Computer Vision

Bogdan Alexe

bogdan.alexe@fmi.unibuc.ro

University of Bucharest, 2nd semester, 2020-2021

Project 1 – good to know

- 3 tasks to solve = 5 points + 1 point (bonus) = 6 points
- deadline: in ~ 4 weeks time, Saturday, 8th of May 2021, 23:59
  - **hard deadline – no late submission policy**
  - upload your source code
  - on Sunday, 9th of May 2021 we will release the test set. You will run your (untouched) code on test images and send us your results. Immediately we will published the intermediate grades (based on your results). Final grades for Project 1 will be published after carefully checking your code and your report (~3 weeks time).

- do not share/copy the solution with/from your colleagues: you + your colleague/s will get 0 points

Project 1

Extracting visual information from Sudoku puzzles

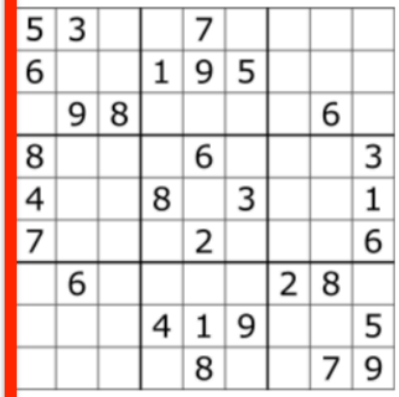
Sudoku

Sudoku is currently one of the most famous puzzles in the world. The most popular version consists on a 9×9 grid made up of 3×3 subgrids, but the general case, an $n^2 \times n^2$ grid with $n \times n$ subgrids is considered. Some cells contain numbers, which can be considered as input data. The goal is to fill in the empty cells, one number in each, so that each column, row, and subgrid contains the numbers 1 through 9 exactly once (numbers 1 to n^2 in the general case). If the input data are correct, the sudoku has one and only one solution.

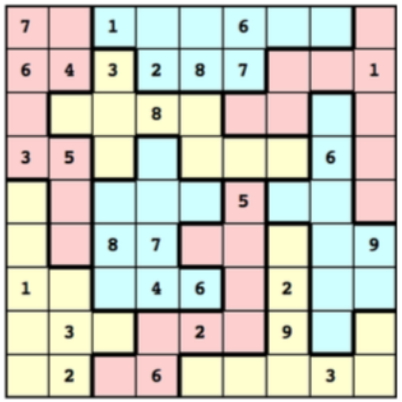
5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

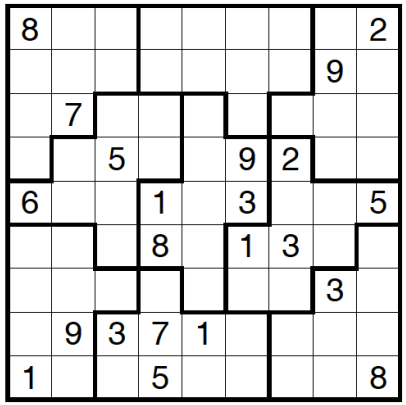
Many variants of Sudoku puzzles



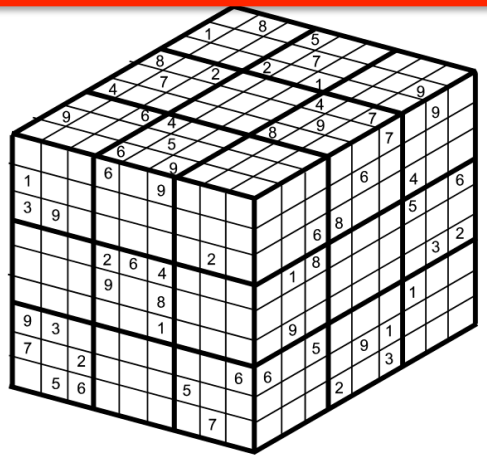
classic



jigsaw

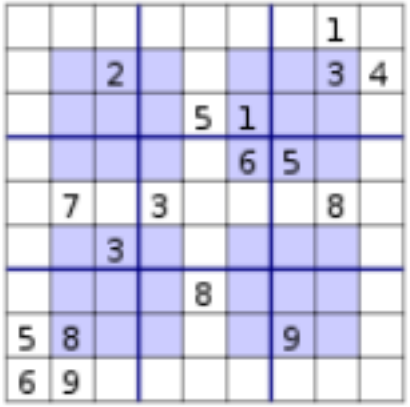


jigsaw

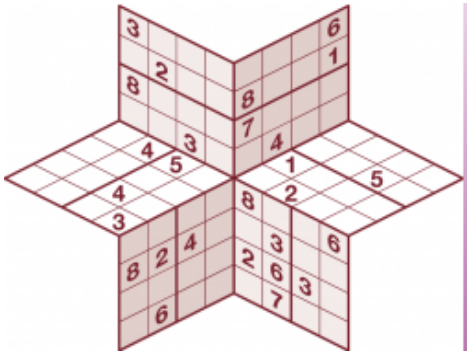


cube

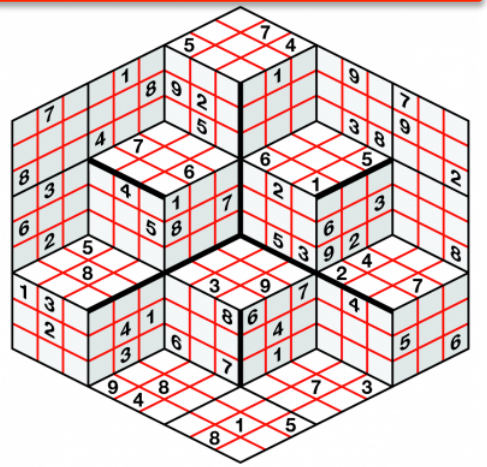
wordoku



hypersudoku



3D star



3D

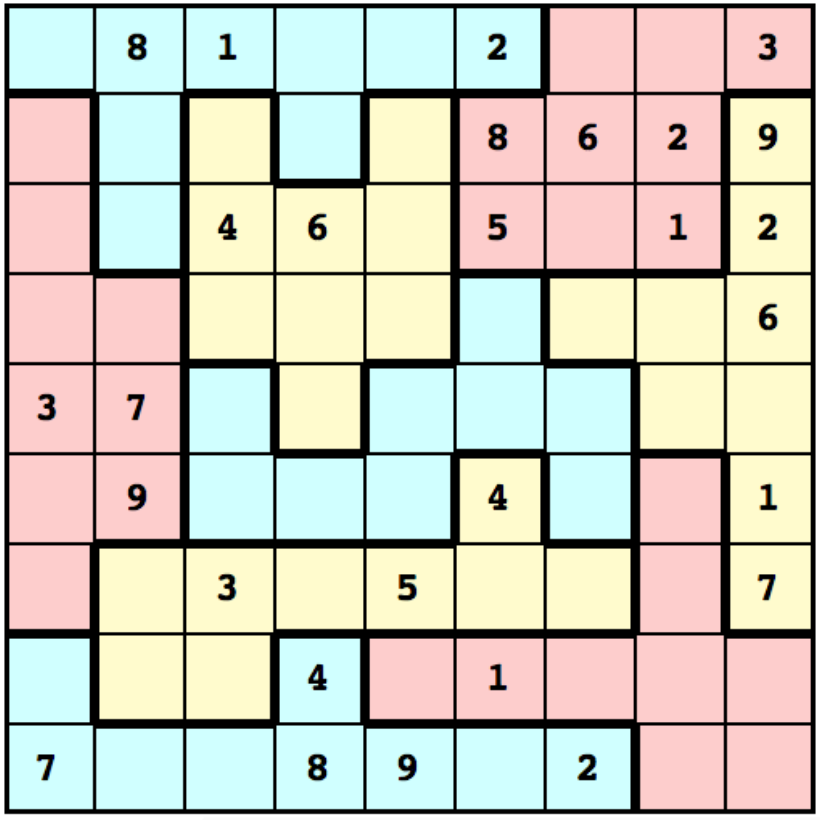
- many more…

Classic Sudoku

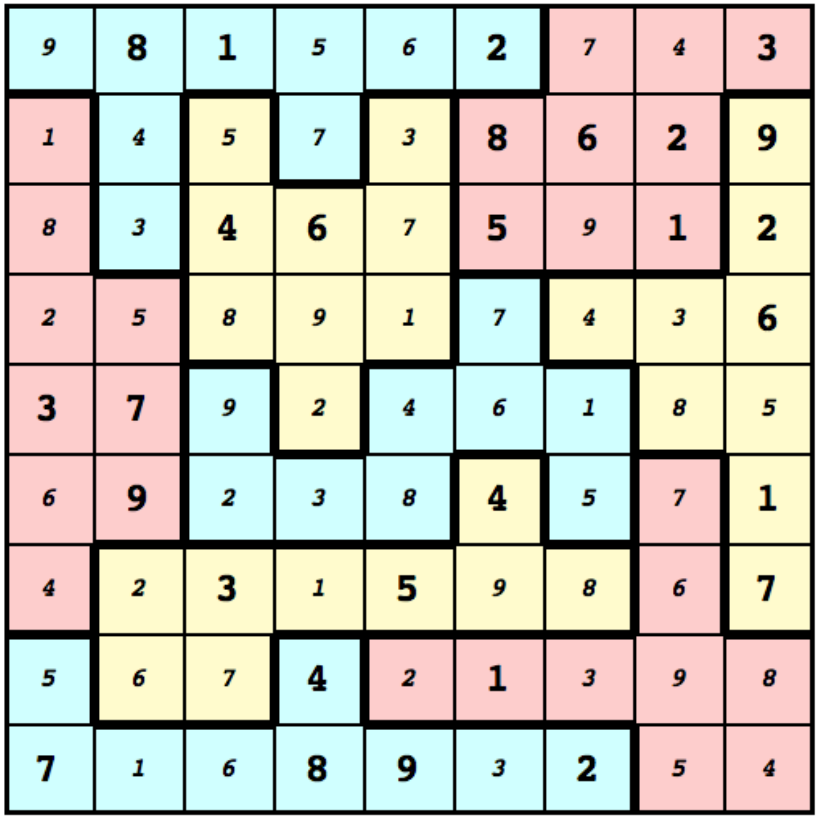
5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Jigsaw Sudoku

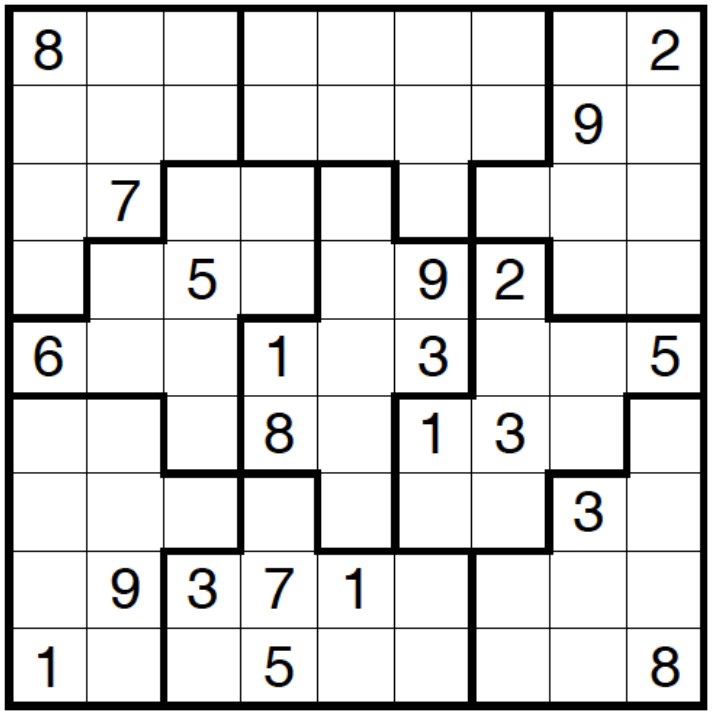


	8	1			2			3
					8	6	2	9
		4	6		5		1	2
								6
3	7							
	9				4			1
		3		5				7
			4		1			
7			8	9		2		



