電腦視覺與應用
Computer Vision and Applications

Lecture09-Feature detection and matching

Tzung-Han Lin
National Taiwan University of Science and Technology
Graduate Institute of Color and Illumination Technology
e-mail: thl@mail.ntust.edu.tw

TAIWAN TECH National Taiwan University of Science and Technology

CIT 色彩與照明科技研究所 Graduate Institute of Color and Illumination Technology

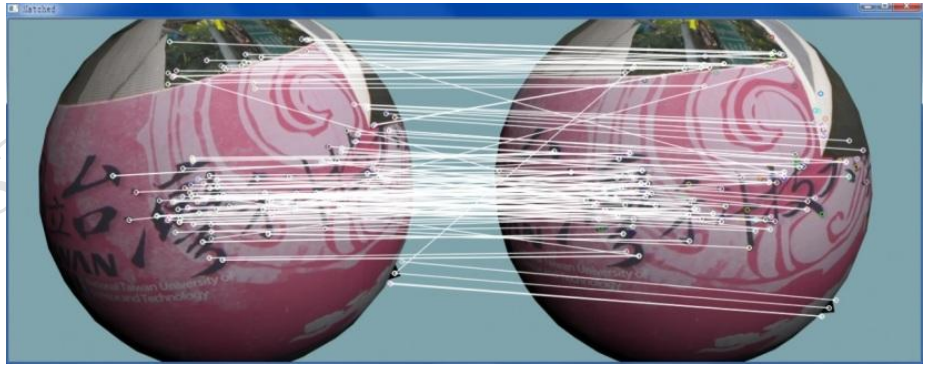
Keyword list

- Feature, feature detection, feature matching, feature tracking
- Algorithm: Harr is corner, Tomasi's, Brown, SIFT, SURF
- SIFT: Scale Invariant Feature Tracking
- SURF: Speeded-Up Robust Features
- Taylor series

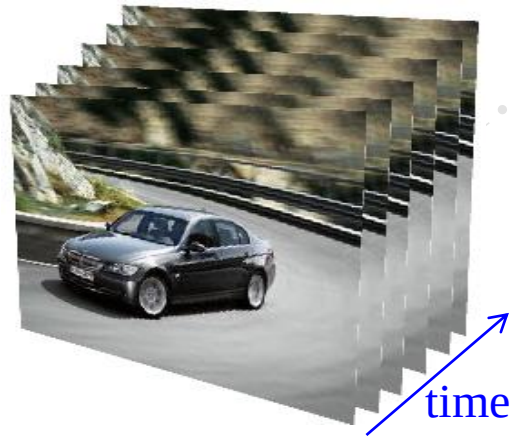
Feature detection and matching

- Brief introduction to
  - Feature Detection→ single frame
  - Feature Matching→ multi images
  - Tracking→ single video

Feature Detection



Feature Matching



Tracking

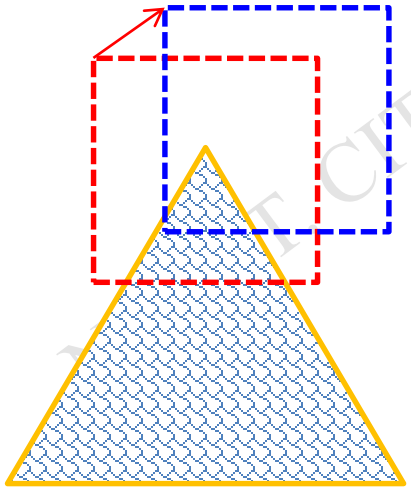
Feature detection and matching

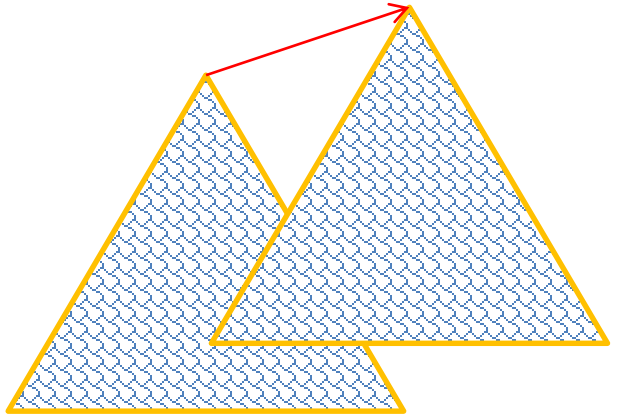
- Feature Detection
  - Harris Corner
  - Tomasi's Good Feature
  - Brown
  - SIFT
  - SURF

- High Level feature → machine/deep learning field (recognition)
  - Face, hand, eye, car, text..
  - codebook, classifier

Feature detection: Harris corner detector

- Recognize the point by looking through a small window
- Shifting a window in any direction should induce a large change in intensity





C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, 1988, pp. 147-151.

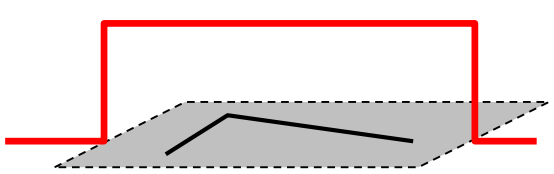
Feature detection: Harris corner detector

- Estimating “the Change” of intensity when having the shift [u,v]

$$E(u,v)=\sum_{x,y} w(x,y)[I(x+u,y+v)-I(x,y)]^2$$

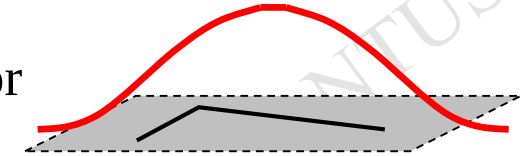
Window function(filter) — Shifted intensity — Intensity

Window function $w(x,y)$ =



1 in window, 0 outside

or



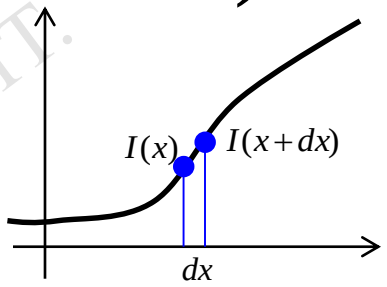
Gaussian

Taylor series (Taylor expansion for approximation)

- 1D case

$$I(x+dx) = I(x) + \frac{\partial I(x)}{\partial x}dx$$

First derivatives



$$I(x+\Delta x) = I(x) + I_x(x)\Delta x + \frac{(\Delta x)^2}{2!}I_{xx}(x) + \frac{(\Delta x)^3}{3!}I_{xxx}(x) + \ldots$$

2nd , 3rd derivatives

- 2D case (in image)

$$I(x+u, y+v) = I(x,y) + uI_x(x,y) + vI_y(x,y) +$$

$$\frac{1}{2!}[u^2 I_{xx}(x,y) + v^2 I_{yy}(x,y) + uvI_{xy}(x,y)] + \ldots.$$

1st partial derivatives

$$I(x+u, y+v) \approx I(x,y) + uI_x(x,y) + vI_y(x,y)$$

- 1st order approximation (for 2D case)

Taylor series (Taylor expansion for approximation)

$$I(x+u, y+v) \approx I(x,y) + uI_x(x,y) + vI_y(x,y)$$

$$\rightarrow\ I(x+u, y+v) - I(x,y) \approx uI_x(x,y) + vI_y(x,y)$$

- Recall Harris corner equation:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2$$

$$\rightarrow E(u,v) \approx \sum_{x,y} w(x,y)[uI_x(x,y) + vI_y(x,y)]^2$$

$$= \sum_{x,y} w(x,y)[u^2 I_x^{\ 2} + v^2 I_y^{\ 2} + 2uvI_xI_y]$$

$$= \sum_{x,y} w(x,y)\begin{bmatrix} u & v \end{bmatrix}\begin{bmatrix} I_x^{\ 2} & I_xI_y \\ I_xI_y & I_y^{\ 2} \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

Feature detection: Harris corner detector

- For small shifts [u, v] we have a bilinear approximation: (from Taylor expansion)

$$E(u,v) \cong \begin{bmatrix} u & v \end{bmatrix} \mathbf{M} \begin{bmatrix} u \\ v \end{bmatrix}$$

- where M is a 2×2 matrix computed from image derivatives:

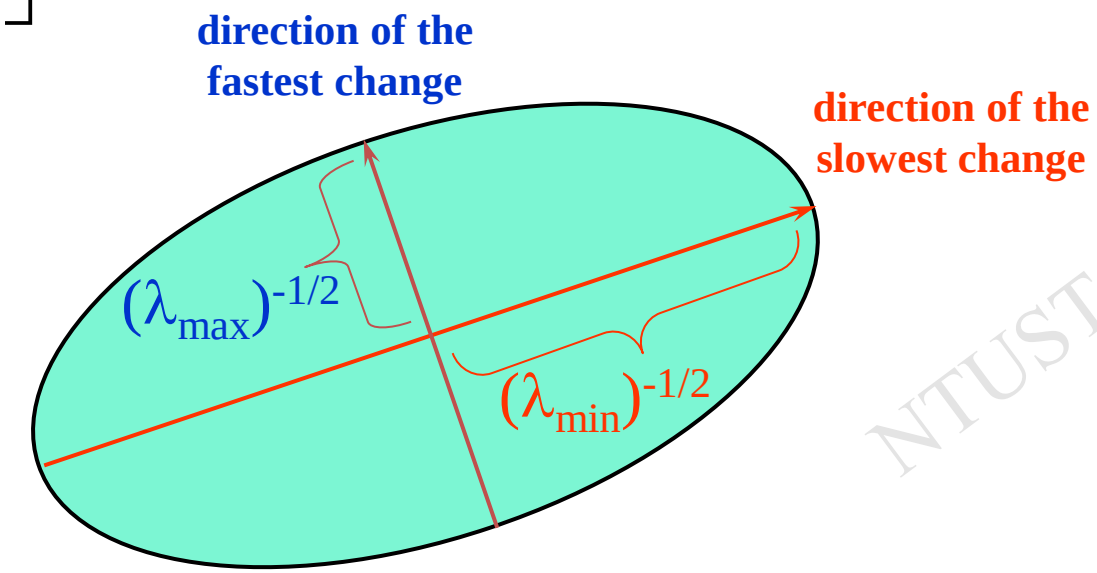
$$\mathbf{M} = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Feature detection: Harris corner detector

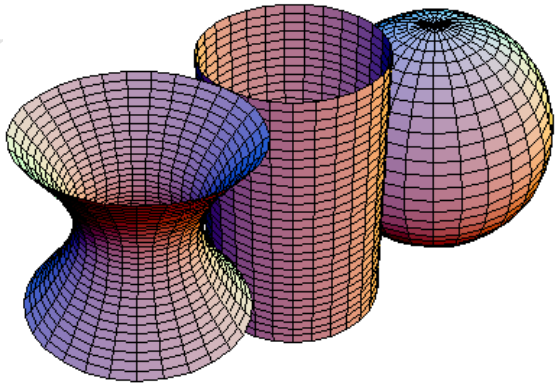
- Intensity change in shifting window: eigenvalue analysis

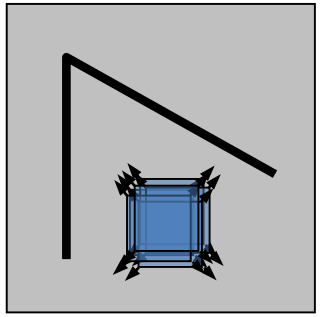
$$E(u,v) \cong \begin{bmatrix} u & v \end{bmatrix} \mathbf{M} \begin{bmatrix} u \\ v \end{bmatrix}$$

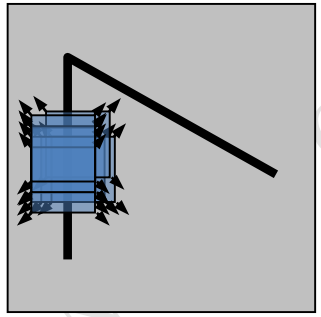
eigenvalue λ_1, λ_2

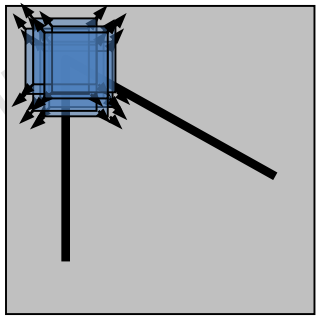


Feature detection: Harris corner detector







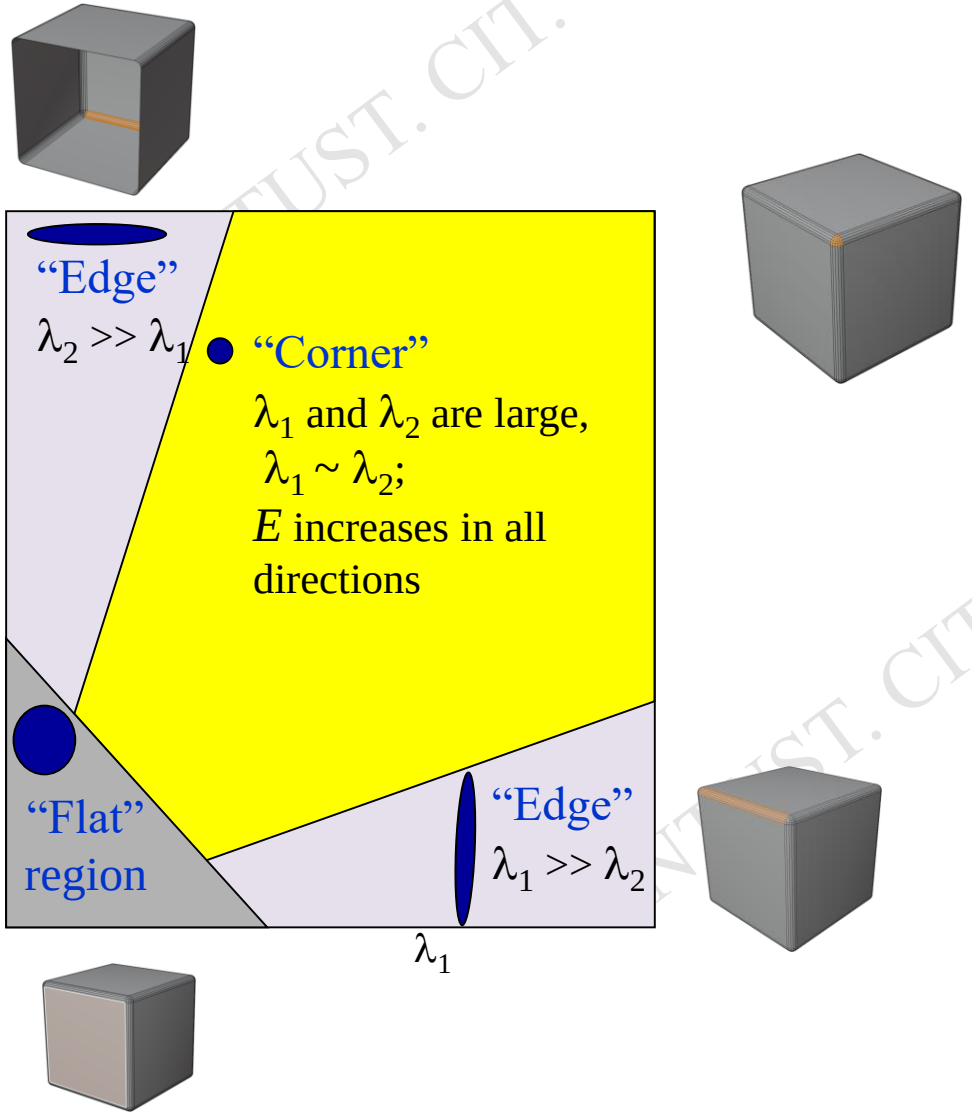


λ_2

"flat" region: no change in all directions

"edge": no change along the edge direction

"corner": significant change in all directions



Feature detection: Harris corner detector

- Measure of corner response:

$$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2$$

$$\det(\mathbf{M}) = \lambda_1\lambda_2$$

$$trace(\mathbf{M}) = (\lambda_1 + \lambda_2)$$

(k – empirical constant, k = 0.04~0.06)

