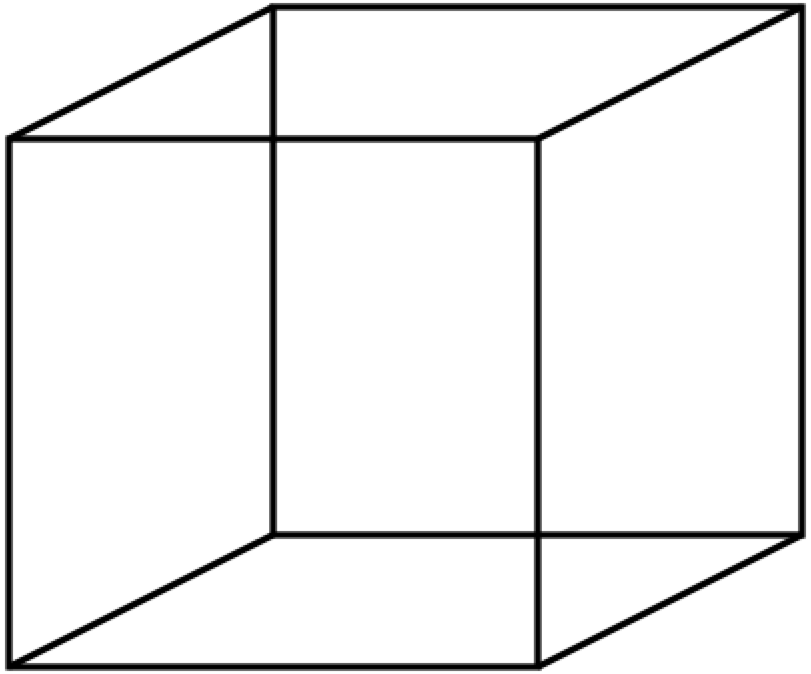
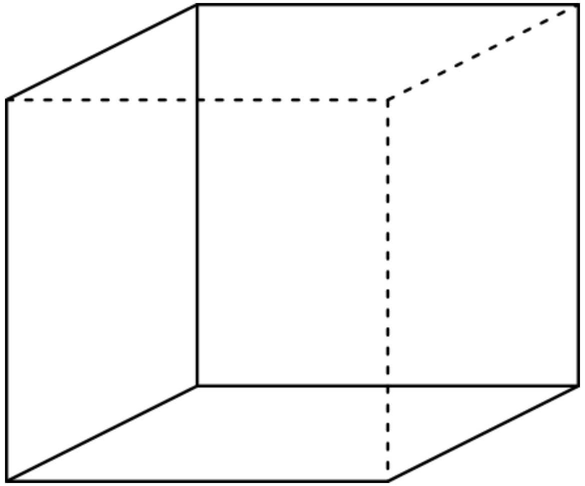
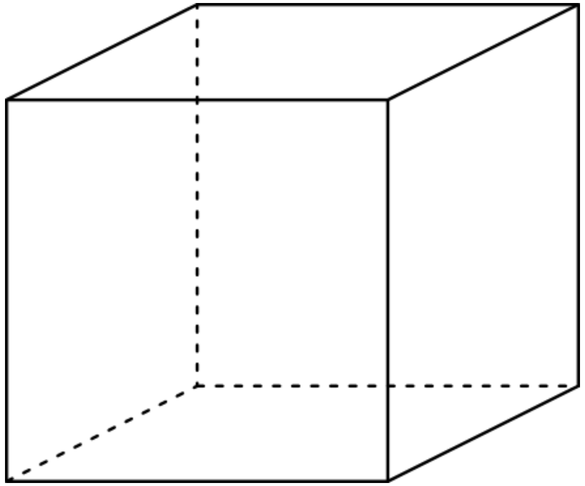
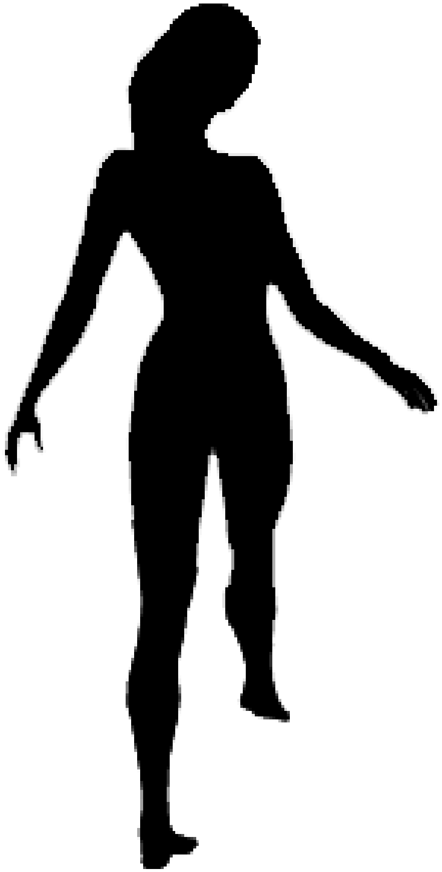
Multi-stable Perception



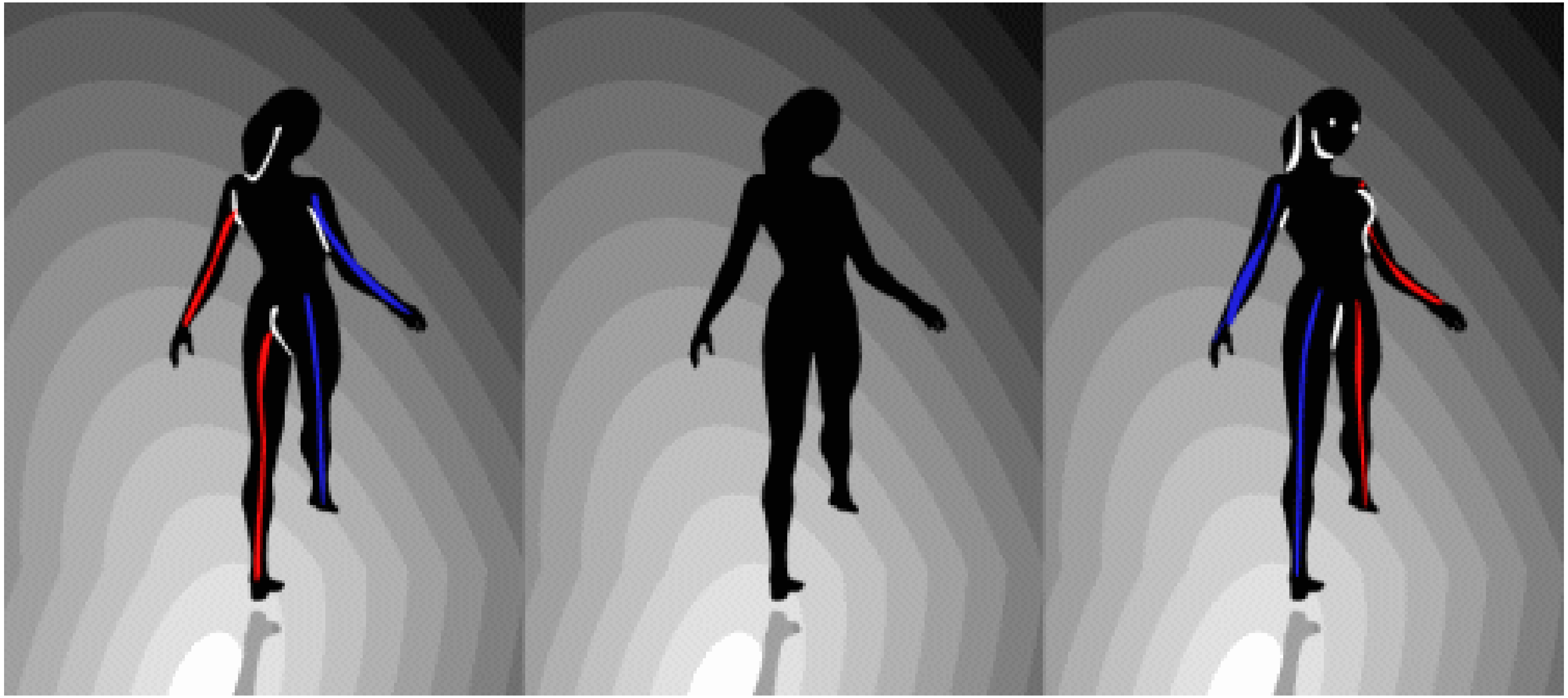
Necker Cube







Spinning dancer illusion, Nobuyuki Kayahara



Feature Matching and Robust Fitting

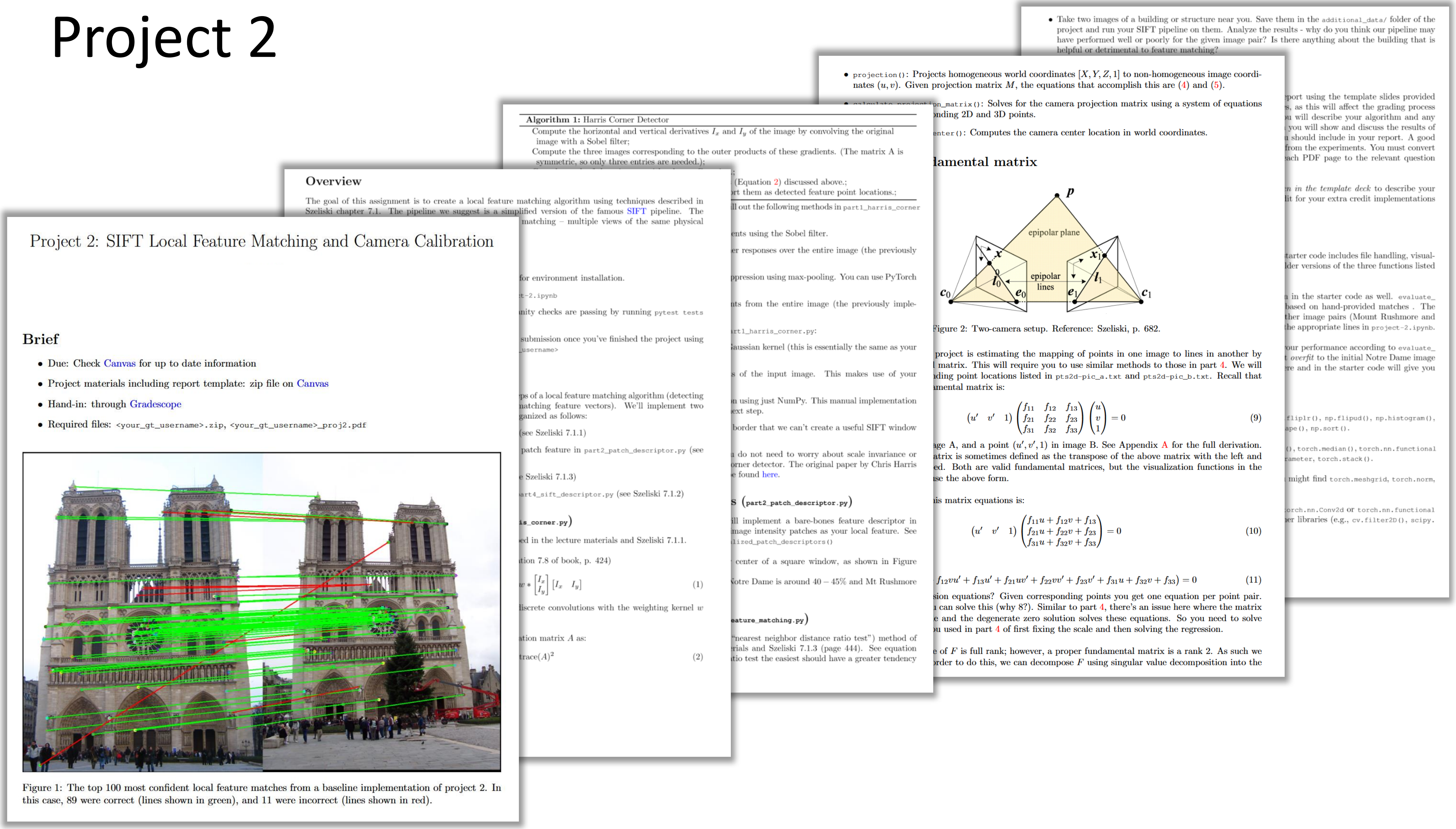
Read Szeliski 7.4.2 and 2.1

Computer Vision

James Hays

Acknowledgment: Many slides from Derek Hoiem and Grauman&Leibe 2008 AAAI Tutorial

Project 2



Project 2: SIFT Local Feature Matching and Camera Calibration

Brief

- Due: Check Canvas for up to date information
- Project materials including report template: zip file on Canvas
- Hand-in: through Gradescope
- Required files: `<your_gt_username>.zip`, `<your_gt_username>_proj2.pdf`

Figure 1: The top 100 most confident local feature matches from a baseline implementation of project 2. In this case, 89 were correct (lines shown in green), and 11 were incorrect (lines shown in red).

Overview

The goal of this assignment is to create a local feature matching algorithm using techniques described in Szeliski chapter 7.1. The pipeline we suggest is a simplified version of the famous SIFT pipeline. The

matching – multiple views of the same physical

for environment installation.

`t-2.ipynb`

nity checks are passing by running `pytest tests`

submission once you've finished the project using

`_username>`

ps of a local feature matching algorithm (detecting

natching feature vectors). We'll implement two

ganized as follows:

(see Szeliski 7.1.1)

patch feature in `part2_patch_descriptor.py` (see

e Szeliski 7.1.3)

`art4_sift_descriptor.py` (see Szeliski 7.1.2)

`is_corner.py`)

ed in the lecture materials and Szeliski 7.1.1.

tion 7.8 of book, p. 424)

$$w * \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} \qquad (1)$$

liscrete convolutions with the weighting kernel w

ation matrix A as:

$$\mathrm{trace}(A)^2 \qquad (2)$$

Algorithm 1: Harris Corner Detector

```
Compute the horizontal and vertical derivatives $I_x$ and $I_y$ of the image by convolving the original
  image with a Sobel filter;
Compute the three images corresponding to the outer products of these gradients. (The matrix A is
  symmetric, so only three entries are needed.);
(Equation 2) discussed above.;
rt them as detected feature point locations.;
```

ill out the following methods in `part1_harris_corner`

ents using the Sobel filter.

er responses over the entire image (the previously

ppression using max-pooling. You can use PyTorch

nts from the entire image (the previously imple-

`art1_harris_corner.py`:

aussian kernel (this is essentially the same as your

s of the input image. This makes use of your

n using just NumPy. This manual implementation

ext step.

border that we can't create a useful SIFT window

u do not need to worry about scale invariance or

orner detector. The original paper by Chris Harris

e found here.

s (`part2_patch_descriptor.py`)

ill implement a bare-bones feature descriptor in

image intensity patches as your local feature. See

`lized_patch_descriptors()`

center of a square window, as shown in Figure

Notre Dame is around 40 – 45% and Mt Rushmore

`eature_matching.py`)

"nearest neighbor distance ratio test") method of

rials and Szeliski 7.1.3 (page 444). See equation

tio test the easiest should have a greater tendency

- `projection()`: Projects homogeneous world coordinates $[X, Y, Z, 1]$ to non-homogeneous image coordinates (u, v). Given projection matrix M, the equations that accomplish this are (4) and (5).
- `on_matrix()`: Solves for the camera projection matrix using a system of equations
nding 2D and 3D points.
- `enter()`: Computes the camera center location in world coordinates.

damental matrix

Figure 2: Two-camera setup. Reference: Szeliski, p. 682.

project is estimating the mapping of points in one image to lines in another by

l matrix. This will require you to use similar methods to those in part 4. We will

nding point locations listed in `pts2d-pic_a.txt` and `pts2d-pic_b.txt`. Recall that

mental matrix is:

$$\begin{pmatrix} u' & v' & 1 \end{pmatrix} \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = 0 \qquad (9)$$

age A, and a point $(u', v', 1)$ in image B. See Appendix A for the full derivation.

atrix is sometimes defined as the transpose of the above matrix with the left and

ed. Both are valid fundamental matrices, but the visualization functions in the

se the above form.

is matrix equations is:

$$\begin{pmatrix} u' & v' & 1 \end{pmatrix} \begin{pmatrix} f_{11}u + f_{12}v + f_{13} \\ f_{21}u + f_{22}v + f_{23} \\ f_{31}u + f_{32}v + f_{33} \end{pmatrix} = 0 \qquad (10)$$

$$f_{12}vu' + f_{13}u' + f_{21}uv' + f_{22}vv' + f_{23}v' + f_{31}u + f_{32}v + f_{33}) = 0 \qquad (11)$$

sion equations? Given corresponding points you get one equation per point pair.

u can solve this (why 8?). Similar to part 4, there's an issue here where the matrix

e and the degenerate zero solution solves these equations. So you need to solve

ou used in part 4 of first fixing the scale and then solving the regression.

e of F is full rank; however, a proper fundamental matrix is a rank 2. As such we

order to do this, we can decompose F using singular value decomposition into the

- Take two images of a building or structure near you. Save them in the `additional_data/` folder of the project and run your SIFT pipeline on them. Analyze the results - why do you think our pipeline may have performed well or poorly for the given image pair? Is there anything about the building that is helpful or detrimental to feature matching?

port using the template slides provided

s, as this will affect the grading process

u will describe your algorithm and any

you will show and discuss the results of

u should include in your report. A good

from the experiments. You must convert

ach PDF page to the relevant question

n in the template deck to describe your

lit for your extra credit implementations

tarter code includes file handling, visual-

lder versions of the three functions listed

n in the starter code as well. `evaluate_`

based on hand-provided matches . The

ther image pairs (Mount Rushmore and

the appropriate lines in `project-2.ipynb`.

our performance according to `evaluate_`

t *overfit* to the initial Notre Dame image

re and in the starter code will give you

`fliplr()`, `np.flipud()`, `np.histogram()`,

`ape()`, `np.sort()`.

`()`, `torch.median()`, `torch.nn.functional`

`rameter`, `torch.stack()`.

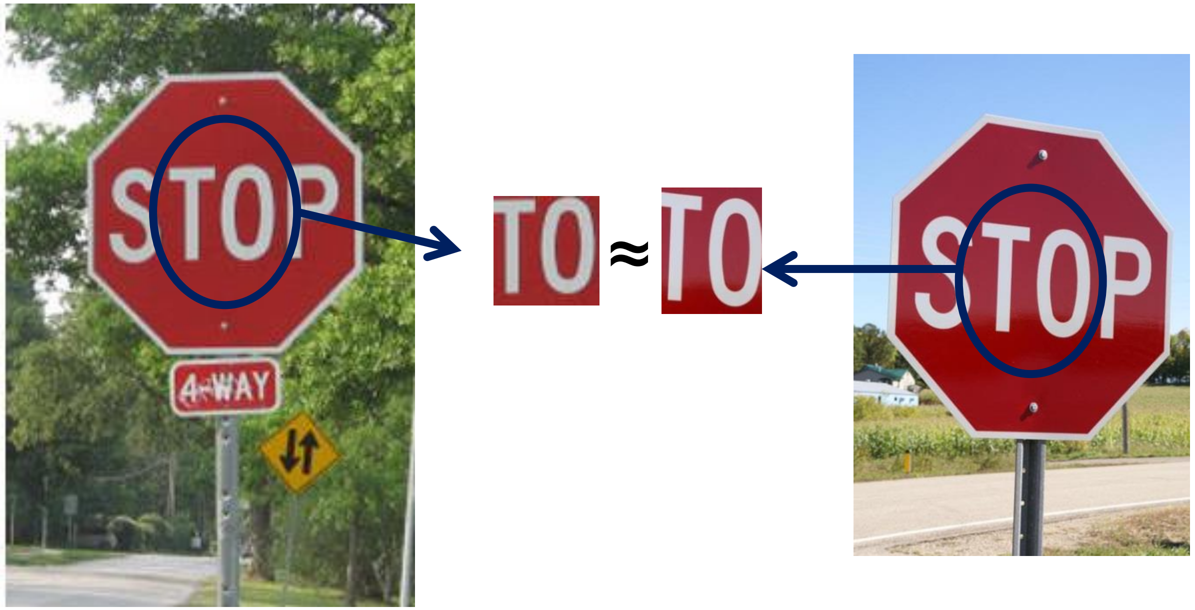
might find `torch.meshgrid`, `torch.norm`,

`orch.nn.Conv2d` or `torch.nn.functional`

er libraries (e.g., `cv.filter2D()`, `scipy.`

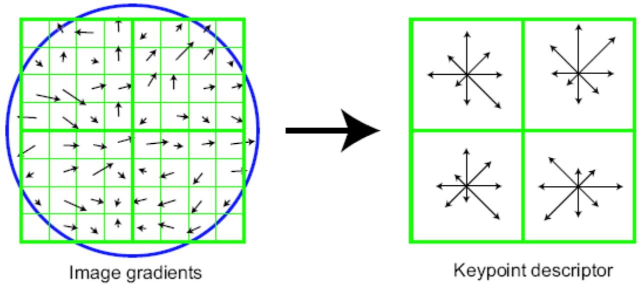
This section: correspondence and alignment

- Correspondence: matching points, patches, edges, or regions across images



Review: Local Descriptors

- Most features can be thought of as templates, histograms (counts), or combinations
- The ideal descriptor should be
  - Robust and Distinctive
  - Compact and Efficient



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

K. Grauman, B. Leibe

How lossy is this? Can we invert SIFT descriptors?