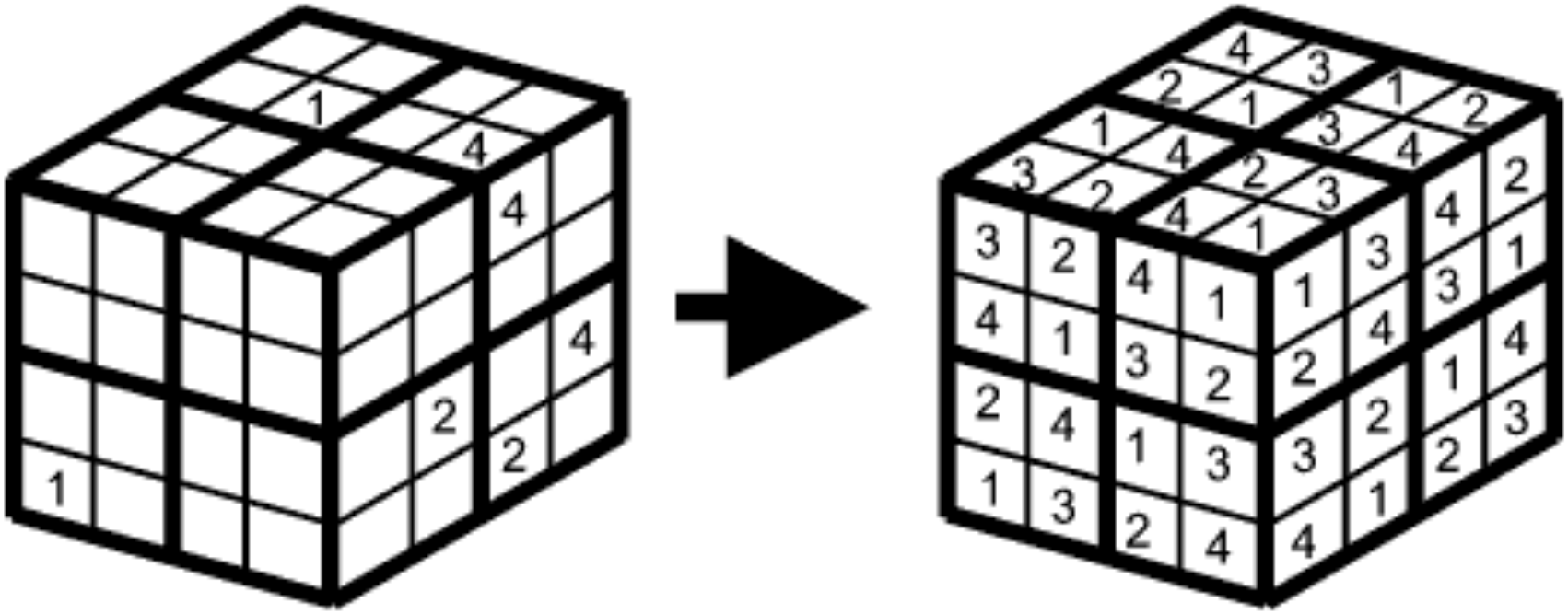
9	8	1	5	6	2	7	4	3
1	4	5	7	3	8	6	2	9
8	3	4	6	7	5	9	1	2
2	5	8	9	1	7	4	3	6
3	7	9	2	4	6	1	8	5
6	9	2	3	8	4	5	7	1
4	2	3	1	5	9	8	6	7
5	6	7	4	2	1	3	9	8
7	1	6	8	9	3	2	5	4

Jigsaw Sudoku

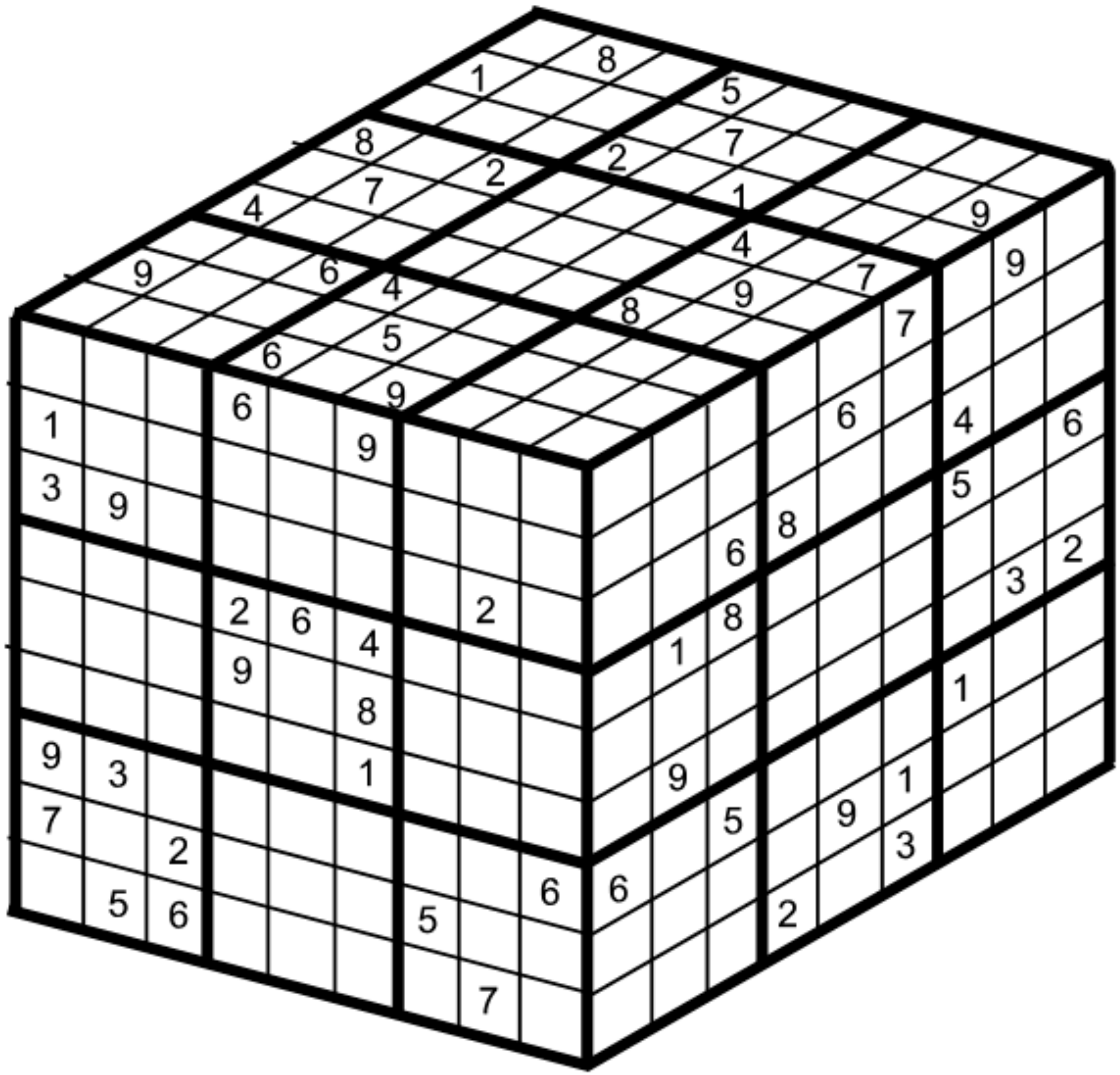


8	5	6	9	3	7	1	4	2
3	1	2	6	8	5	4	9	7
9	7	1	4	6	2	5	8	3
4	8	5	3	7	9	2	1	6
6	2	9	1	4	3	8	7	5
5	4	7	8	2	1	3	6	9
7	6	8	2	5	4	9	3	1
2	9	3	7	1	8	6	5	4
1	3	4	5	9	6	7	2	8

Sudoku Cube


1
4
4
4
2
2
1
4 3 1 2
2 1 3 4
1 4 2 3
3 2 4 1
3 2 4 1 1 3 4 2
4 1 3 2 2 4 3 1
2 4 1 3 3 2 1 4
1 3 2 4 4 1 2 3

Sudoku Cube



First task - Classic Sudoku

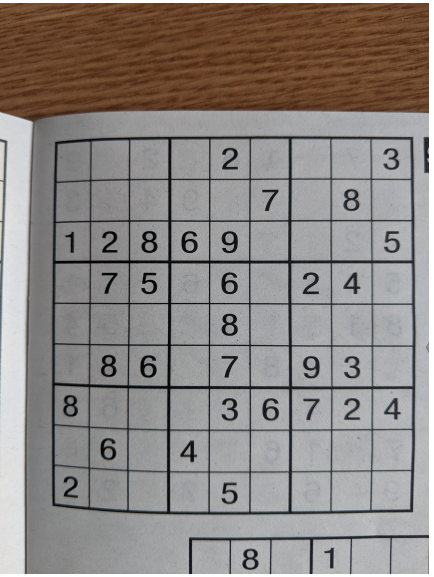
In the first task you are asked to write a program that processes an input image containing a Classic Sudoku puzzle and outputs the configuration of the puzzle by marking the empty cells with letter 'o' and the filled in cells with letter 'x'. The training data consists of 50 training examples. Each training example (an image obtained by taking a photo with the mobile phone) contains one Classic Sudoku puzzle, centered, usually axis aligned or with small rotations with respect to the Ox and Oy axis. Figure 4 shows two training examples of Clasic Sudoku puzzles and their corresponding configurations.

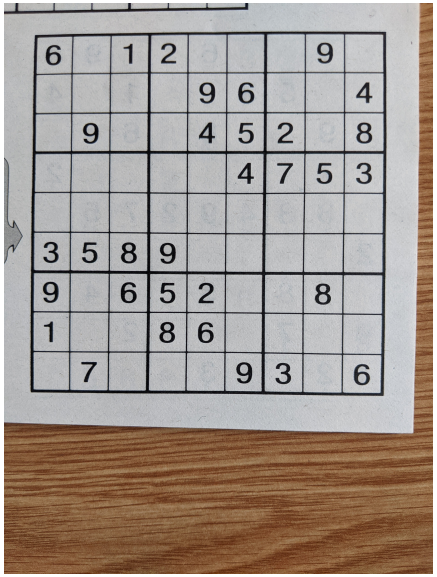
```
OOXXOXXOO
XOXOOOOXX
OOOOXOXXO
XOXOXOOXO
XOOOXOOOX
OXOOXOXOX
OXXOXOOOO
XXOOOOXOX
OOXXOXXOO
```

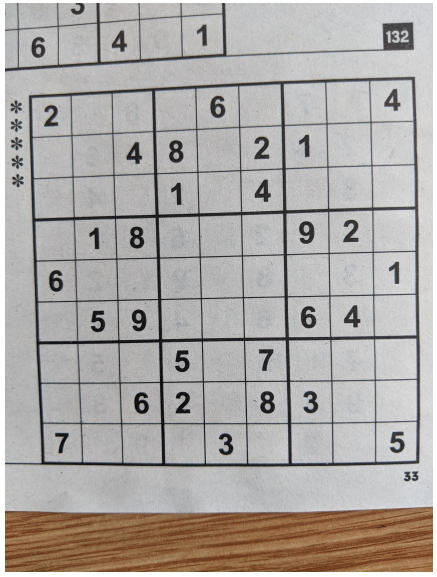
```
OXOOXOOXO
XOOXOXOOX
OOXOOOXOO
OXOOXOOXO
XOXXOXXOX
OXOOXOOXO
OOXOOOXOO
XOOXOXOOX
OXOOXOOXO
```

First task - Classic Sudoku

-training data: 50 images
-testing data: 50 images (will be made available after the deadline)
-each correctly solved puzzle worth 0.05points
-total: 50 x 0.05 = 2.5 points for task 1



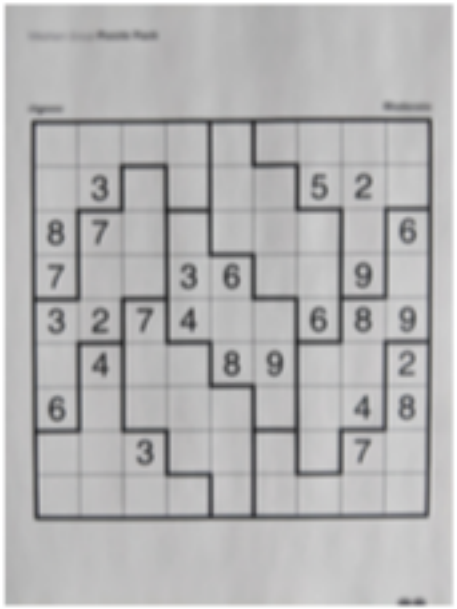




Second task - Jigsaw Sudoku

In the second task you are asked to write a program that processes an input image containing a Jigsaw Sudoku puzzle and outputs the configuration of the puzzle by: (1) determining the irregular shape regions in the puzzle and marking them with digits from 1 to 9; (2) marking the empty cells with letter 'o' and the filled in cells with letter 'x'. The irregular shape regions from the puzzle are separated by bold borders and sometimes (in the colored puzzles) contain cells with the same color.

```
1o1x1o1o1x1o1o2o2o
3o3o1o4x4x1o4x4o2o
3o3x3o3x4o4o4x2x2x
5o5o3x3o3o6o4o4o2o
5x5o5x6o6x6x2o2x2o
5o5o6x6o6o7o7o7o8o
9o5o6o7o7o7o7o8o8o
9x5o6x7o9o9o7o8x8x
9x9o9o9o9o8o8o8o8o
```

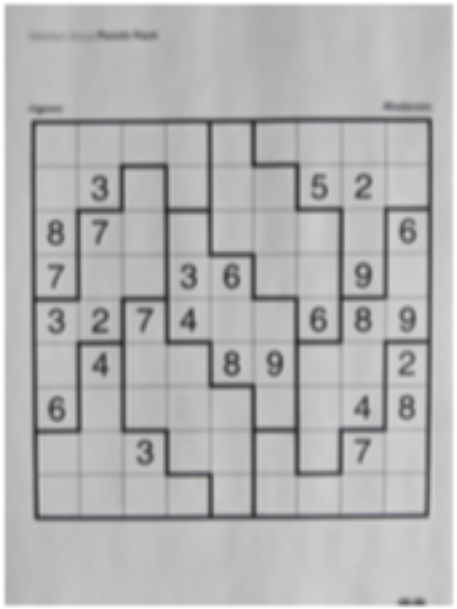


```
1o1o1o1o2o3o3o3o3o
1o1x4o1o2o2o3x3x3o
1x4x4o5o2o2o2o3o6x
1x4o4o5x5x2o2o3x6o
4x4x7x5xmo5o2x6x6x
4o8x7o7o5x5x6o6o9x
4x8o7o7o7o5o6o6x9x
8o8o8x7o7o9o6o9x9o
8o8o8o8o7o9o9o9o9o
```

Second task - Jigsaw Sudoku

red puzzles) contain cells with the same color. For marking the irregular shape regions in a Jigsaw puzzle we use the following simple algorithm: (i) we process the cells from left to right and top to bottom; (ii) the top left cell gets digit 1 as it is part of region 1; (iii) we assign the same digit for all cells in the same region; (iv) the first cell in the next region gets the increased digit.