$\det(\mathbf{M}) = \lambda_1\lambda_2$ → Gaussion

$trace(\mathbf{M}) = (\lambda_1 + \lambda_2)$ → Mean *2

Harris corners compared to other criterions

$$\mathbf{M} = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

- Harris corner→ finding max. R $R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2$
- Good features to track→ finding the smallest eigenvalue $\min(\lambda_1, \lambda_2)$
- Brown 2005→ finding max. of harmonic mean $\frac{\det(\mathbf{M})}{\mathrm{trace}(\mathbf{M})} = \frac{\lambda_1\lambda_2}{\lambda_1 + \lambda_2}$

[1] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, 1988, pp. 147-151.
[2] J. Shi and C. Tomasi, "Good features to track," *IEEE Conference on Computer Vision and Pattern Recognition*, 1994.
[3] M. Brown, R. Szeliski, and S. Winder, "Multi-image matching using multi-scale oriented patches," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, vol. 1, pp. 510-517

Corner detection: Summary

Step 1 : Filter images with Gaussian to reduce noise

Step 2 : Compute magnitude of *x* and *y* gradient at each pixel

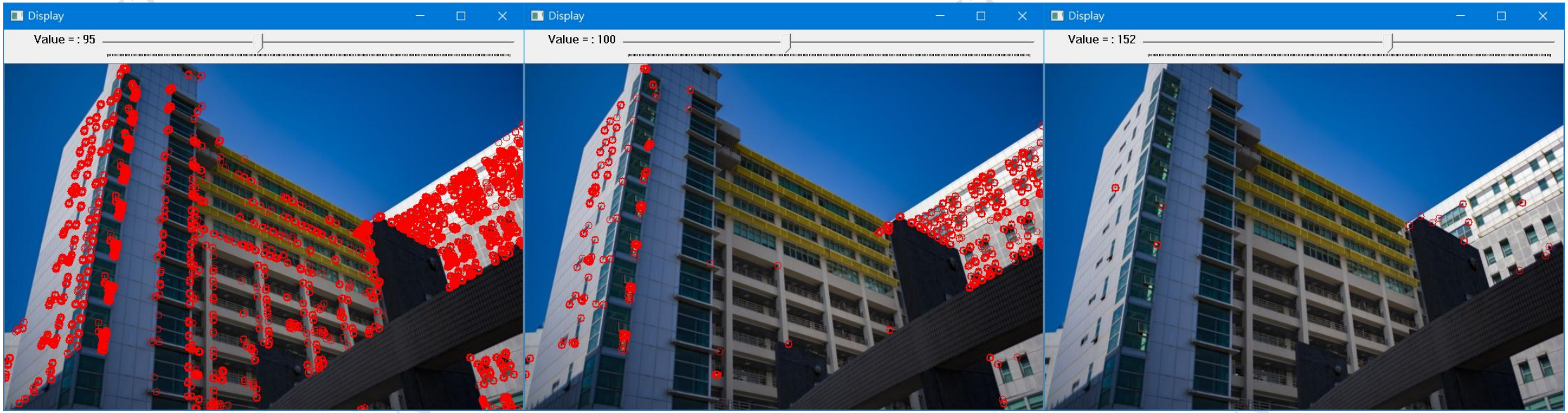
Step 3 : Construct **M** matrix and Corner response **R**

Step 4 : Preserve pixels with **R** > threshold

Step 5 : Local maximum suppression

Feature detection in openCV

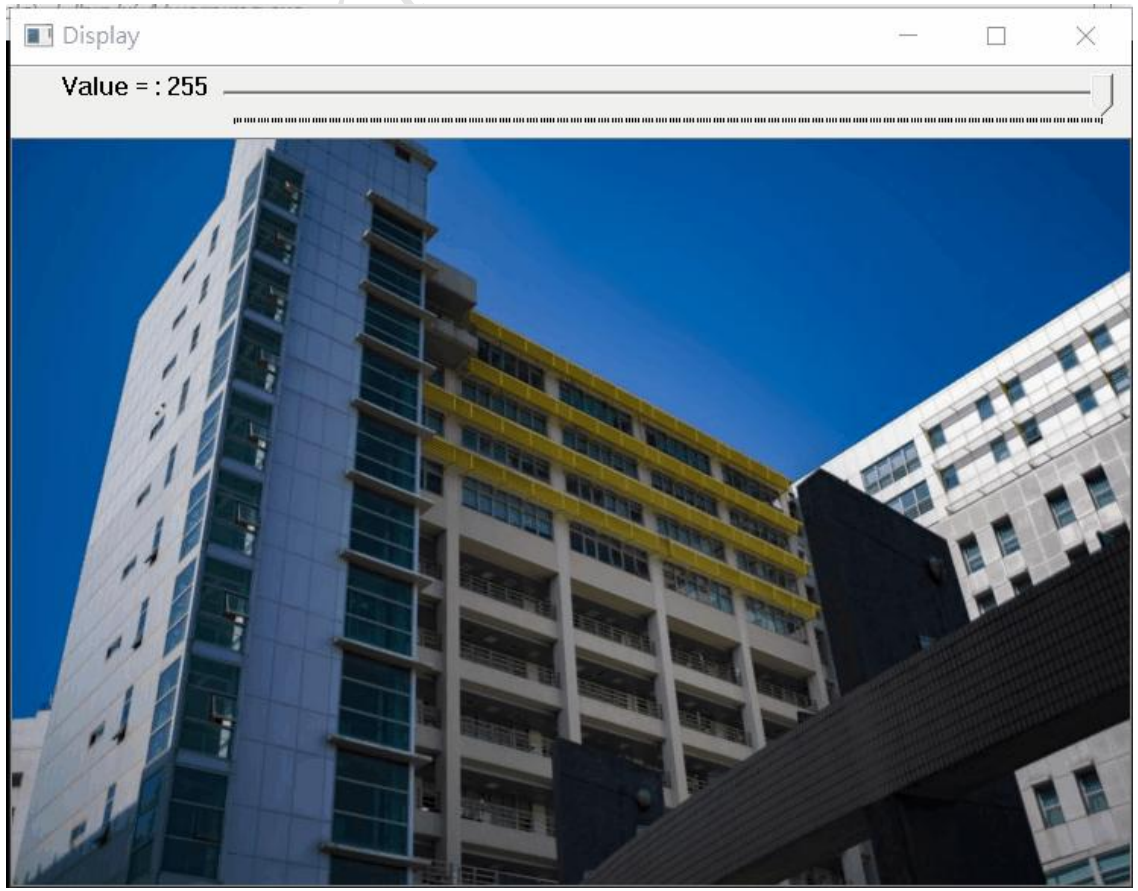
- Example: openCV simple code

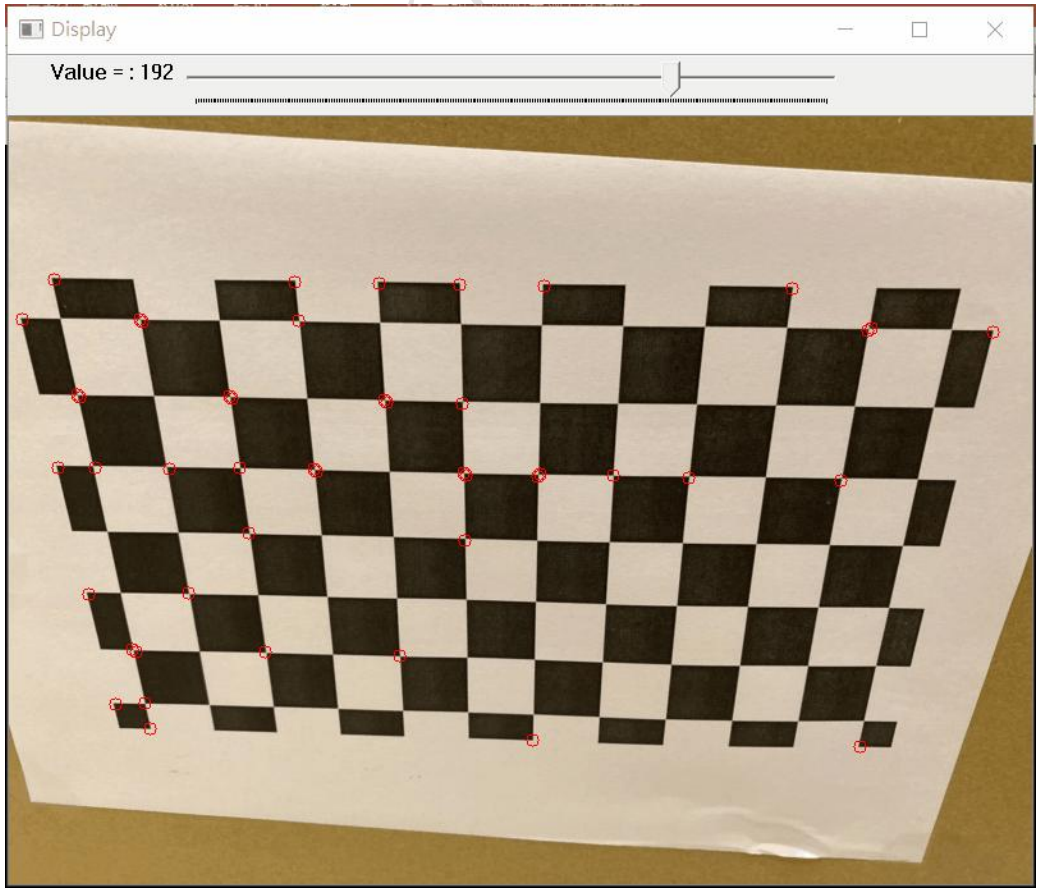


lower threshold middle threshold higher threshold

(Google “Harris corner openCV”)
http://www.opencv.org.cn/opencvdoc/2.3.1/html/doc/tutorials/features2d/trackingmotion/harris_detector/harris_detector.html

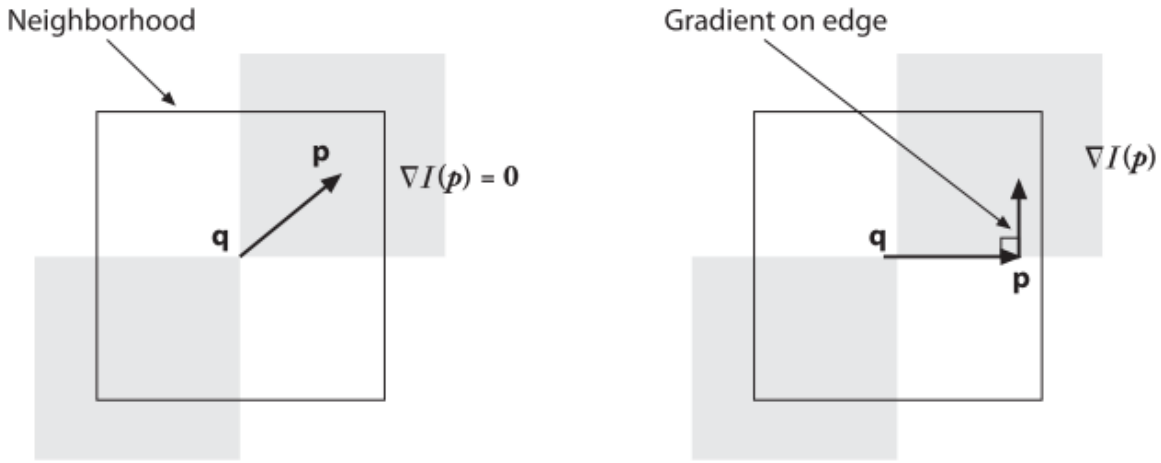
Harris corner in openCV





Feature detection in openCV

- Subpixel Corners (for determine precious 2D positions)

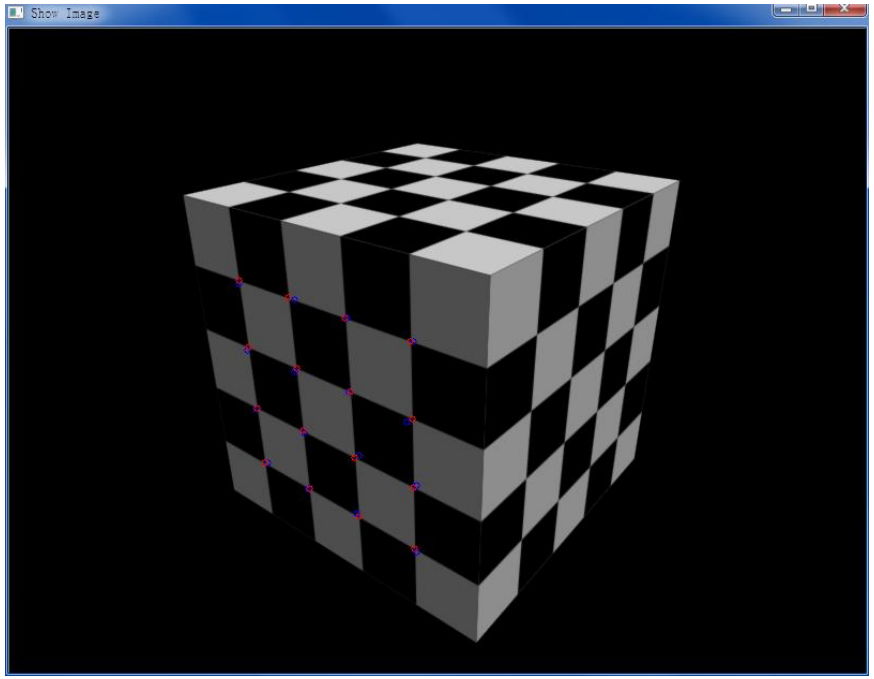


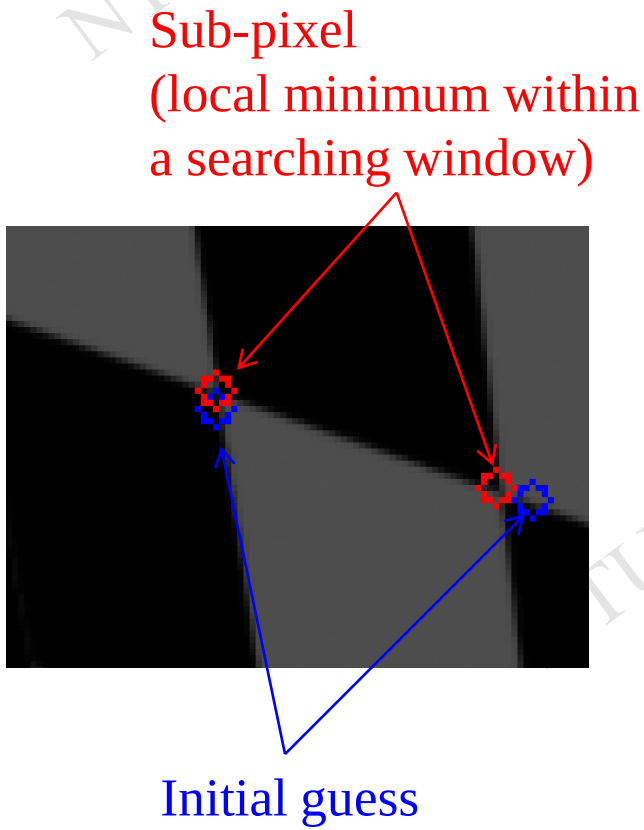

```
//OpenCV sample code
cvFindCornerSubPix(imgB, p, corner_count, cvSize(11,11),cvSize(-1,-1),
cvTermCriteria( CV_TERMCRIT_ITER | CV_TERMCRIT_EPS , 20, 0.03 ) );
```

D. Chen and G. Zhang, "A new sub-pixel detector for x-corners in camera calibration targets," in *Central Europe on Computer Graphics Visualization and Computer Vision*, 2005.

