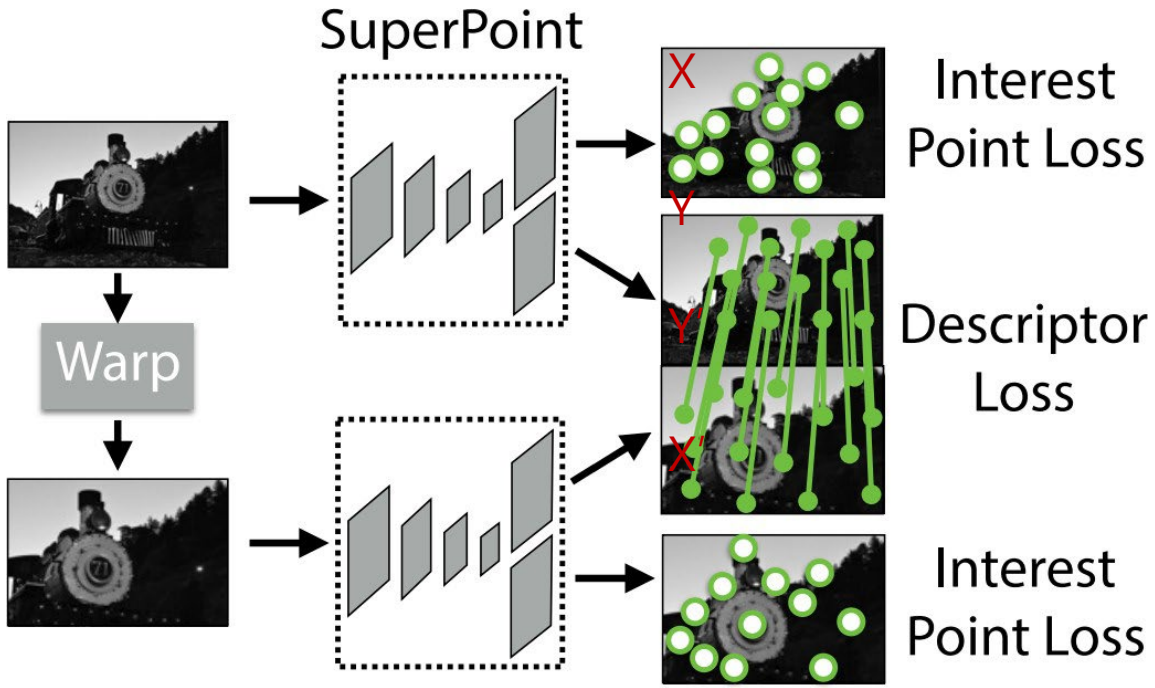
SuperPoint (2017)

- Key idea: **Self-supervised training with homography transformation**
  - Training data generation: The base detector, *MagicPoint* → Ground truth interest points



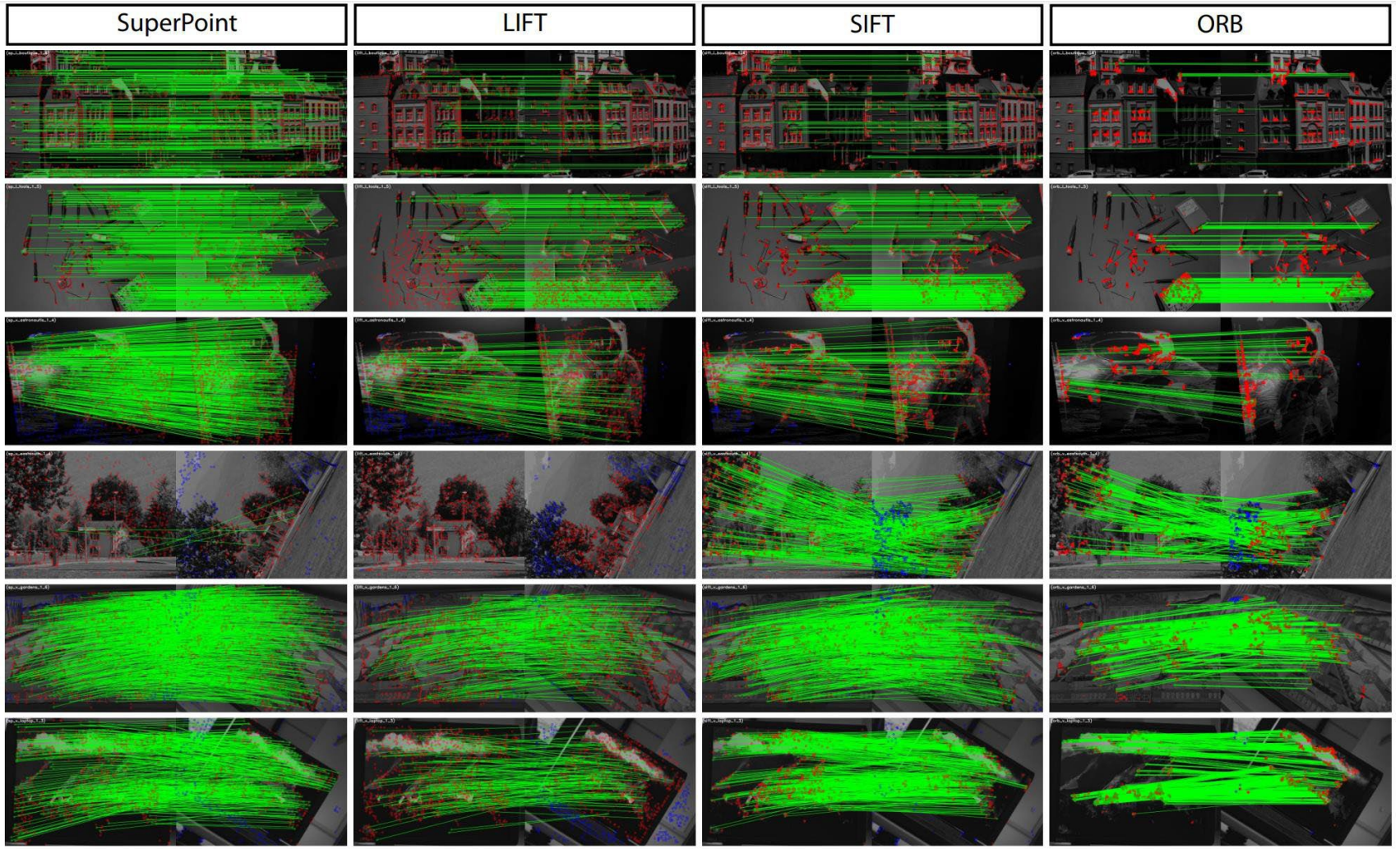
SuperPoint (2017)

- Key idea: **Self-supervised training with homography transformation**
  - Training data augmentation: Random homography transformation
  - Loss functions: Interest Point Loss (X, Y) + Interest Point Loss (X′, Y′) + Descriptor Loss (Y, Y′)



SuperPoint (2017)