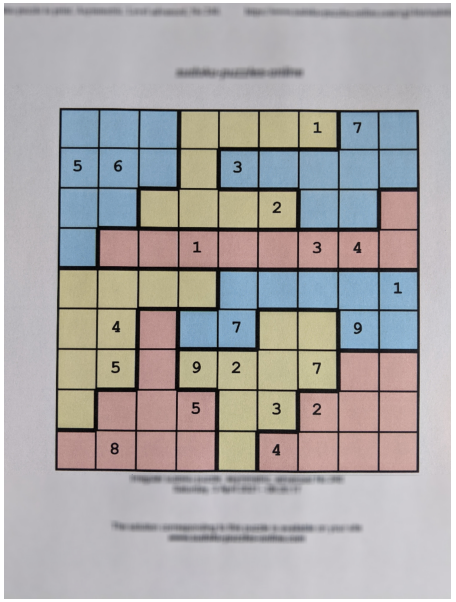
```
1o1x1o1o1x1o1o2o2o
3o3o1o4x4x1o4x4o2o
3o3x3o3x4o4o4x2x2x
5o5o3x3o3o6o4o4o2o
5x5o5x6o6x6x2o2x2o
5o5o6x6o6o7o7o7o8o
9o5o6o7o7o7o7o8o8o
9x5o6x7o9o9o7o8x8x
9x9o9o9o9o8o8o8o8o
```

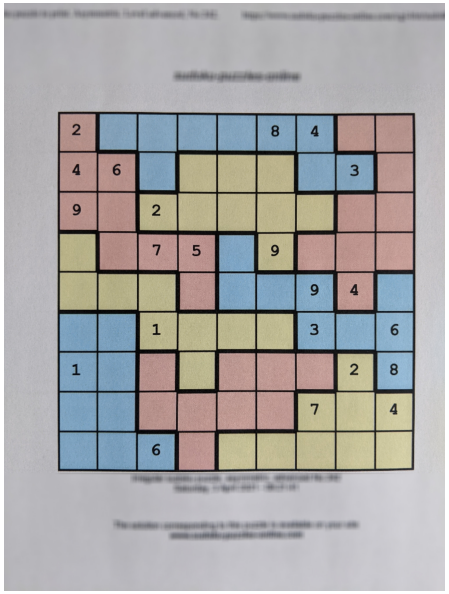


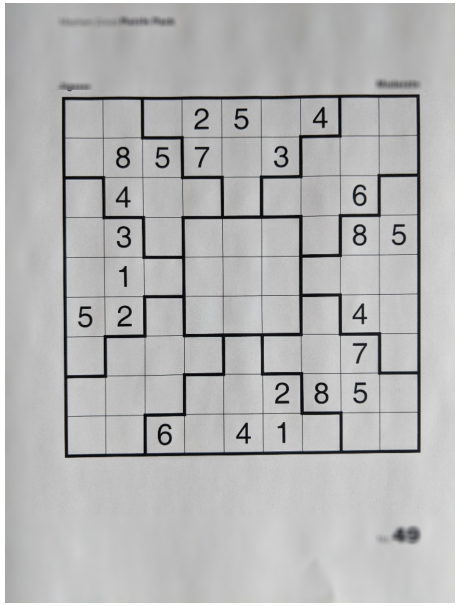
```
1o1o1o1o2o3o3o3o3o
1o1x4o1o2o2o3x3x3o
1x4x4o5o2o2o2o3o6x
1x4o4o5x5x2o2o3x6o
4x4x7x5xmo5o2x6x6x
4o8x7o7o5x5x6o6o9x
4x8o7o7o7o5o6o6x9x
8o8o8x7o7o9o6o9x9o
8o8o8o8o7o9o9o9o9o
```

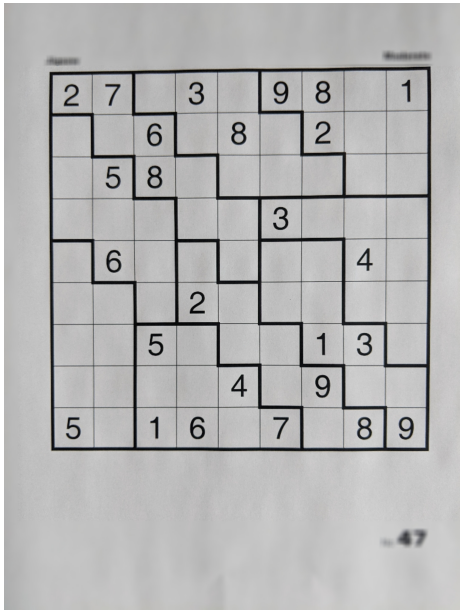
Second task - Jigsaw Sudoku

-training data: 40 images
-testing data: 40 images (will be made available after the deadline)
-each correctly solved puzzle worth 0.05points
-total: 40 x 0.05 = 2 points for task 2



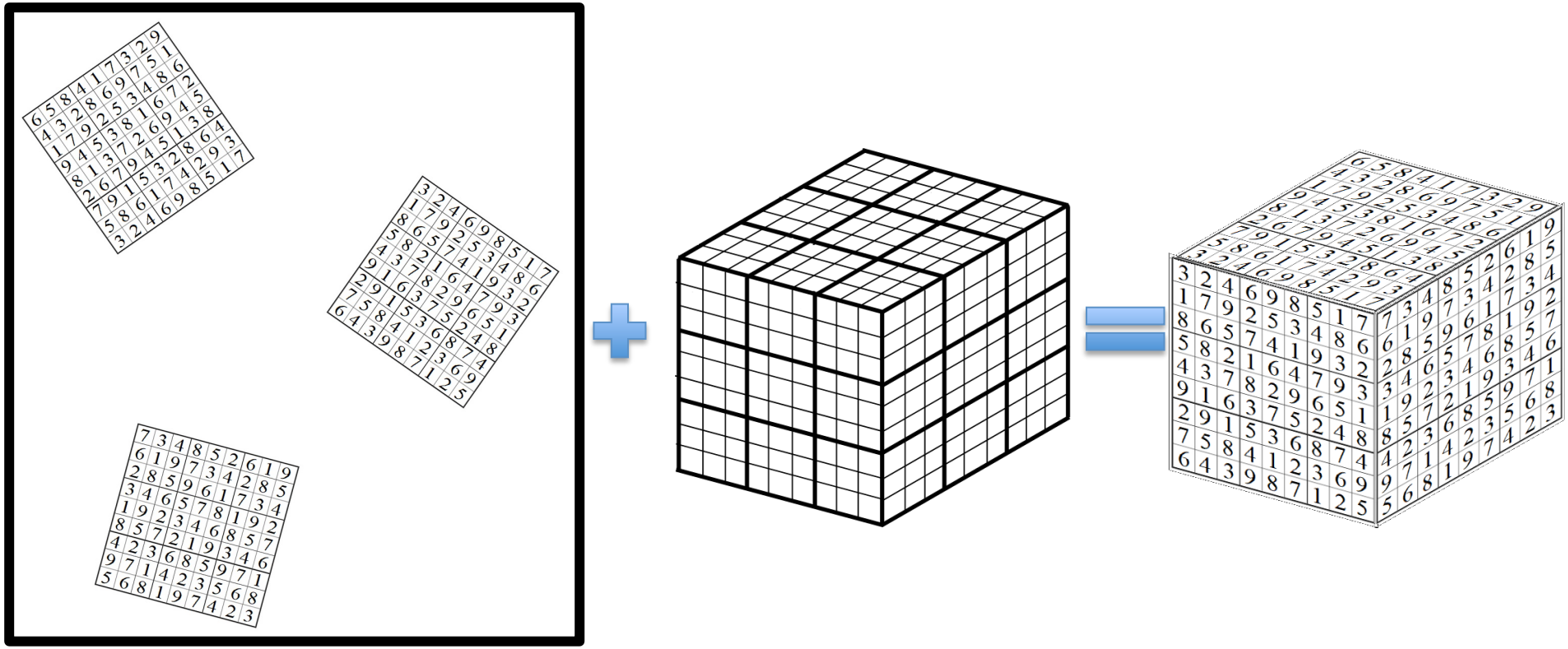






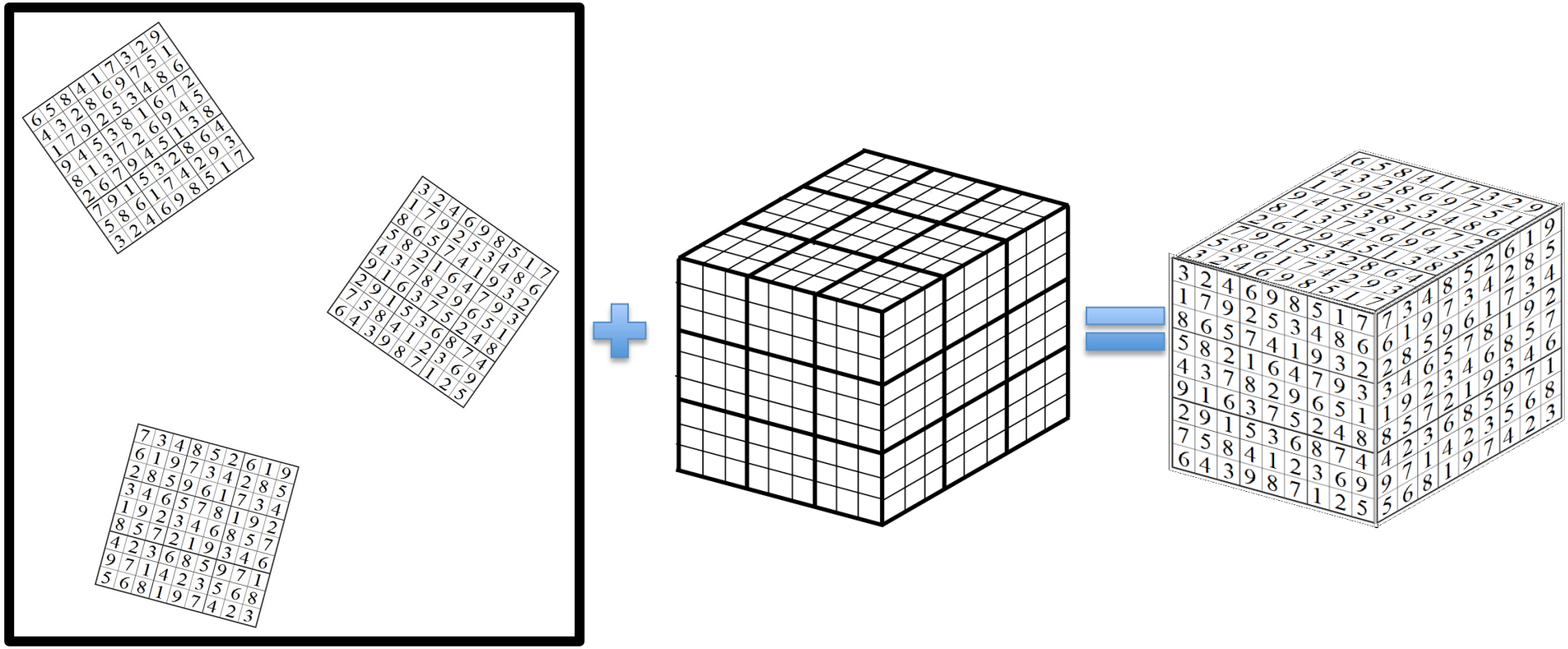
Third task – Sudoku Cube

In the third task you are asked to write a program that processes an input image containing three sides (sudoku puzzles) of a Sudoku Cube and outputs the coresponding Sudoku Cube by: (1) localizing the three sudoku puzzles in the image that form the sides of the Sudoku Cube; (2) inferring their position in the Sudoku Cube using the constraint that the digits on the common edge of two sides must be the same number; (3) warping each side on the corresponding side of a given template for the Sudoku Cube. The training data