Feature detection in openCV

- Subpixel Corners (for determine precious 2D positions)

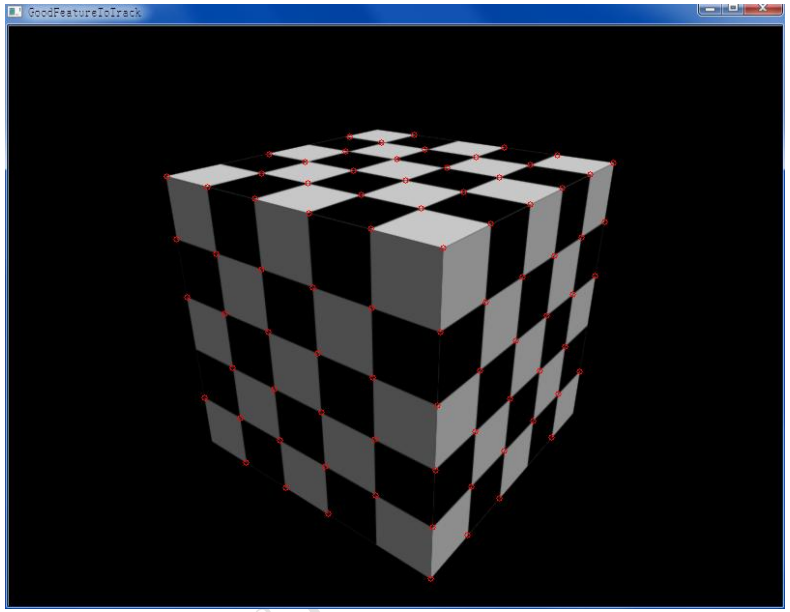




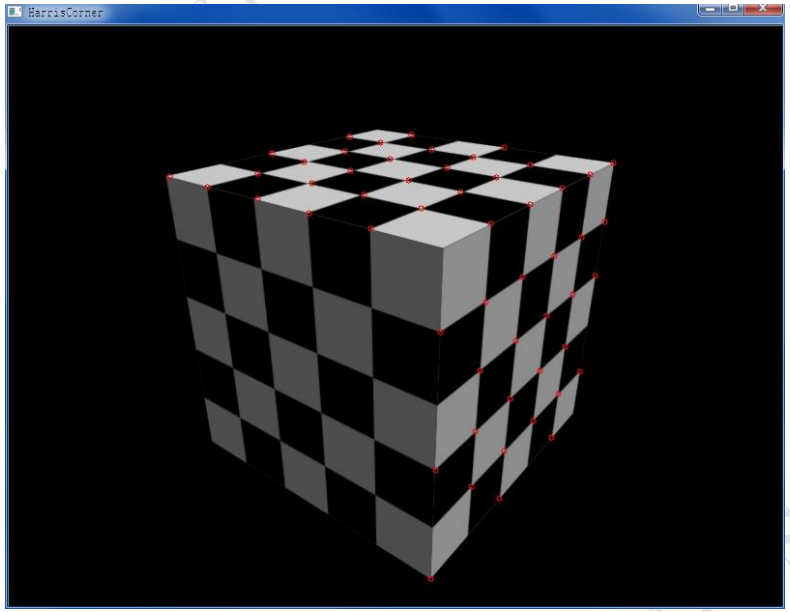
D. Chen and G. Zhang, "A new sub-pixel detector for x-corners in camera calibration targets," in *Central Europe on Computer Graphics Visualization and Computer Vision*, 2005.

Feature detection in openCV

- Subpixel Corners (coupled with Harris Corner)



cvGoodFeaturesToTrack
+cvFindCornerSubPix



cvGoodFeaturesToTrack
(Harris criterion)
+cvFindCornerSubPix

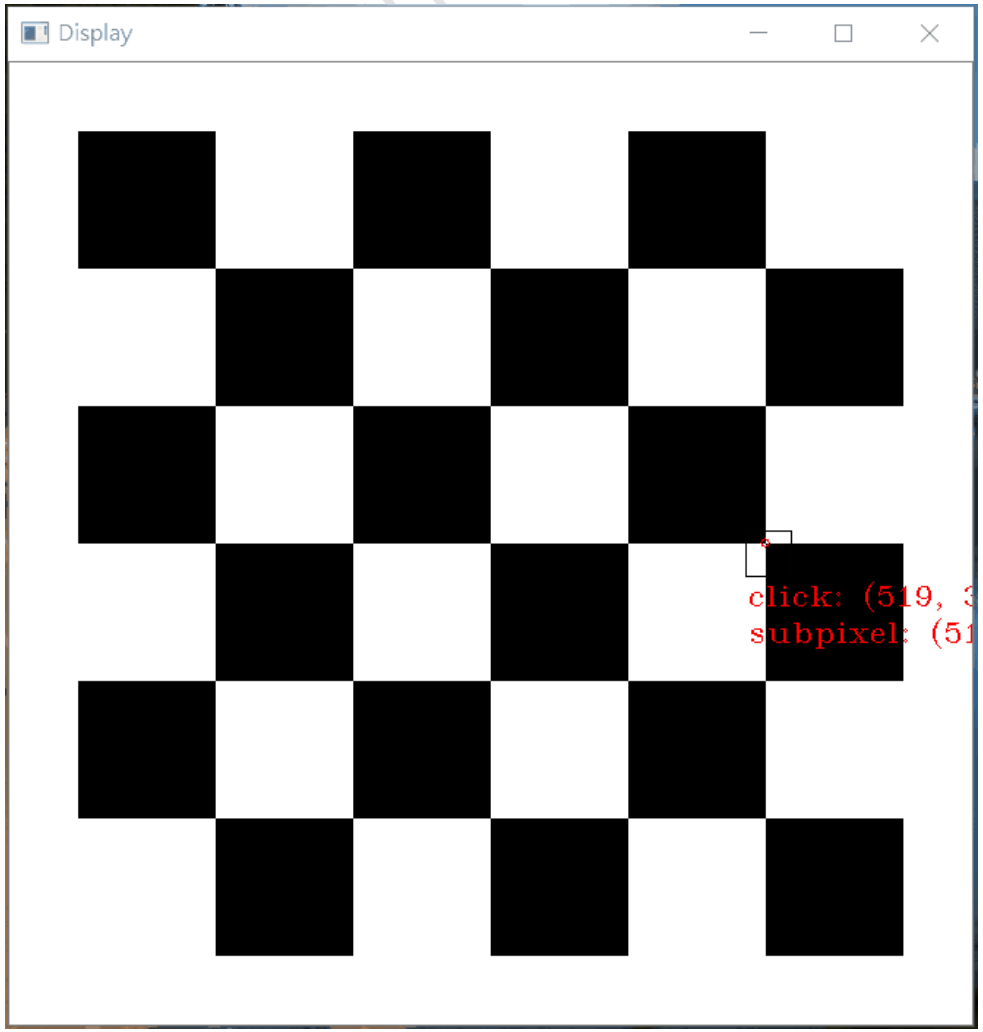
Feature detection in openCV

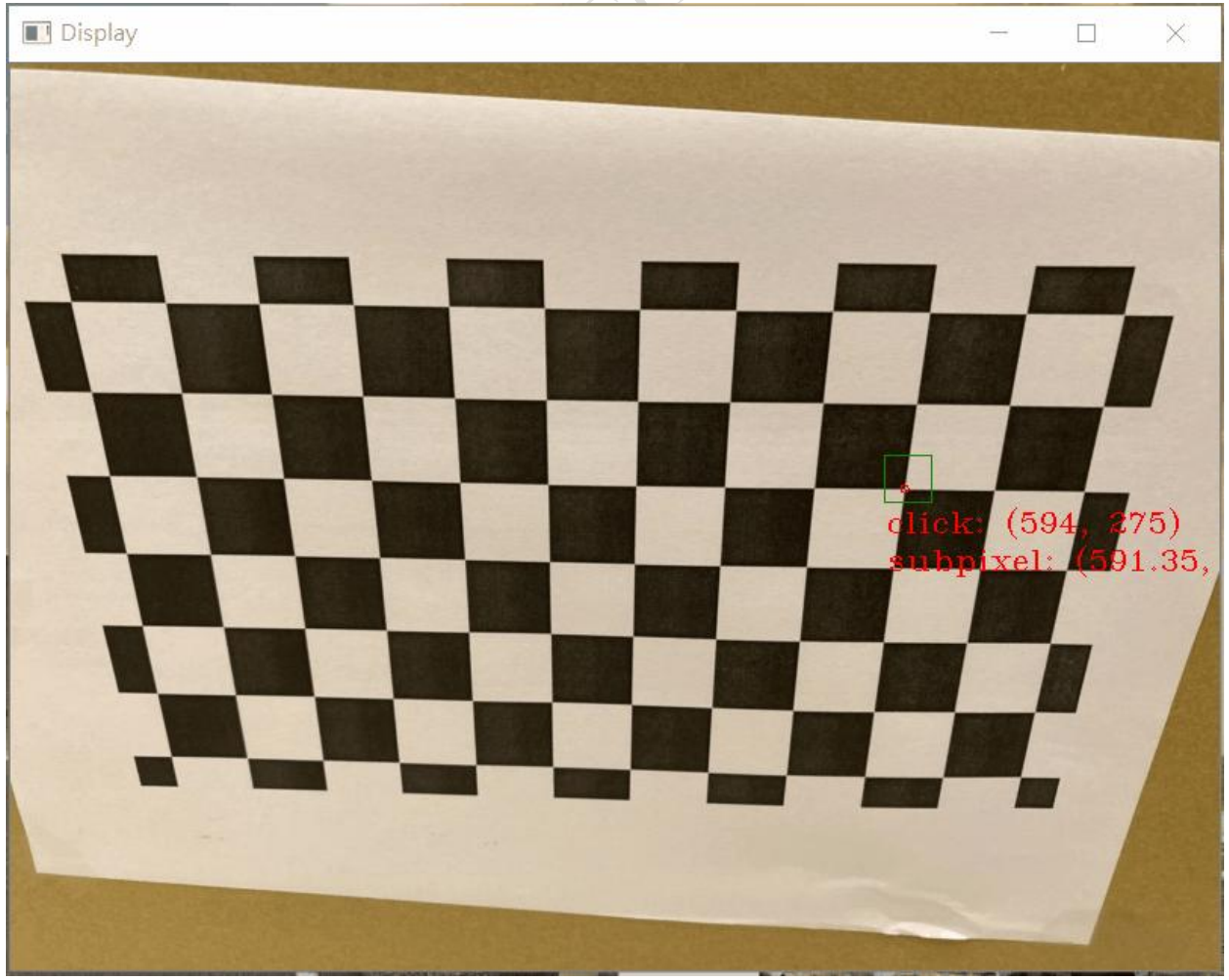
- Subpixel Corners (coupled with Harris Corner)

cvGoodFeaturesToTrack
+cvFindCornerSubPix

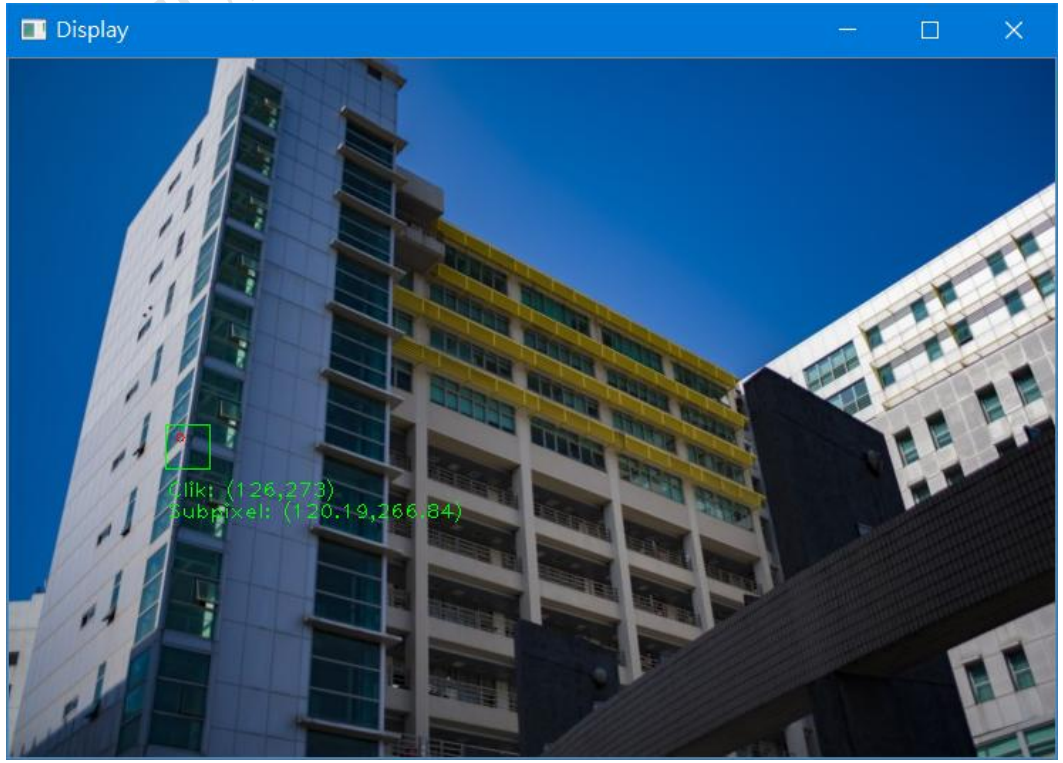
cvGoodFeaturesToTrack
(Harris criterion)
+cvFindCornerSubPix