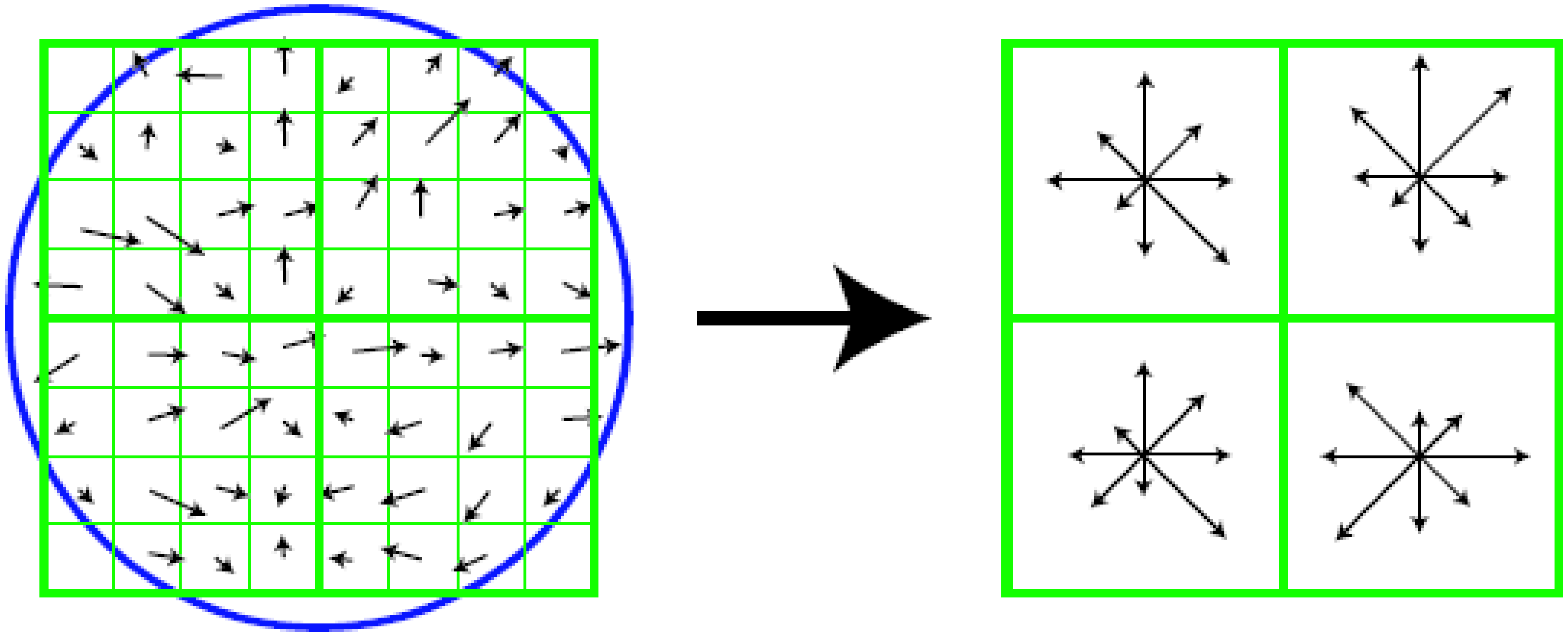
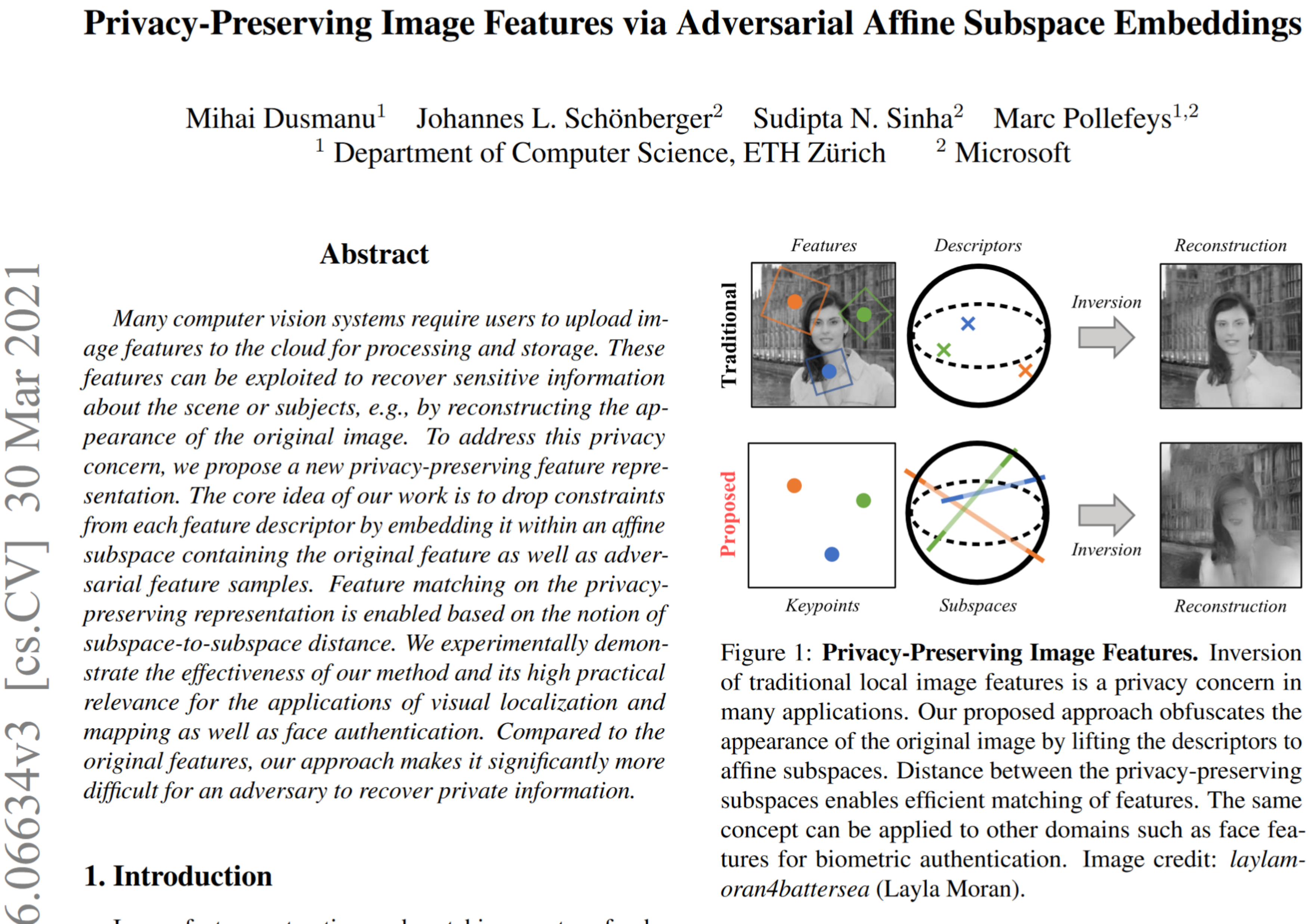


Image gradients

Keypoint descriptor

Can we invert SIFT descriptors?



Privacy-Preserving Image Features via Adversarial Affine Subspace Embeddings

Mihai Dusmanu[1] Johannes L. Schönberger[2] Sudipta N. Sinha[2] Marc Pollefeys[1,2]
[1] Department of Computer Science, ETH Zürich [2] Microsoft

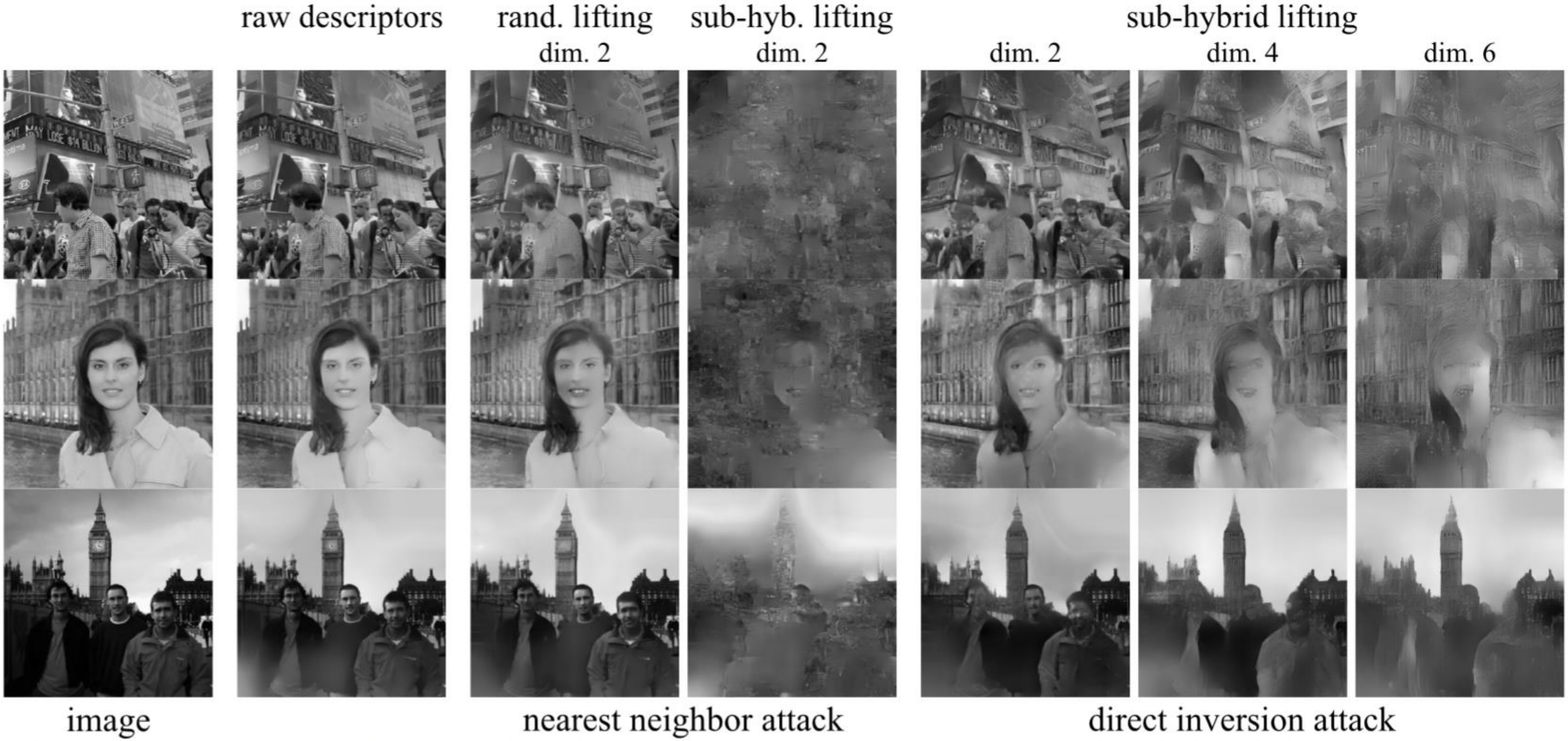
Abstract

Many computer vision systems require users to upload image features to the cloud for processing and storage. These features can be exploited to recover sensitive information about the scene or subjects, e.g., by reconstructing the appearance of the original image. To address this privacy concern, we propose a new privacy-preserving feature representation. The core idea of our work is to drop constraints from each feature descriptor by embedding it within an affine subspace containing the original feature as well as adversarial feature samples. Feature matching on the privacy-preserving representation is enabled based on the notion of subspace-to-subspace distance. We experimentally demonstrate the effectiveness of our method and its high practical relevance for the applications of visual localization and mapping as well as face authentication. Compared to the original features, our approach makes it significantly more difficult for an adversary to recover private information.

1. Introduction

Figure 1: **Privacy-Preserving Image Features.** Inversion of traditional local image features is a privacy concern in many applications. Our proposed approach obfuscates the appearance of the original image by lifting the descriptors to affine subspaces. Distance between the privacy-preserving subspaces enables efficient matching of features. The same concept can be applied to other domains such as face features for biometric authentication. Image credit: *laylamoran4battersea* (Layla Moran).

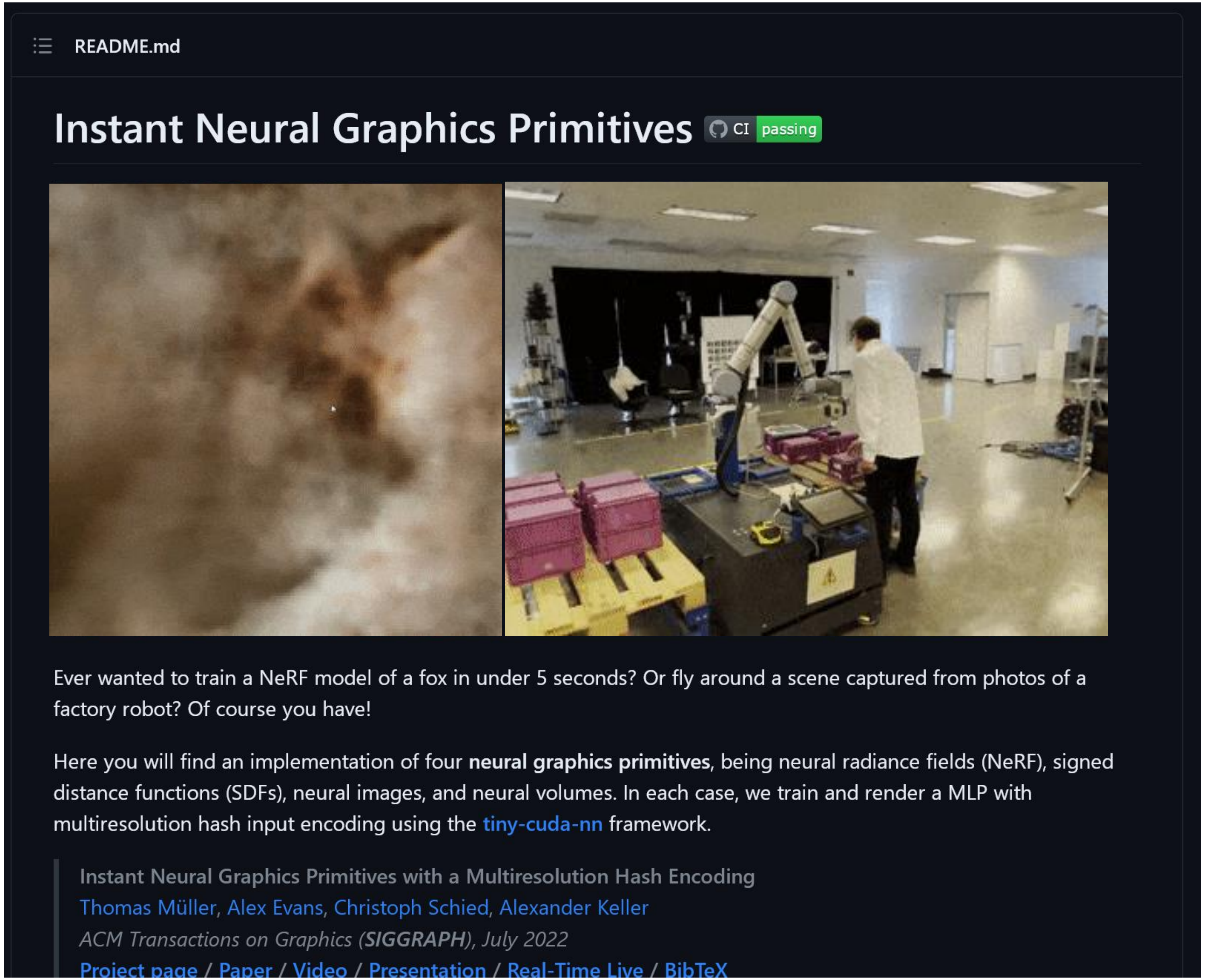
Can we invert SIFT descriptors?



SIFT is 20+ years old. Is it still useful?

SIFT is 20+ years old. Is it still useful?

- Let’s look at some trendy research on Neural Radiance Fields (NeRF)



README.md

Instant Neural Graphics Primitives

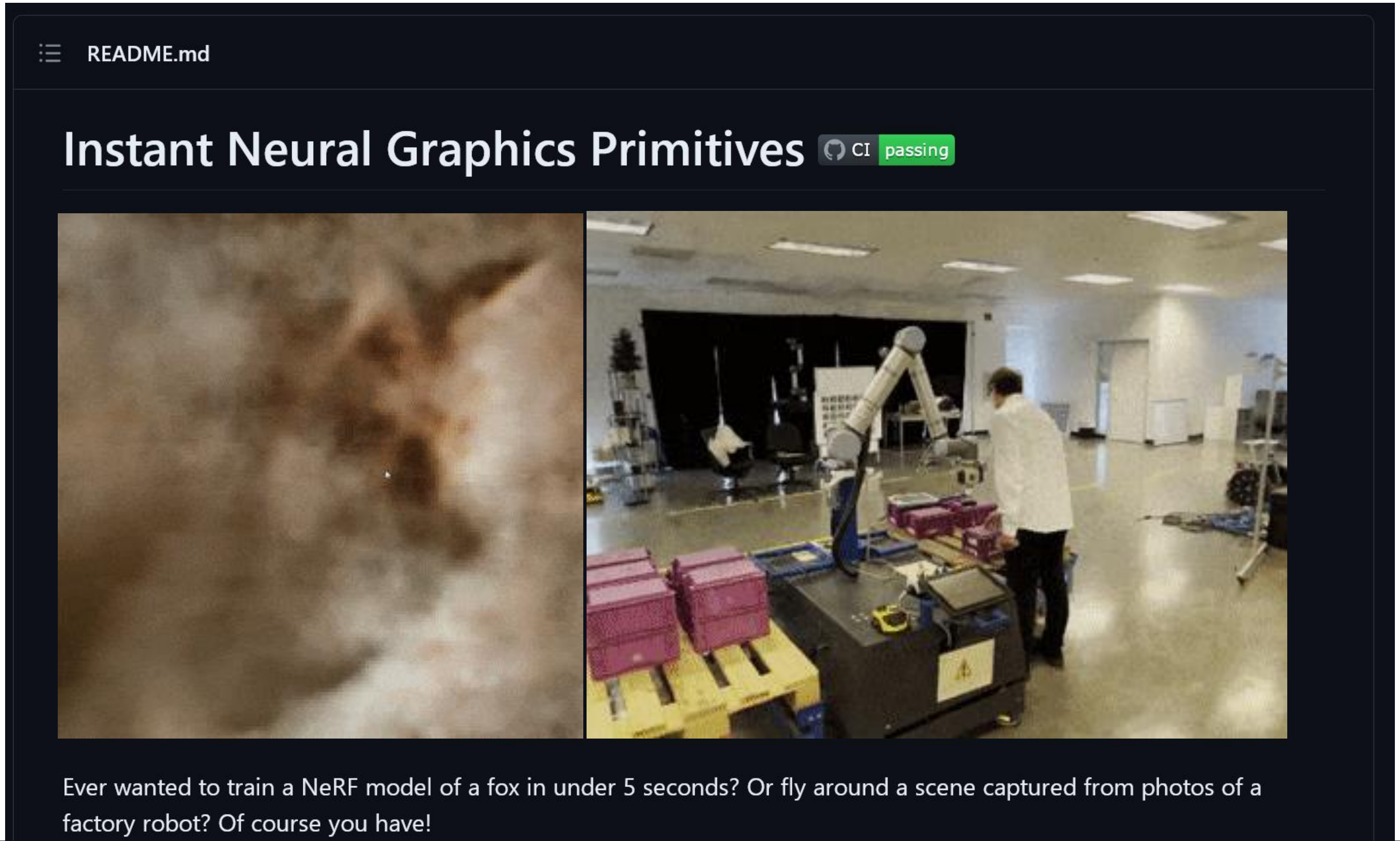
Ever wanted to train a NeRF model of a fox in under 5 seconds? Or fly around a scene captured from photos of a factory robot? Of course you have!