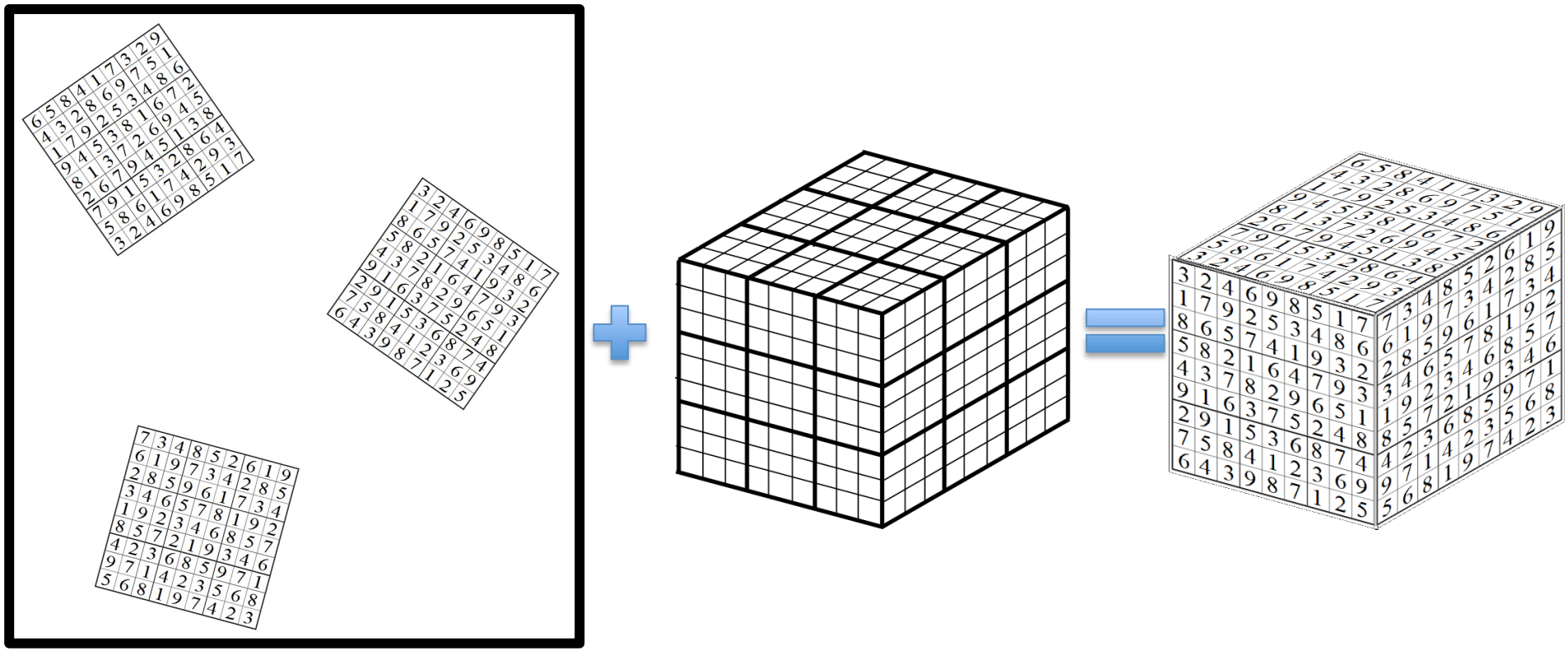
Third task – Sudoku Cube

on the corresponding side of a given template for the Sudoku Cube. The training data consists of 10 training examples. Each training example (an image obtained by taking a photo with the mobile phone) contains three sides of a Sudoku Cube. They are scattered around the image and usually rotated with respect to the axis. Figure 6 shows the Sudoku Cube template, the input image containing the three sides and the desired output. You are allowed to annotate points on the template for warping.



Third task – Sudoku Cube

-training data: 10 images + template with the Sudoku Cube
-testing data: 10 images (will be made available after the deadline)
-each correctly assembled puzzle worth 0.1 points
-total: 10 x 0.1 = 1 point for task 3



Summary

Task 1 – extracting configurations of Classic Sudoku puzzles
-training data: 50 images
-testing data: 50 images (will be made available after the deadline)
-each correctly solved puzzle worth 0.05points
-total: 50 x 0.05 = 2.5 points for task 1

Task 2 - extracting configurations of Jigsaw Sudoku puzzles
-training data: 40 images
-testing data: 40 images (will be made available after the deadline)
-each correctly solved puzzle worth 0.05points
-total: 40 x 0.05 = 2 points for task 2

Task 3 – assembling a Sudoku Cube
-training data: 10 images + template with the Sudoku Cube
-testing data: 10 images (will be made available after the deadline)
-each correctly assembled puzzle worth 0.1 points
-total: 10 x 0.1 = 1 point for task 3

Total = 2.5 points + 2 points + 1 points + 0.5 points (ex officio) = 6 points

All data will be made available this week here: **https://tinyurl.com/CV-2021-Project1**

Lecture 8

Course structure

1. **Features and filters: low-level vision**

   Linear filters, color, texture, edge detection

2. **Grouping and fitting: mid-level vision**

   Fitting curves and lines, robust fitting, RANSAC, Hough transform, segmentation

3. **Multiple views**

   Local invariant feature and description, epipolar geometry and stereo, object instance recognition

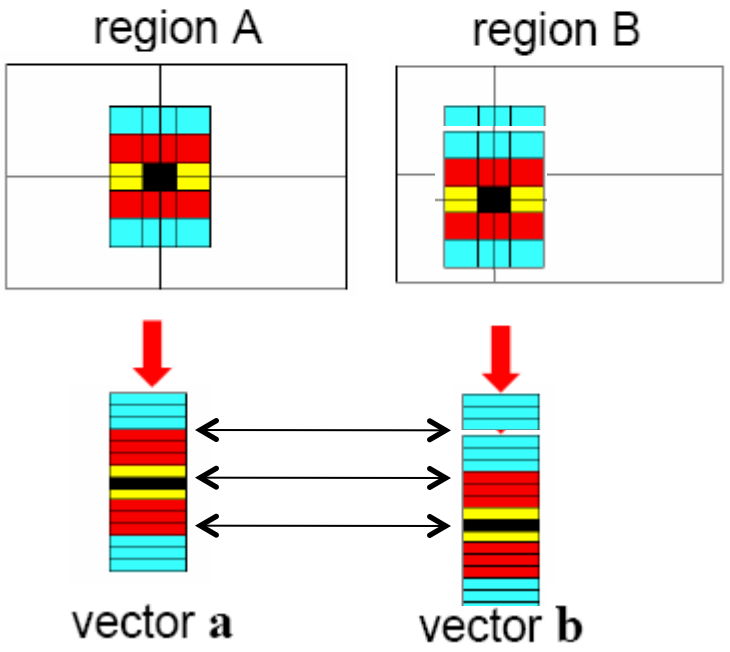
4. **Object Recognition: high – level vision**

   Object classification, object detection, part based models, bovw models

5. **Video understanding**

   Object tracking, background subtraction, motion descriptors, optical flow

Raw patches as local descriptors



The simplest way to describe the neighborhood around an interest point is to write down the list of intensities to form a feature vector.

But this is very sensitive to even small shifts, rotations.

Figure: Andrew Zisserman

Local Descriptors

- The ideal descriptor should be
  - Robust
  - Distinctive
  - Compact
  - Efficient

- Most available descriptors focus on edge/gradient information
  - Capture texture information
  - Color rarely used

David Lowe

FOLLOW

Computer Science Dept., University of British Columbia
Verified email at cs.ubc.ca - Homepage

Computer Vision Object Recognition

TITLE	CITED BY	YEAR
Distinctive image features from scale-invariant keypoints DG Lowe International journal of computer vision 60 (2), 91-110	54091	2004
Object recognition from local scale-invariant features DG Lowe International Conference on Computer Vision, 1999, 1150-1157	18041	1999
Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. M Muja, DG Lowe VISAPP (1) 2, 331-340	2876	2009
Automatic panoramic image stitching using invariant features M Brown, DG Lowe International Journal of Computer Vision 74 (1), 59-73	2427	2007

Distinctive Image Features from Scale-Invariant Keypoints

Abstract

This paper presents a method for extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene. The features are invariant to image scale and rotation, and are shown to provide robust matching across a a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. The features are highly distinctive, in the sense that a single feature can be correctly matched with high probability against a large database of features from many images. This paper also describes an approach to using these features for object recognition. The recognition proceeds by matching individual features to a database of features from known objects using a fast nearest-neighbor algorithm, followed by a Hough transform to identify clusters belonging to a single object, and finally performing verification through least-squares solution for consistent pose parameters. This approach to recognition can robustly identify objects among clutter and occlusion while achieving near real-time performance.

http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf

SIFT properties

- Invariant to
  - Scale
  - Rotation

- Partially invariant to
  - Illumination changes
  - Camera viewpoint
  - Occlusion, clutter

Distinctive Image Features from Scale-Invariant Keypoints

1. **Scale-space extrema detection:** The first stage of computation searches over all scales and image locations. It is implemented efficiently by using a difference-of-Gaussian function to identify potential interest points that are invariant to scale and orientation.

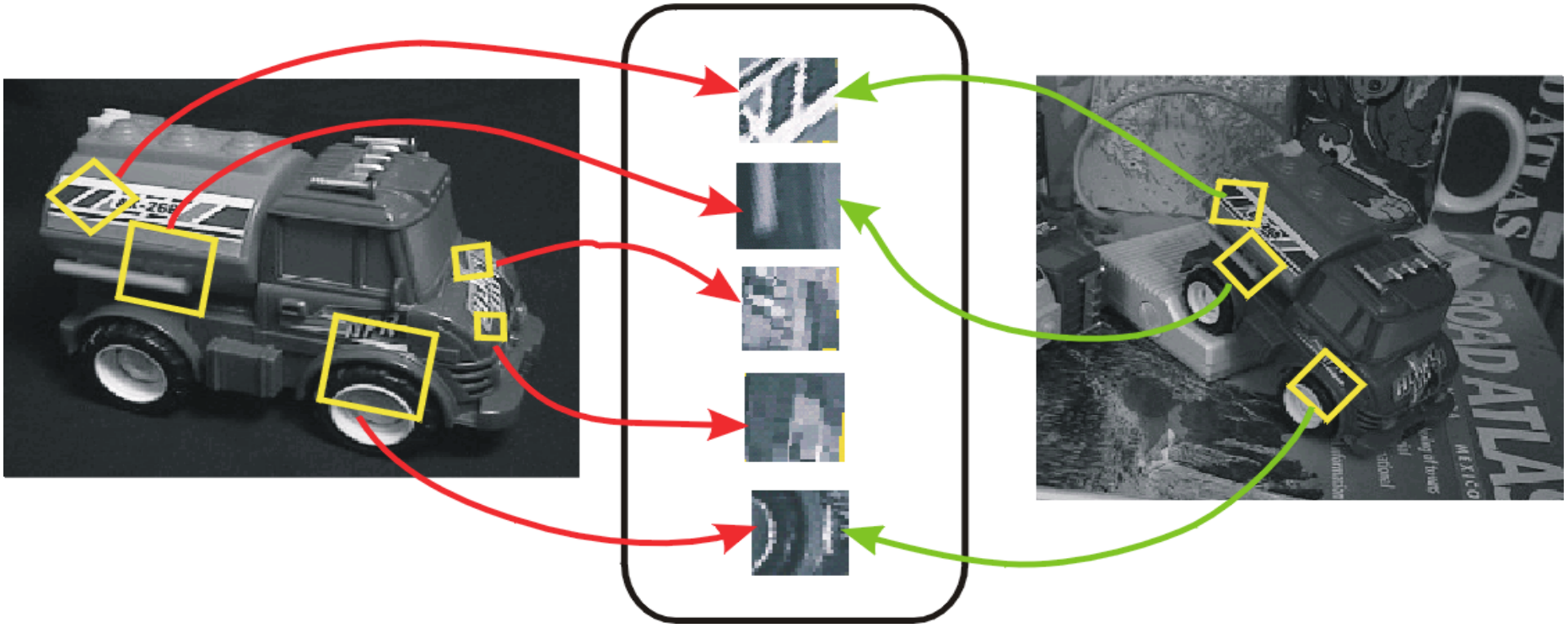
2. **Keypoint localization:** At each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability.

3. **Orientation assignment:** One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.

4. **Keypoint descriptor:** The local image gradients are measured at the selected scale in the region around each keypoint. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination.

This approach has been named the Scale Invariant Feature Transform (SIFT), as it transforms image data into scale-invariant coordinates relative to local features.

From keypoint detection to feature description



- Detection is *covariant*:

  features(transform(image)) = transform(features(image))

- Description is *invariant*:

  features(transform(image)) = features(image)

Details of Lowe's SIFT algorithm

- Run DoG detector
  - Find maxima in location/scale space
  - Remove edge points

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

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}$$

4.1 Eliminating edge responses

For stability, it is not sufficient to reject keypoints with low contrast. The difference-of-Gaussian function will have a strong response along edges, even if the location along the edge is poorly determined and therefore unstable to small amounts of noise.

A poorly defined peak in the difference-of-Gaussian function will have a large principal curvature across the edge but a small one in the perpendicular direction. The principal curvatures can be computed from a 2x2 Hessian matrix, **H**, computed at the location and scale of the keypoint:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \quad (4)$$

The derivatives are estimated by taking differences of neighboring sample points.

The eigenvalues of **H** are proportional to the principal curvatures of D. Borrowing from the approach used by Harris and Stephens (1988), we can avoid explicitly computing the eigenvalues, as we are only concerned with their ratio. Let α be the eigenvalue with the largest magnitude and β be the smaller one. Then, we can compute the sum of the eigenvalues from the trace of **H** and their product from the determinant:

$$\mathrm{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$
$$\mathrm{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

Lowe IJCV 2004

Details of Lowe's SIFT algorithm

- Run DoG detector
  - Find maxima in location/scale space
  - Remove edge points

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

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}$$

In the unlikely event that the determinant is negative, the curvatures have different signs so the point is discarded as not being an extremum. Let r be the ratio between the largest magnitude eigenvalue and the smaller one, so that $\alpha = r\beta$. Then,

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

which depends only on the ratio of the eigenvalues rather than their individual values. The quantity $(r+1)^2/r$ is at a minimum when the two eigenvalues are equal and it increases with r. Therefore, to check that the ratio of principal curvatures is below some threshold, r, we only need to check

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

This is very efficient to compute, with less than 20 floating point operations required to test each keypoint. The experiments in this paper use a value of $r = 10$, which eliminates keypoints that have a ratio between the principal curvatures greater than 10. The transition from Figure 5 (c) to (d) shows the effects of this operation.