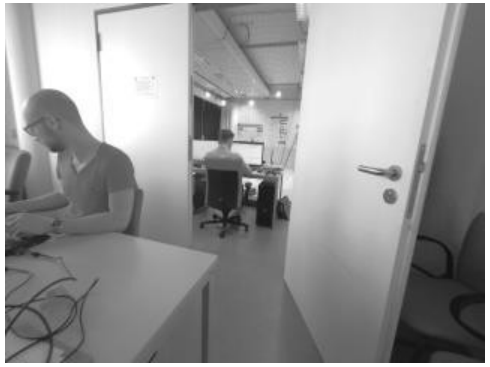
- Key idea: **Self-supervised training with homography transformation**
  - Real-time performance: **70 FPS** (13 msec) on 480 x 640 images with NVIDIA Titan X GPU



Freiburg RGBD:

Microsoft 7 Scenes:

MonoVO:



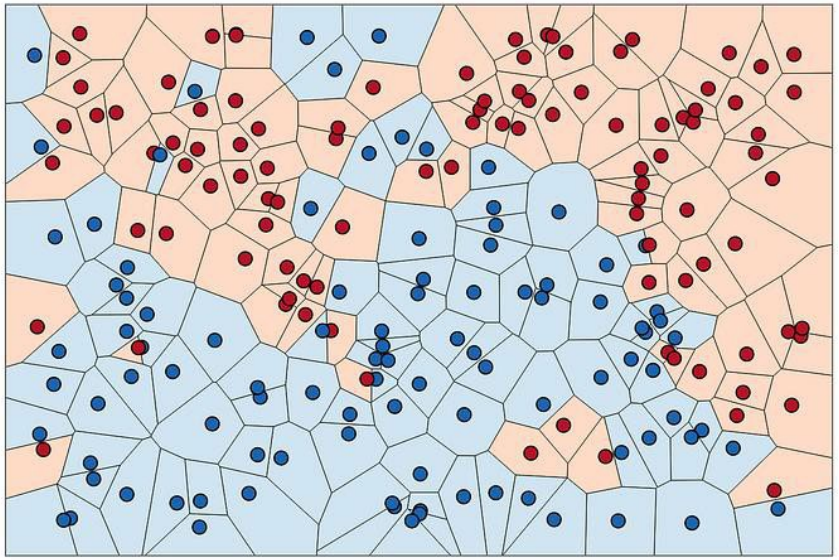
Feature Points and Descriptors

Feature Points	**Gradient-based** ▪ Harris ▪ GFTT (a.k.a. Shi-Tomasi) ▪ SIFT ▪ SURF	**Intensity-based** ▪ FAST	**DL-based** ▪ LIFT ▪ SuperPoint
Feature Descriptor	**Real-valued** ▪ SIFT ▪ SURF	**Binary-valued** ▪ BRIEF ▪ ORB (FAST+BRIEF)	**Real-valued (DL-based)** ▪ LIFT ▪ SuperPoint
Advantages Disadvantages	(+) Accurate (—) Slow	(+) Fast (—) Inaccurate (+) Less storage	(+) Accurate (+) Fast (—) GPU requirement

Feature Matching

For Real-valued Descriptors

- **Real-valued descriptors**
  - Distance measures
    - Euclidean distance: $l_2(\mathbf{d}, \mathbf{d}') = \|\mathbf{d} - \mathbf{d}'\|_2$ ($\downarrow$: similar)
    - Cosine similarity: $s_c(\mathbf{d}, \mathbf{d}') = \frac{\mathbf{d} \cdot \mathbf{d}'}{\|\mathbf{d}\| \|\mathbf{d}'\|}$ ($\uparrow$: similar)
    - Note: Matching measures can be combined or advanced.
      - e.g. The ratio of the best and second best similarity > threshold
        - It may select more distinguishable feature matching.
  - Matching algorithms
    - Brute-force search
      - Time complexity: $O(N)$ for N descriptors
    - Approximated nearest neighborhood search (ANN search)
      - Time complexity: $O(\log N)$ or less for N descriptors
      - Note: Big-ANN Competition (recent: NeurIPS 2023)