Here you will find an implementation of four **neural graphics primitives**, being neural radiance fields (NeRF), signed distance functions (SDFs), neural images, and neural volumes. In each case, we train and render a MLP with multiresolution hash input encoding using the tiny-cuda-nn framework.

> Instant Neural Graphics Primitives with a Multiresolution Hash Encoding
> Thomas Müller, Alex Evans, Christoph Schied, Alexander Keller
> *ACM Transactions on Graphics (**SIGGRAPH**), July 2022*
> Project page / Paper / Video / Presentation / Real-Time Live / BibTeX

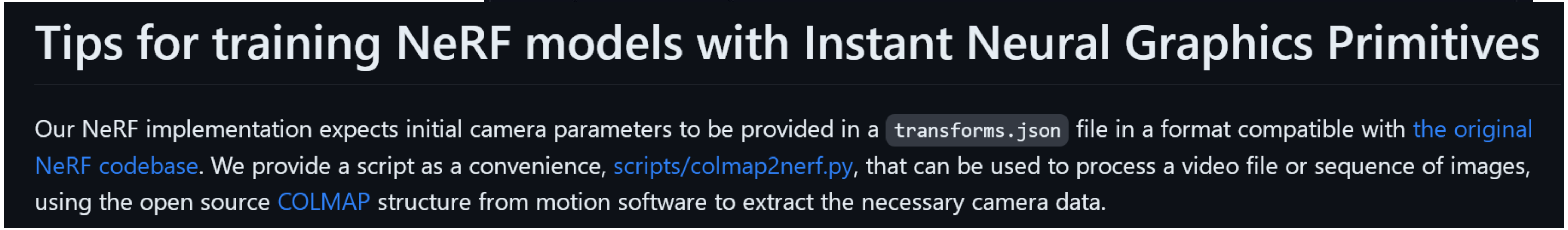
SIFT is 20+ years old. Is it still useful?

- Let’s look at some trendy research on Neural Radiance Fields (NeRF)
- Let’s look under the hood



Instant Neural Graphics Primitives

Ever wanted to train a NeRF model of a fox in under 5 seconds? Or fly around a scene captured from photos of a factory robot? Of course you have!

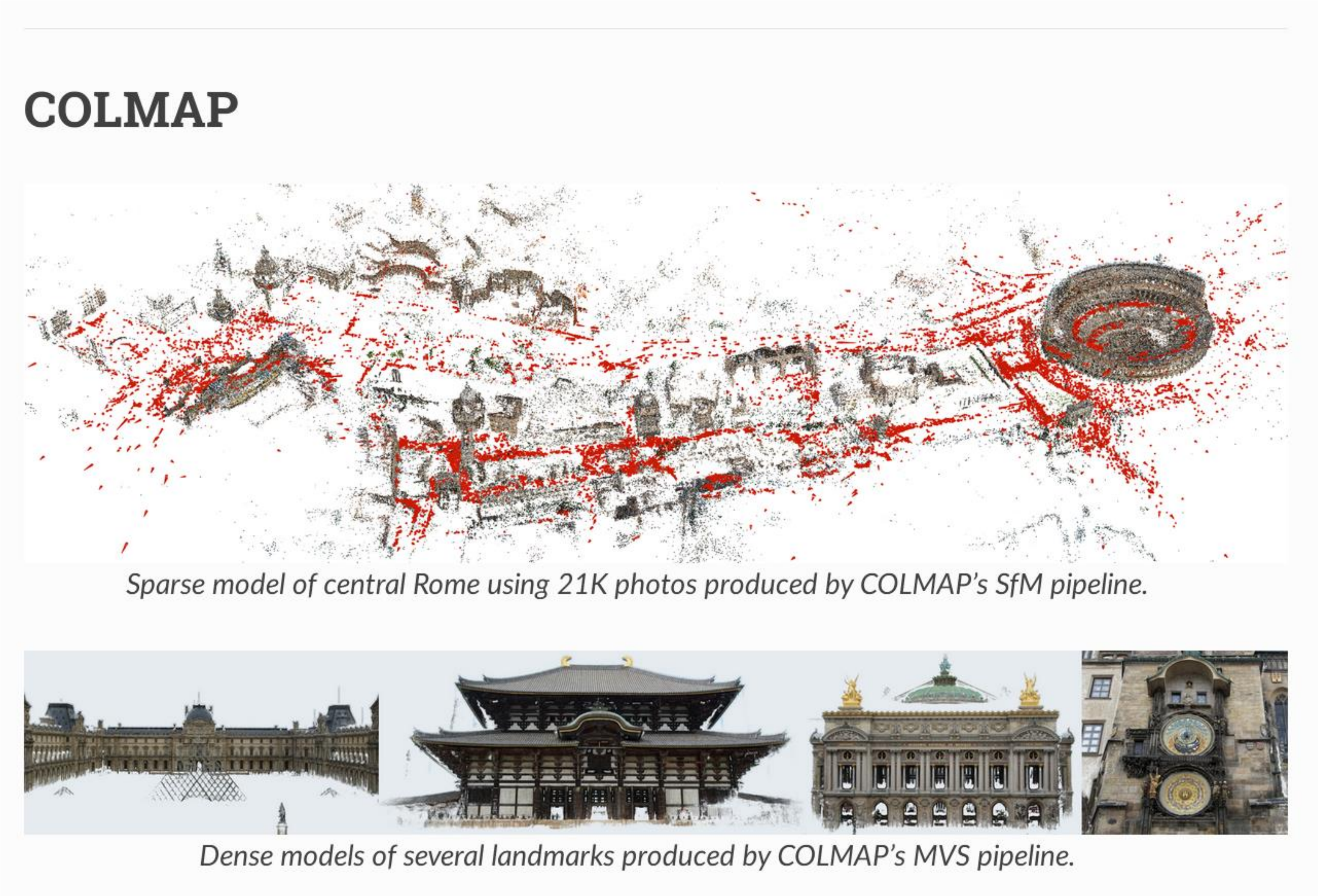


Tips for training NeRF models with Instant Neural Graphics Primitives

Our NeRF implementation expects initial camera parameters to be provided in a `transforms.json` file in a format compatible with the original NeRF codebase. We provide a script as a convenience, scripts/colmap2nerf.py, that can be used to process a video file or sequence of images, using the open source COLMAP structure from motion software to extract the necessary camera data.

SIFT is 20+ years old. Is it still useful?

- COLMAP is the "standard" way to do structure from motion these days



COLMAP

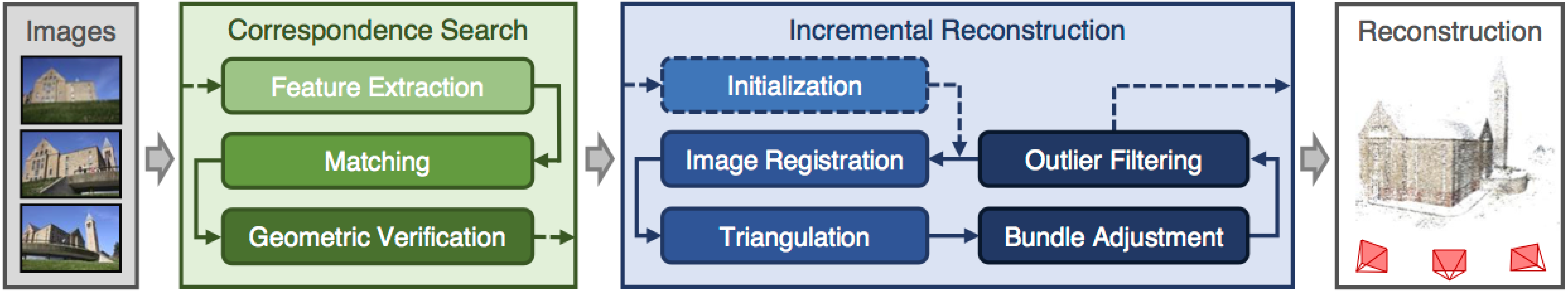
Sparse model of central Rome using 21K photos produced by COLMAP's SfM pipeline.

Dense models of several landmarks produced by COLMAP's MVS pipeline.

"Structure-From-Motion Revisited". Johannes L. Schonberger, Jan-Michael Frahm; CVPR 2016
5k+ citations

SIFT is 20+ years old. Is it still useful?

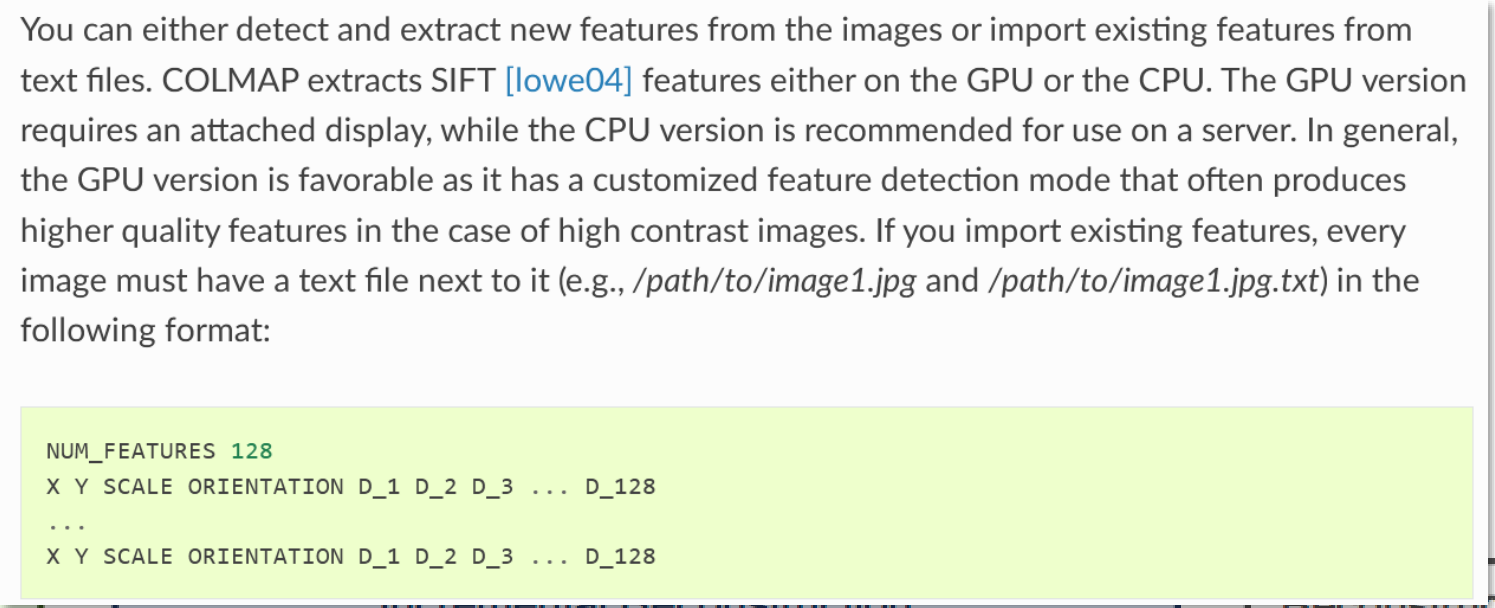
- COLMAP is the "standard" way to do structure from motion these days



"Structure-From-Motion Revisited". Johannes L. Schonberger, Jan-Michael Frahm; CVPR 2016
5k+ citations

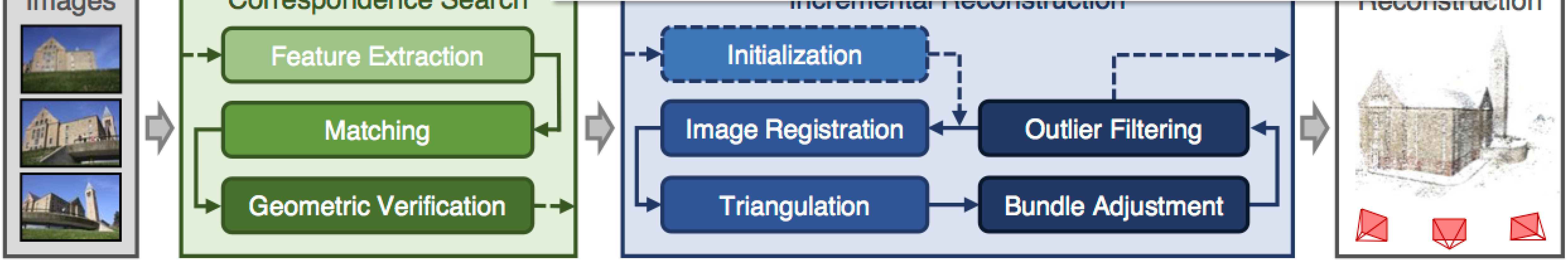
SIFT is 20+ years old. Is it still useful?

- COLMAP is the "standard" way to do structure from motion these days



You can either detect and extract new features from the images or import existing features from text files. COLMAP extracts SIFT [lowe04] features either on the GPU or the CPU. The GPU version requires an attached display, while the CPU version is recommended for use on a server. In general, the GPU version is favorable as it has a customized feature detection mode that often produces higher quality features in the case of high contrast images. If you import existing features, every image must have a text file next to it (e.g., */path/to/image1.jpg* and */path/to/image1.jpg.txt*) in the following format:

```
NUM_FEATURES 128
X Y SCALE ORIENTATION D_1 D_2 D_3 ... D_128
...
X Y SCALE ORIENTATION D_1 D_2 D_3 ... D_128
```



"Structure-From-Motion Revisited". Johannes L. Schonberger, Jan-Michael Frahm; CVPR 2016
5k+ citations

Distributed Global Structure-from-Motion with a Deep Front-End

Ayush Baid *† John Lambert*† Travis Driver* Akshay Krishnan*
Hayk Stepanyan Frank Dellaert
Georgia Tech

Abstract

While initial approaches to Structure-from-Motion (SfM) revolved around both global and incremental methods, most recent applications rely on incremental systems to estimate camera poses due to their superior robustness. Though there has been tremendous progress in SfM 'front-ends' powered by deep models learned from data, the state-of-the-art (incremental) SfM pipelines still rely on classical SIFT features, developed in 2004. In this work, we investigate whether leveraging the developments in feature extraction and matching helps global SfM perform on par with the SOTA incremental SfM approach (COLMAP). To do so, we design a modular SfM framework that allows us to easily combine developments in different stages of the SfM pipeline. Our experiments show that while developments in deep-learning based two-view correspondence estimation do translate to improvements in point density for scenes reconstructed with global SfM, none of them outperform SIFT when comparing with incremental SfM results on a range of datasets. Our SfM system is designed from the ground up to leverage distributed computation, enabling us to parallelize computation on multiple machines and scale to large scenes. Our code is publicly available at github.com/borglab/gtsfm.

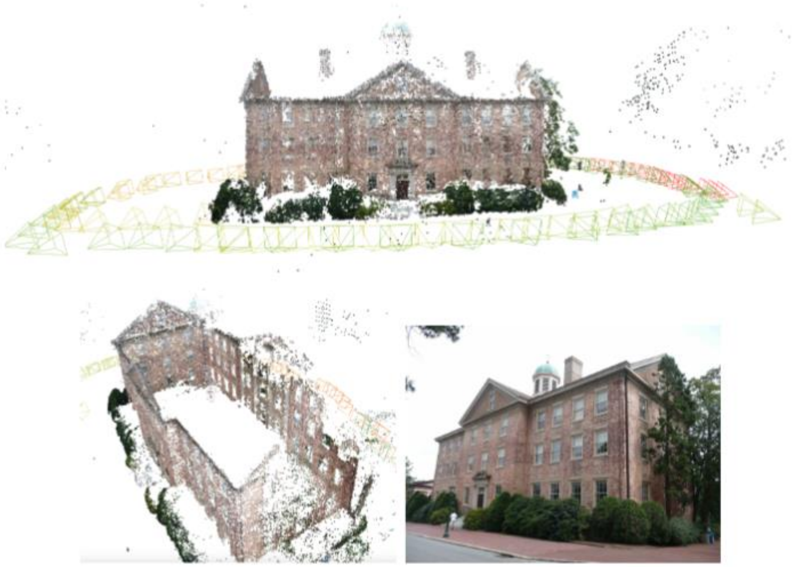


Figure 1. A sparse reconstruction of the UNC South Building using GTSfM with a deep LoFTR-based [64] front-end, with an example image input. Multi-view stereo is not used.

[53, 57], Gaussian Splatting [32], accurate monocular depth predictions for humans [39], and more.

Incremental SfM is the dominant paradigm, as global SfM suffers from a lack of accuracy, largely due to difficulty in reasoning about outliers globally in a single pass. However, to our knowledge, almost all global SfM systems

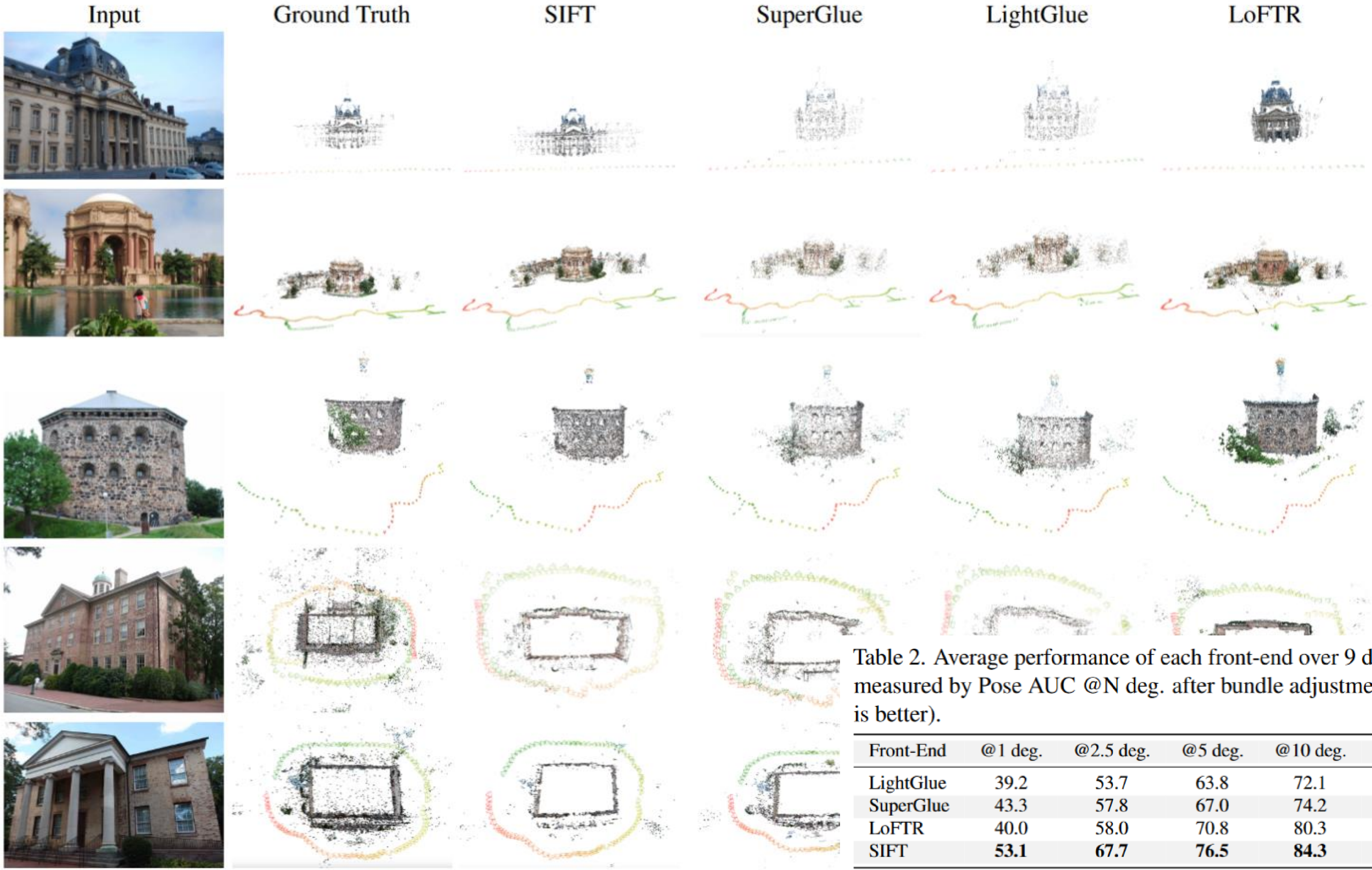
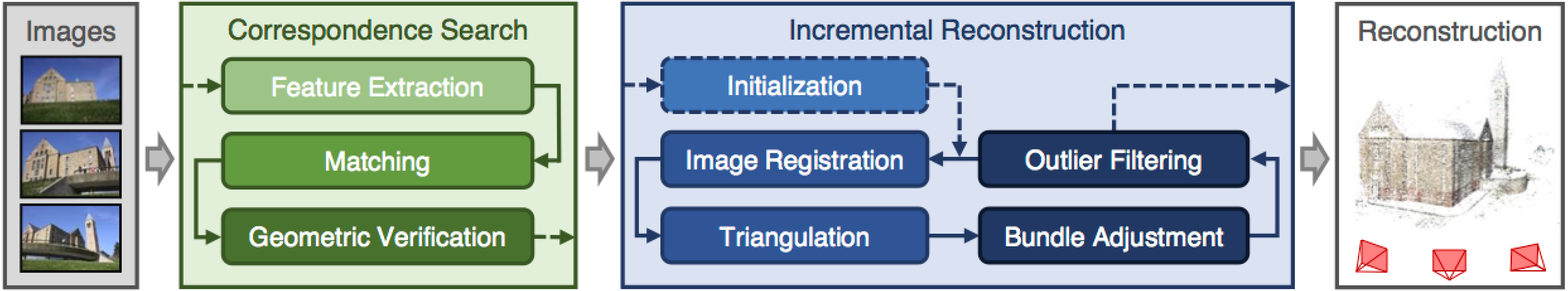


Table 2. Average performance of each front-end over 9 datasets, as measured by Pose AUC @N deg. after bundle adjustment (higher is better).