Lowe IJCV 2004

Detecting keypoints with DoG

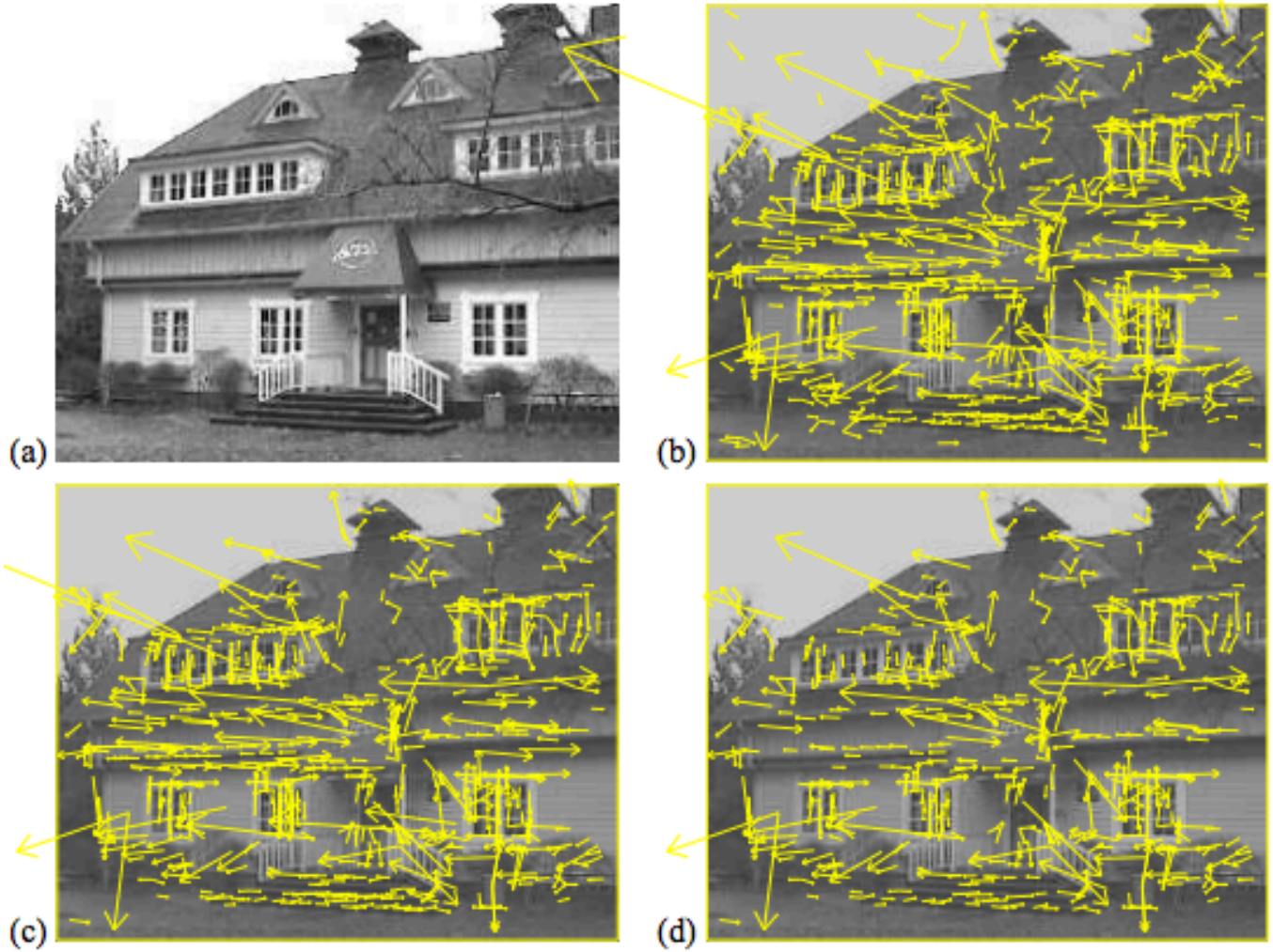


Figure 5: This figure shows the stages of keypoint selection. (a) The 233x189 pixel original image. (b) The initial 832 keypoints locations at maxima and minima of the difference-of-Gaussian function. Keypoints are displayed as vectors indicating scale, orientation, and location. (c) After applying a threshold on minimum contrast, 729 keypoints remain. (d) The final 536 keypoints that remain following an additional threshold on ratio of principal curvatures.

Details of Lowe's SIFT algorithm

- Run DoG detector
  - Find maxima in location/scale space
  - Remove edge points

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

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}$$

- Find all major orientations
  - Bin orientations into 36 bin histogram
    - Weight by gradient magnitude
    - Weight by distance to center (Gaussian-weighted mean)
  - Return orientations within 0.8 of peak
    - Use parabola for better orientation fit

Lowe IJCV 2004

Following experimentation with a number of approaches to assigning a local orientation, the following approach was found to give the most stable results. The scale of the keypoint is used to select the Gaussian smoothed image, L, with the closest scale, so that all computations are performed in a scale-invariant manner. For each image sample, $L(x, y)$, at this scale, the gradient magnitude, $m(x, y)$, and orientation, $\theta(x, y)$, is precomputed using pixel differences:

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

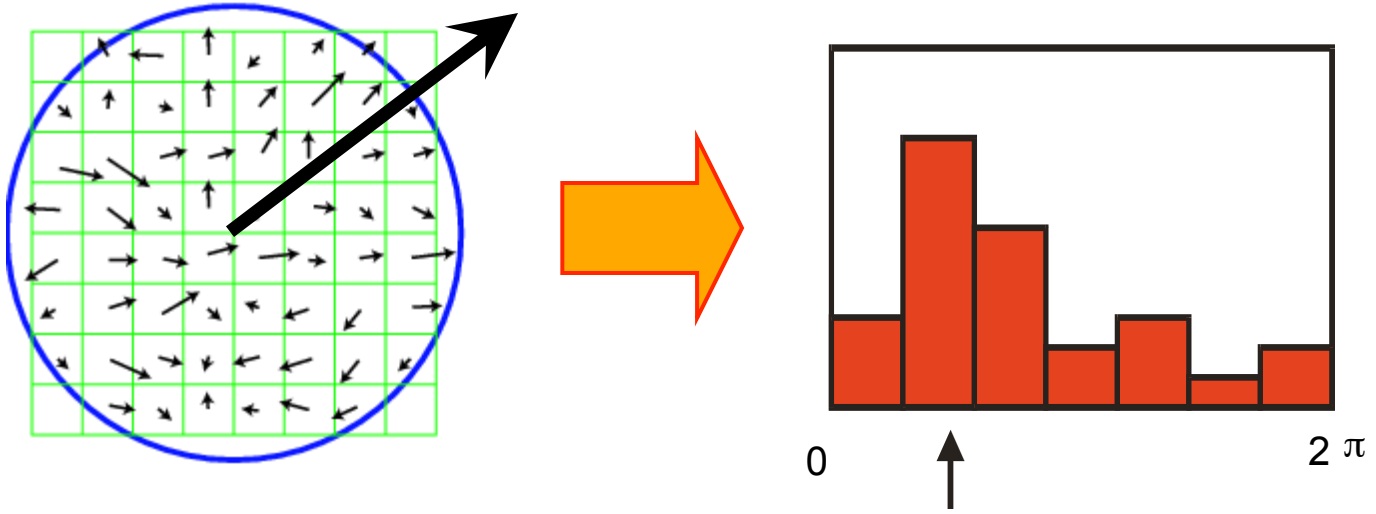
$$\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1))/(L(x+1, y) - L(x-1, y)))$$

An orientation histogram is formed from the gradient orientations of sample points within a region around the keypoint. The orientation histogram has 36 bins covering the 360 degree range of orientations. Each sample added to the histogram is weighted by its gradient magnitude and by a Gaussian-weighted circular window with a σ that is 1.5 times that of the scale of the keypoint.

Peaks in the orientation histogram correspond to dominant directions of local gradients. The highest peak in the histogram is detected, and then any other local peak that is within 80% of the highest peak is used to also create a keypoint with that orientation. Therefore, for locations with multiple peaks of similar magnitude, there will be multiple keypoints created at the same location and scale but different orientations. Only about 15% of points are assigned multiple orientations, but these contribute significantly to the stability of matching. Finally, a parabola is fit to the 3 histogram values closest to each peak to interpolate the peak position for better accuracy.

Finding a reference orientation

- Create histogram of local gradient directions in the patch
- Assign reference orientation at peak of smoothed histogram



Details of Lowe's SIFT algorithm

- Run DoG detector
  - Find maxima in location/scale space
  - Remove edge points

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

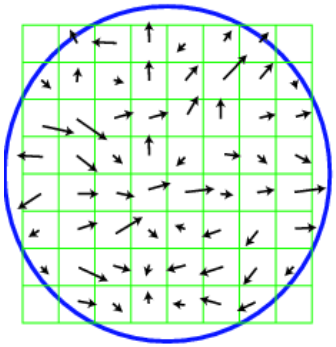
$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}$$

- Find all major orientations
  - Bin orientations into 36 bin histogram
    - Weight by gradient magnitude
    - Weight by distance to center (Gaussian-weighted mean)
  - Return orientations within 0.8 of peak
    - Use parabola for better orientation fit
- For each (x,y,scale,orientation), create descriptor:
  - Sample 16x16 gradient mag. and rel. orientation
  - Bin 4x4 samples into 4x4 histograms
  - Threshold values to max of 0.2, divide by L2 norm
  - Final descriptor: 4x4x8 normalized histograms

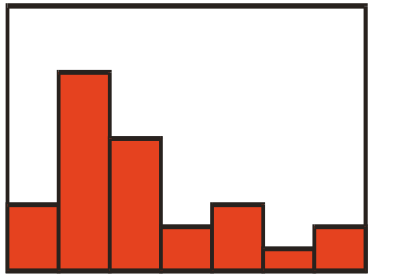
Lowe IJCV 2004

SIFT descriptor

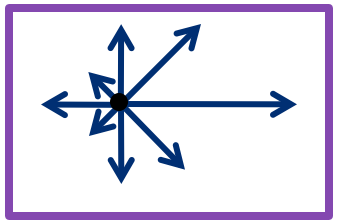
- Use histograms to bin pixels within sub-patches according to their orientation.