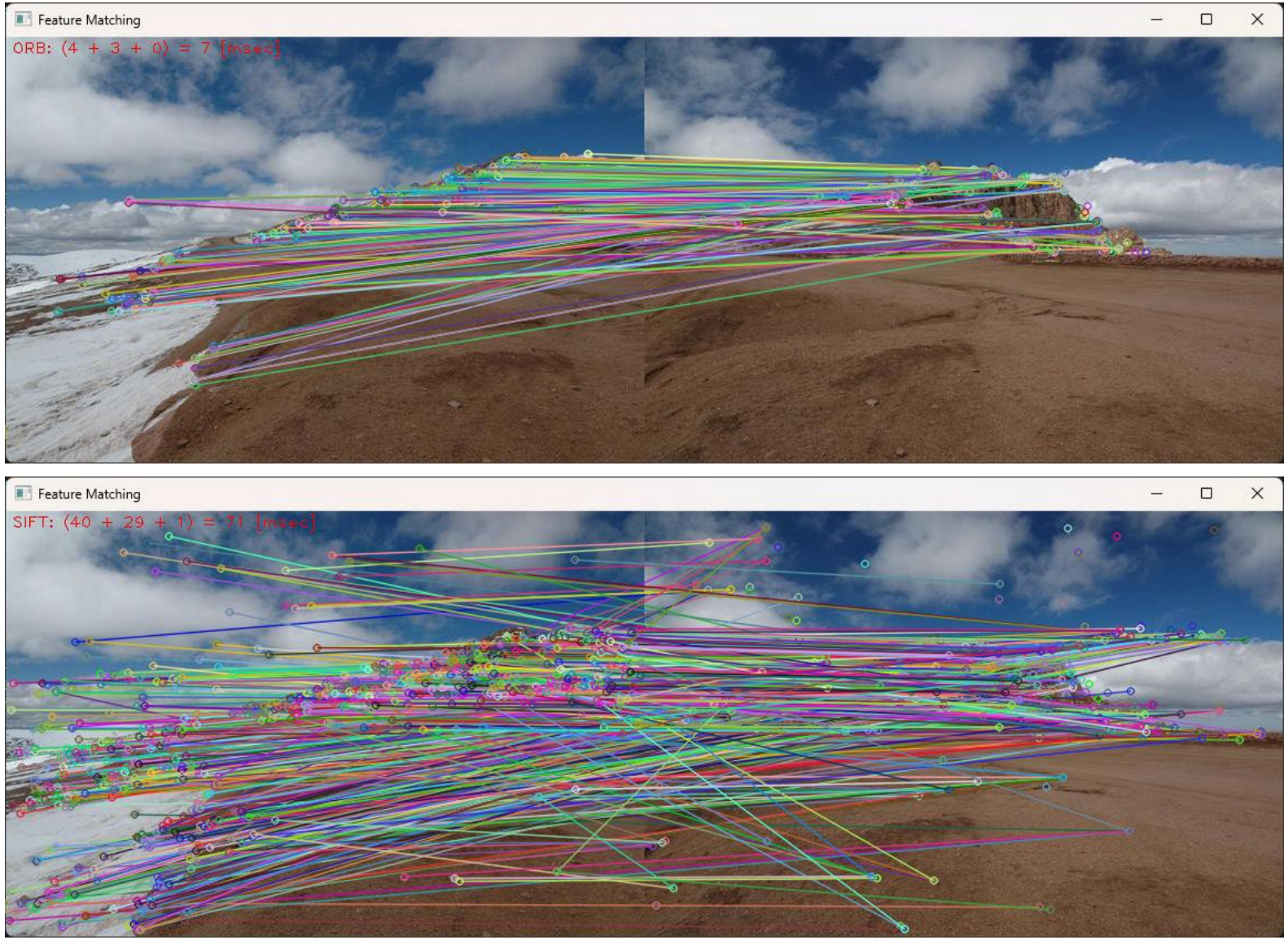


For Binary-valued Descriptors

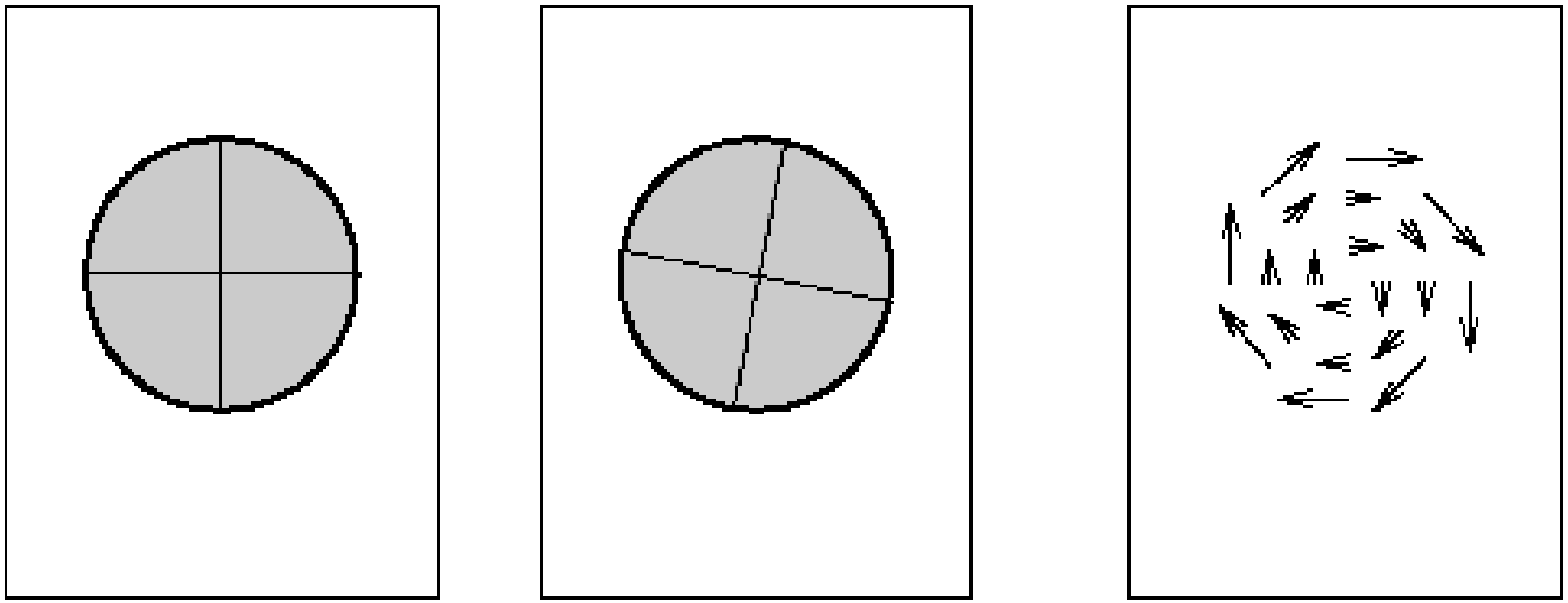
- **Binary-valued descriptors**
  - Distance measures
    - Hamming distance: $l_h(\mathbf{d}, \mathbf{d}') = \sigma_i(d_i \neq d_i')$
      - e.g. 4-bit descriptors
        - $l_h(0110, 1110) = 1$ vs. (6, 14)
        - $l_h(0110, 0111) = 1$ vs. (6, 7)
        - $l_h(0110, 0101) = 2$ vs. (6, 5)
      - Note: Hamming distance is the L_1-norm with binary-valued descriptors
  - Matching algorithms
    - Brute-force search
      - Time complexity: $O(N)$ for N descriptors

Feature Matching

- Example: **Feature matching comparison**



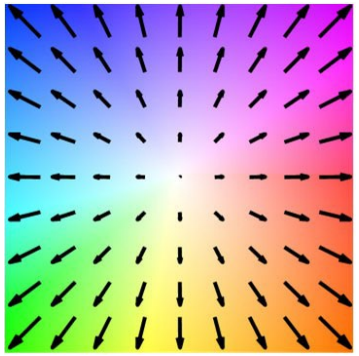
Optical Flow



Optical Flow: Dense Correspondence Over Time

Input [Liu *et al.* CVPR'08]

Optical flow (2D motion vector)



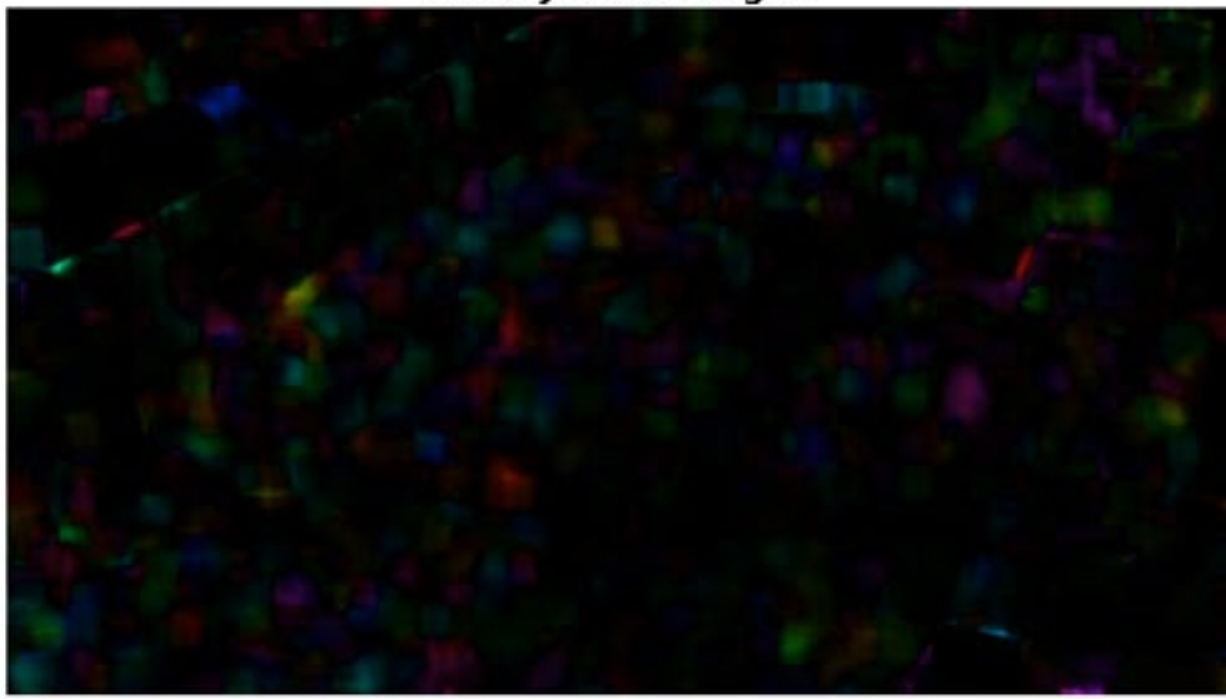
Color key
[Baker *et al.* IJCV'11]