Front-End	@1 deg.	@2.5 deg.	@5 deg.	@10 deg.	@20 deg.
LightGlue	39.2	53.7	63.8	72.1	77.9
SuperGlue	43.3	57.8	67.0	74.2	79.0
LoFTR	40.0	58.0	70.8	80.3	86.2
SIFT	**53.1**	**67.7**	**76.5**	**84.3**	**90.3**

What is “Geometric Verification”?

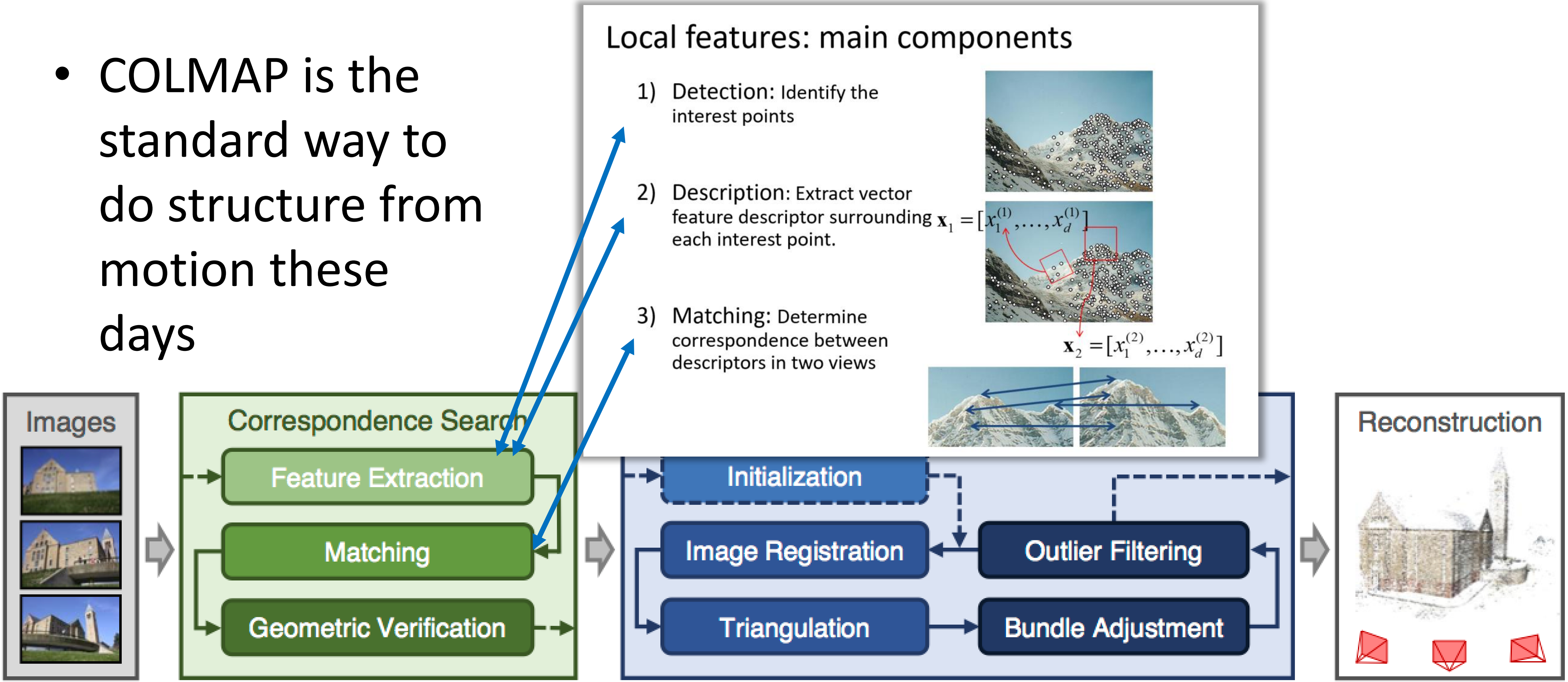
- COLMAP is the standard way to do structure from motion these days



“Structure-From-Motion Revisited”. Johannes L. Schonberger, Jan-Michael Frahm; CVPR 2016
5k+ citations

What is “Geometric Verification”?

- COLMAP is the standard way to do structure from motion these days



Local features: main components

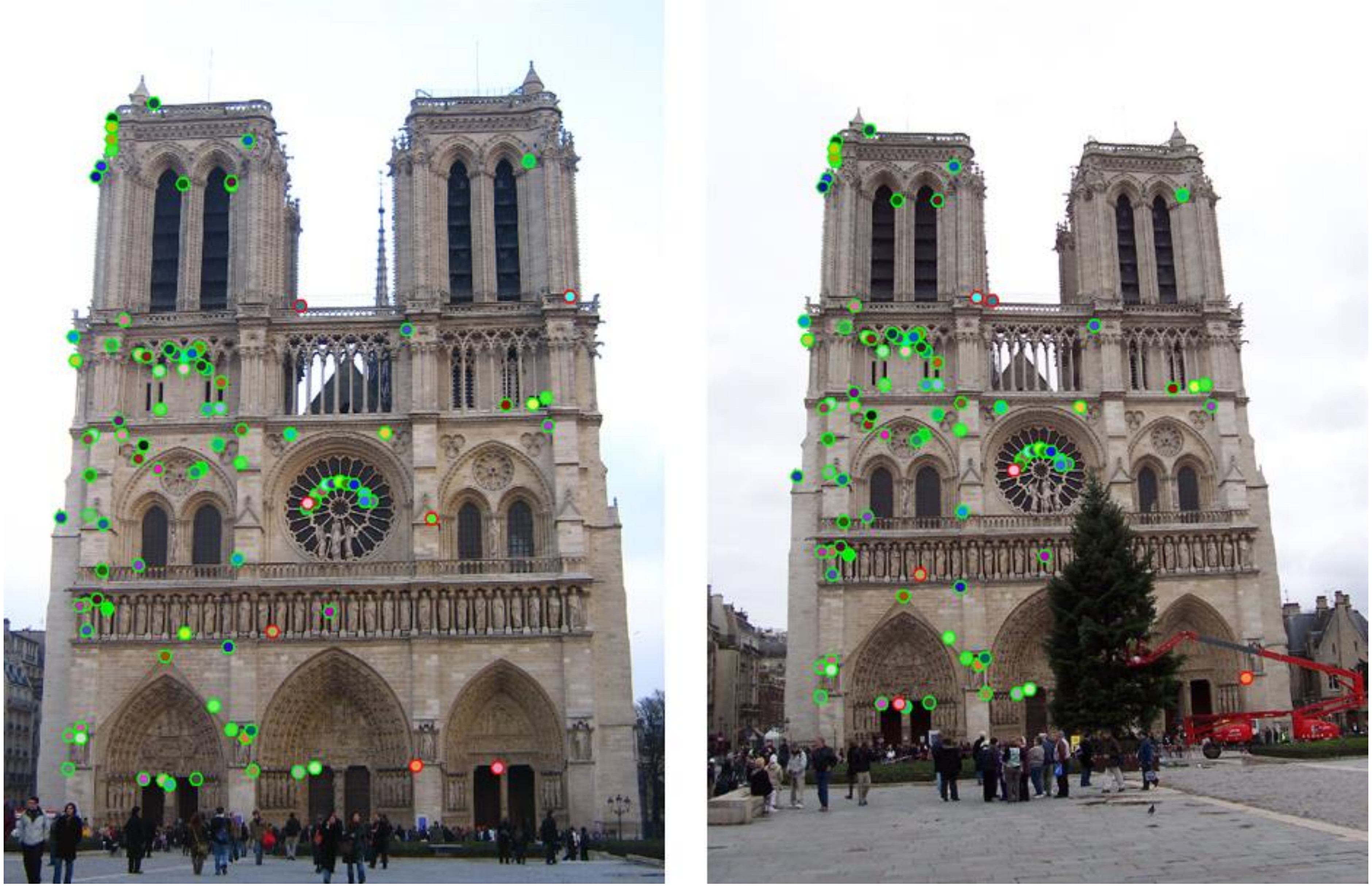
1) Detection: Identify the interest points
2) Description: Extract vector feature descriptor surrounding each interest point.
3) Matching: Determine correspondence between descriptors in two views

$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$

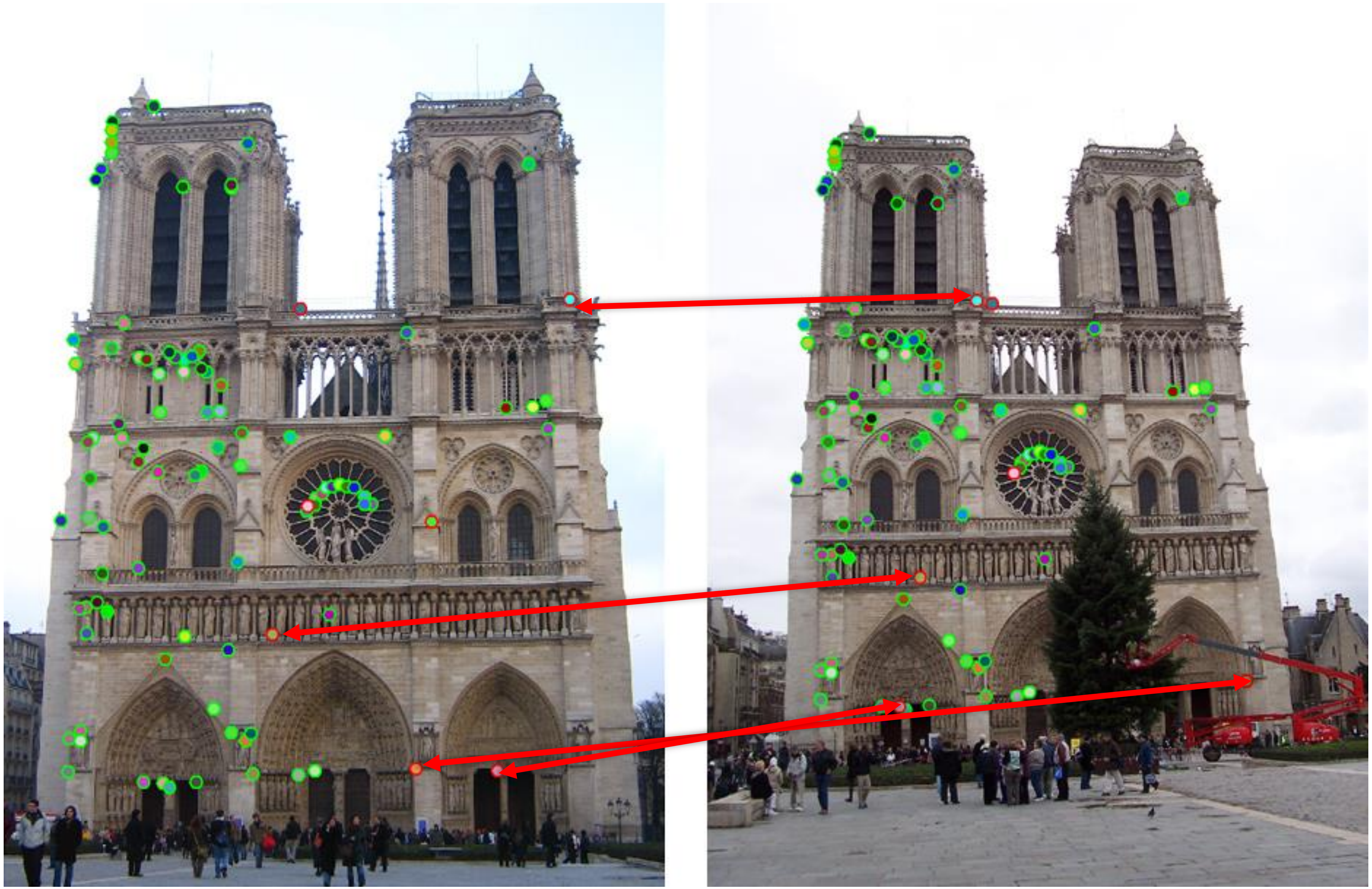
$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$

“Structure-From-Motion Revisited”. Johannes L. Schonberger, Jan-Michael Frahm; CVPR 2016
5k+ citations

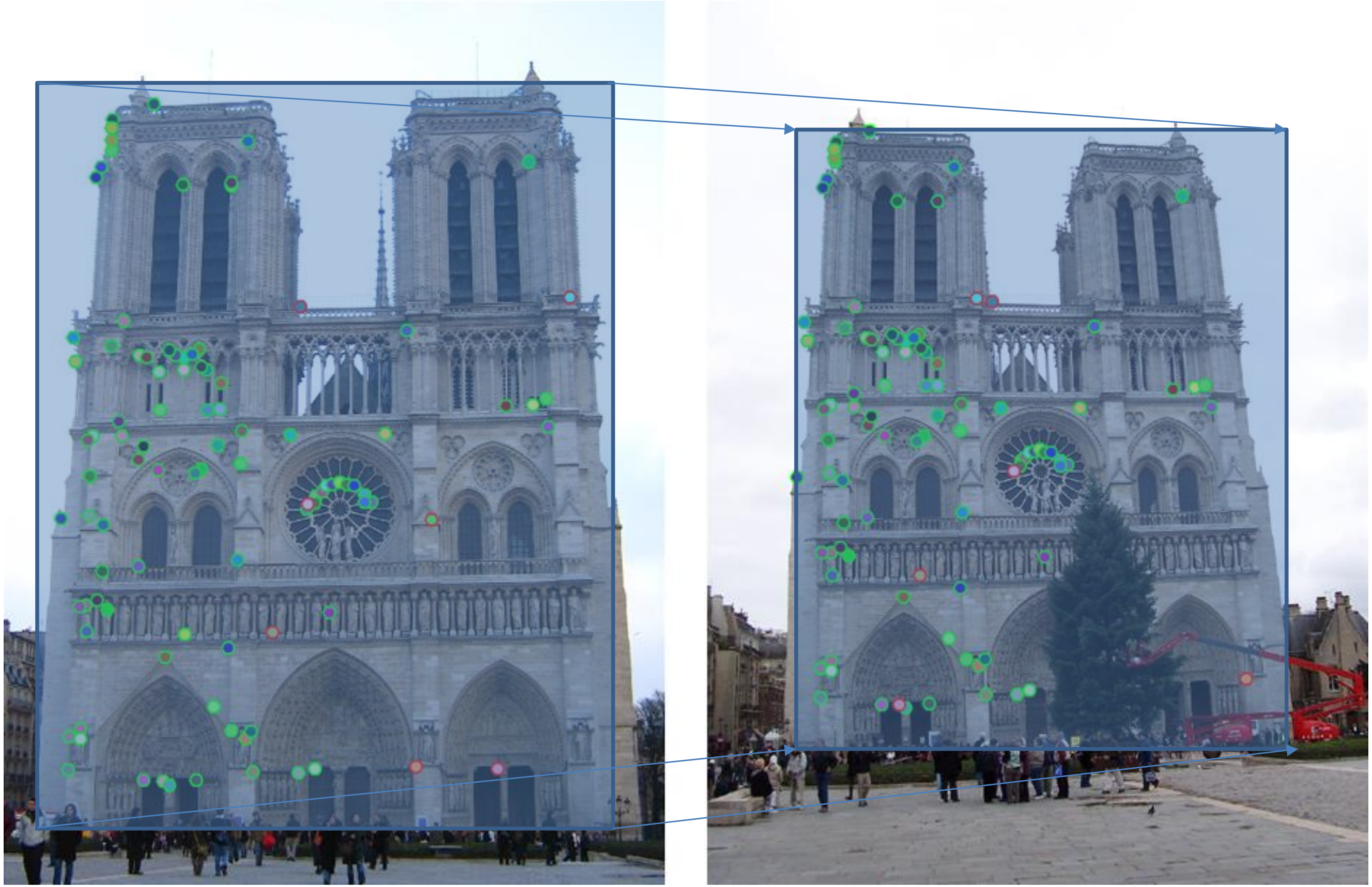
Can we refine this further?



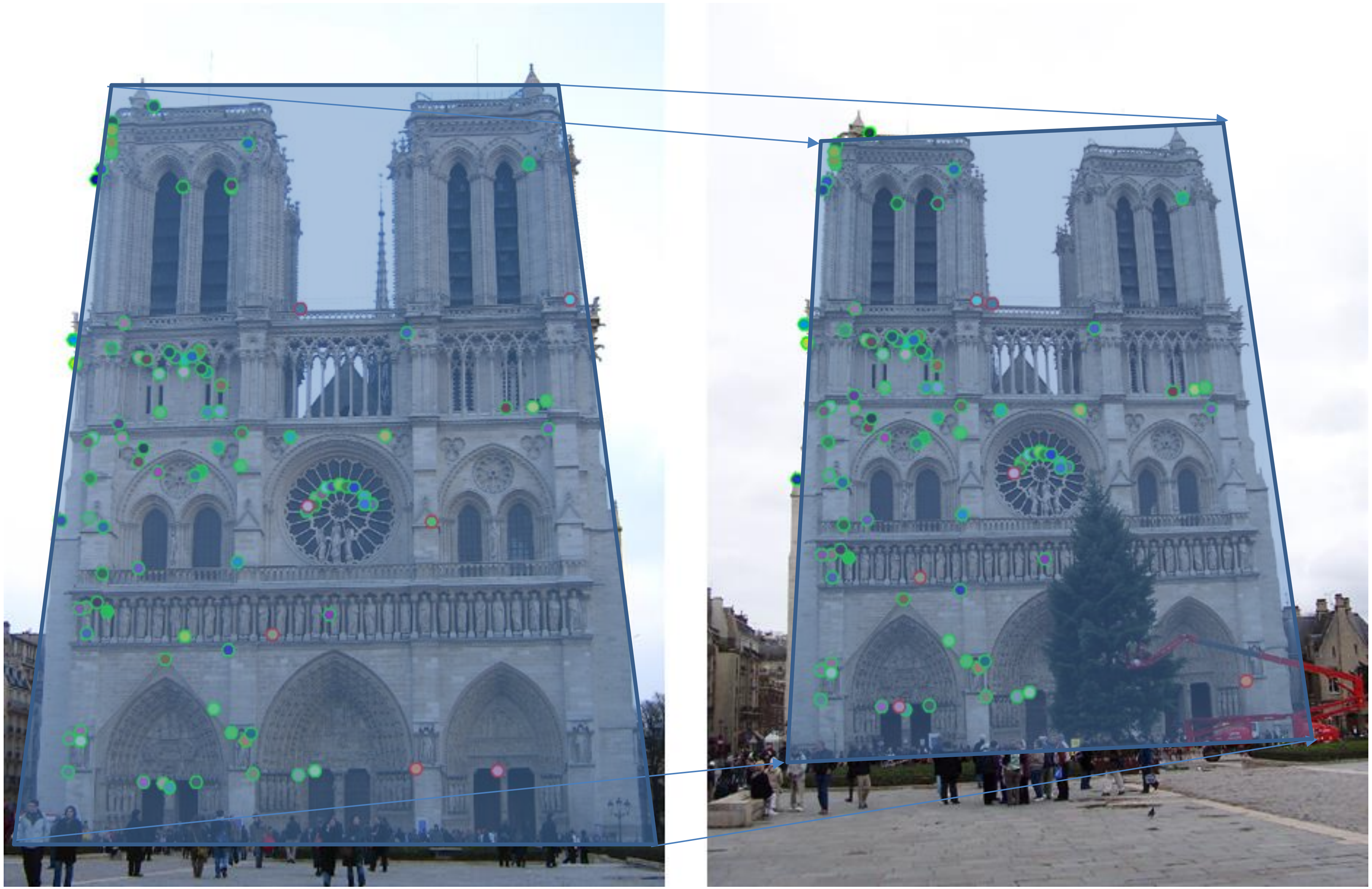
Can we refine this further?