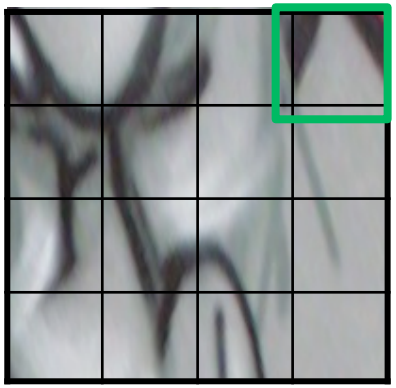


gradients

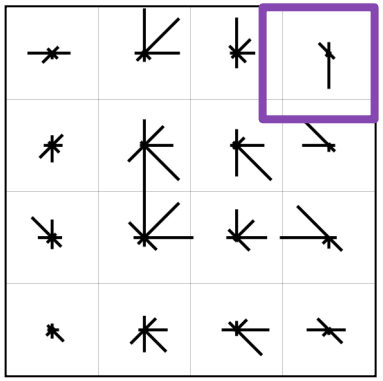


binned by orientation





subdivided local patch



histogram per grid cell

Final descriptor = concatenation of all histograms, normalize

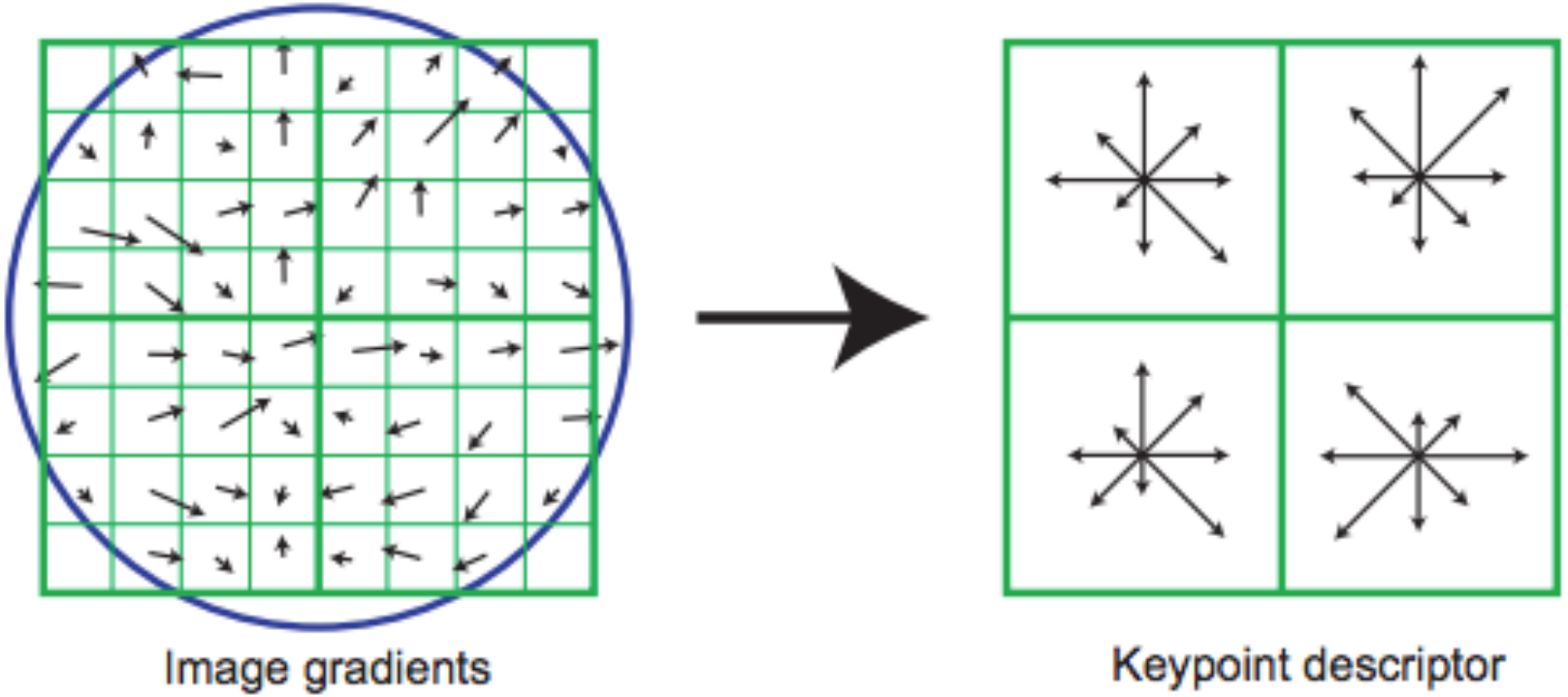
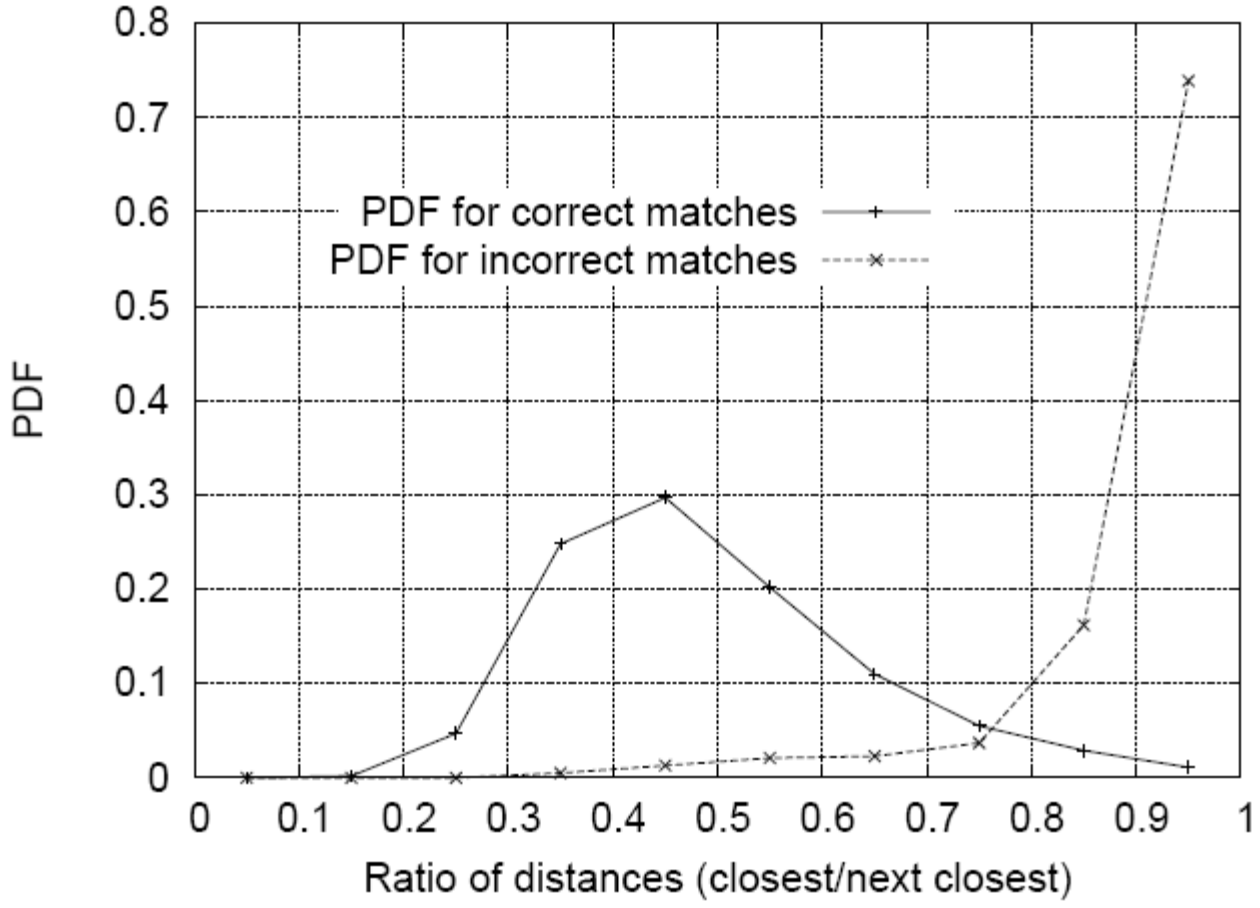


Figure 7: A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2x2 descriptor array computed from an 8x8 set of samples, whereas the experiments in this paper use 4x4 descriptors computed from a 16x16 sample array.

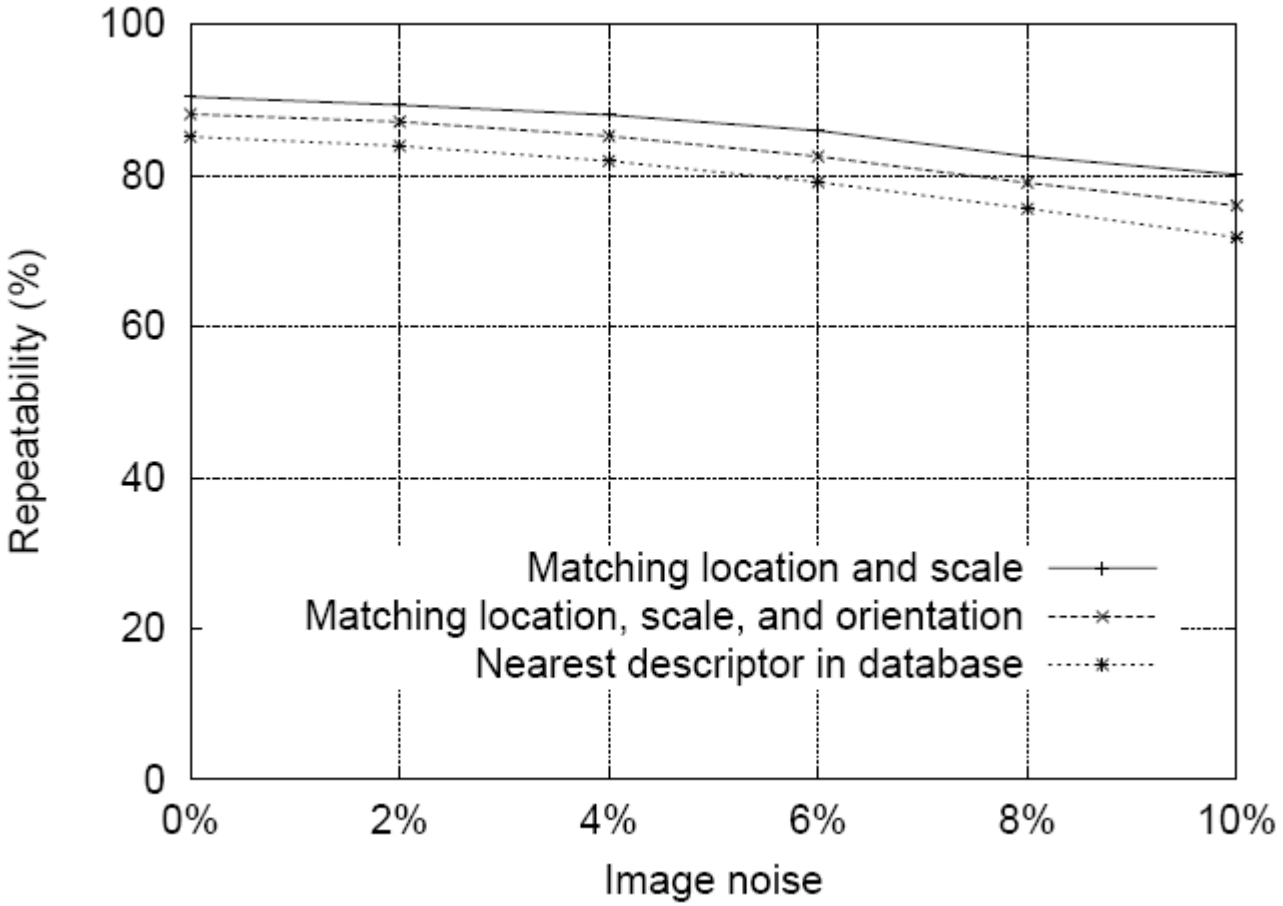
Matching SIFT Descriptors

- Nearest neighbor (Euclidean distance)
- Threshold ratio of nearest to 2nd nearest descriptor



Lowe IJCV 2004

SIFT Repeatability



Lowe IJCV 2004

SIFT Repeatability

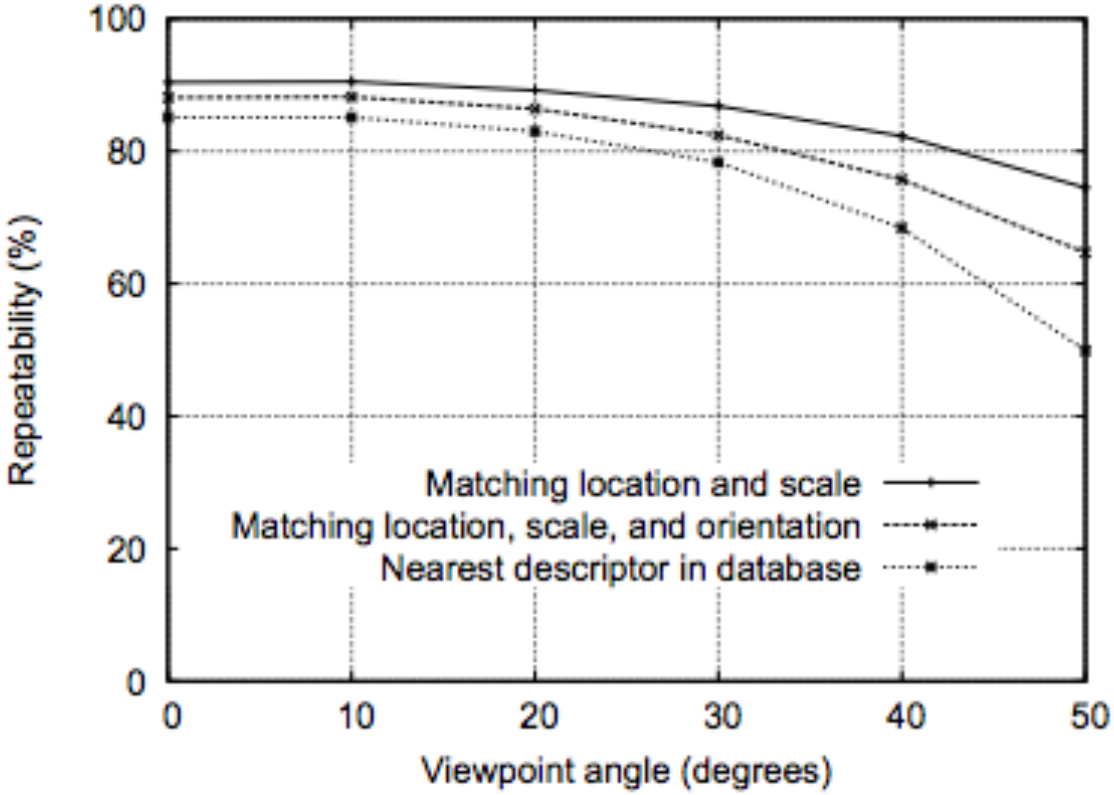


Figure 9: This graph shows the stability of detection for keypoint location, orientation, and final matching to a database as a function of affine distortion. The degree of affine distortion is expressed in terms of the equivalent viewpoint rotation in depth for a planar surface.