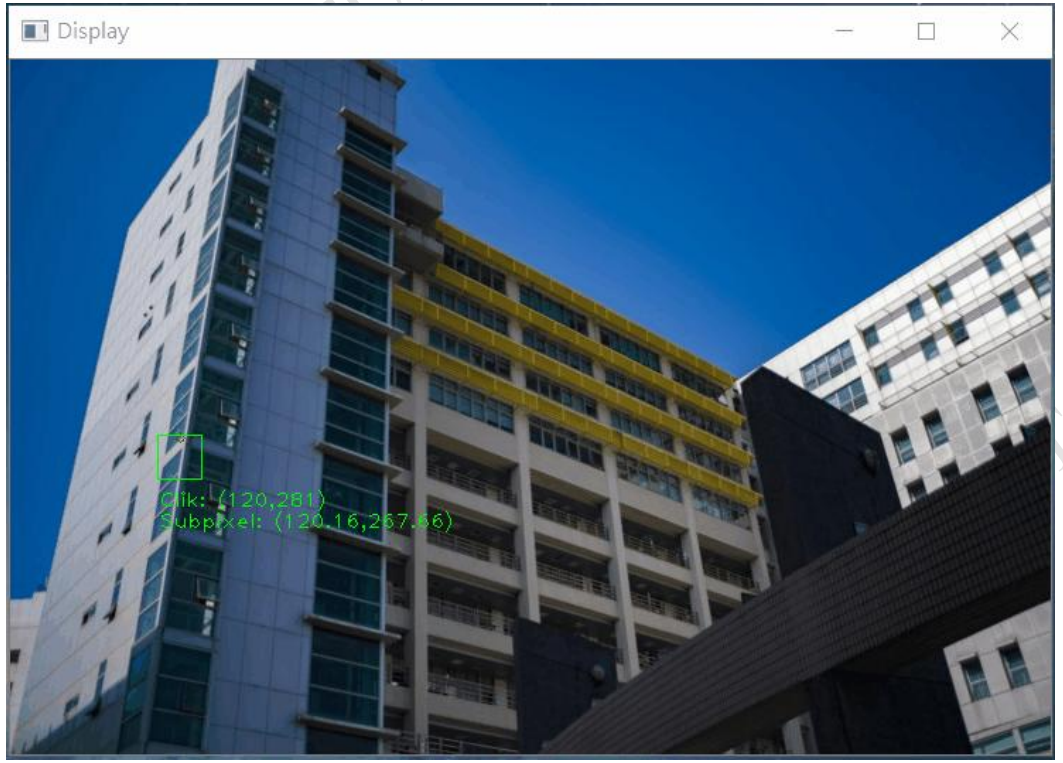
Subpixel in openCV





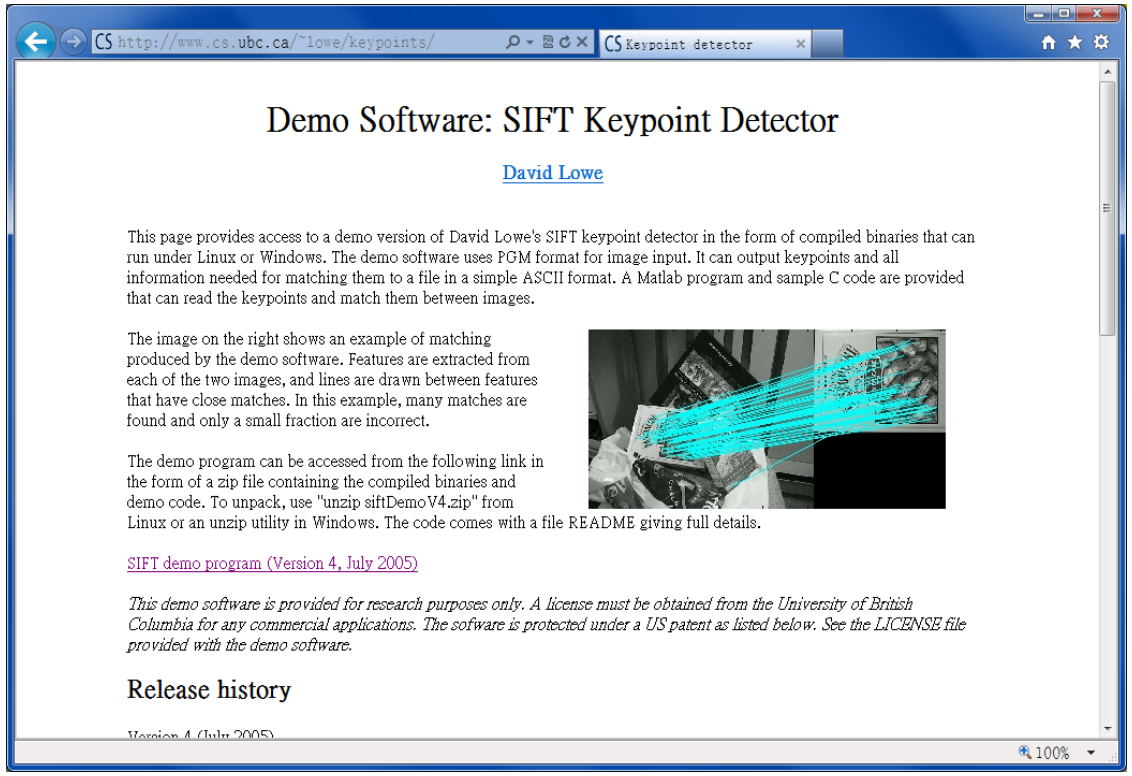
Subpixel in openCV





Feature detection

- SIFT: Scale Invariant Feature Transform
- Other type: PCA-SIFT, GLOH



Hessian Matrix

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Reject when

$$\frac{[trace(\mathbf{H})]^2}{\det(\mathbf{H})} > threshold$$

Demo code: http://www.cs.ubc.ca/~lowe/keypoints/

D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, Nov. 2004.

Summary of SIFT (note: this is a patterned algorithm)

then examined in further detail. The first stage of key-point detection is to identify locations and scales that can be repeatably assigned under differing views of the same object. Detecting locations that are invariant to scale change of the image can be accomplished by searching for stable features across all possible scales, using a continuous function of scale known as scale space (Witkin, 1983).

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D}{\partial \mathbf{x}}^T \hat{\mathbf{x}}.$$

Eliminated point
If D(x) is less than threshold

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

Input Image (Grey image) → Apply Gaussian convolution σ → Neighboring pixels comparison → Several Candidate Keypoint (extremum) → Apply Contrast threshold D → Apply Eigenvalue-Ratio (curvature) threshold r → Keypoint Histogram Approximation → SIFT Features descriptor

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y),$$
$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

Difference of Gaussian function k

$$D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y)$$
$$= L(x,y,k\sigma) - L(x,y,\sigma). \quad (1)$$

$$\sigma\nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x,y,k\sigma) - G(x,y,\sigma)}{k\sigma - \sigma}$$
$$G(x,y,k\sigma) - G(x,y,\sigma) \approx (k-1)\sigma^2\nabla^2 G.$$

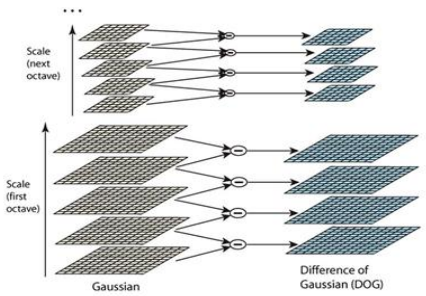
Approximated form (efficient)

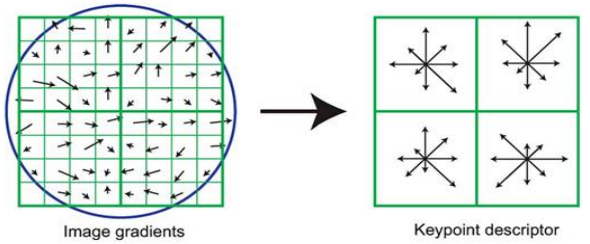
Once a keypoint candidate has been found by comparing a pixel to its neighbors, the next step is to perform a detailed fit to the nearby data for location, scale, and ratio of principal curvatures. This information allows points to be rejected that have low contrast (and are therefore sensitive to noise) or are poorly localized along an edge.

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r},$$
$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}.$$

Eliminated point (edge feature)
If ratio is great than threshold

BinWidth → Supposed to be 4x4 descript
Nobin → Supposed to be 8 (directions)





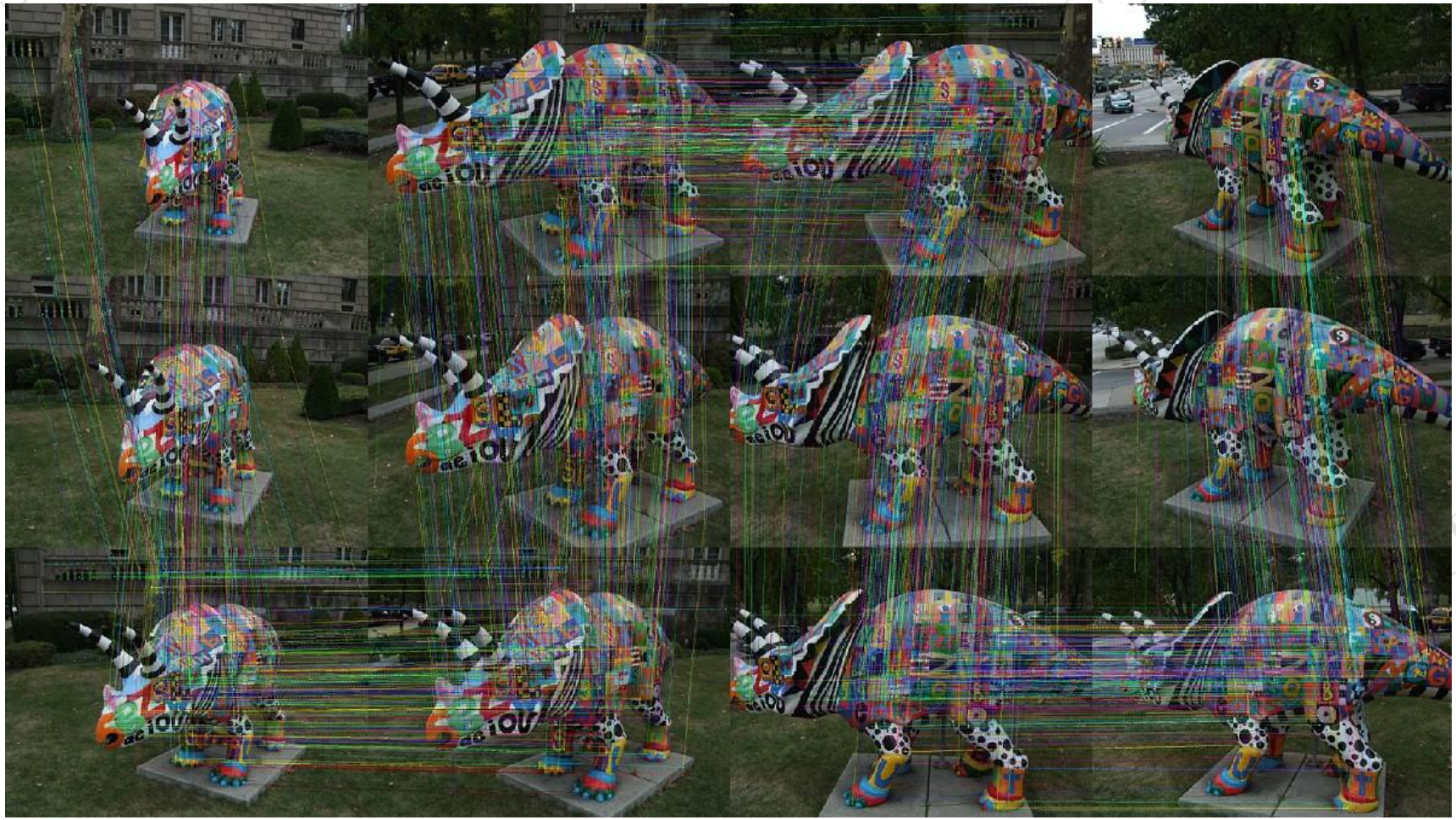
Optimized / Suggestion parameter

tave of scale space. The image doubling increases the number of stable keypoints by almost a factor of 4, but no significant further improvements were found with a larger expansion factor. *imgDBL*

D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, Nov. 2004.

Feature detection

- SIFT: example



Feature detection

- SURF: Speeded Up Robust Features (nonfree)

Hessian Matrix

$$\mathbf{H}(x,\sigma)=\begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix}$$

//Sample code in openCV

```
cv::SurfFeatureDetector surf(2500);
surf.detect(image1, keypoints1);
drawKeypoints(image1, keypoints1, outImg1, Scalar(255,255,255), DrawMatchesFlags::DRAW_RICH_KEYPOINTS);
```

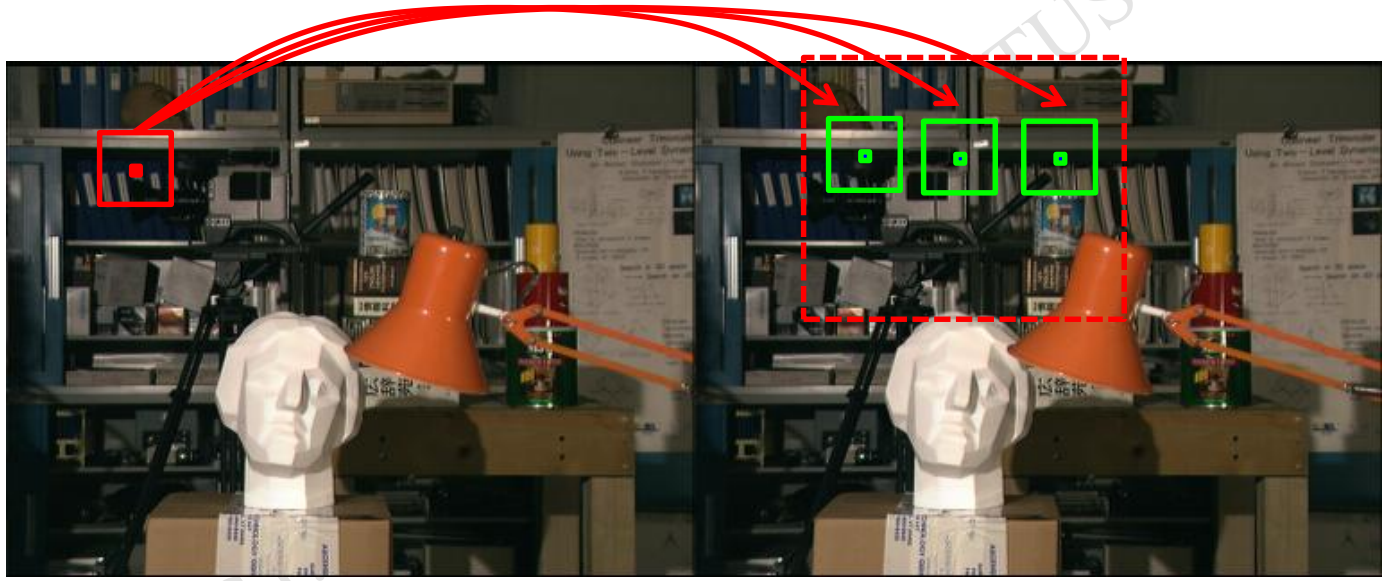
H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *European Conference on Computer Vision*, 2006, pp. 404-417.