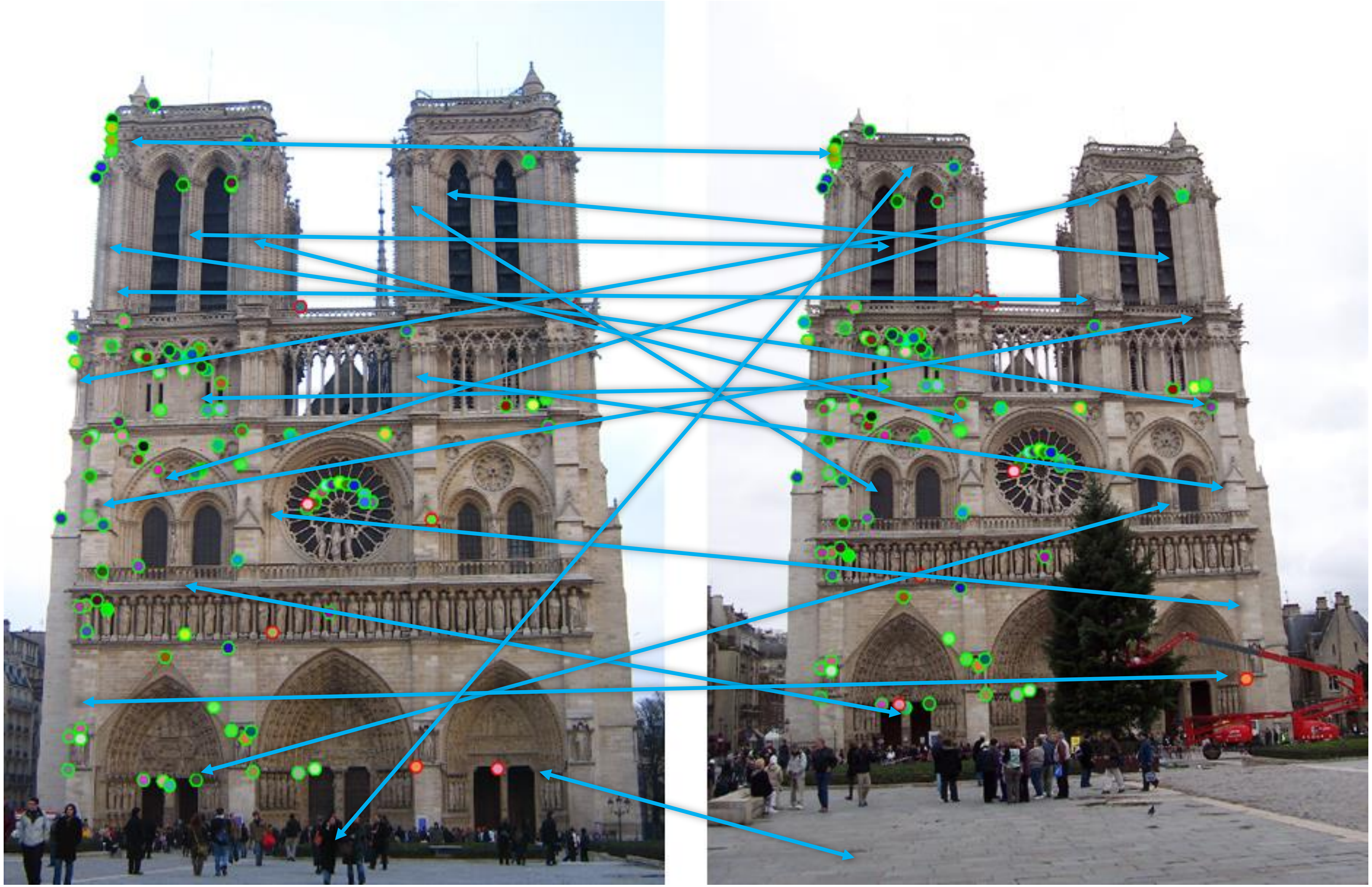
Can we refine this further?

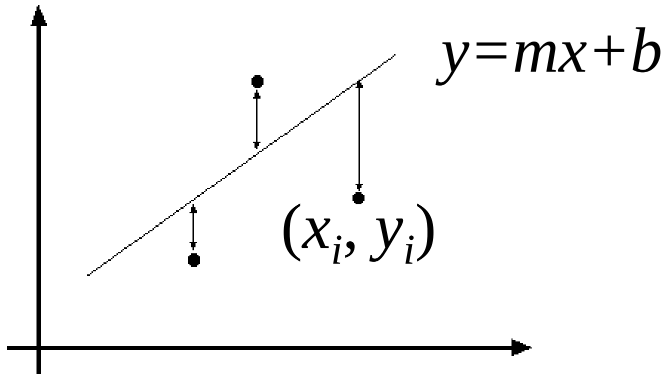


Can we refine this further?



What is the space of allowable correspondences?



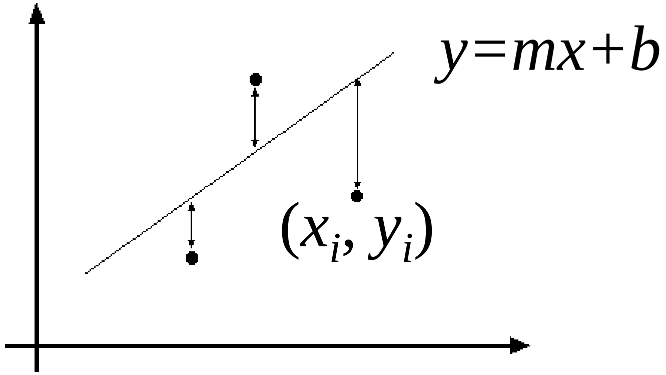


Fitting: find the parameters of a model that best fit the data

Alignment: find the parameters of the transformation that best align matched points

Fitting and Alignment

- Design challenges
  - Design a suitable **goodness of fit** measure
    - Similarity should reflect application goals
    - Encode robustness to outliers and noise
  - Design an **optimization** method
    - Avoid local optima
    - Find best parameters quickly



Fitting and Alignment: Methods

- Global optimization / Search for parameters
  - Least squares fit
  - Robust least squares
  - Other parameter search methods
- Hypothesize and test
  - Generalized Hough transform
  - RANSAC
- Iterative Closest Points (ICP)

Fitting and Alignment: Methods

- Global optimization / Search for parameters
  - Least squares fit
  - Robust least squares
  - Other parameter search methods

- Hypothesize and test
  - Generalized Hough transform
  - RANSAC

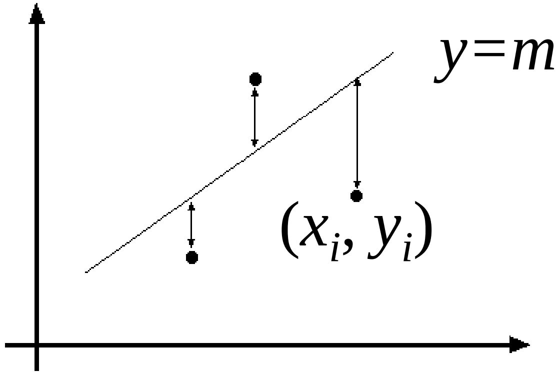
- Iterative Closest Points (ICP)

Simple example: Fitting a line

Least squares line fitting

- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation: $y_i = m x_i + b$
- Find (m, b) to minimize

$$\boxed{E = \sum_{i=1}^{n} (y_i - mx_i - b)^2}$$



$$E = \sum_{i=1}^{n} \left(\begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - y_i \right)^2 = \left\| \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right\|^2 = \|\mathbf{Ap} - \mathbf{y}\|^2$$

$$= \mathbf{y}^T \mathbf{y} - 2(\mathbf{Ap})^T \mathbf{y} + (\mathbf{Ap})^T (\mathbf{Ap})$$

$$\frac{dE}{dp} = 2\mathbf{A}^T \mathbf{Ap} - 2\mathbf{A}^T \mathbf{y} = 0$$

Matlab: `p = A \ y;`

Python: `p = numpy.linalg.lstsq(A, y)`

$$\mathbf{A}^T \mathbf{Ap} = \mathbf{A}^T \mathbf{y} \Rightarrow \mathbf{p} = \left(\mathbf{A}^T \mathbf{A}\right)^{-1} \mathbf{A}^T \mathbf{y}$$

Modified from S. Lazebnik

Least squares (global) optimization

Good

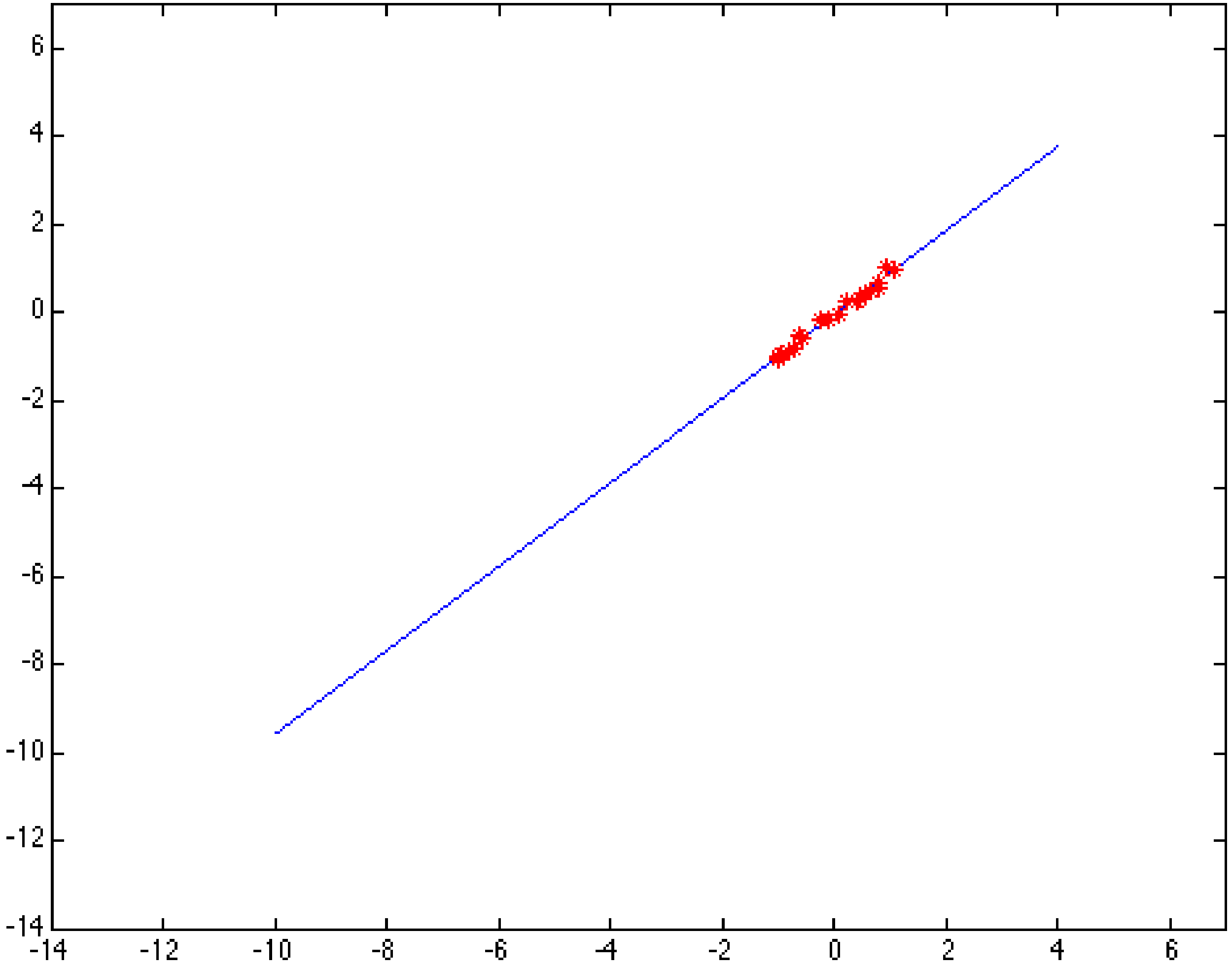
- Clearly specified objective
- Optimization is easy

Bad

- May not be what you want to optimize
- Sensitive to outliers
  - Bad matches, extra points
- Doesn’t allow you to get multiple good fits
  - Detecting multiple objects, lines, etc.

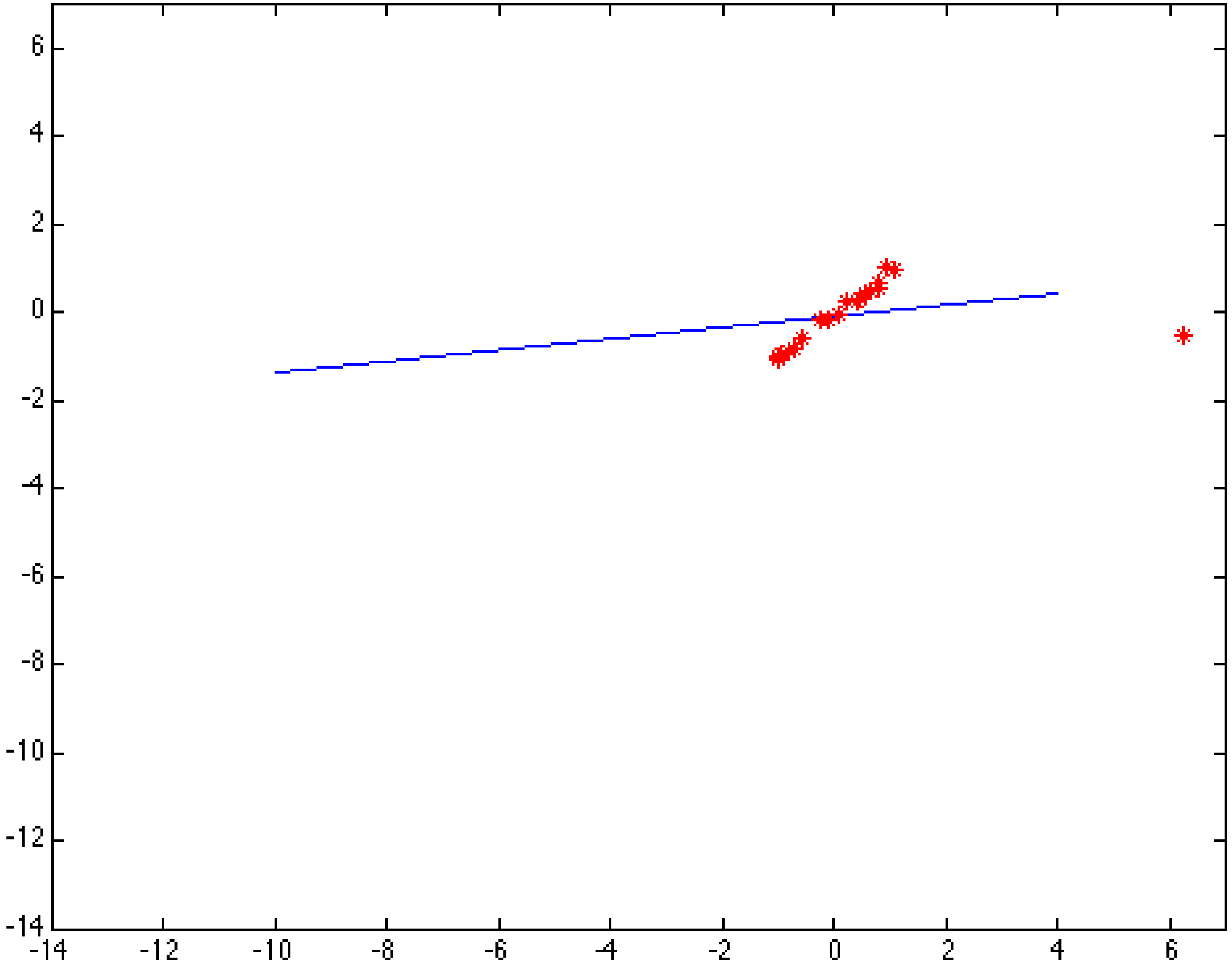
Least squares: Robustness to noise

- Least squares fit to the red points:



Least squares: Robustness to noise

- Least squares fit with an outlier:



Problem: squared error heavily penalizes outliers

Fitting and Alignment: Methods

- Global optimization / Search for parameters
  - Least squares fit
  - Robust least squares
  - Other parameter search methods
- Hypothesize and test
  - Generalized Hough transform
  - RANSAC
- Iterative Closest Points (ICP)

Robust least squares (to deal with outliers)

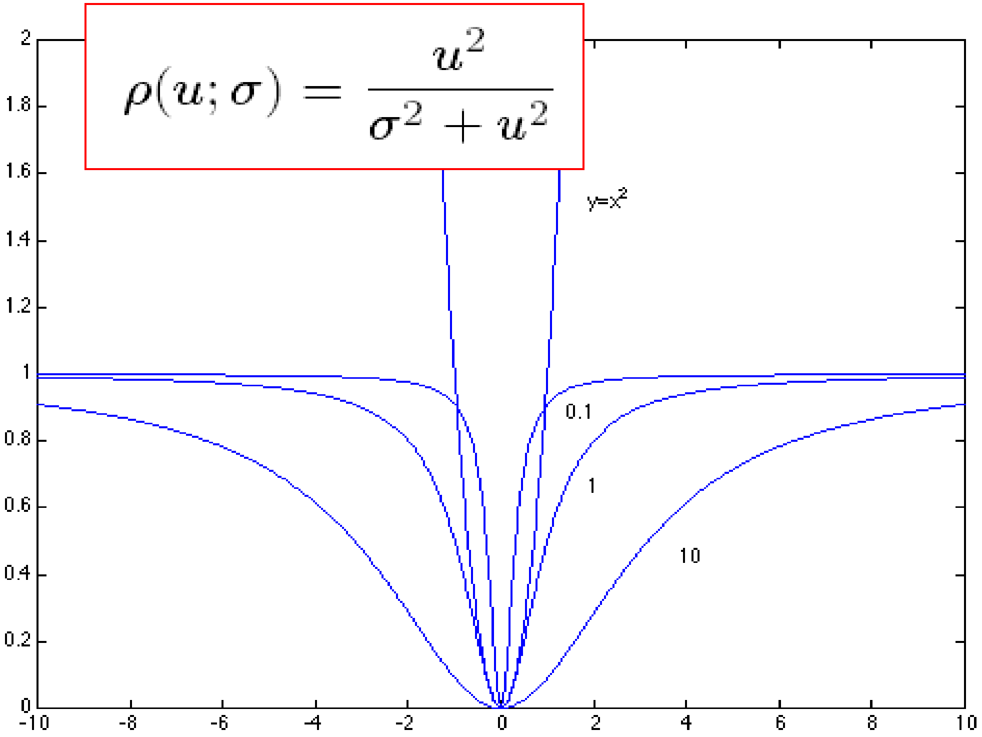
General approach:

minimize

$$\sum_{\mathrm{i}} \rho\left(\mathrm{u_i}\left(\mathrm{x_i}, \theta\right); \sigma\right) \qquad u^2 = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$

$u_i\,(x_i,\, \theta)$ – residual of i$^{\text{th}}$ point w.r.t. model parameters ϑ