Lowe IJCV 2004

SIFT Repeatability

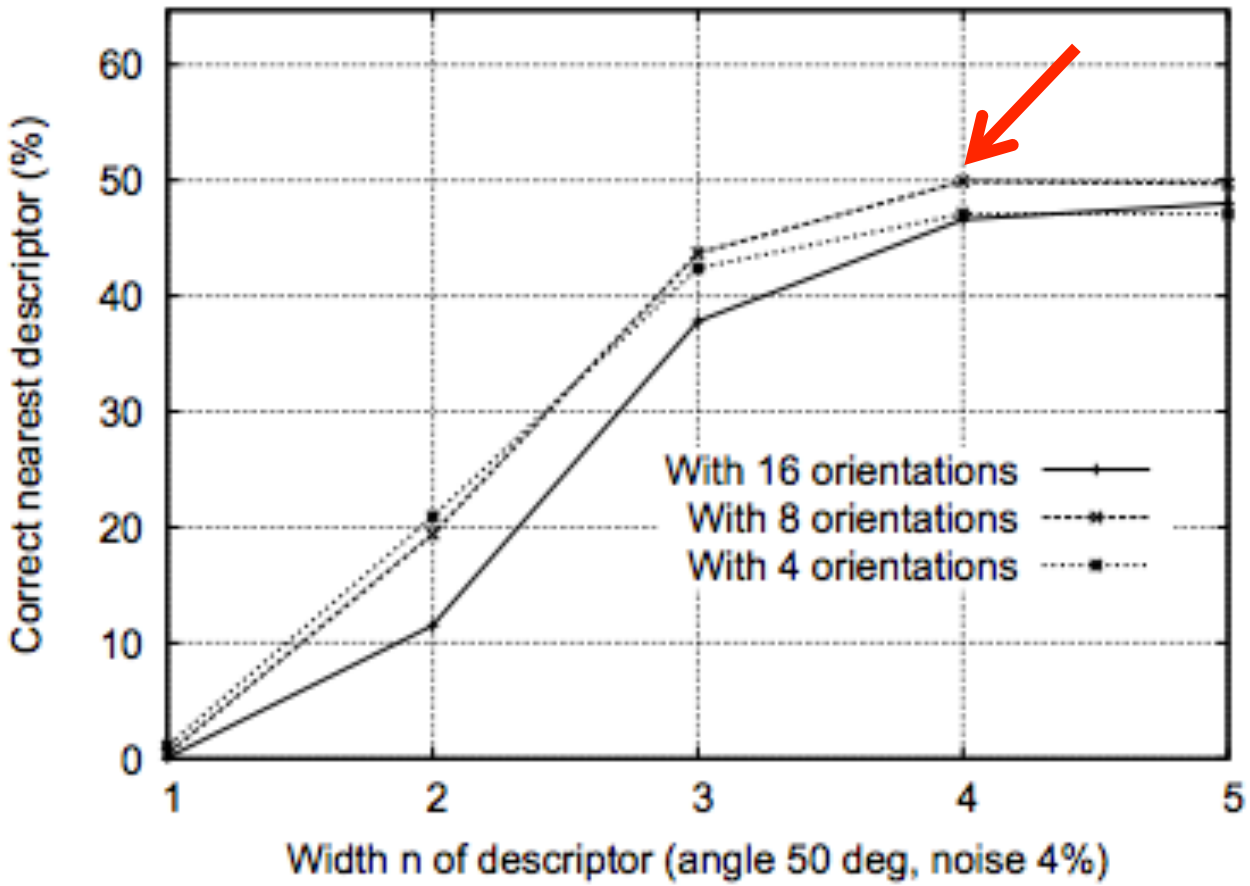


Figure 8: This graph shows the percent of keypoints giving the correct match to a database of 40,000 keypoints as a function of width of the $n \times n$ keypoint descriptor and the number of orientations in each histogram. The graph is computed for images with affine viewpoint change of 50 degrees and addition of 4% noise.

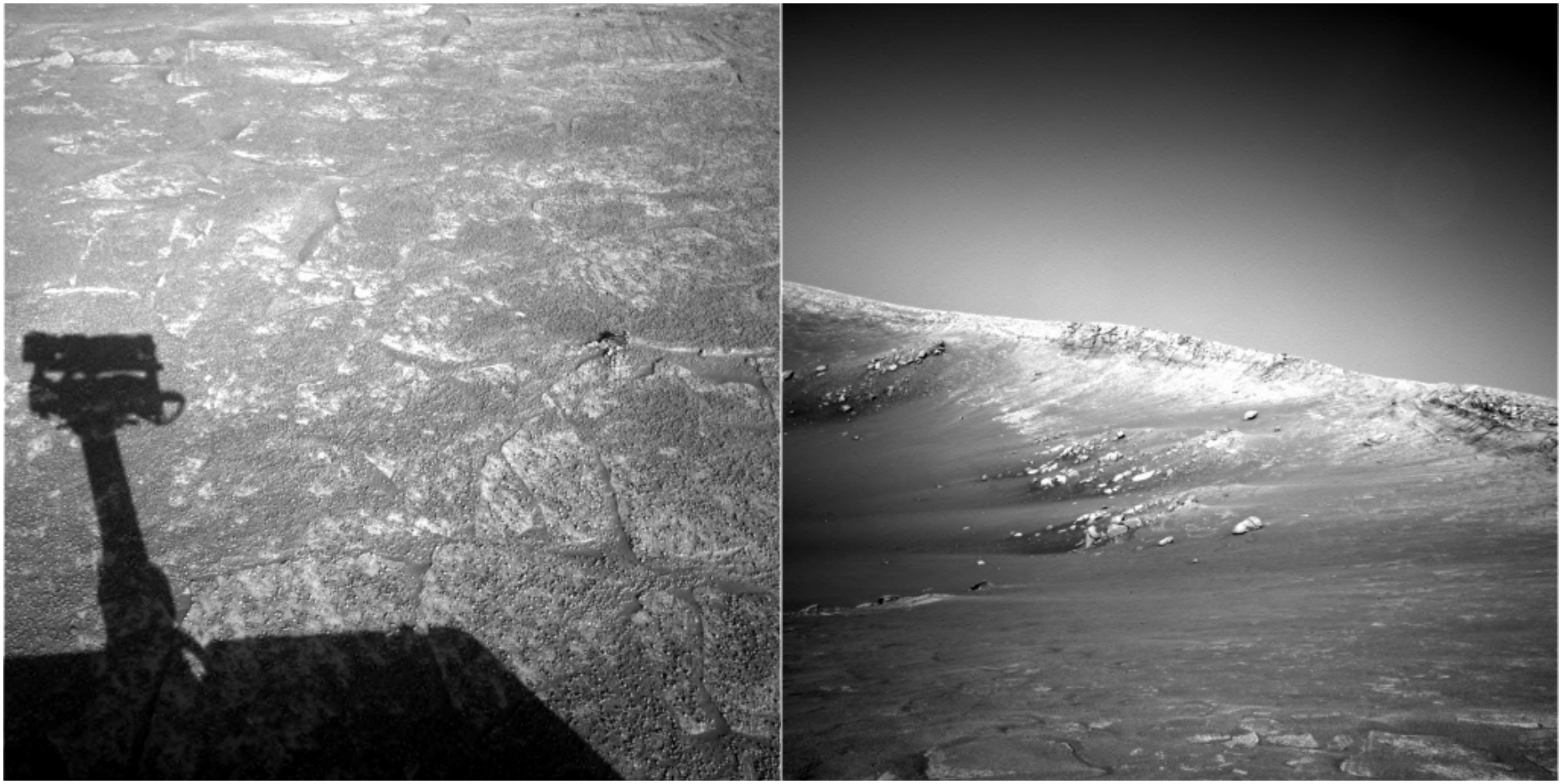
Lowe IJCV 2004

SIFT descriptor

- Extraordinarily robust matching technique
  - Can handle changes in viewpoint
    - Up to about 60 degree out of plane rotation
  - Can handle significant changes in illumination
    - Sometimes even day vs. night (below)
  - Fast and efficient—can run in real time
  - Lots of code available, e.g. http://www.vlfeat.org/overview/

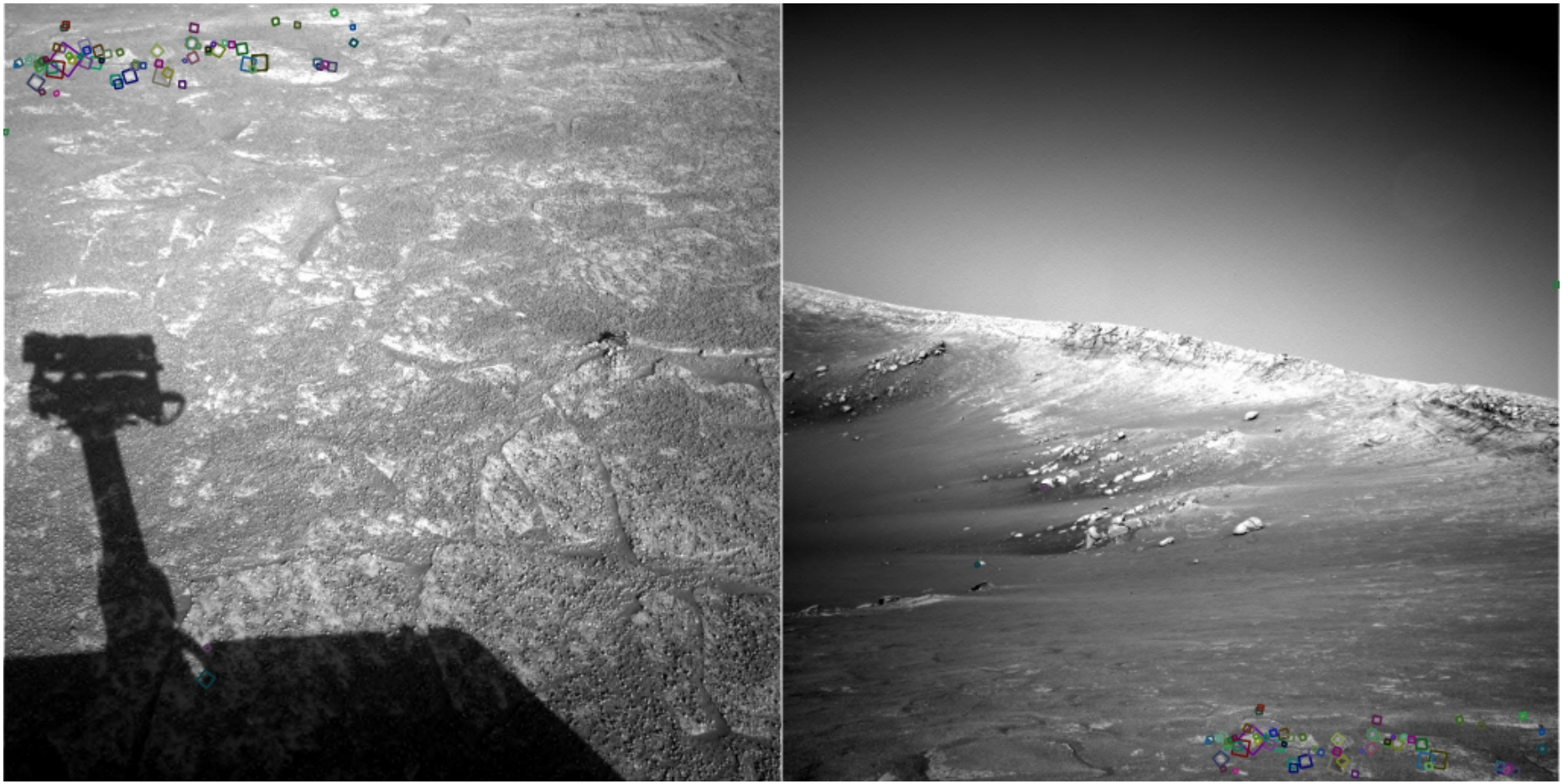
Steve Seitz

Example



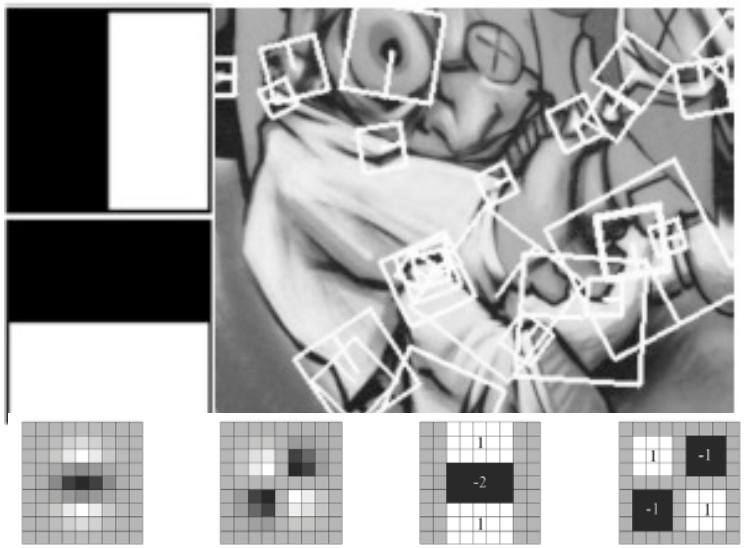
NASA Mars Rover images

Example



NASA Mars Rover images
with SIFT feature matches
Figure by Noah Snavely

Local Descriptors: SURF



Fast approximation of SIFT idea

Efficient computation by 2D box filters & integral images

⇒ 6 times faster than SIFT

Equivalent quality for object identification

GPU implementation available

Feature extraction @ 200Hz (detector + descriptor, 640×480 img)

http://www.vision.ee.ethz.ch/~surf

Many other efficient descriptors are also available

[Bay, ECCV'06], [Cornelis, CVGPU'08]

Local Descriptors: ORB

- Many similarities to SIFT/SURF
- Designed for efficiency and robustness to orientation
- Not designed for scale robustness
- Used for tracking and long-range matching in ORB-SLAM

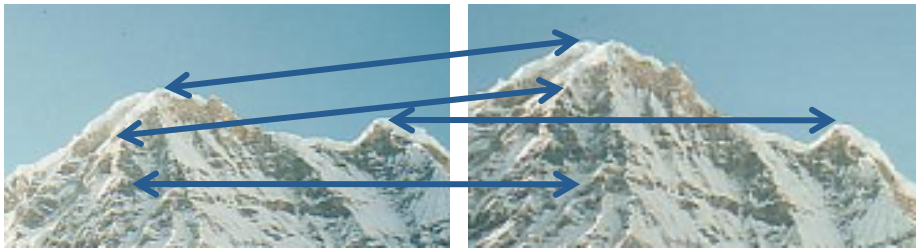
http://www.willowgarage.com/sites/default/files/orb_final.pdf (ICCV 2011)
http://webdiis.unizar.es/~raulmur/orbslam/

Local features: main components

1) Detection: Identify the interest points

2) Description:Extract vector feature descriptor surrounding each interest point.