Sandipan Dey (UMBC)

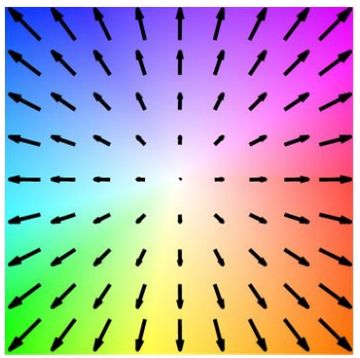
Optical Flow to find the Walk Direction of people in a Video with OpenCV-python

input

Velocity Vector Angles

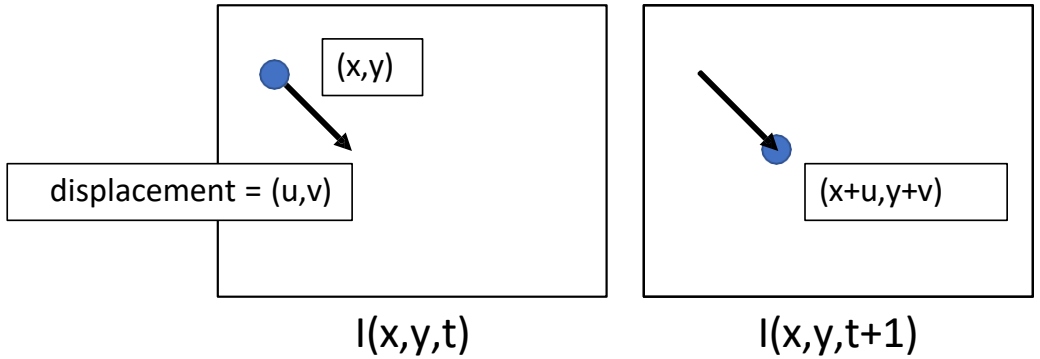


https://www.youtube.com/watch?v=e_TeY6QRp4c



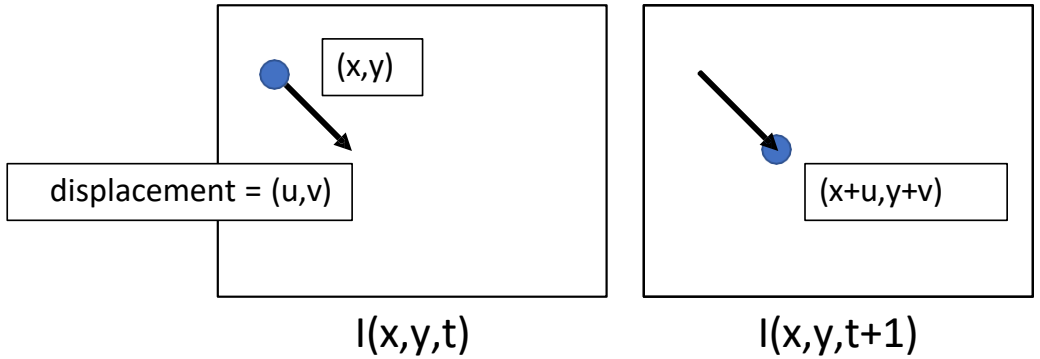
Color key

Optical Flow



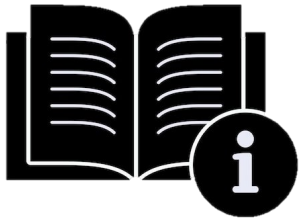
Brightness constancy: $I(x, y, t) = I(x + u, y + v, t + 1)$

Optical Flow



Brightness constancy: $I(x, y, t) = I(x + u, y + v, t + 1)$

Recall Taylor Expansion: $I(x + u, y + v, t + 1) = I(x, y, t) + I_x u + I_y v + I_t \ldots$



Optical Flow Equation

$$
\begin{aligned}
I(x+u, y+v, t+1) &= I(x,y,t) \\
0 &= I(x+u, y+v, t+1) - I(x,y,t) \\
&\approx I(x,y,t) + I_t + I_x u + I_y v - I(x,y,t) \\
&= I_t + I_x u + I_y v \\
&= I_t + \nabla I \cdot [u, v]
\end{aligned}
$$

Taylor Expansion

When is this approximation **bad**?

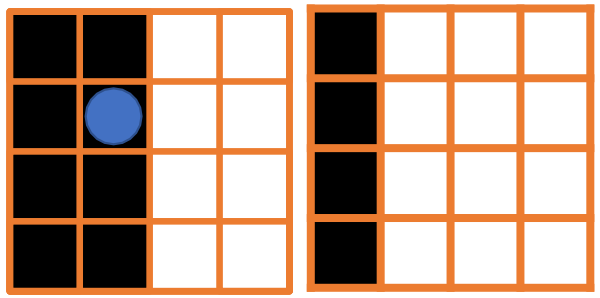
u or v big

Brightness constancy equation

$$I_x u + I_y v + I_t = 0$$

Brightness Constancy Example

$$I_x u + I_y v + I_t = 0$$



t t+1

It = 1-0 = 1

@  Iy = 0

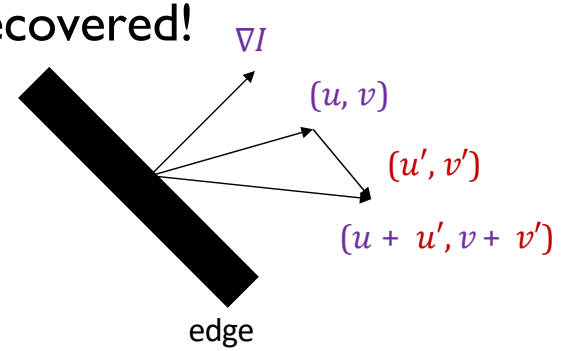
Ix = 1-0 = 1

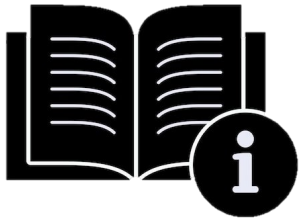
What's u? What's v?