Feature matching

- Comparison of patches (block matching)
  - brute force
  - strategy
  - transform → frequency domain FFT
  - GPU (acceleration issue)
- Comparison of feature descriptor (keypoint)

Feature matching

- Template matching by a cropped patch



For estimating the disparity of ONE pixel (in the left image)
Transfer the image into grey level, then,
1.Extend(crop) a window on the left image
2.Searching from the same pixel position, compare the difference in-between the upper / lower bound region
3.Pick out the most similar region as the matching position.

Feature matching

- Matching Criteria – Difference
  - Common matching criteria
  - SSD - Sum of Squared Differences (min. for match)

$$d(p_1, p_2) = \sum_{j=-k}^{k}\sum_{i=-k}^{k}\left(I_1(x_1+i, y_1+j) - I_2(x_2+i, y_2+j)\right)^2$$

k - the size of the window.

$p_1 = (x_1, y_1)$ is the center of the window in I_1.

$p_2 = (x_2, y_2)$ is the center of the window in I_2.

Feature matching

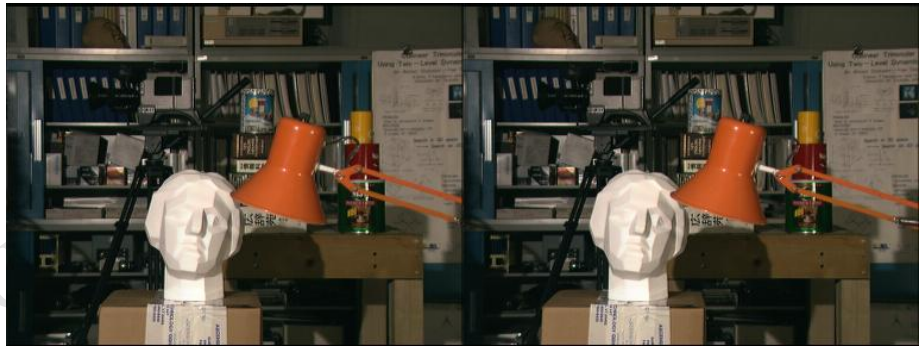
- Matching Criteria – Difference
  - Common matching criteria
  - SAD - Sum of Absolute Differences (min. for match)
  - (MAD - Minimum Absolute Difference)

$$d(p_1, p_2) = \sum_{j=-k}^{k}\sum_{i=-k}^{k}\left|I_1(x_1+i, y_1+j) - I_2(x_2+i, y_2+j)\right|$$

Feature matching

- Matching Criteria – Difference
  - Common matching criteria
  - Cross correlation (the max. value for the matched point)

$$d(p_1, p_2) = \sum_{j=-k}^{k}\sum_{i=-k}^{k} I_1(x_1+i, y_1+j)\cdot I_2(x_2+i, y_2+j)$$



- In SSD, SAD and Cross Correlation, the brightness is assumed to be similar.

Feature matching

- Matching Criteria – Difference
  - NCC (Normalized Cross Correlation)
    - When ordering the pixels in the windows in vectors v_1, v_2:

$$SSD(\vec{v}_1,\vec{v}_2)=\sum_i(\vec{v}_1(i)-\vec{v}_2(i))^2=\|\vec{v}_1-\vec{v}_2\|^2$$ ← Squared vector norm

$$Corr(\vec{v}_1,\vec{v}_2)=\sum_i\vec{v}_1(i)\cdot\vec{v}_2(i)=\langle\vec{v}_1,\vec{v}_2\rangle$$ ← Inner product

    - Normalized Cross Correlation is:

$$NCC(\vec{v}_1,\vec{v}_2)=\frac{\sum_i\vec{v}_1(i)\cdot\vec{v}_2(i)}{\sqrt{\sum_i\vec{v}_1(i)\cdot\vec{v}_1(i)}\sqrt{\sum_i\vec{v}_2(i)\cdot\vec{v}_2(i)}}=\frac{\langle\vec{v}_1,\vec{v}_2\rangle}{\|\vec{v}_1\|\|\vec{v}_2\|}$$

Feature matching

- Limitations of Matching
  - Accuracy:
    - A pixel is always matched to integer location on the grid. The image motion is usually not integer.
  - Neighborhood/Scene constraints:
    - High level knowledge about the scene/camera may help in limiting the search, and reducing errors.

Middlebury stereo benchmarks (stereo matching)

vision.middlebury.edu/stereo/

vision.middlebury.edu

stereo • mview • MRF • flow • color

Stereo Evaluation • Datasets • Code • Submit

Daniel Scharstein • Richard Szeliski • Heiko Hirschmüller

Welcome to the Middlebury Stereo Vision Page. This website accompanies our taxonomy and comparison of two-frame stereo correspondence algorithms [1], extending our initial paper with Ramin Zabih [2]. It contains:

- An on-line evaluation of current algorithms
- Many stereo datasets with ground-truth disparities
- Our stereo correspondence software
- An on-line submission script that allows you to evaluate your stereo algorithm in our framework

How to cite the materials on this website:
We grant permission to use and publish all images and numerical results on this website. If you report performance results, we request that you cite our paper [1]. Instructions on how to cite our datasets are listed on the datasets page. If you want to cite this website, please use the URL "**vision.middlebury.edu/stereo/**".

References:

[1] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1/2/3):7-42, April-June 2002.
Microsoft Research Technical Report MSR-TR-2001-81, November 2001.

[2] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In Workshop on Stereo and Multi-Baseline Vision (in conjunction with IEEE CVPR 2001), pages 131-140, Kauai, Hawaii, December 2001.

Other online stereo benchmarks:

- Robust Vision Challenge
- KITTI Stereo 2012 evaluation
- KITTI Stereo 2015 evaluation
- ETH3D 2-view stereo benchmark
- Heidelberg HD1K Stereo benchmark

Support for this work was provided in part by NSF CAREER grant 9984485 and NSF grant IIS-0413169. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Stereo Evaluation • Datasets • Code • Submit

Middlebury Stereo Evaluation - Version 3

Mouseover the table cells to see the produced disparity map. Clicking a cell will blink the ground truth for comparison. To change the table type, click the links below. For more information, please see the description of new features.

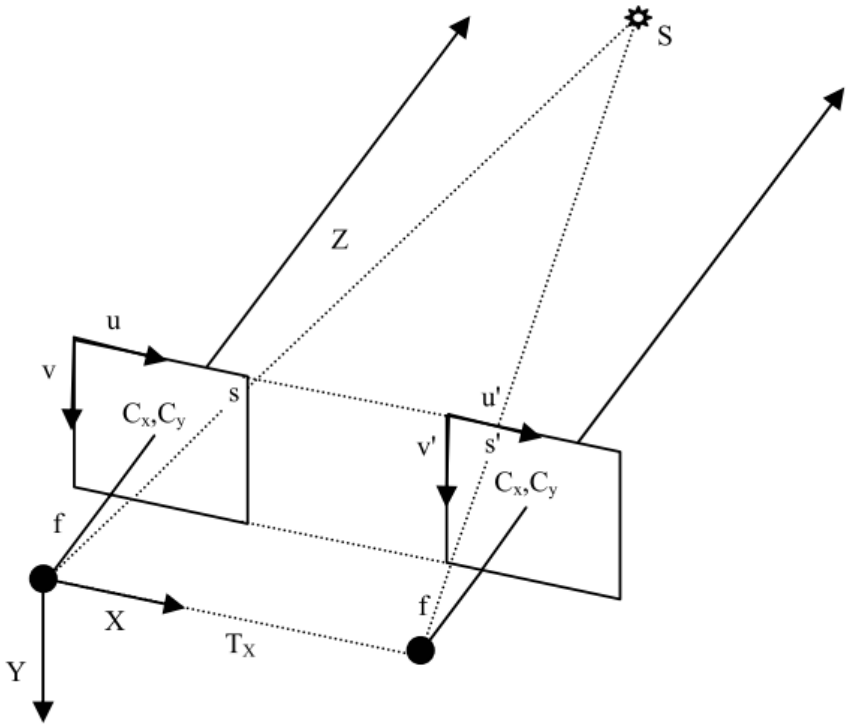
Submit and evaluate your own results. See snapshots of previous results. See the evaluation v.2 (no longer active).

Set: test dense test sparse training dense training sparse
Metric: bad 0.5 bad 1.0 bad 2.0 bad 4.0 avgerr rms A50 A90 A95 A99 time time/MP time/GD
Mask: nonocc all
plot selected show invalid Reset sort Reference list

Date	Name	Res	Weight Avg	Austr MP: 5.6 nd: 290 im0 im1 GT nonocc	AustrP MP: 5.6 nd: 290 im0 im1 GT nonocc	Bicyc2 MP: 5.6 nd: 250 im0 im1 GT nonocc	Class MP: 5.7 nd: 610 im0 im1 GT nonocc	ClassE MP: 5.7 nd: 610 im0 im1 GT nonocc	Compu MP: 1.5 nd: 256 im0 im1 GT nonocc	Crusa MP: 5.5 nd: 800 im0 im1 GT nonocc	CrusaP MP: 5.5 nd: 800 im0 im1 GT nonocc	Djemb MP: 5.7 nd: 320 im0 im1 GT nonocc	DjembL MP: 5.7 nd: 320 im0 im1 GT nonocc	Hoops MP: 5.7 nd: 410 im0 im1 GT nonocc	Livgrm MP: 5.9 nd: 320 im0 im1 GT nonocc	Nkuba MP: 5.5 nd: 570 im0 im1 GT nonocc	Plants MP: 5.6 nd: 320 im0 im1 GT nonocc	Stairs MP: 5.2 nd: 450 im0 im1 GT nonocc
11/10/21	CREStereo	F	**3.71** 1	4.73 21	3.94 21	5.07 22	1.96 4	**3.02** 1	**1.42** 1	**2.28** 1	**2.05** 1	1.51 2	6.86 2	**6.35** 1	**4.25** 1	**6.01** 1	4.60 2	**5.49** 1
07/26/21	RAFT-Stereo	F	4.74 2	4.19 17	3.44 15	3.11 4	**1.51** 1	7.30 15	2.79 2	2.67 2	2.59 2	**1.39** 1	7.46 3	10.2 14	5.86 2	13.0 11	**3.59** 1	9.38 14
05/26/18	NOSS_ROB	H	5.01 3	3.57 4	2.84 3	3.99 10	1.93 3	5.15 3	3.34 4	3.32 7	3.15 5	2.32 10	8.55 5	7.45 3	7.06 8	12.5 8	5.20 9	9.06 9
07/23/21	HBP_ISP	H	5.20 4	3.70 9	3.05 10	3.57 6	2.34 10	7.80 18	3.79 10	3.34 8	3.09 4	1.87 4	9.85 9	10.1 13	7.82 13	11.2 4	5.26 11	7.86 4
06/22/17	LocalExp	H	5.43 5	3.65 6	2.87 5	2.98 2	1.99 6	5.59 6	3.37 5	3.48 11	3.35 8	2.05 6	10.3 14	9.75 9	8.57 16	14.4 27	5.40 13	9.55 17
03/09/19	3DMST-CM	H	5.47 6	4.10 16	3.37 14	2.99 3	2.95 20	7.63 17	4.55 14	3.26 5	3.95 15	2.16 7	10.2 11	8.28 4	6.37 4	13.2 13	5.86 19	9.35 13
06/23/21	ERW-LocalExp	H	5.53 7	3.64 5	2.84 3	**2.66** 1	1.97 5	5.68 7	4.87 15	3.27 6	3.25 7	2.36 13	10.5 16	11.5 19	7.46 11	14.7 30	5.55 17	9.18 10
03/05/21	LocalExp-RC	H	5.54 8	3.78 13	3.02 9	3.85 7	2.08 7	5.95 8	3.48 9	3.61 13	3.65 10	2.52 18	10.3 15	6.85 2	7.25 9	16.1 45	5.12 7	10.2 21
08/30/20	LE_PC	H	5.58 9	3.52 3	2.99 8	4.24 13	1.92 2	5.39 5	3.42 8	3.16 4	3.72 12	2.30 9	7.83 4	9.90 11	7.79 12	17.4 55	4.74 5	9.51 15
07/16/20	HLocalExp-CM	H	5.68 10	3.68 8	2.95 7	3.92 8	2.45 11	8.12 20	3.41 7	3.74 14	3.53 9	2.17 8	10.2 11	10.0 12	8.75 17	14.1 25	5.12 7	9.61 18
12/19/19	CRLE	H	5.75 11	3.66 7	3.11 11	5.92 31	2.14 8	6.01 9	3.39 6	3.49 12	3.68 11	2.34 11	10.2 11	9.63 8	8.04 14	14.9 33	5.45 15	9.26 12
01/24/17	3DMST	H	5.92 12	3.71 10	2.78 2	4.75 17	2.72 14	7.36 16	4.28 12	3.44 9	3.76 13	2.35 12	12.6 25	11.5 18	8.56 15	14.0 24	5.35 12	8.87 7
03/10/17	MC-CNN+TDSR	F	6.35 13	5.45 27	4.45 34	6.80 43	3.46 27	10.7 32	6.05 29	5.01 22	5.19 24	2.62 20	10.8 18	9.62 7	6.59 6	11.4 5	6.01 21	7.04 2
10/28/20	HITNet	H	6.46 14	6.25 39	4.67 41	4.51 14	2.17 9	6.52 12	5.18 16	2.92 3	2.66 3	2.37 14	36.7 89	9.28 5	6.27 3	11.2 3	4.61 3	9.54 16
11/21/21	Gwc_CoAtRS	H	6.50 15	6.92 42	6.82 74	4.55 15	3.48 28	5.12 2	5.80 26	4.88 20	4.96 20	2.69 24	15.3 36	12.8 21	6.40 5	10.2 2	7.13 31	8.48 6
05/12/16	PMSC	H	6.71 16	**3.46** 1	**2.68** 1	6.19 36	2.54 12	6.92 13	4.54 13	3.96 15	4.04 17	2.37 14	13.1 27	12.3 20	12.2 29	16.2 46	5.88 20	10.8 25
04/22/21	LESC	H	6.78 17	4.07 14	3.46 16	3.26 5	3.36 25	9.15 25	4.08 11	4.76 18	5.21 25	2.80 27	11.7 22	13.0 22	10.2 21	17.0 49	5.52 16	12.5 38
10/19/16	LW-CNN	H	7.04 18	4.65 18	3.95 22	5.30 24	2.63 13	11.2 37	5.41 20	4.32 17	4.22 18	2.43 17	12.2 24	13.4 25	13.6 40	14.8 31	4.72 4	12.0 36
05/28/20	LEAStereo	H	7.15 19	7.56 46	4.52 35	4.62 16	4.64 34	8.83 22	5.66 22	5.86 27	6.03 33	3.30 44	13.1 28	11.3 17	10.3 22	12.1 7	7.06 30	9.90 19
10/12/17	FEN-D2DRR	H	7.23 20	4.68 19	4.11 25	5.03 21	3.03 21	8.42 21	6.05 29	4.90 21	5.32 27	3.20 38	11.5 21	14.1 27	13.4 38	13.9 22	5.06 6	14.3 45
05/28/16	APAP-Stereo	H	7.26 21	5.43 26	4.01 46	5.11 33	5.17 38	21.6 64	6.99 36	4.31 16	4.23 19	3.24 40	14.3 33	9.78 10	7.32 10	13.4 15	6.30 22	8.46 5