ρ – robust function with scale parameter σ



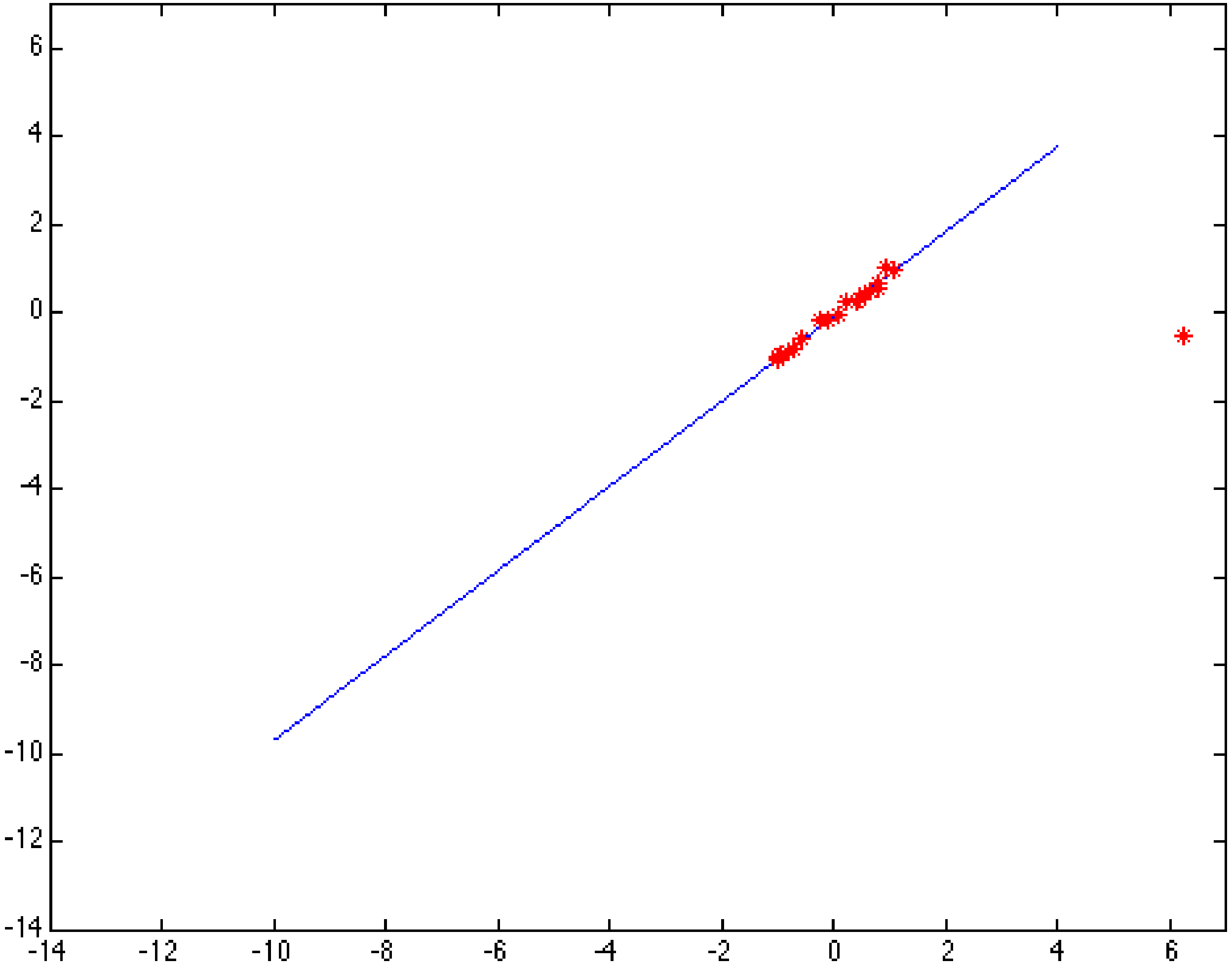
$$\rho(u;\sigma) = \frac{u^2}{\sigma^2 + u^2}$$

The robust function *ρ*

- Favors a configuration with small residuals
- Constant penalty for large residuals

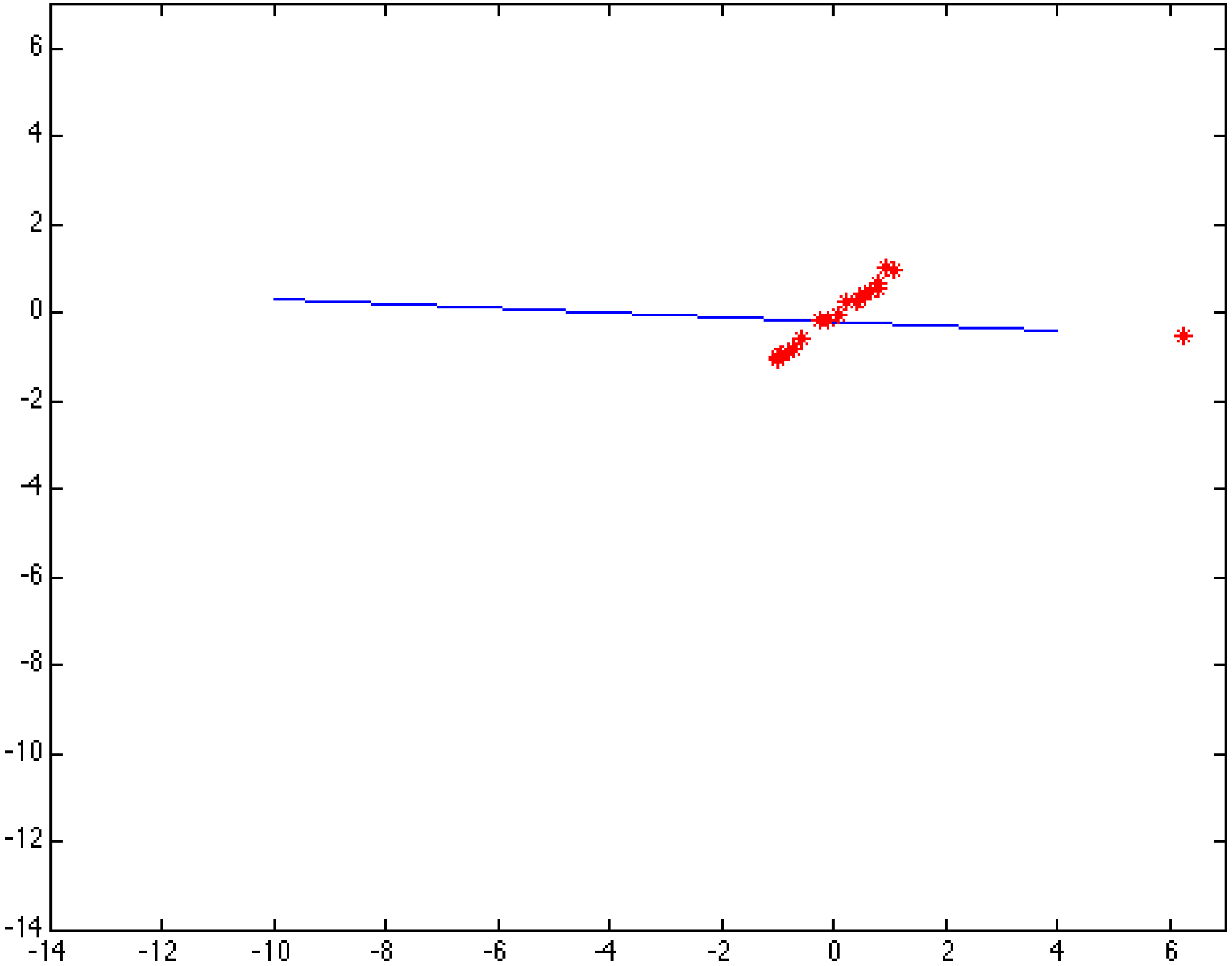
Slide from S. Savarese

Choosing the scale: Just right



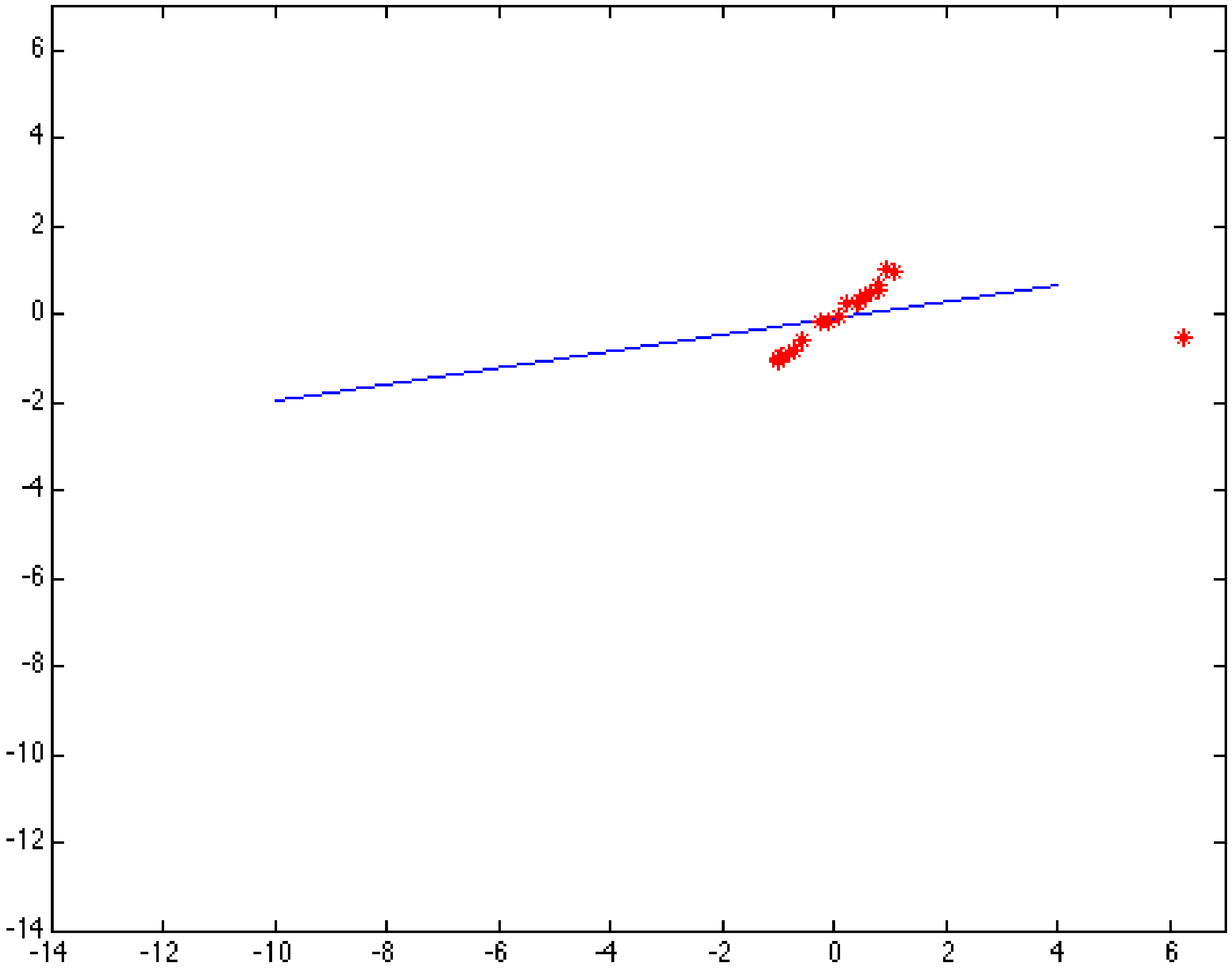
The effect of the outlier is minimized

Choosing the scale: Too small



The error value is almost the same for every point and the fit is very poor

Choosing the scale: Too large



Behaves much the same as least squares

Robust estimation: Details

- Robust fitting is a nonlinear optimization problem that must be solved iteratively
- Least squares solution can be used for initialization
- Scale of robust function should be chosen adaptively based on median residual

Fitting and Alignment: Methods

- Global optimization / Search for parameters
  - Least squares fit
  - Robust least squares
  - Other parameter search methods

- Hypothesize and test
  - Generalized Hough transform
  - RANSAC

- Iterative Closest Points (ICP)

Other ways to search for parameters (for when no closed form solution exists)

- Line search (see also“coordinate descent”)
  1. For each parameter, step through values and choose value that gives best fit
  2. Repeat (1) until no parameter changes

- Grid search
  1. Propose several sets of parameters, evenly sampled in the joint set
  2. Choose best (or top few) and sample joint parameters around the current best; repeat

- Gradient descent
  1. Provide initial position (e.g., random)
  2. Locally search for better parameters by following gradient

Fitting and Alignment: Methods

- Global optimization / Search for parameters
  - Least squares fit
  - Robust least squares
  - Other parameter search methods

- Hypothesize and test
  - Generalized Hough transform
  - RANSAC

- Iterative Closest Points (ICP)

Fitting and Alignment: Methods

- Global optimization / Search for parameters
  - Least squares fit
  - Robust least squares
  - Other parameter search methods

- Hypothesize and test
  - Generalized Hough transform
  - RANSAC

- Iterative Closest Points (ICP)

Hough Transform: Outline

1. Create a grid of parameter values

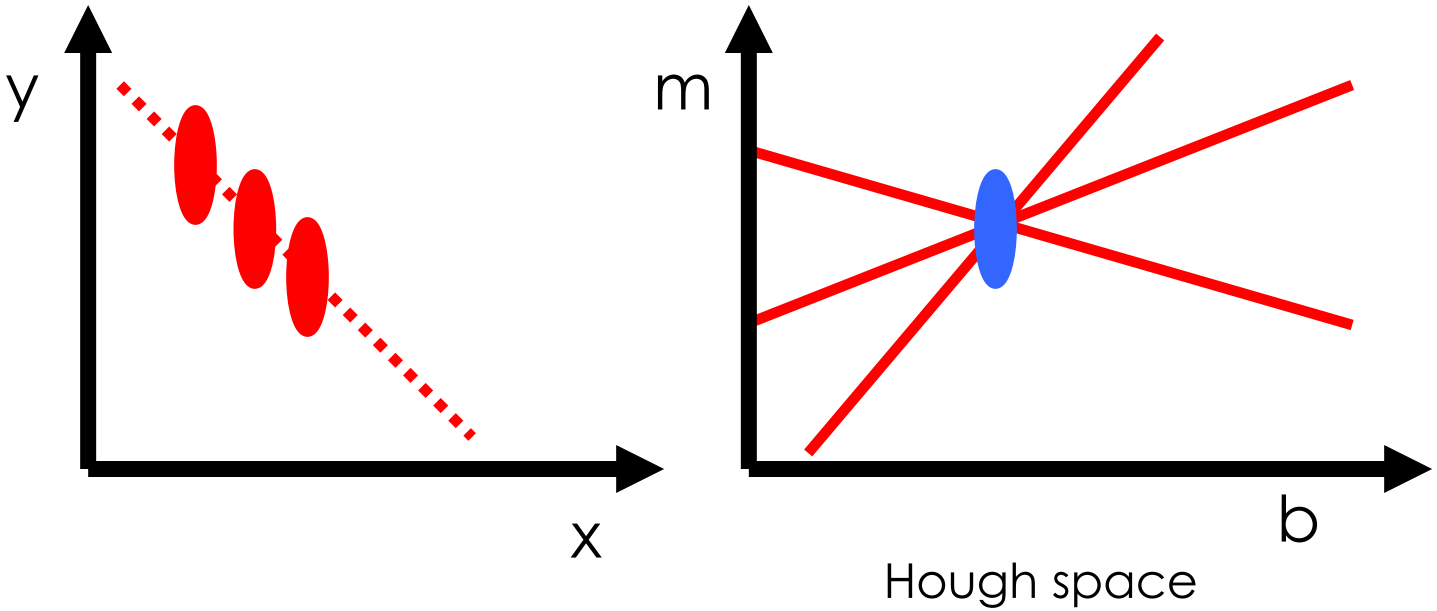
2. Each point votes for a set of parameters, incrementing those values in grid

3. Find maximum or local maxima in grid

Hough transform

P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

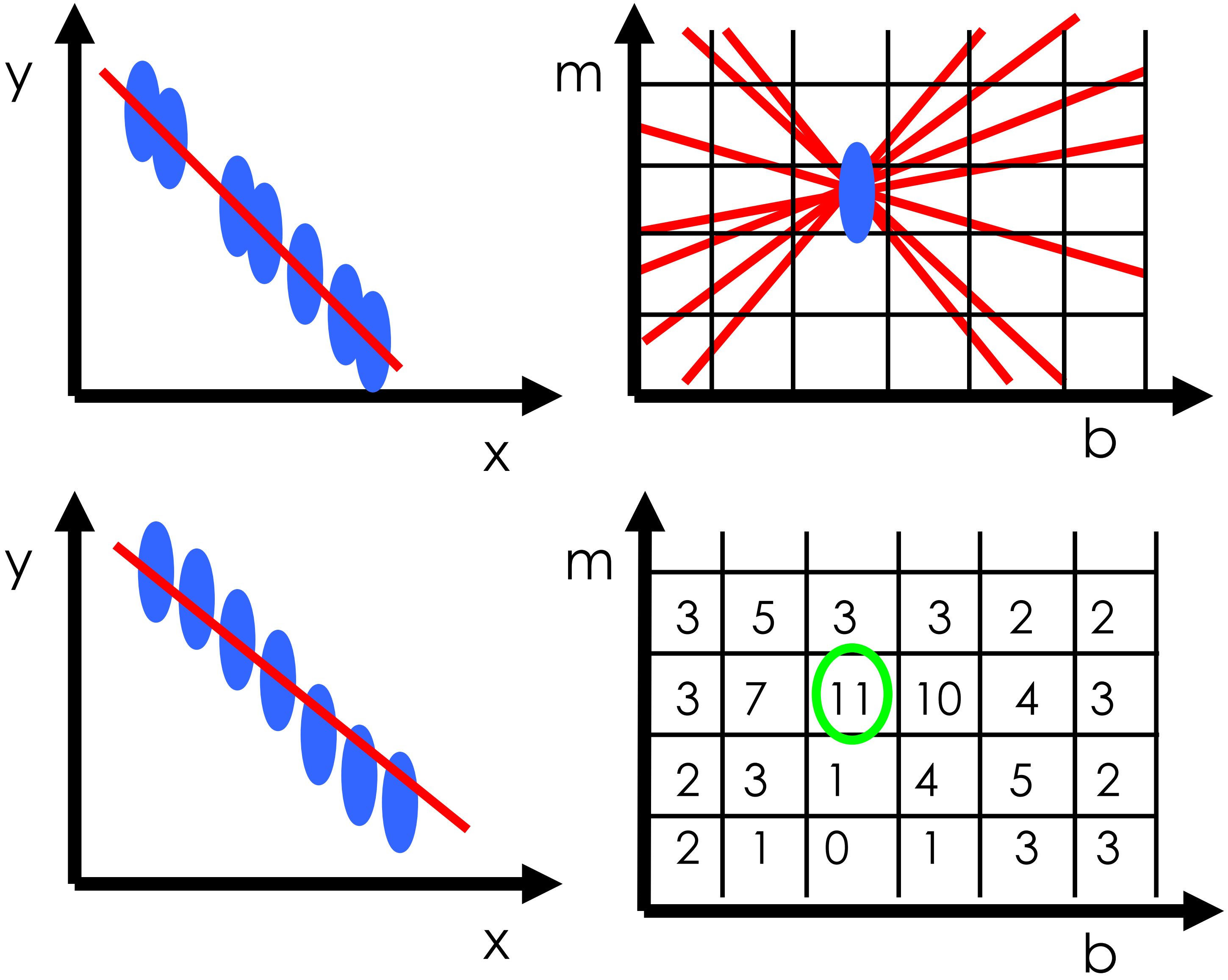
Given a set of points, find the curve or line that explains the data points best



$$y = m\,x + b$$

Slide from S. Savarese

Hough transform



3	5	3	3	2	2
3	7	11	10	4	3
2	3	1	4	5	2
2	1	0	1	3	3

Slide from S. Savarese

Hough transform

P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Issue : parameter space [m,b] is unbounded…

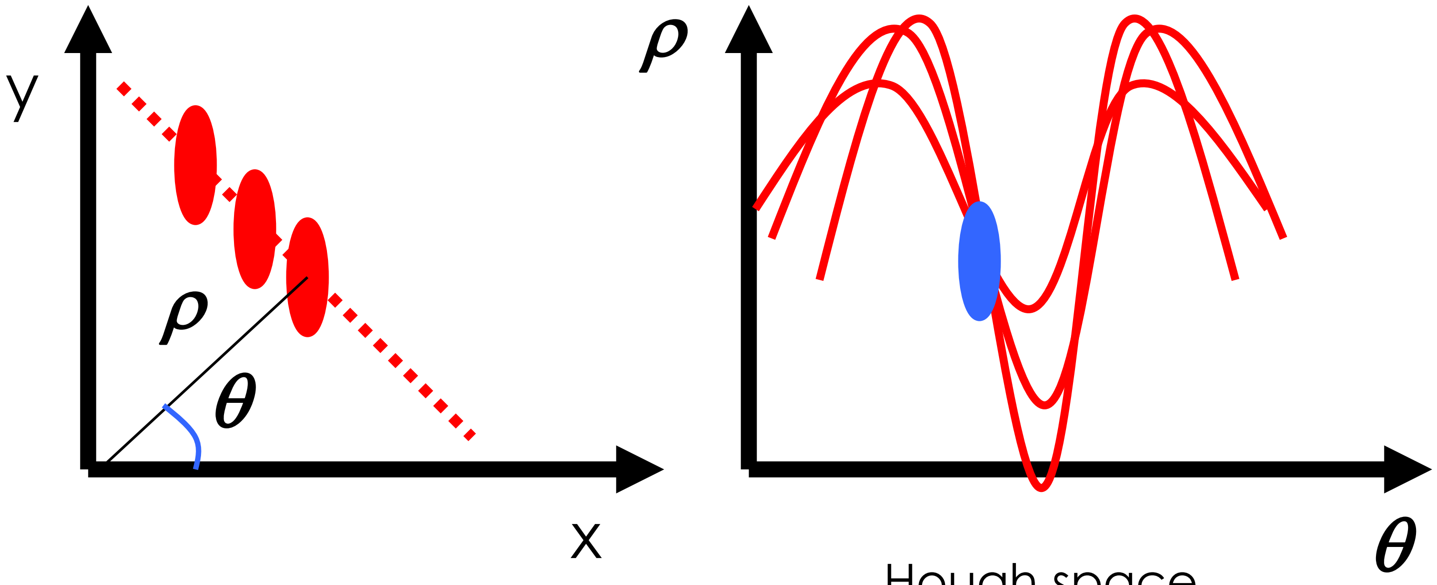
Slide from S. Savarese

Hough transform

P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Issue : parameter space [m,b] is unbounded…