The Brightness Constancy Constraint

$$I_x u + I_y v + I_t = 0$$

- Given the gradients I_x, I_y and I_t, can we uniquely recover the motion (u, v)?
  - One equation, two unknowns
  - Suppose (u, v) satisfies the constraint: $\nabla I \cdot (u, v) + I_t = 0$
  - Then $\nabla I \cdot (u + u', v + v') + I_t = 0$ for any (u', v') s. t. $\nabla I \cdot (u', v') = 0$
  - Interpretation: the component of the flow perpendicular to the gradient (i.e., parallel to the edge) cannot be recovered!





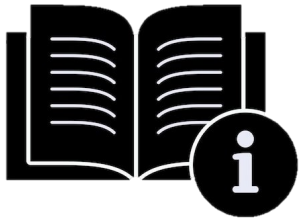
Solving Ambiguity – Lucas Kanade

- 2 unknowns [u,v], 1 eqn per pixel How do we get more equations?
- Assume **spatial coherence**: pixel's neighbors have move together / have same [u,v]
- 5x5 window gives 25 new equations

$$I_t + I_x u + I_y v = 0$$

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}$$

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.



Solving for u,v

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ \vdots \\ I_t(p_{25}) \end{bmatrix} \qquad \underset{25x2}{\boldsymbol{A}} \; \underset{2x1}{\boldsymbol{d}} = \underset{25x1}{\boldsymbol{b}}$$

What's the solution?

$$(\boldsymbol{A}^T\boldsymbol{A})\boldsymbol{d} = \boldsymbol{A}^T\boldsymbol{b} \quad \rightarrow \quad \boldsymbol{d} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$$

Intuitively, need to solve (sum over pixels in window)