3) Matching: Determine correspondence between descriptors in two views



Slide credit: Kristen Grauman

Matching local features

Slide credit: Kristen Grauman

Matching local features

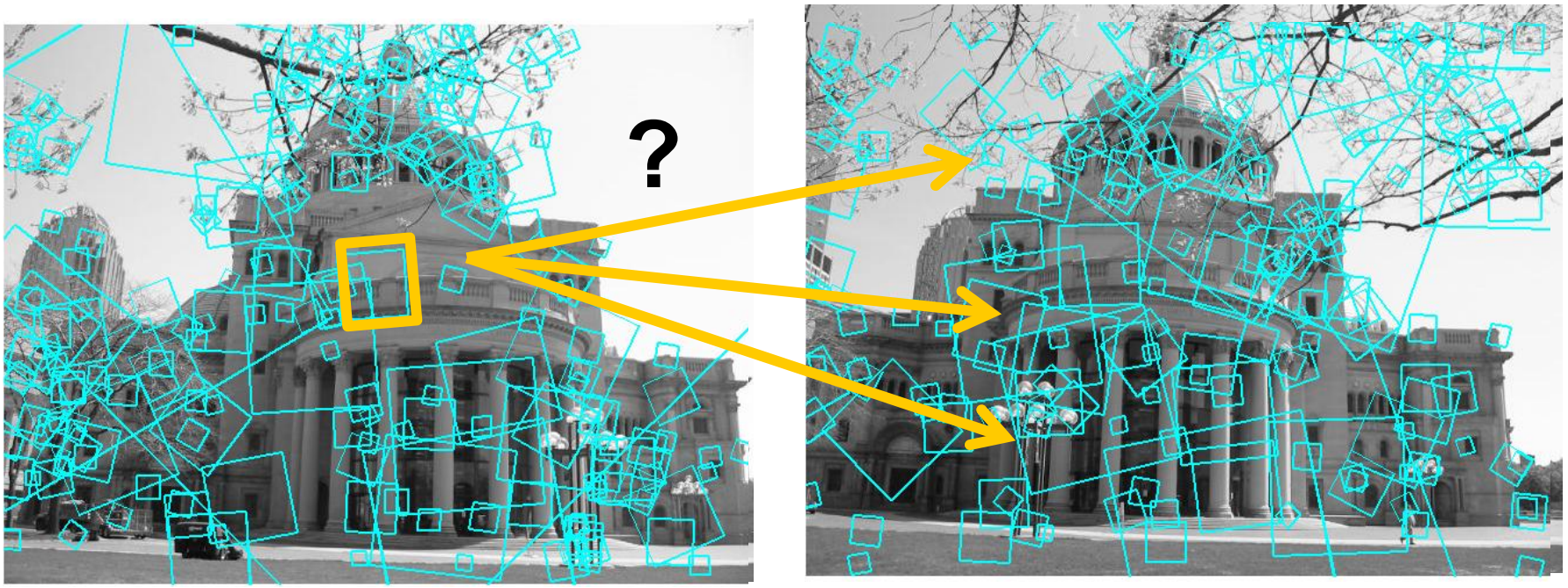


Image 1 Image 2

To generate **candidate matches**, find patches that have the most similar appearance (e.g., lowest SSD)

Simplest approach: compare them all, take the closest (or closest k, or within a thresholded distance)

Slide credit: Kristen Grauman

Ambiguous matches

Image 1

Image 2

At what SSD value do we have a good match?

To add robustness to matching, consider **ratio** :

dist to best match / dist to second best match

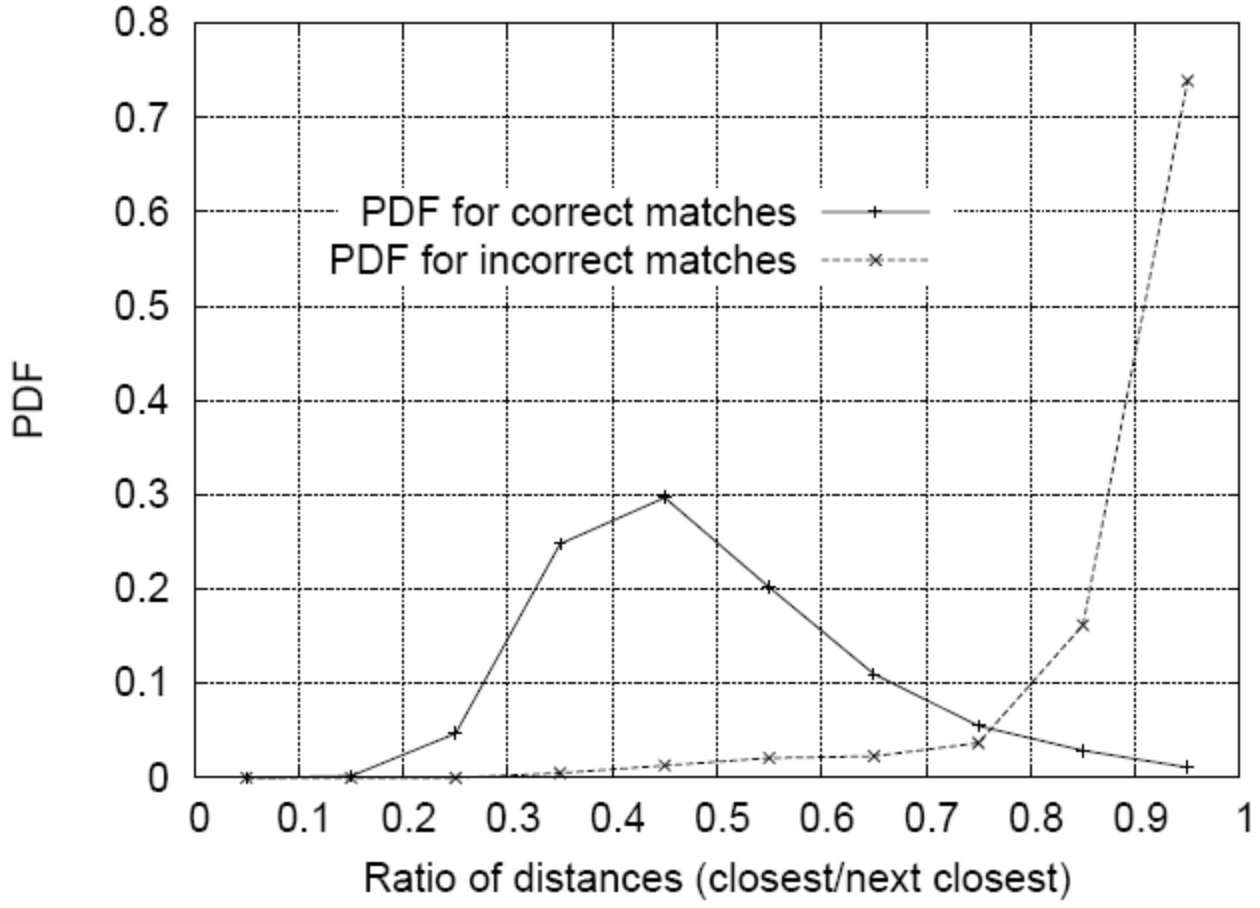
If low, first match looks good.

If high, could be ambiguous match.

Slide credit: Kristen Grauman

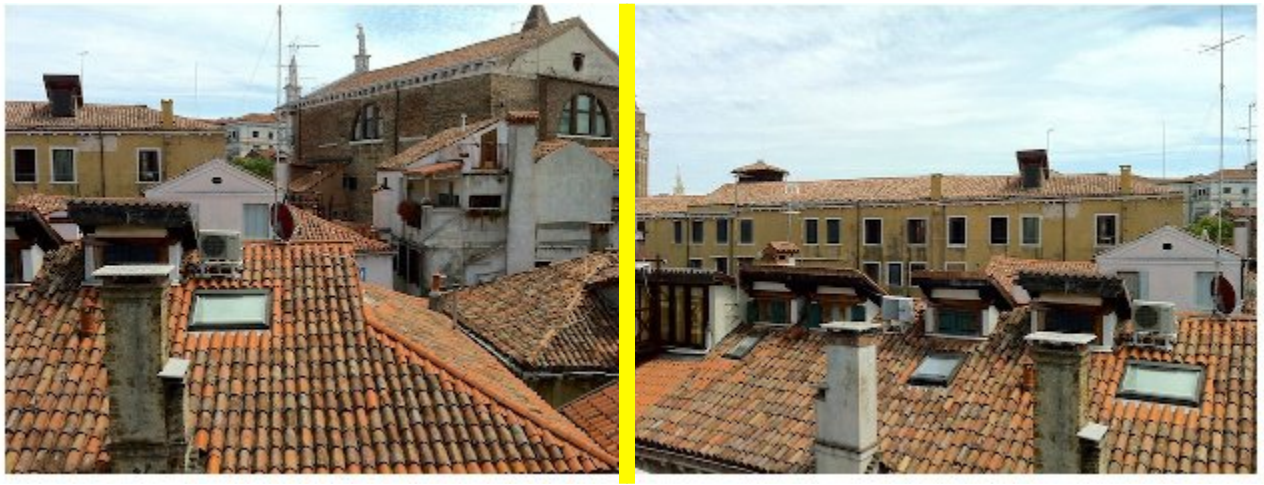
Matching SIFT Descriptors

- Nearest neighbor (Euclidean distance)
- Threshold ratio of nearest to 2nd nearest descriptor

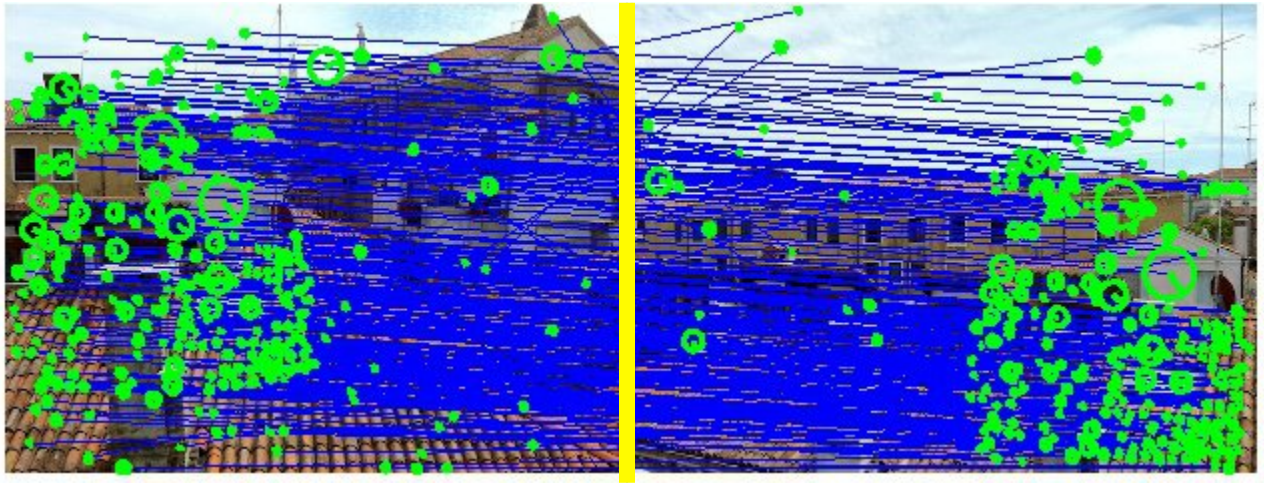


Lowe IJCV 2004

SIFT (preliminary) matches



img1 img2



img1 img2

http://www.vlfeat.org/overview/sift.html

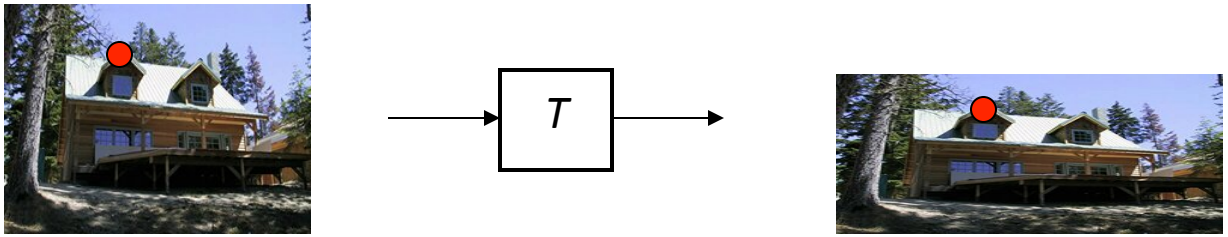
Image alignment

- Alignment: find parameters of model that maps one set of points to another

- Typically want to solve for a global transformation that accounts for most true correspondences

- Difficulties
  - Noise (typically 1-3 pixels)
  - Outliers (often 30-50%)
  - Many-to-one matches or multiple objects

Parametric (global) warping



p = (x,y)

p' = (x',y')

Transformation T is a coordinate-changing function:

$$p' = T(p)$$

What does it mean that *T* is global?

- Is the same for any point p
- can be described by just a few numbers (parameters)

For linear transformations, we can represent T as a matrix

$$p' = \mathbf{T}p$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

Common transformations

original

Transformed

translation

rotation

aspect

affine

perspective

Slide credit A. Efros and/or S. Seitz