Use a polar representation for the parameter space

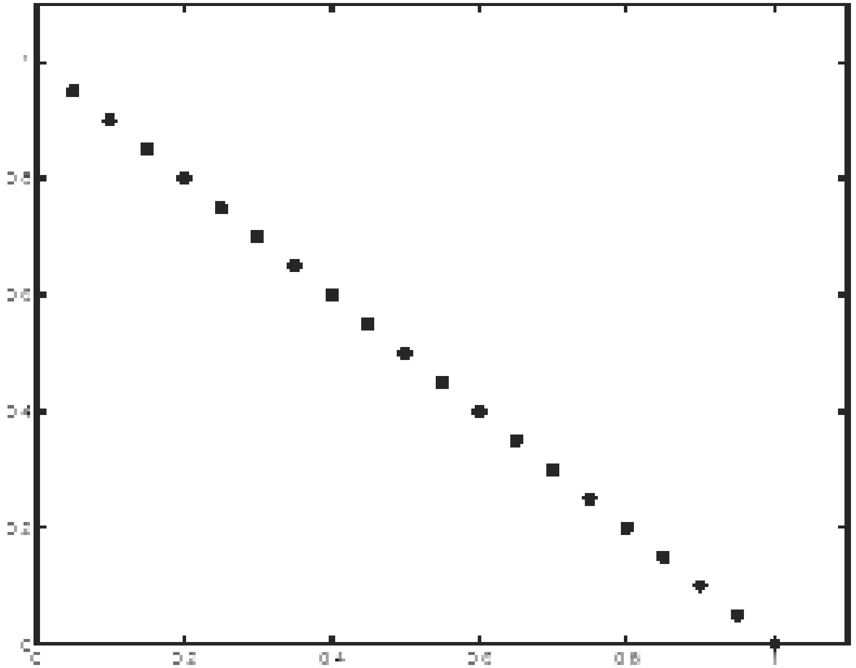


Hough space

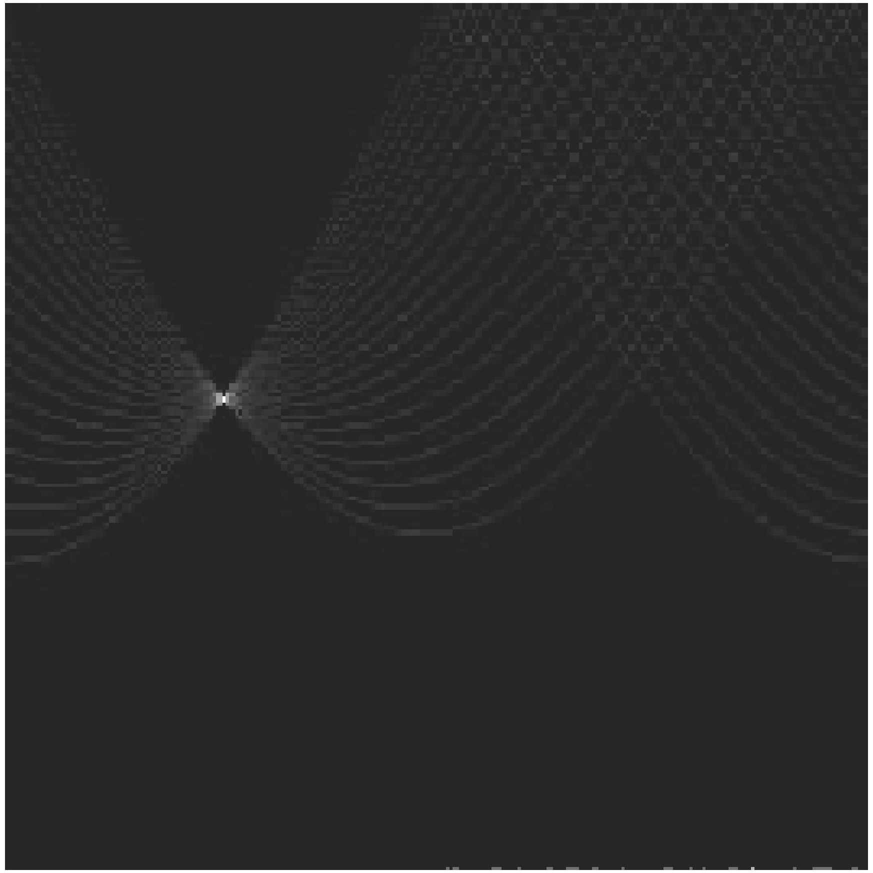
$$x\cos\theta + y\sin\theta = \rho$$

Slide from S. Savarese

Hough transform - experiments



features

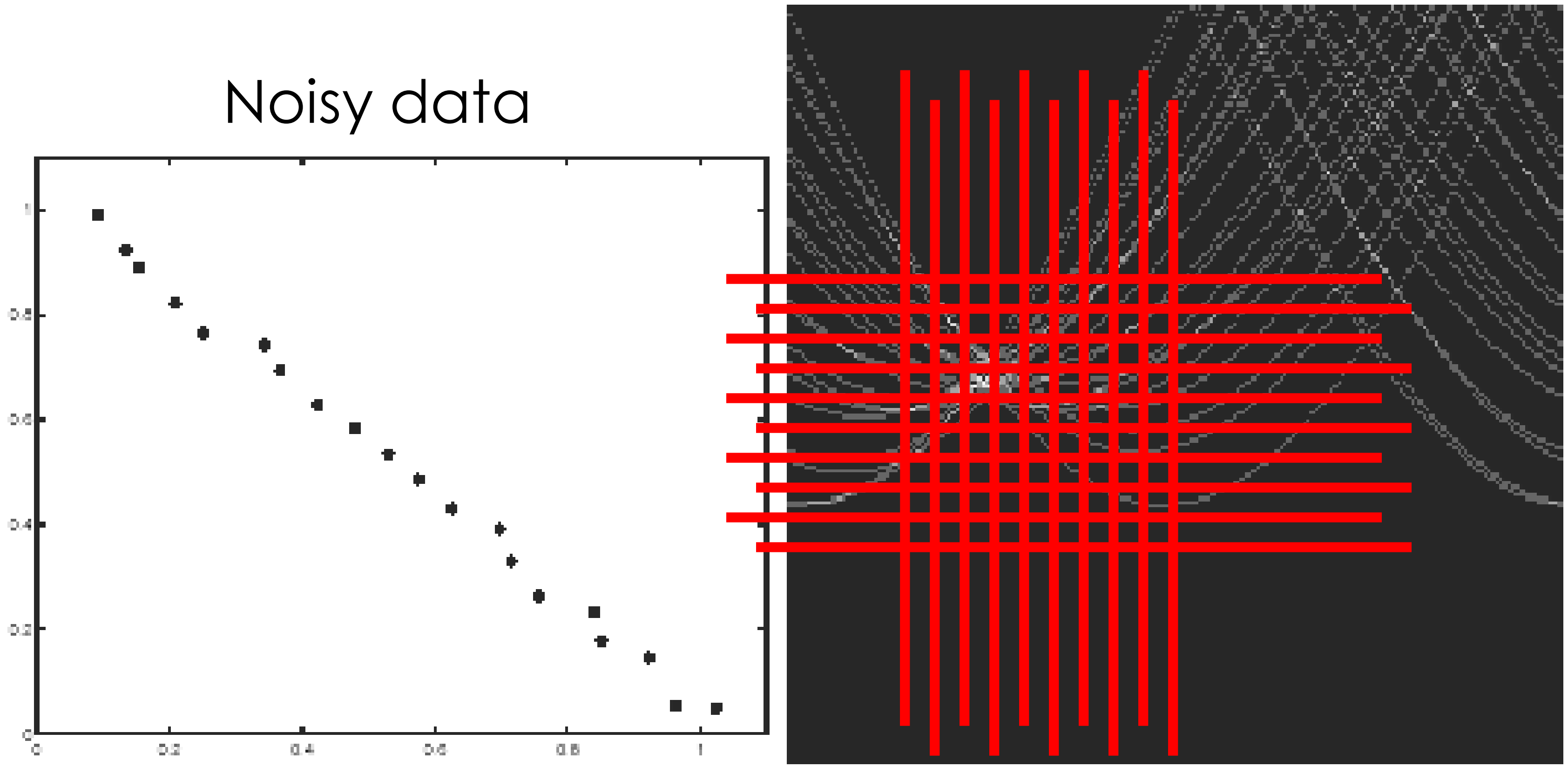


votes

Slide from S. Savarese

Hough transform - experiments

Noisy data



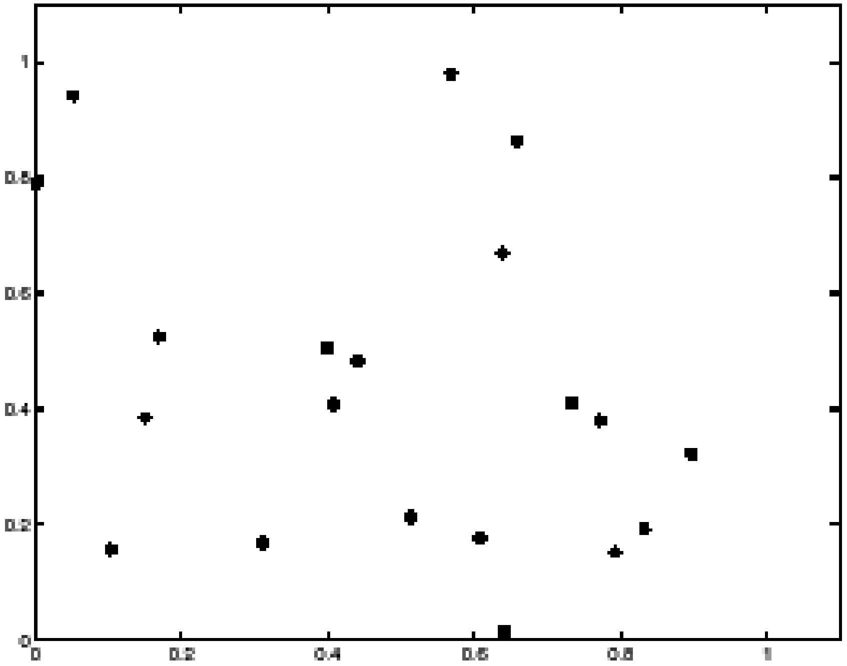
features

votes

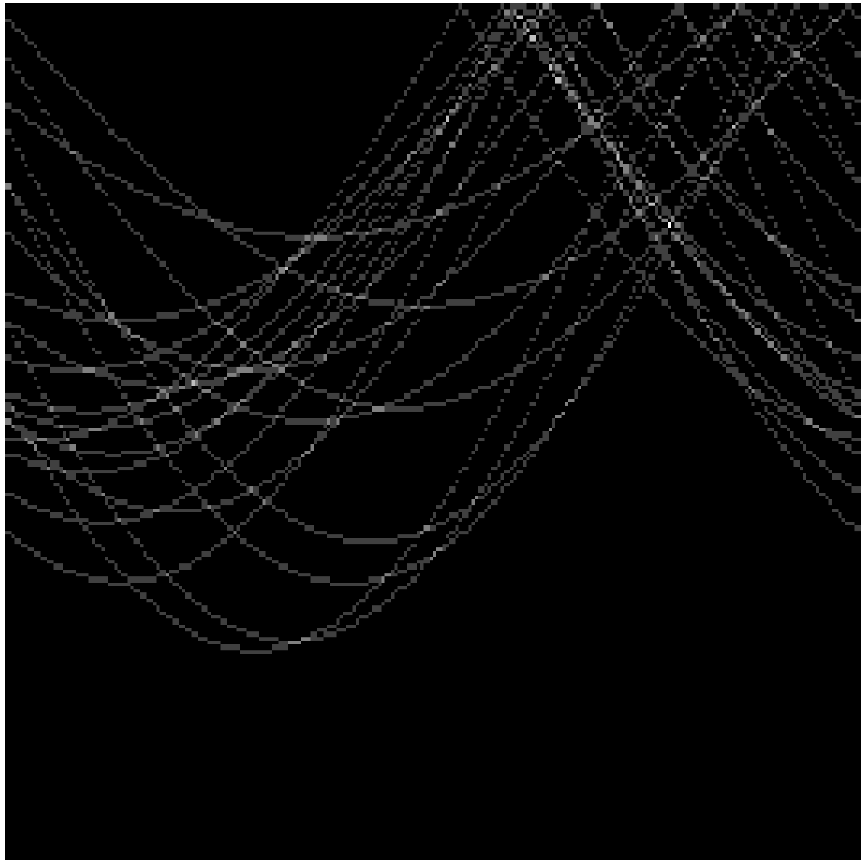
Need to adjust grid size or smooth

Slide from S. Savarese

Hough transform - experiments



features

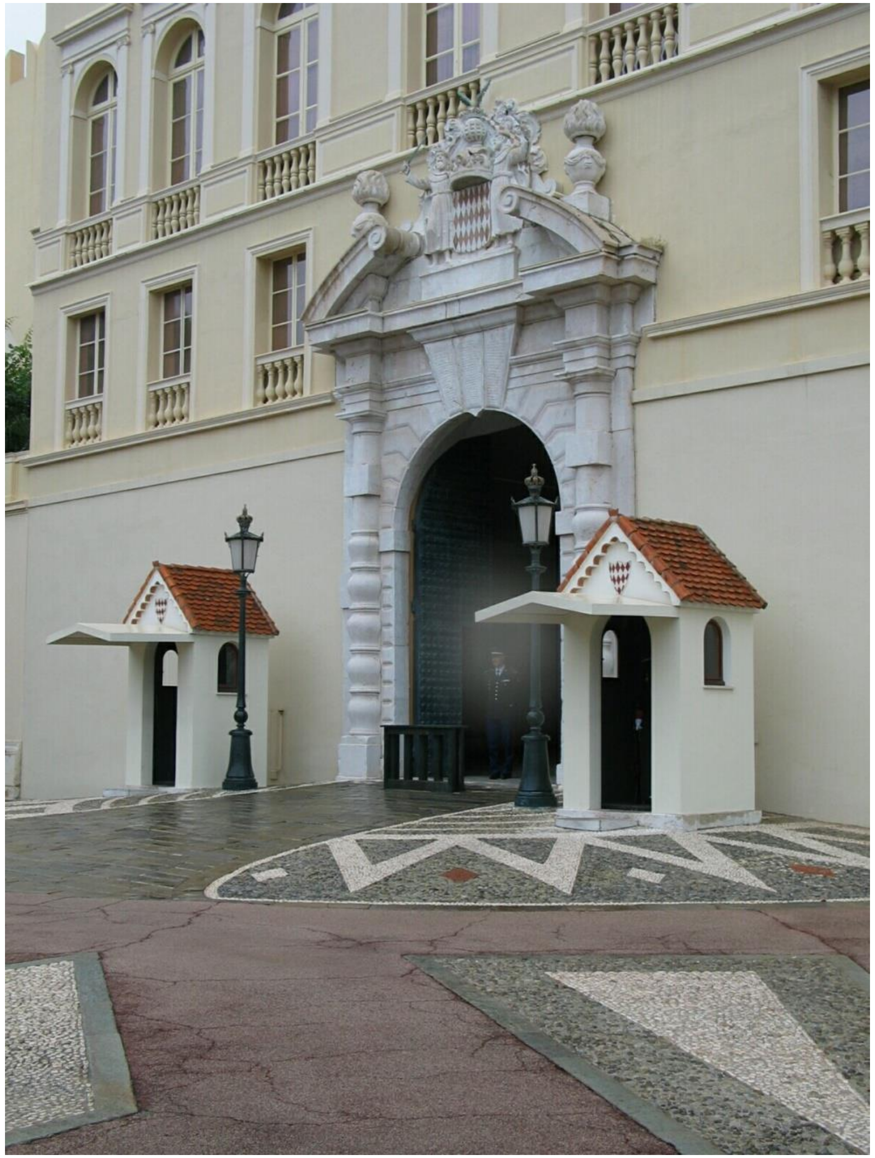


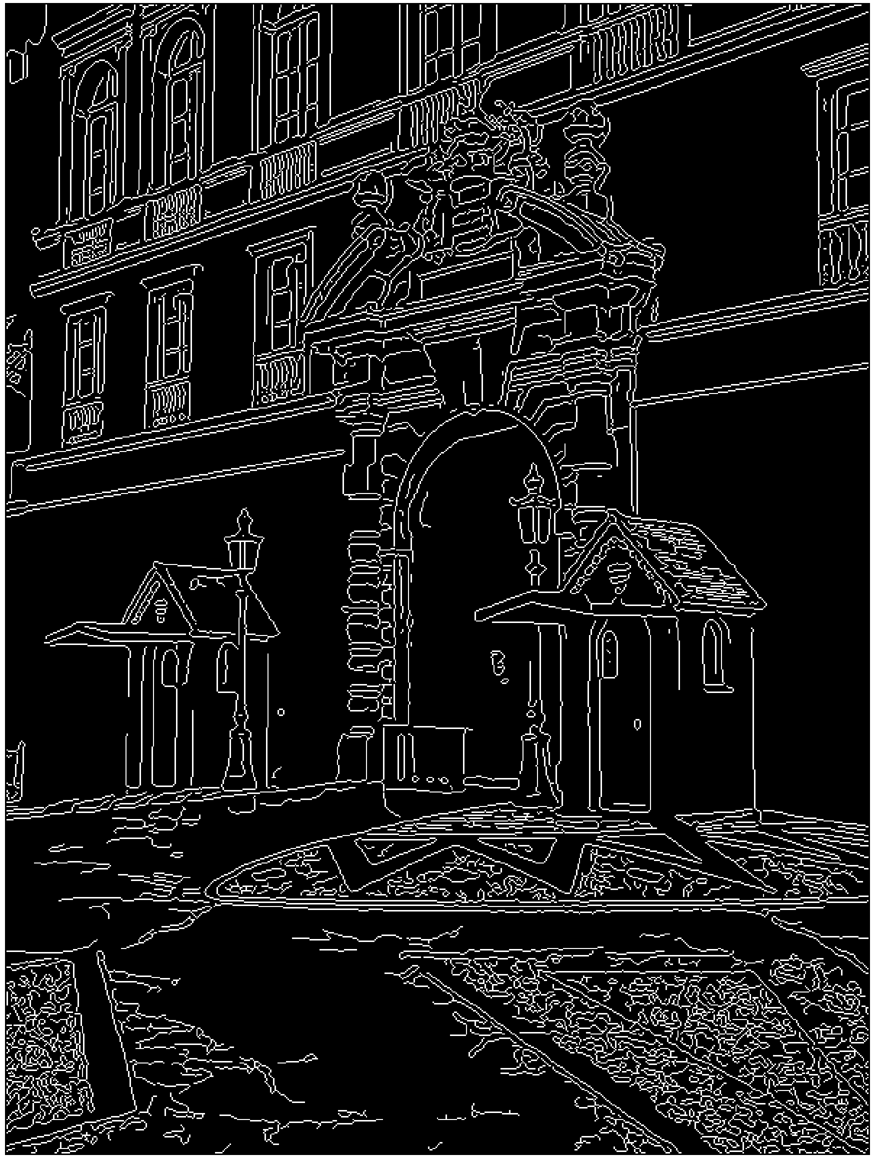
votes

Issue: spurious peaks due to uniform noise

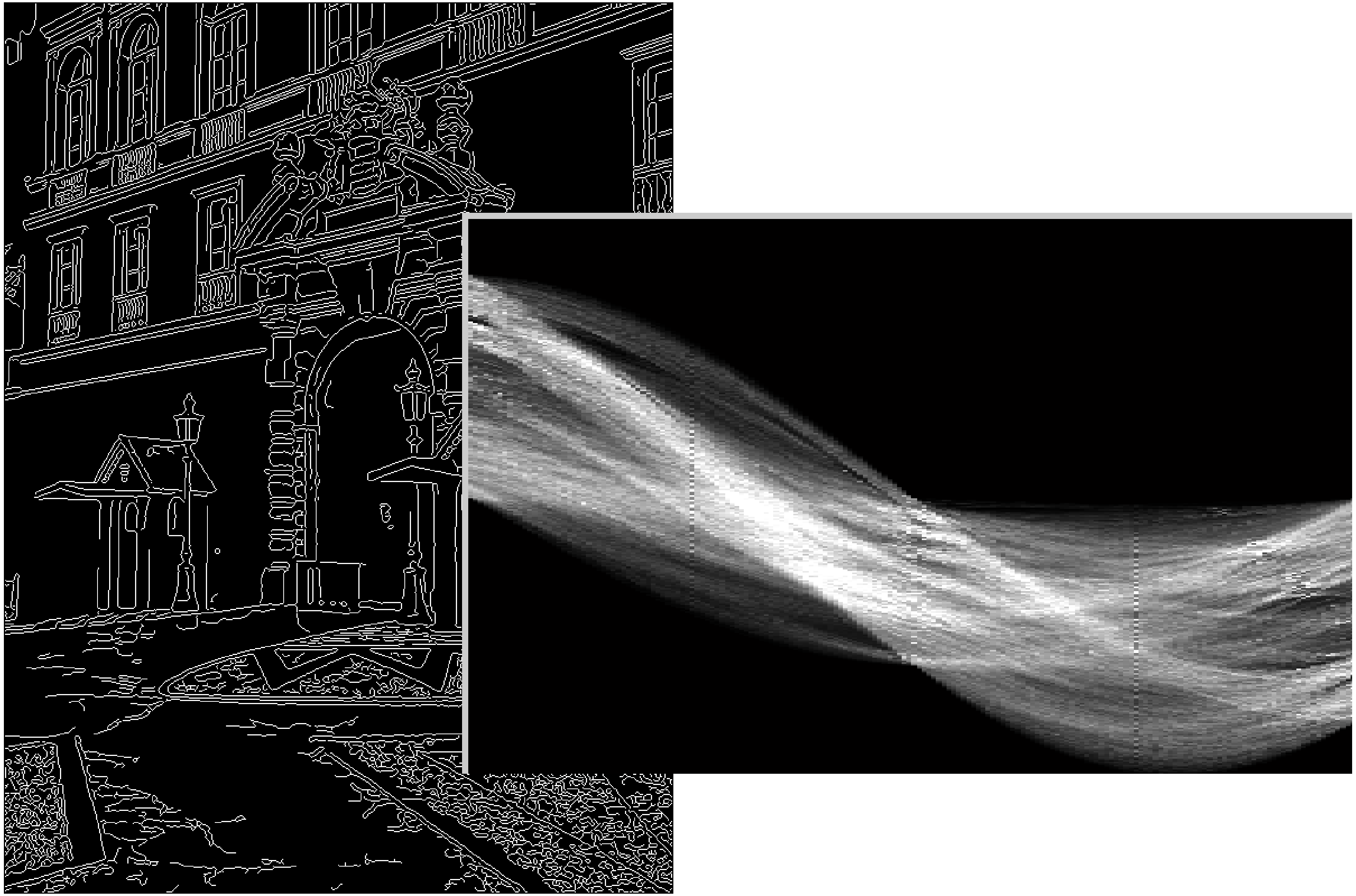
Slide from S. Savarese

1. Image → Canny Edge Detection



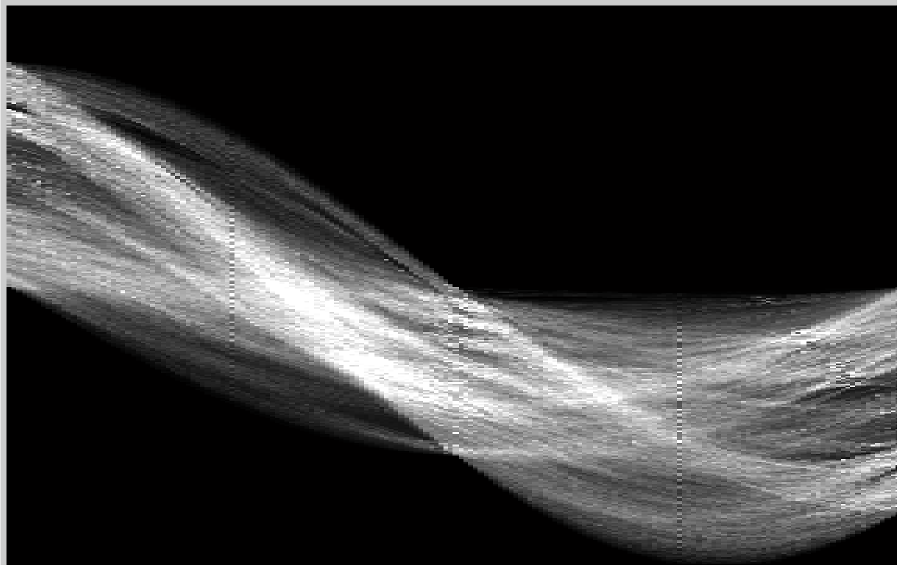


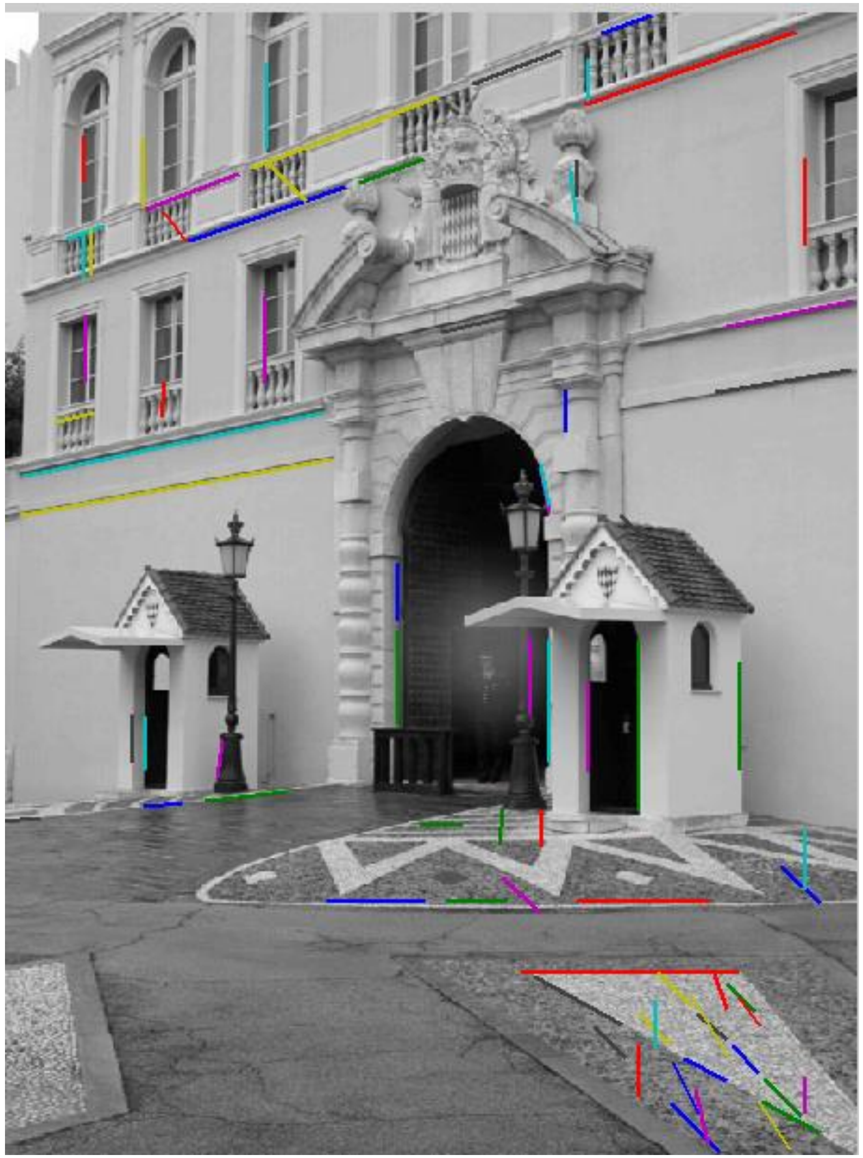
2. Canny → Hough votes



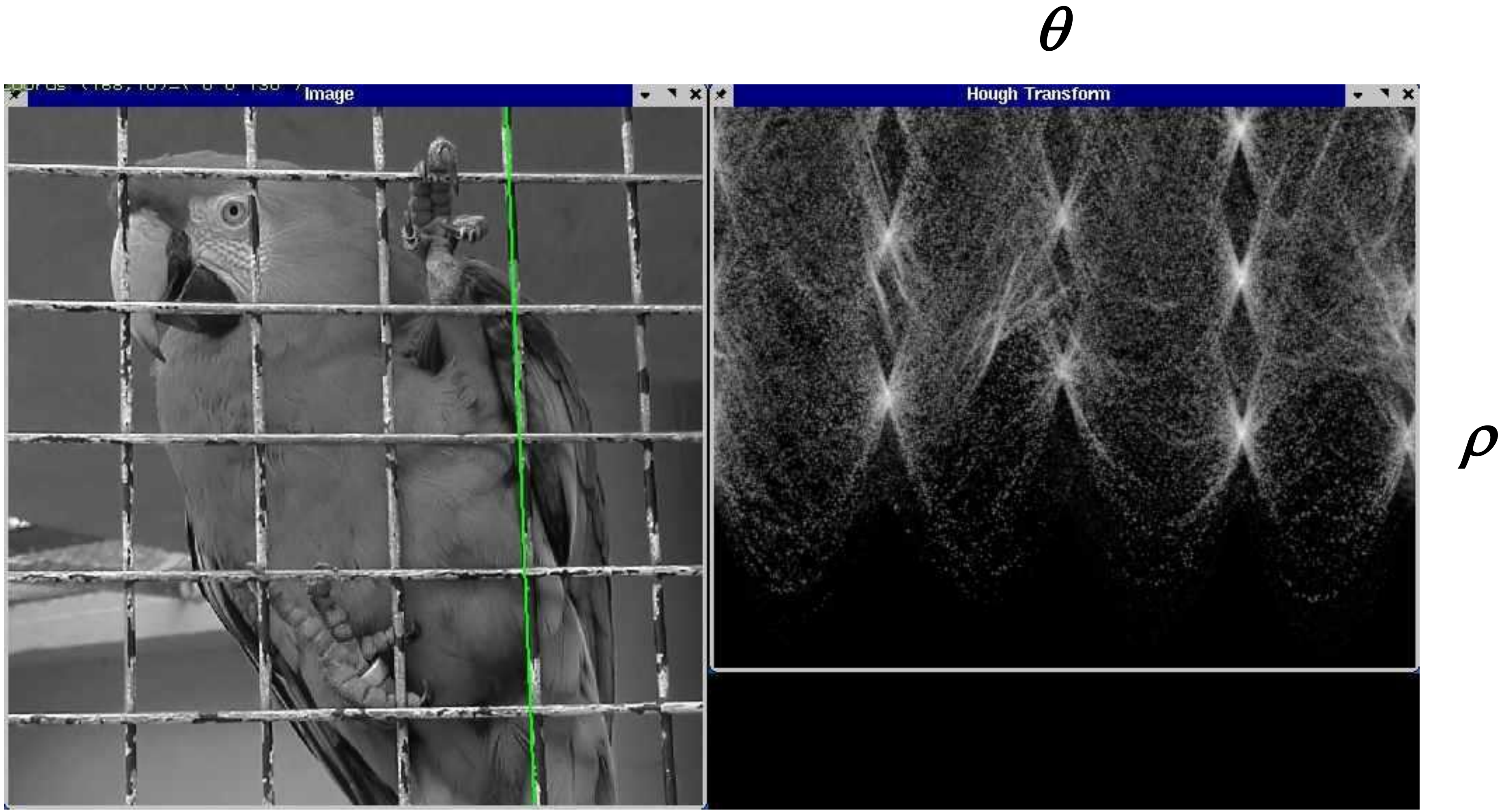
3. Hough votes → Edges

Find peaks and post-process





Hough transform example



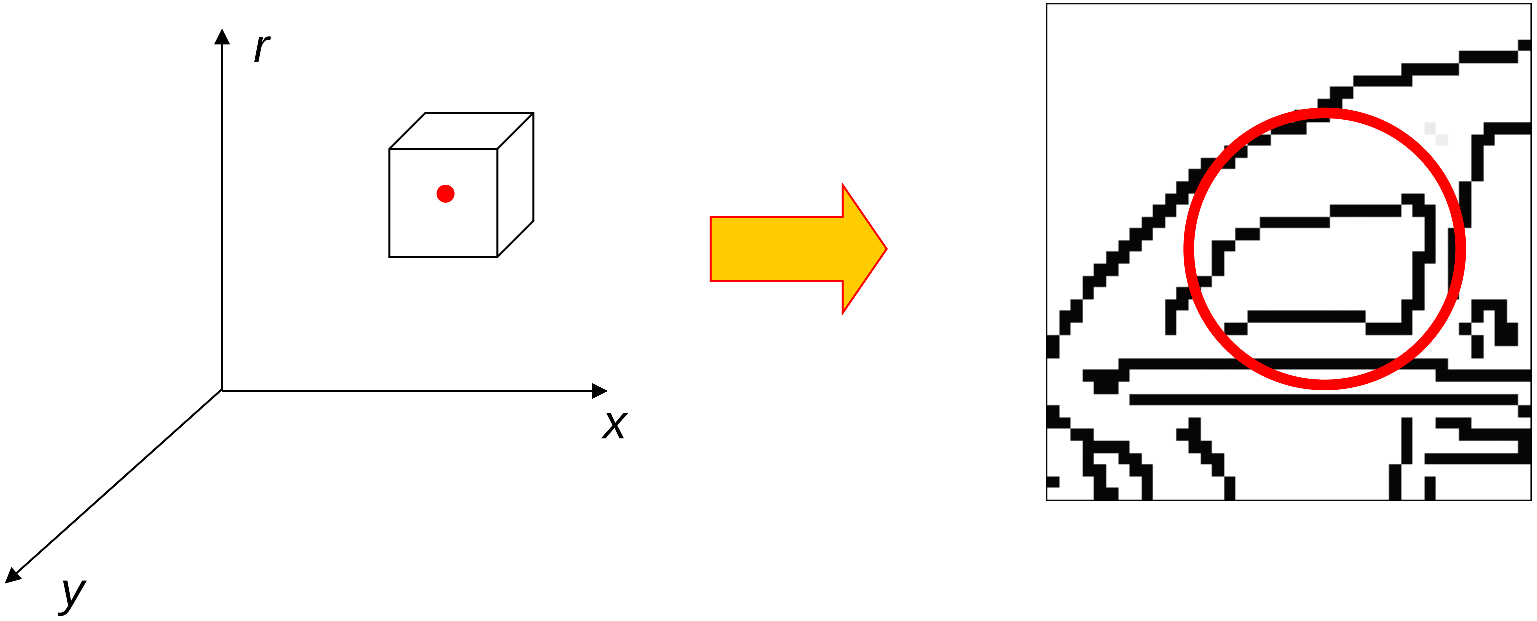
http://ostatic.com/files/images/ss_hough.jpg

Hough Transform

- How would we find circles?
  - Of fixed radius
  - Of unknown radius
  - Of unknown radius but with known edge orientation