Stereoscopic image to 3D data

- Reprojecting of disparity into 3D space
  - Reprojection matrix in homogenous coordinate



$$\mathbf{Q} \equiv \begin{bmatrix} 1 & 0 & 0 & -C_x \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{-1}{T_x} & \frac{C_x - C'_x}{T_x} \end{bmatrix}$$

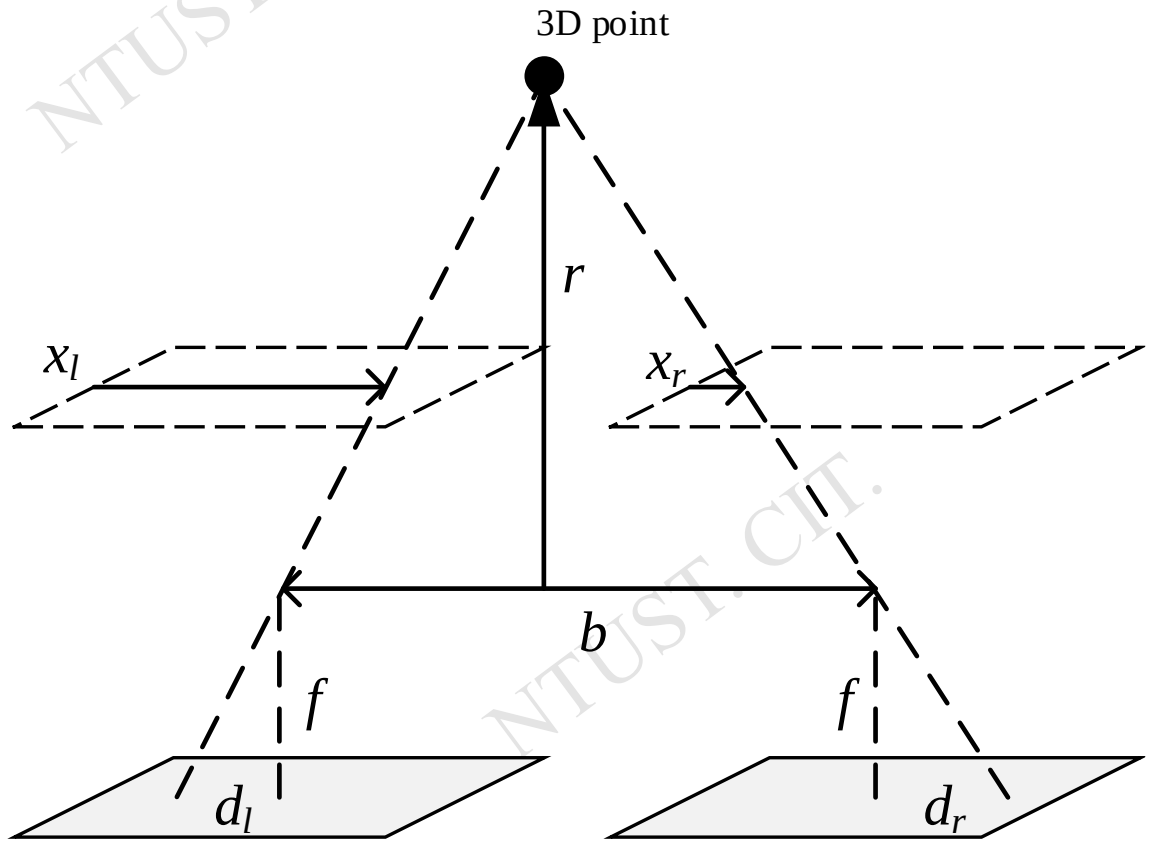
$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \mathbf{Q} \begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix}$$

The actual 3D point is $(X/W, Y/W, Z/W)$.

Stereoscopic image to 3D data (How to improve)

- Smooth depth image (to suppress the noise)
- Get into subpixel level (time consuming)

Stereoscopic image performance estimation

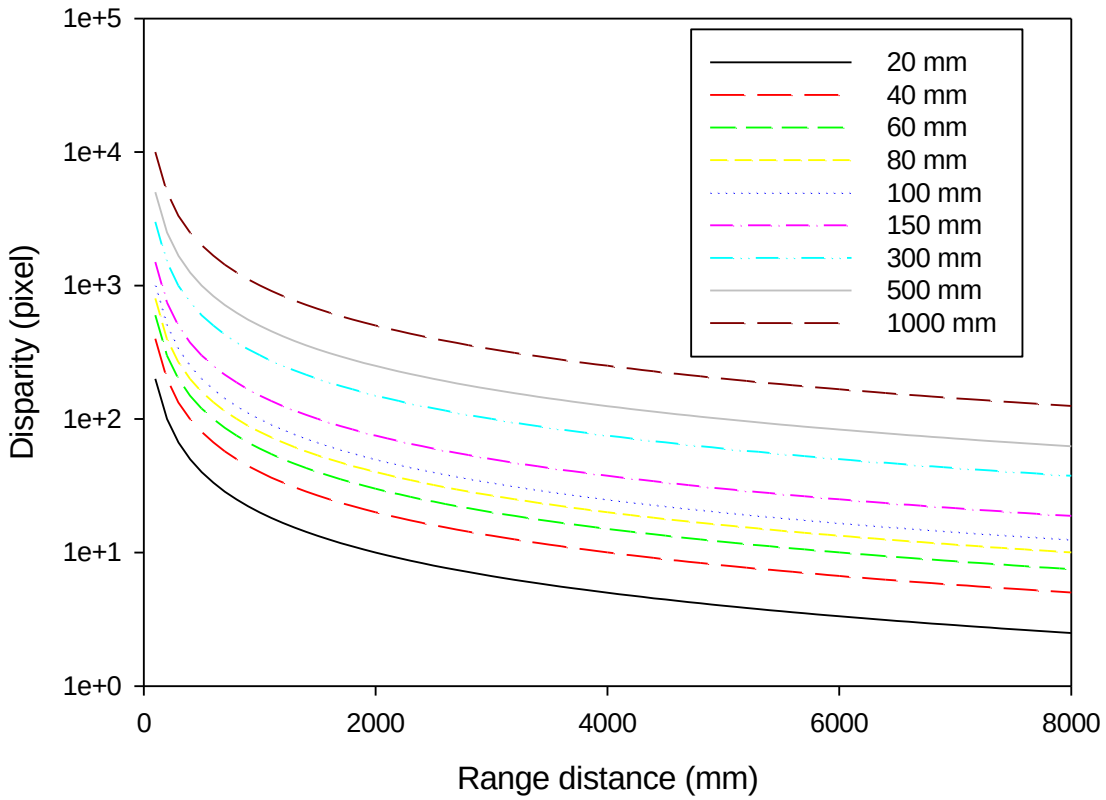


$$\frac{r}{b} = \frac{f}{d_l + d_r}$$

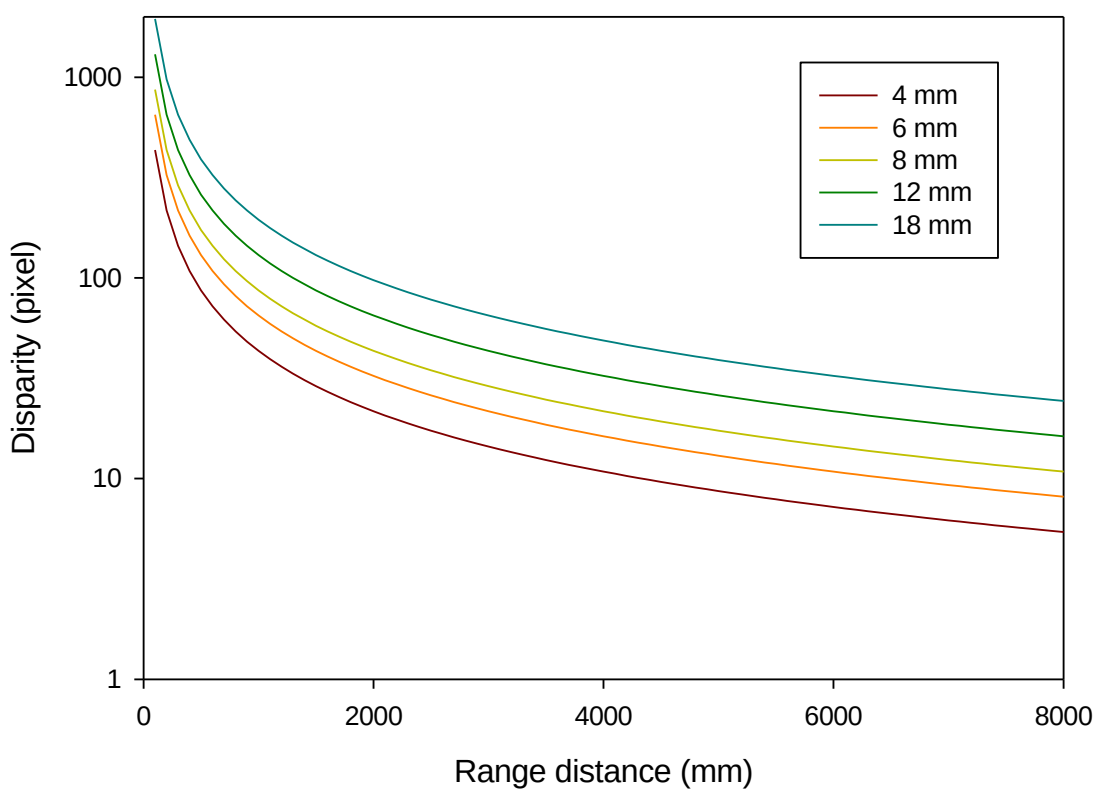
$$r = \frac{fb}{d_l + d_r} = \frac{fb}{x_l - x_r} = \frac{fb}{d}$$

Stereoscopic image performance estimation

Different baseline
(under Focal length = 6 mm)



Different Focal length
(under baseline = 65 mm)



for pixel size = 6.0 µm CMOS

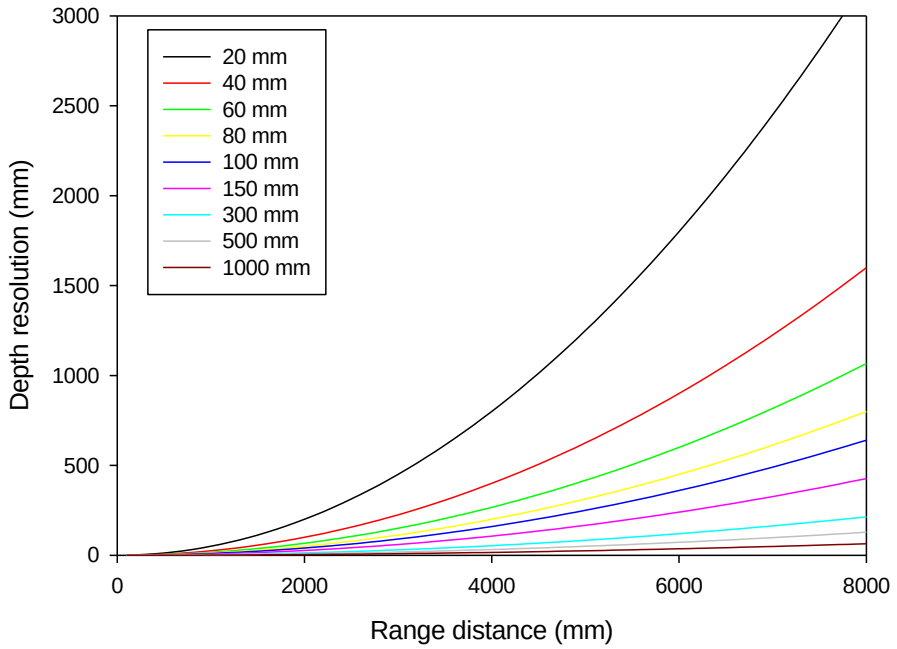
Stereoscopic image performance estimation

$$\Delta d = \frac{fb}{r+\Delta r} - \frac{fb}{r}$$

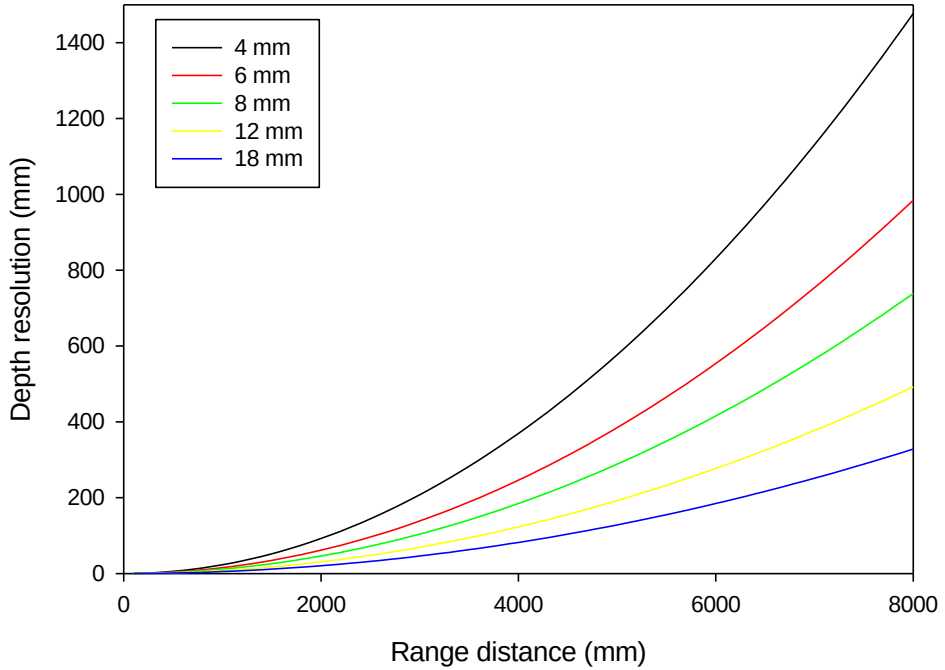
$$\frac{\Delta d}{fb} = \frac{1}{r+\Delta r} - \frac{1}{r} = \frac{-\Delta r}{(r+\Delta r)r}$$

$$\Delta r \cong -\frac{r^2}{fb}\Delta d$$

Different baseline
(under Focal length = 6 mm)