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

Challenges for traditional methods

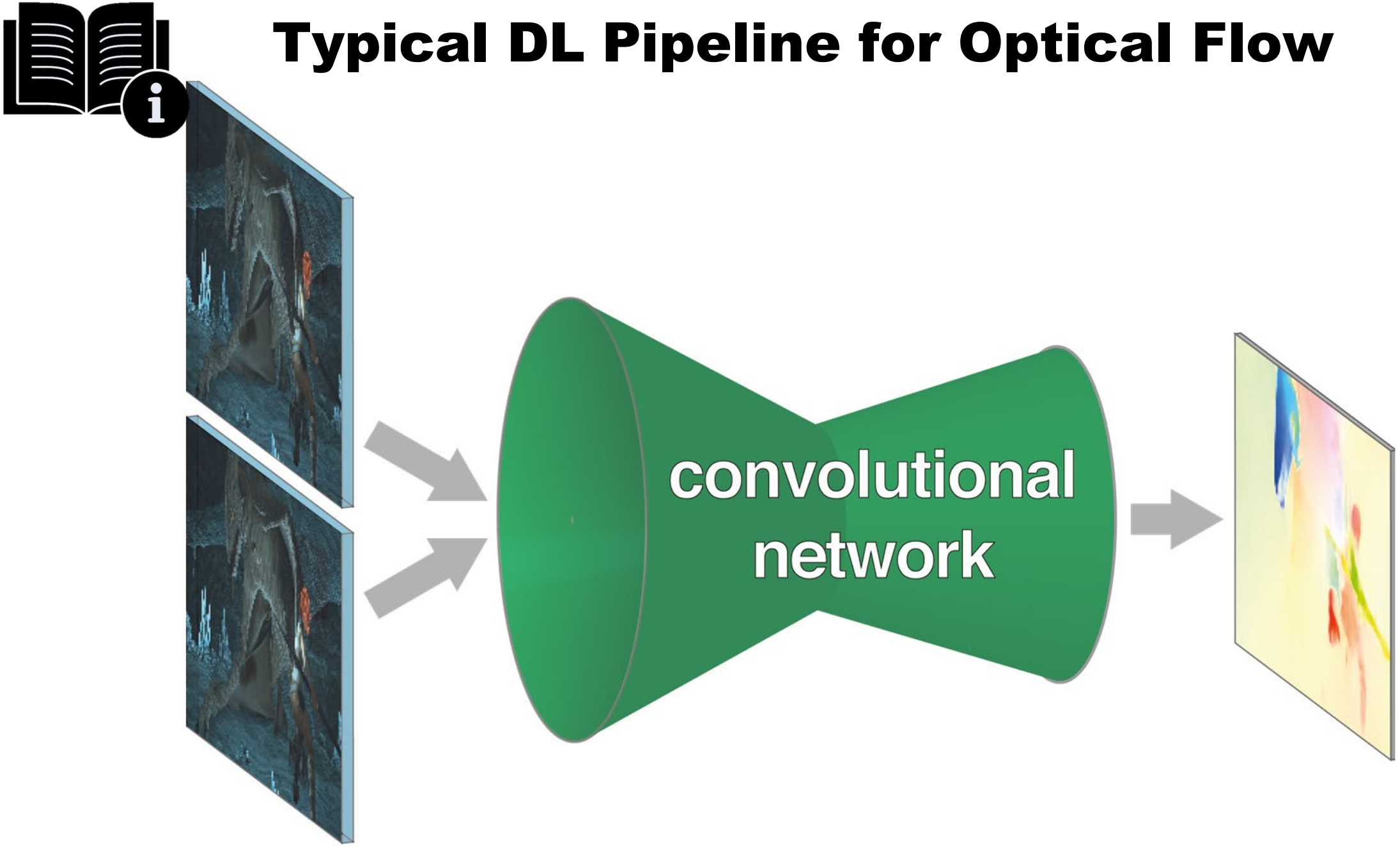
Large motion, motion blur

Textureless regions

Occlusions

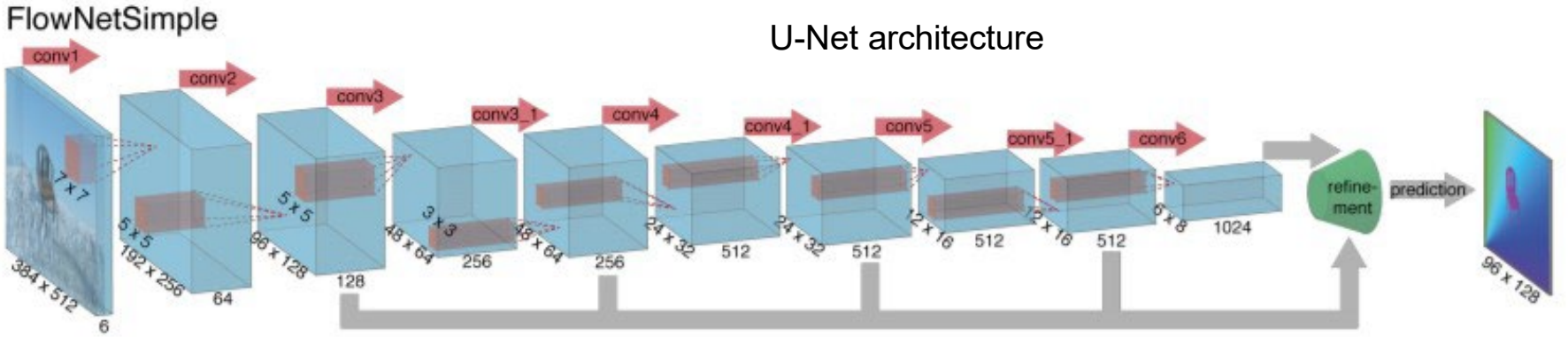
Lighting changes, noise …

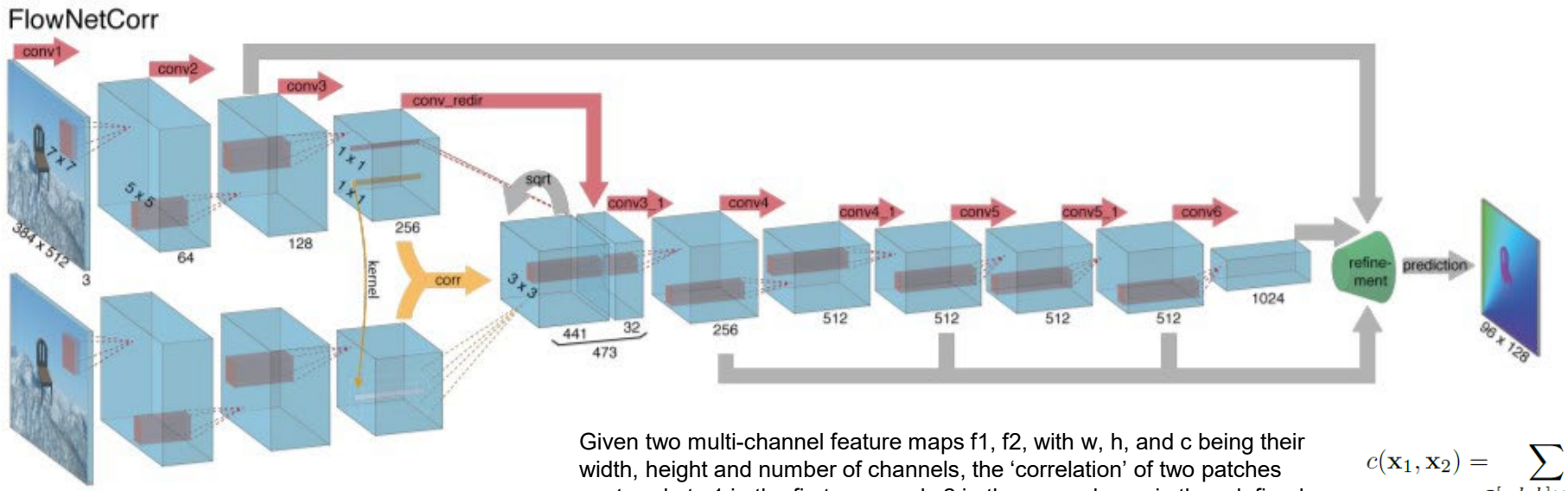
Typical DL Pipeline for Optical Flow



FlowNetS: mapping from images to flow

[Dosovitskiy *et al.* ICCV'15]

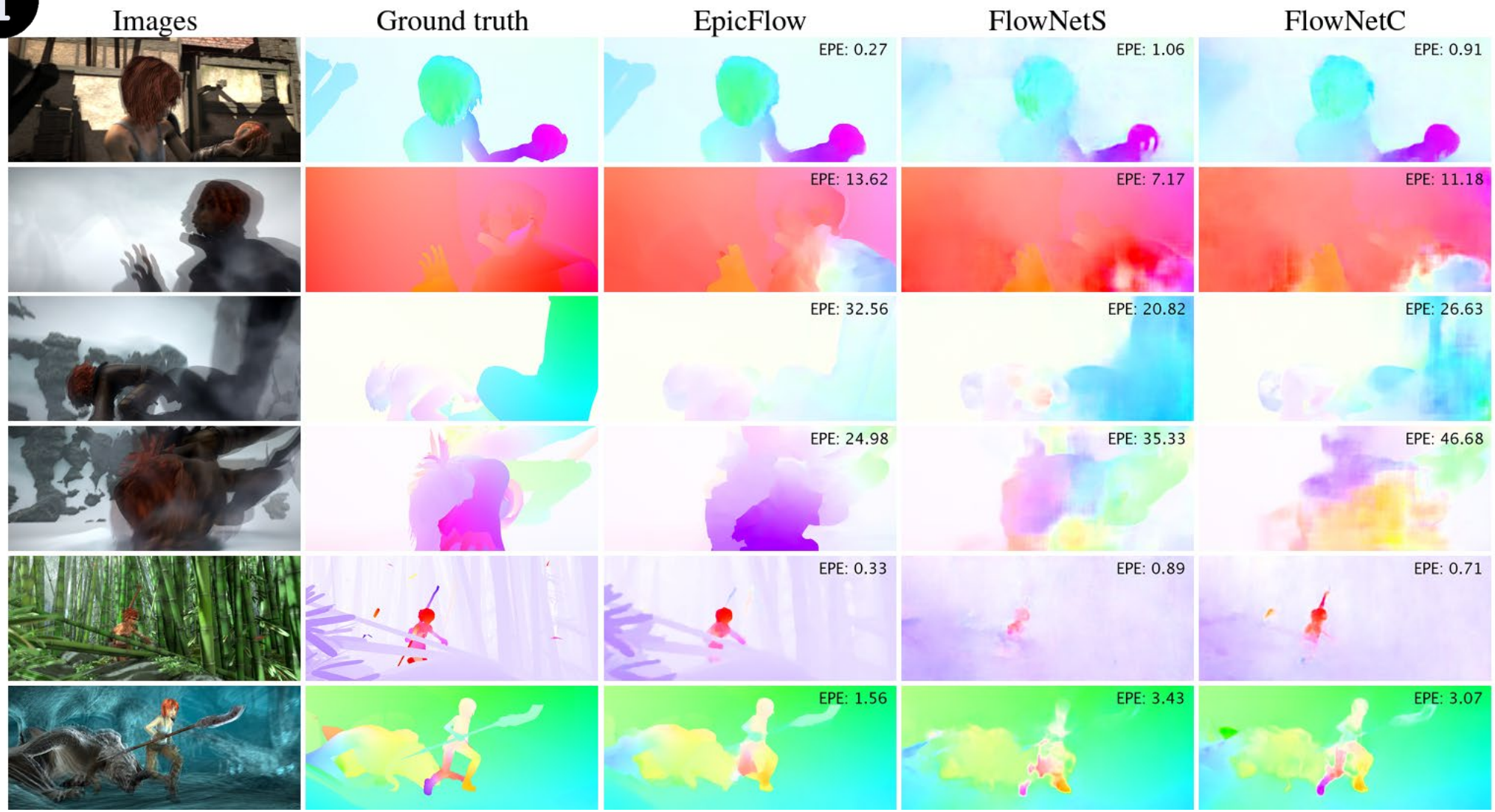




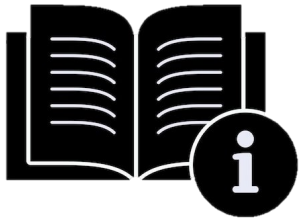
Given two multi-channel feature maps f1, f2, with w, h, and c being their width, height and number of channels, the 'correlation' of two patches centered at x1 in the first map and x2 in the second map is then defined as:

$$c(\mathbf{x}_1, \mathbf{x}_2) = \sum_{\mathbf{o} \in [-k,k] \times [-k,k]} \langle \mathbf{f}_1(\mathbf{x}_1 + \mathbf{o}), \mathbf{f}_2(\mathbf{x}_2 + \mathbf{o}) \rangle$$

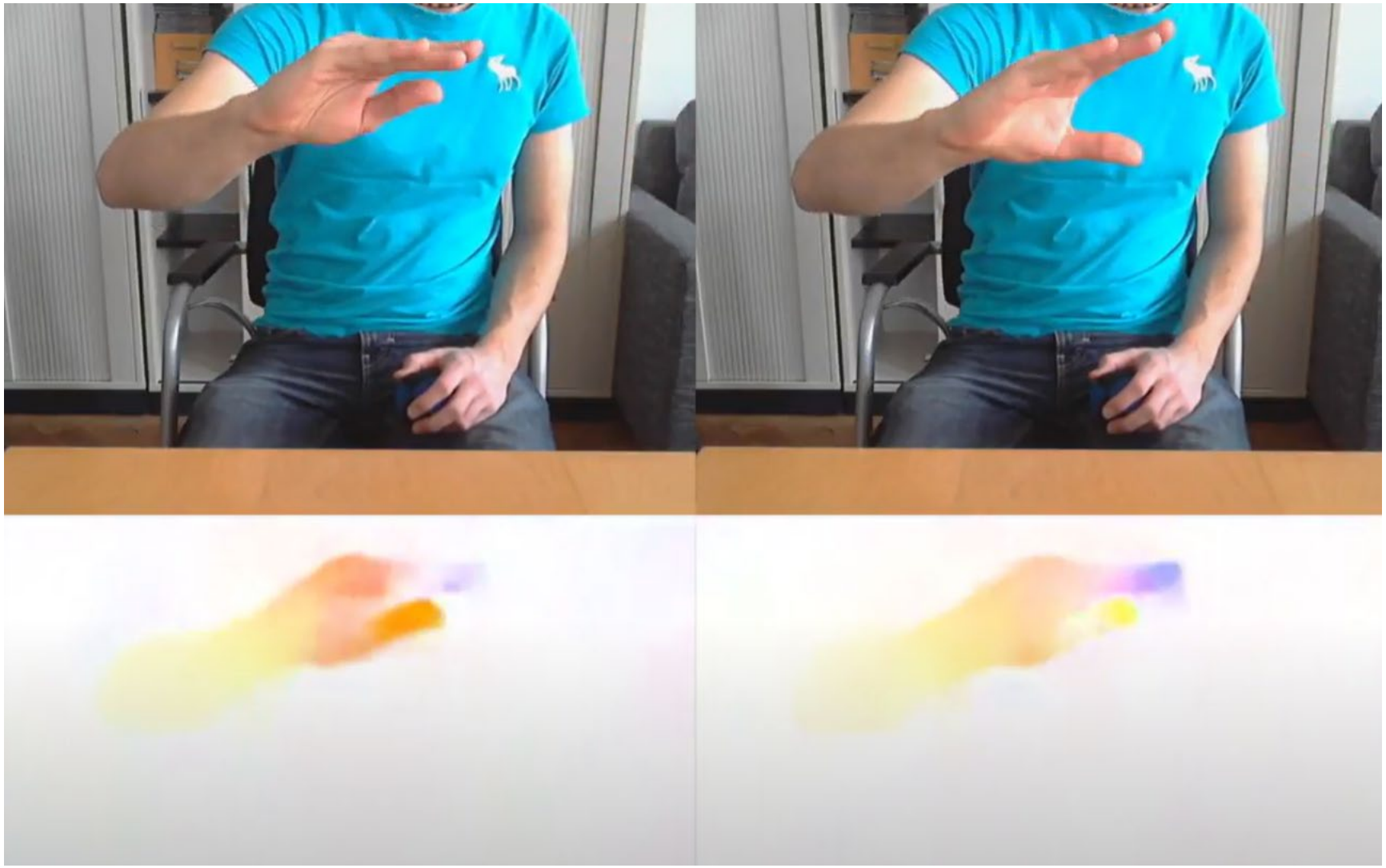
FlowNetS: mapping from images to flow



EPE: endpoint error



FlowNetS: mapping from images to flow



https://youtu.be/k_wkDLJ8lJE?si=wJj1RePbw-NA75lU

Lukas-Kanade Optical Flow (1981)

- Key idea: **Finding movement of a patch whose pixel values are same**
  - Brightness constancy constraint: $I(x, y, t) = I(x + \Delta_x, y + \Delta_y, t + \Delta_t)$

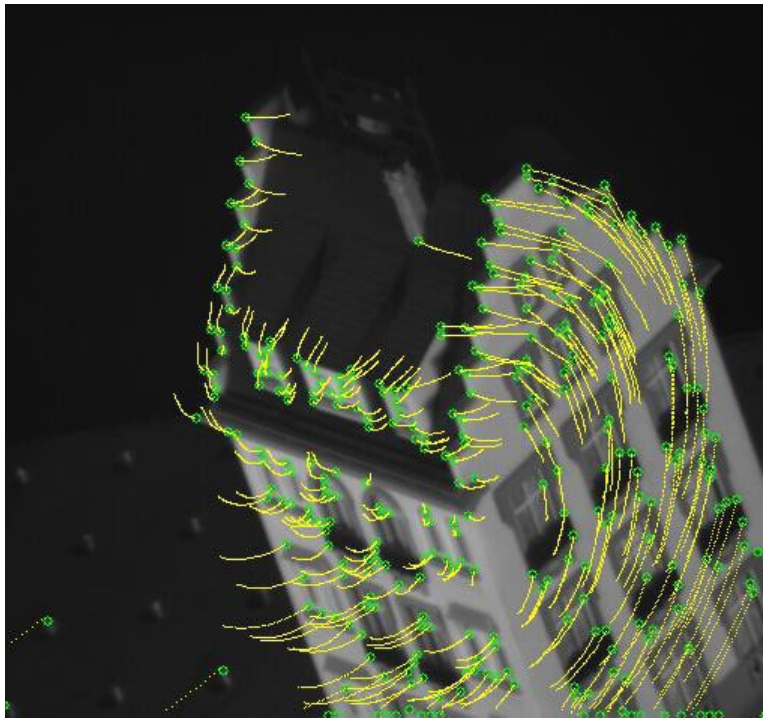
$$I_x \frac{\Delta_x}{\Delta_t} + I_y \frac{\Delta_y}{\Delta_t} + I_t = 0 \quad \leftarrow \quad I(x + \Delta_x, y + \Delta_y, t + \Delta_t) \approx I(x, y, t) + I_x \Delta_x + I_y \Delta_y + I_t \Delta_t$$