Hough transform for circles

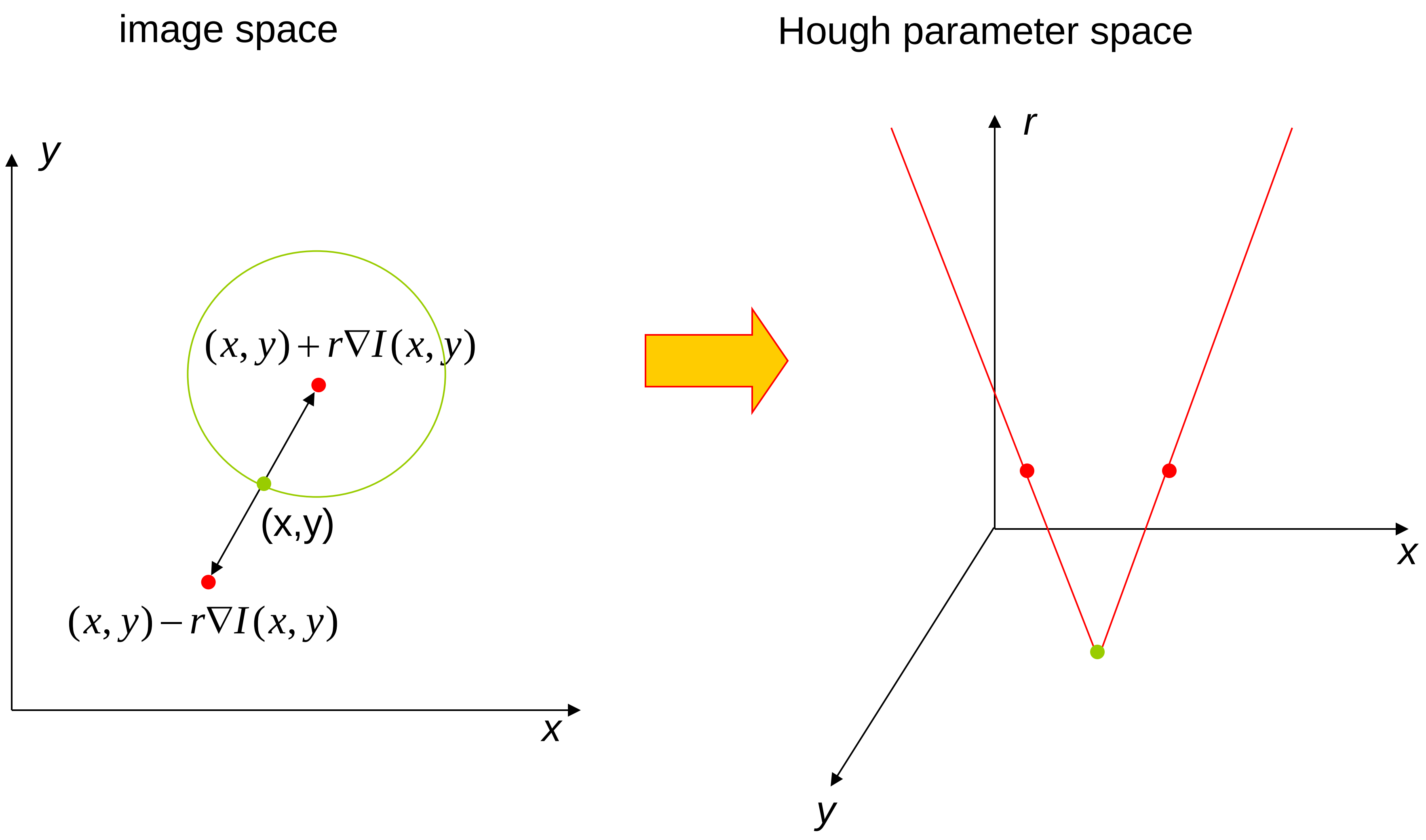
- Grid search equivalent procedure: for each (x,y,r), draw the corresponding circle in the image and compute its "support"



Hough Transform

- How would we find circles?
  - Of fixed radius
  - Of unknown radius
  - Of unknown radius but with known edge orientation

Hough transform for circles



Hough transform conclusions

Good

- Robust to outliers: each point votes separately
- Fairly efficient (often faster than trying all sets of parameters)
- Provides multiple good fits

Bad

- Some sensitivity to noise
- Bin size trades off between noise tolerance, precision, and speed/memory
  - Can be hard to find sweet spot
- Not suitable for more than a few parameters
  - grid size grows exponentially

Common applications

- Line fitting (also circles, ellipses, etc.)
- Object instance recognition (parameters are affine transform)
- Object category recognition (parameters are position/scale)

Fitting and Alignment: Methods

- Global optimization / Search for parameters
  - Least squares fit
  - Robust least squares
  - Other parameter search methods
- Hypothesize and test
  - Generalized Hough transform
  - RANSAC
- Iterative Closest Points (ICP)