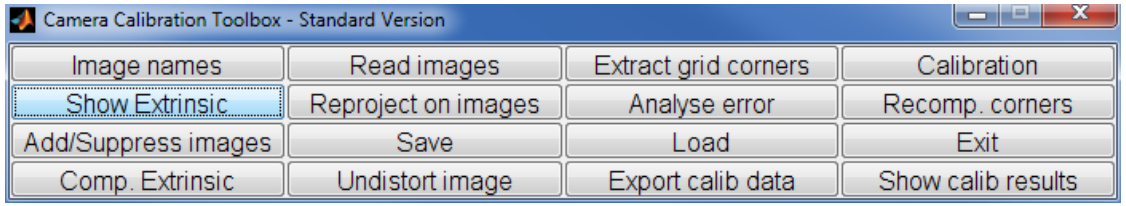
Different Focal length
(under baseline = 65 mm)

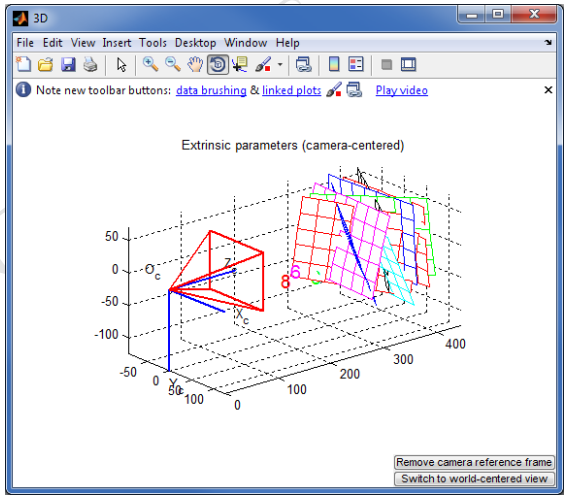


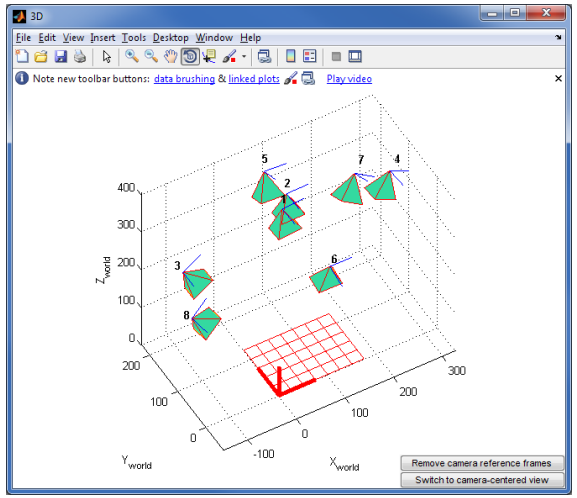
for pixel size = 6.0 μm CMOS

Camera calibration

- Calibration toolbox for Matlab
  - To obtain intrinsic and extrinsic parameter
  - To obtain lens distortion compensation



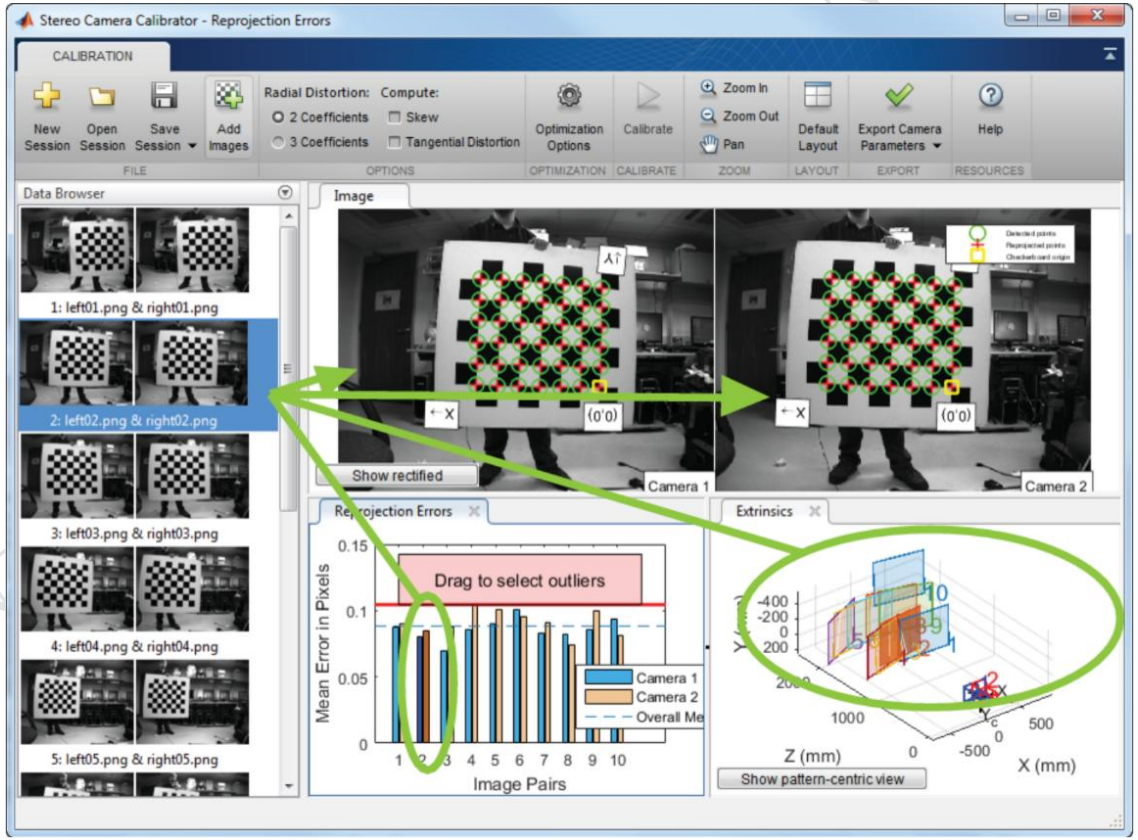




http://www.vision.caltech.edu/bouguetj/calib_doc/

Stereo camera calibration (by Matlab)

- Build-in toolbox
  - stereoCameraCalibrator



https://www.mathworks.com/help/vision/ug/stereo-camera-calibrator-app.html

Feature matching

- Block Matching in openCV

openCV function: cvMatchTemplate

```
void cvMatchTemplate(const CvArr* image,
                     const CvArr* templ,
                     CvArr* result,
                     int method );
```

Online resource
http://dasl.mem.drexel.edu/~noahKuntz/openCVTut6.html

Feature matching

- Block Matching in openCV

method=CV_TM_SQDIFF

→ sum of square difference

method=CV_TM_SQDIFF NORMED

→ normalized sum of square difference

method=CV_TM_CCORR

→ cross correlation

method=CV_TM_CCORR_NORMED

→ normalized cross correlation

method=CV_TM_CCOEFF

→ correlation coefficient

method=CV_TM_CCOEFF_NORMED

→ fast normalized correlation coefficient (FFT)

http://www.idiom.com/~zilla/Papers/nvisionInterface/nip.html#tex2html1

Feature matching

- Stereo-Block matching example

Video source from:
http://www.youtube.com/watch?v=FX7AMktf24E

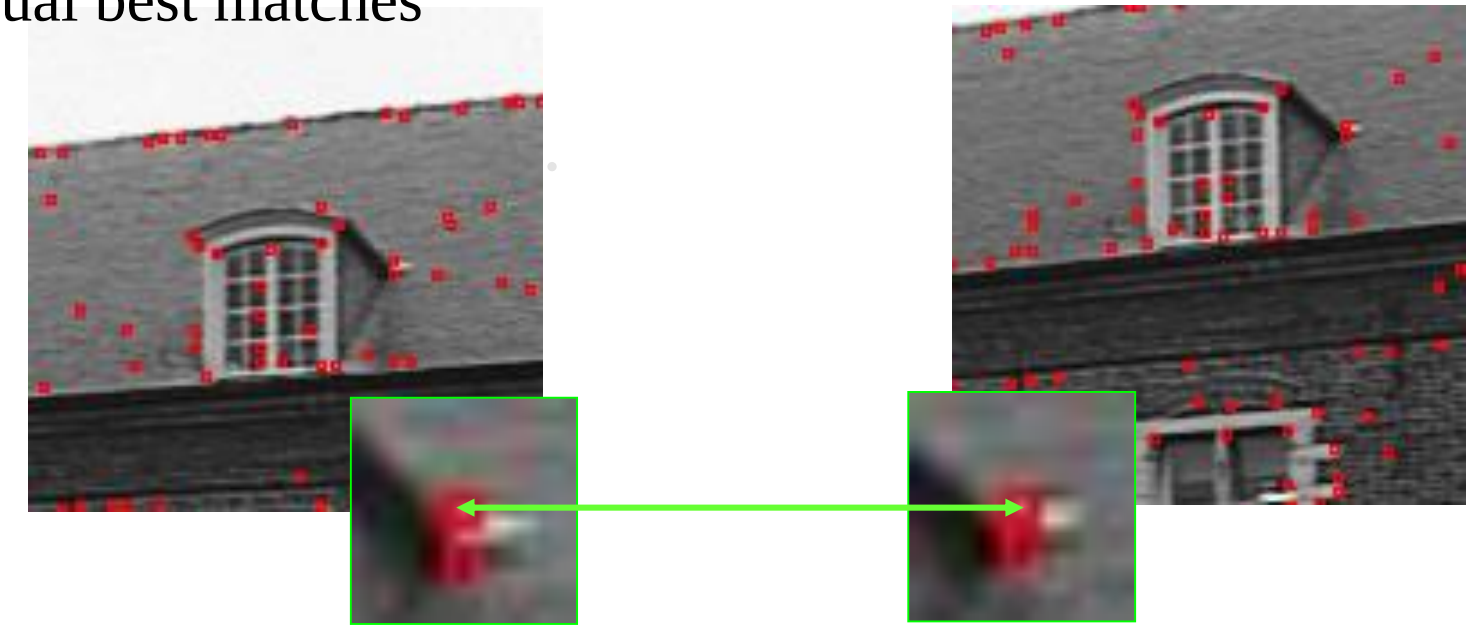
Feature matching

- Matching for known features

  for each corner in image 1 find the corner in image 2 that is most similar (using SSD or NCC) and vice-versa

  Only compare geometrically compatible points

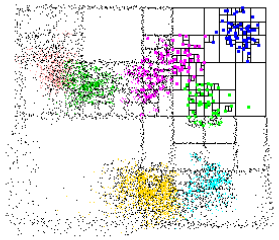
  Keep mutual best matches



Feature matching (wider region)

- Fast Searching Issue
  - ANN or FLANN (Fast Approximate Nearest Neighbor Search), KD-Tree

ANN: A Library for Approximate Nearest Neighbor Searching
David M. Mount and Sunil Arya
Version 1.1.2
Release Date: Jan 27, 2010



What is ANN?

ANN is a library written in C++, which supports data structures and algorithms for both exact and approximate nearest neighbor searching in arbitrarily high dimensions.

In the nearest neighbor problem a set of data points in d-dimensional space is given. These points are preprocessed into a data structure, so that given any query point q, the nearest or generally k nearest points of P to q can be reported efficiently. The distance between two points can be defined in many ways. ANN assumes that distances are measured using any class of distance functions called Minkowski metrics. These include the well known Euclidean distance, Manhattan distance, and max distance.

Based on our own experience, ANN performs quite efficiently for point sets ranging in size from thousands to hundreds of thousands, and in dimensions as high as 20. (For applications in significantly higher dimensions, the results are rather spotty, but you might try it anyway.)

The library implements a number of different data structures, based on kd-trees and box-decomposition trees, and employs a couple of different search strategies.

The library also comes with test programs for measuring the quality of performance of ANN on any particular data sets, as well as programs for visualizing the structure of the geometric data structures.

ANN: http://www.cs.umd.edu/~mount/ANN/

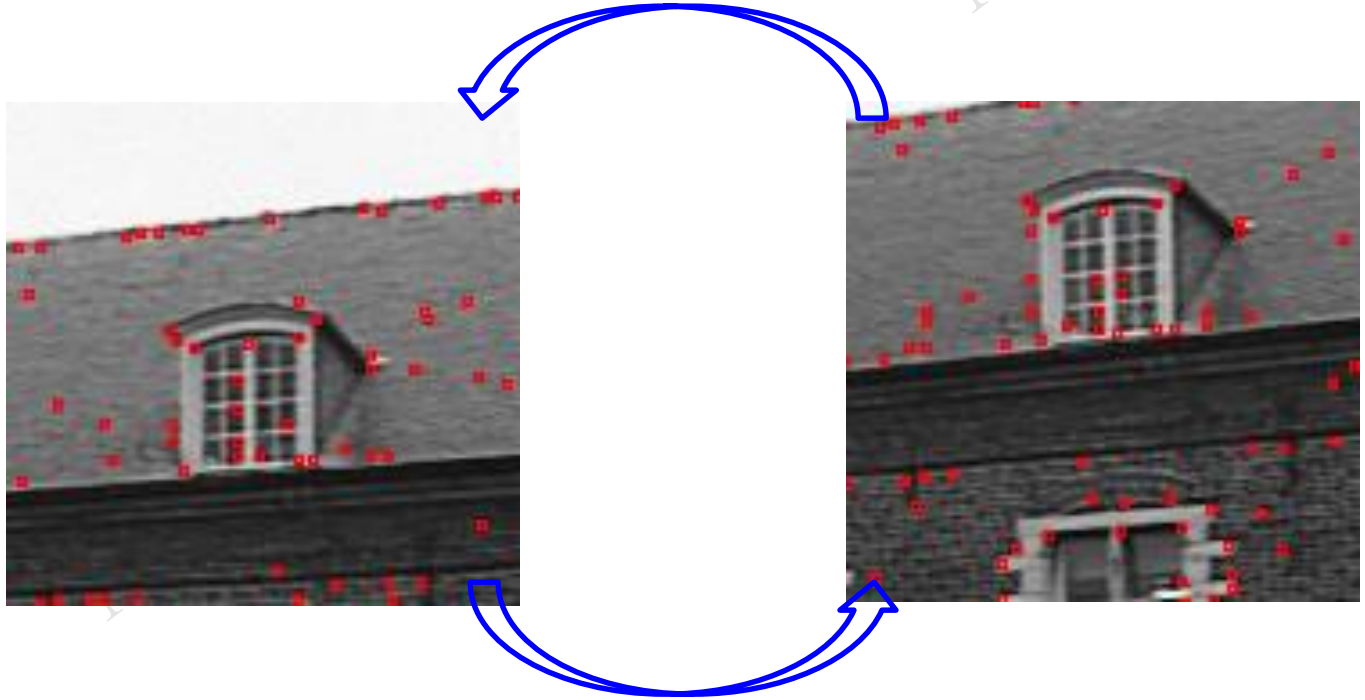
Harris + FLANN

- openCV example please See: Intel, “The OpenCV Tutorials,” 2011. Chapter 6

[1] K. Hajebi, Y. Abbasi-yadkori, H. Shahbazi, and H. Zhang, “Fast approximate nearest-neighbor search with k-nearest neighbor graph,” in *International Joint Conference on Artificial Intelligence*, 2009, no. C, pp. 1312-1317.
[2] M. Muja and D. G. Lowe, “Fast approximate nearest neighbors with automatic algorithm configuration,” in *International Conference on Computer Vision Theory and Applications*, 2009, pp. 331-340.