$$\mathrm{A}\mathbf{x} = \mathbf{b} \quad \text{where} \quad \mathrm{A} = [I_x \quad I_y],\ \mathbf{x} = \left[\Delta_x,\ \Delta_y\right]^{\mathsf{T}},\ \text{and}\ \mathbf{b} = [-I_t] \quad (\Delta_t = 1)$$

$$\therefore \mathbf{x} = \mathrm{A}^{\dagger}\,\mathbf{b}$$

- Combination: **KLT tracker**
  - GFTT (a.k.a. Shi-**T**omasi) detector + **L**ukas-**K**anade optical flow

- Advantages and disadvantages (feature tracking vs. matching)
  - (+) No descriptor required (→ fast and compact)
  - (–) Continuous feature tracking causes drift errors
  - (–) Not working in wide-baseline cases
  - (+) Able to control matching range



Lukas-Kanade Optical Flow (1981)

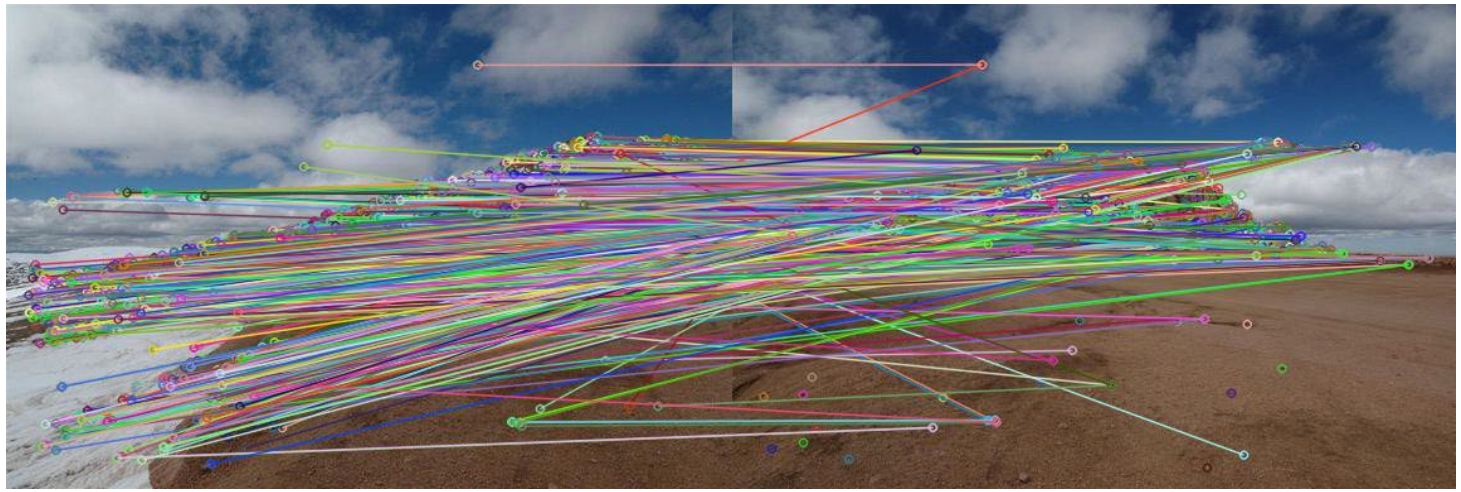
- Example: Lukas-Kanade tracker

Lukas-Kanade Optical Flow (1981)

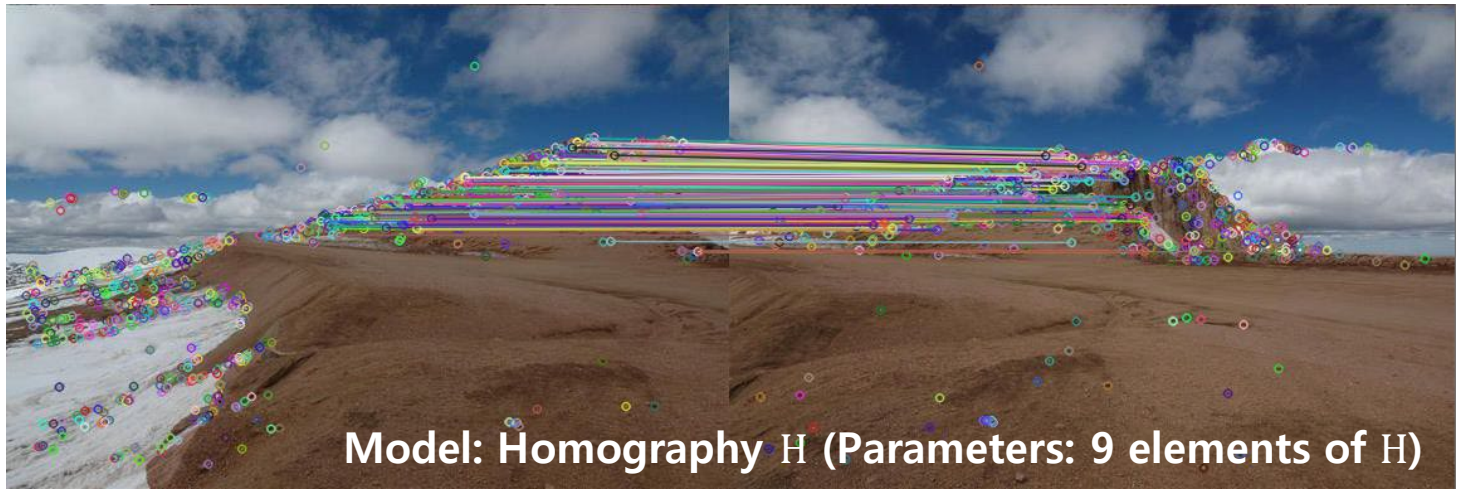
- Example: Lukas-Kanade tracker

How to Remove Outliers?

Putative feature matches (inliers + outliers)



After applying RANSAC (inliers)



References

- **Slides:** Photogrammetry I & II Course (https://www.ipb.uni-bonn.de/photo12-2021/index.html)
- **Slides:** Mines Computer Vision Course (by Thomas Williams)
- **Slides:** MIT's Advances in Computer Vision (http://6.869.csail.mit.edu/fa19/materials.html)
- **OpenCV camera calibration**: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html