Feature matching

- Feature → epipolar geometry(F matrix) ?
  epipolar geometry → Feature ?



Feature matching (auto-stitching, panorama)

- Example: SIFT→RANSAC

(a) Image 1

(b) Image 2

(c) SIFT matches 1

(d) SIFT matches 2

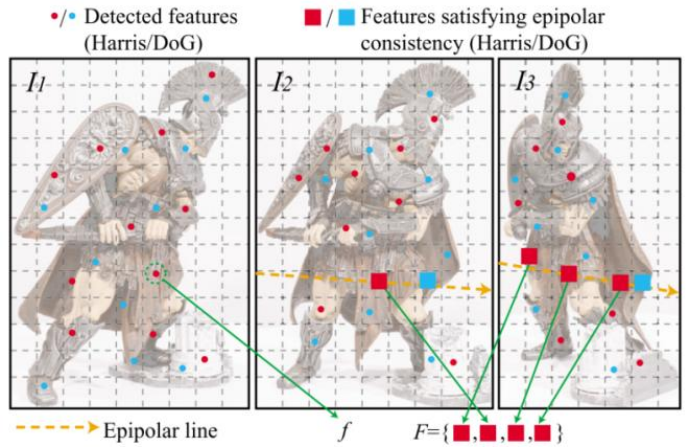
(e) RANSAC inliers 1

(f) RANSAC inliers 2

M. Brown and D. G. Lowe, "Automatic Panoramic Image Stitching using Invariant Features," *International Journal of Computer Vision*, vol. 74, no. 1, pp. 59-73, Dec. 2006.

Feature matching

- Example:SIFT→RANSAC(epipolar geometry) →Harris corner detection→….



Y. Furukawa and J. Ponce, “Accurate, dense, and robust multiview stereopsis,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, vol. 32, no. 8, pp. 1362-76.

Feature tracking

- Tracking in “Single Video Stream”
  - Optical flow
    - by block matching(template matching)
    - by Horn-Schunck
    - by Lucas-Kanade
    - by KLT
  - Kalman filter

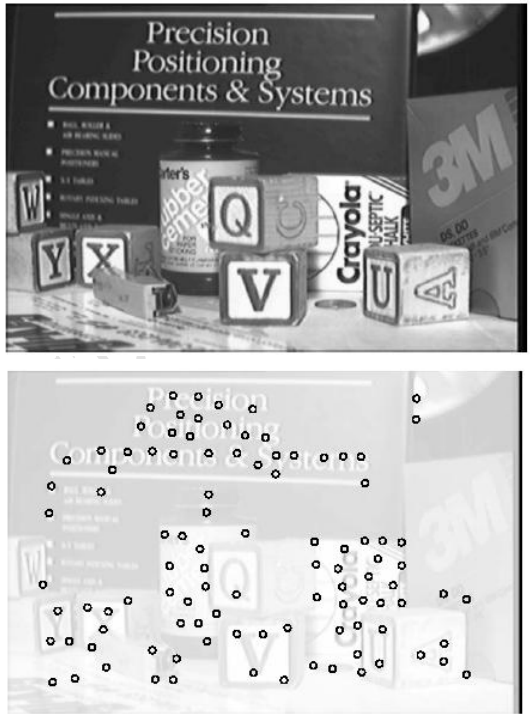
Feature tracking

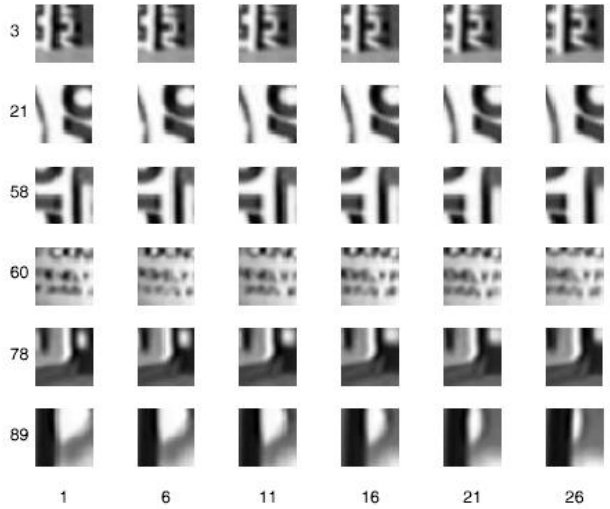
- Identify features and track them over video
  - Small difference between frames
  - potential large difference overall

- Standard approach:
  - KLT (Kanade-Lukas-Tomasi)

Feature tracking

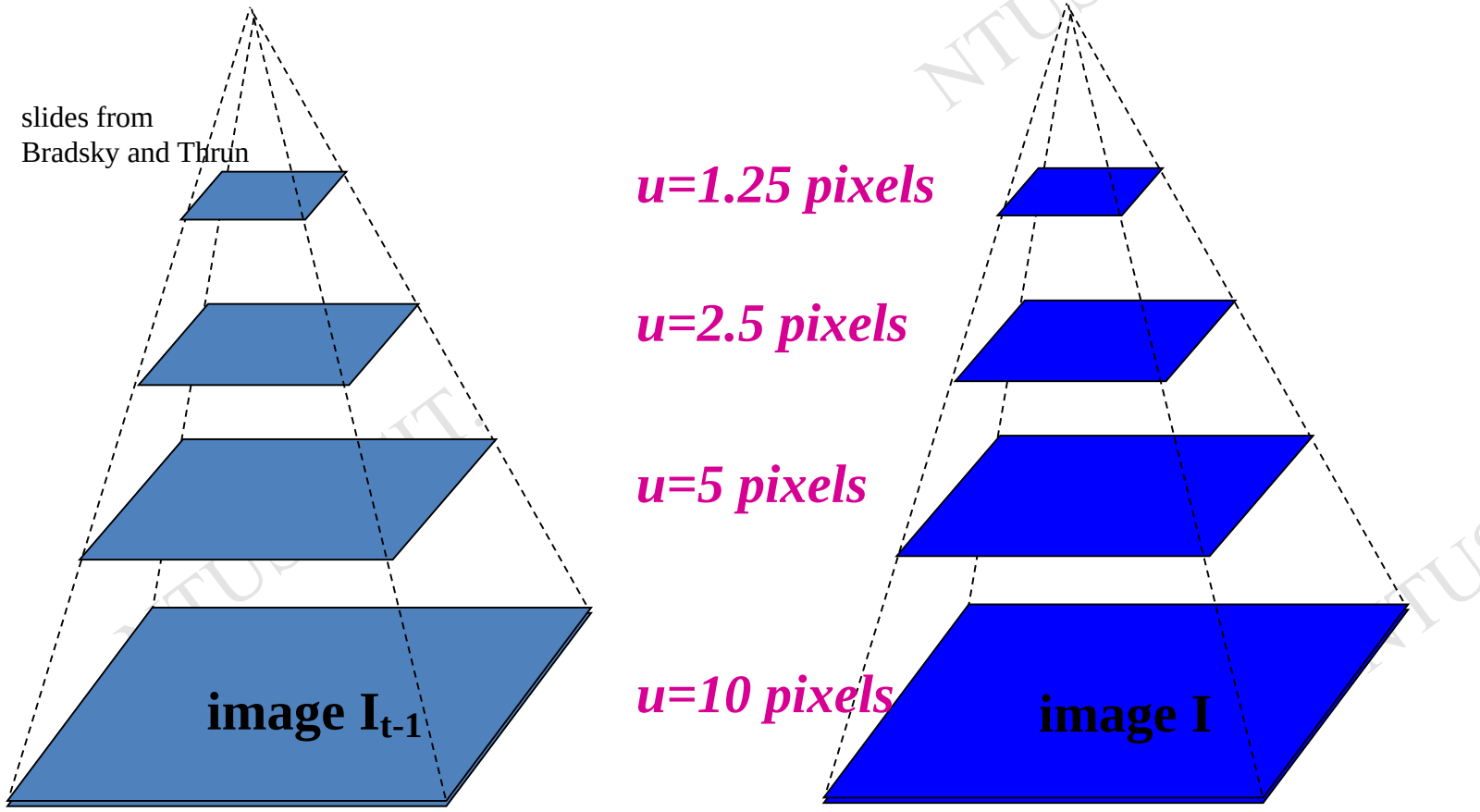
- Good features to keep tracking
  - Perform affine alignment between first and last frame. Stop tracking features with too large errors





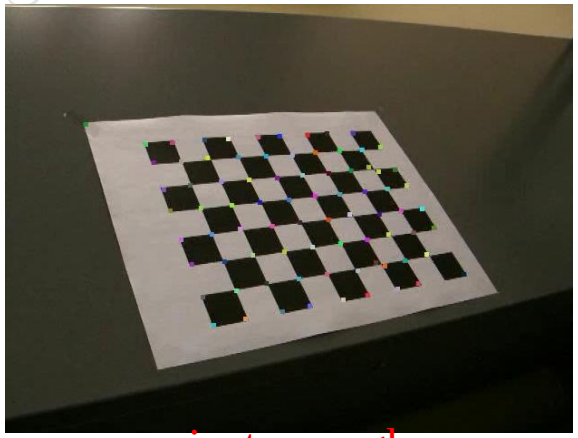
Feature tracking

- Coarse-to-fine optical flow estimationldk-ps-blendmode-up-osx



Feature tracking

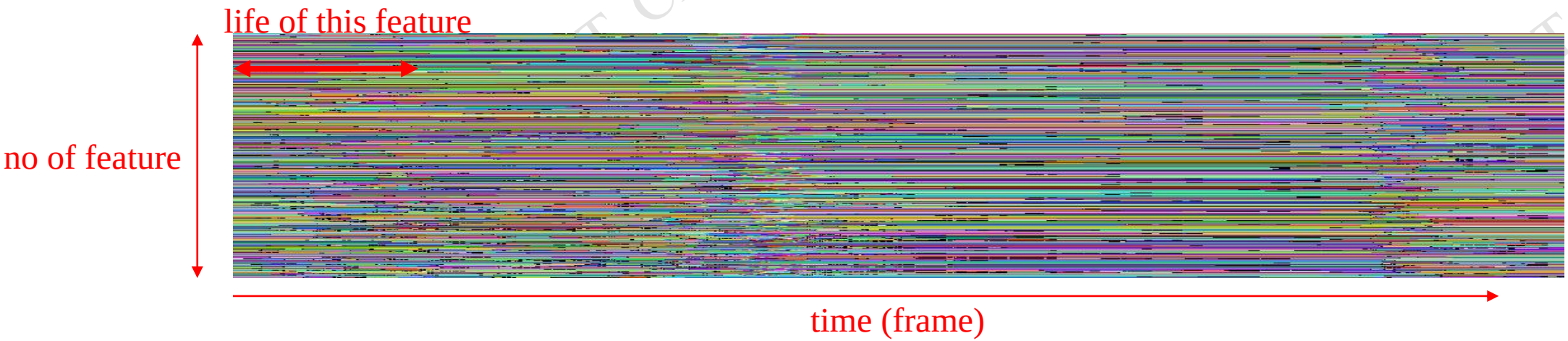
- KLT tracking example



just enough

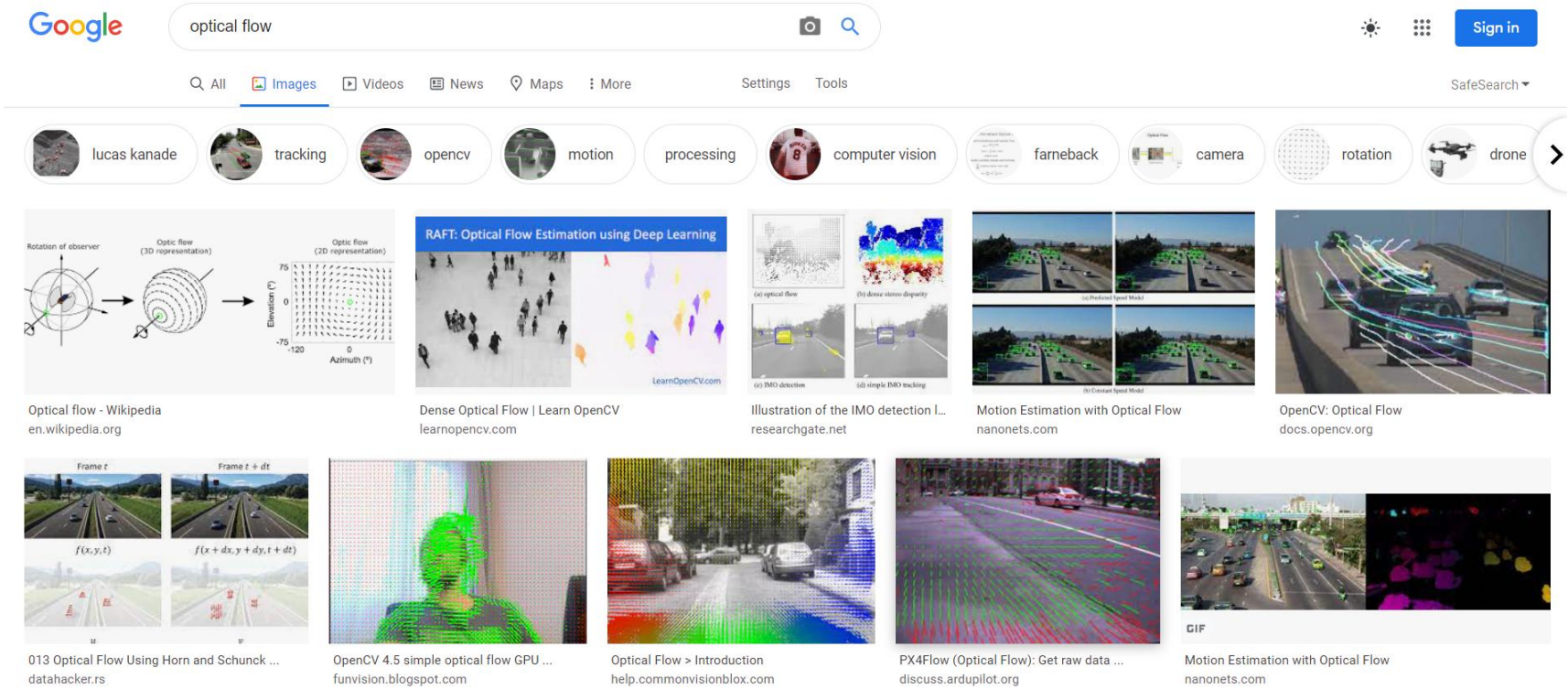
sparse feature

dense feature



Feature tracking

- Optical flow (a prediction tracking model)



Optical flow function in openCV

cvCalcOpticalFlowBM
→using block matching

cvCalcOpticalFlowHS
→using Horn and Schunck

cvCalcOpticalFlowLK
using Lucas-Kanade
cvCalcOpticalFlowPyrLK
→using Lucas-Kanade (LoD)

色彩與照明科技研究所
Graduate Institute of
Color and Illumination Technology

Find us on: facebook