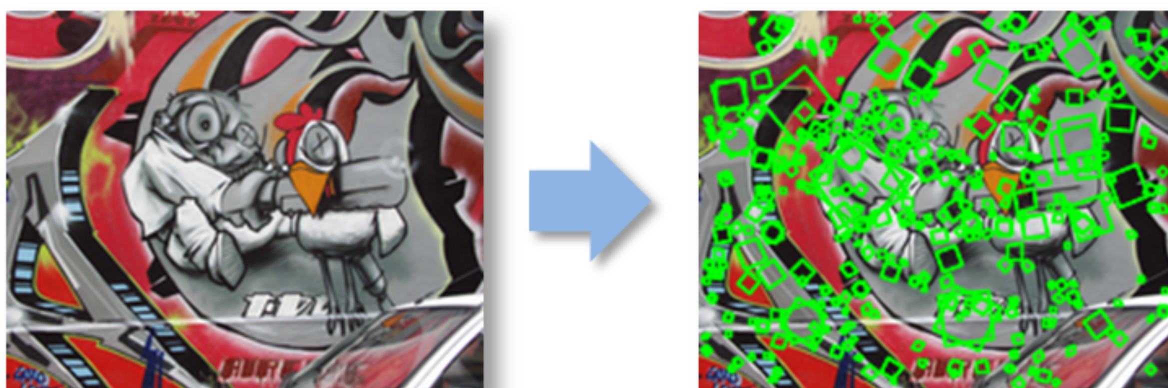
CSCI 497/597P: Computer Vision

Scott Wehrwein

Resampling: Upsampling

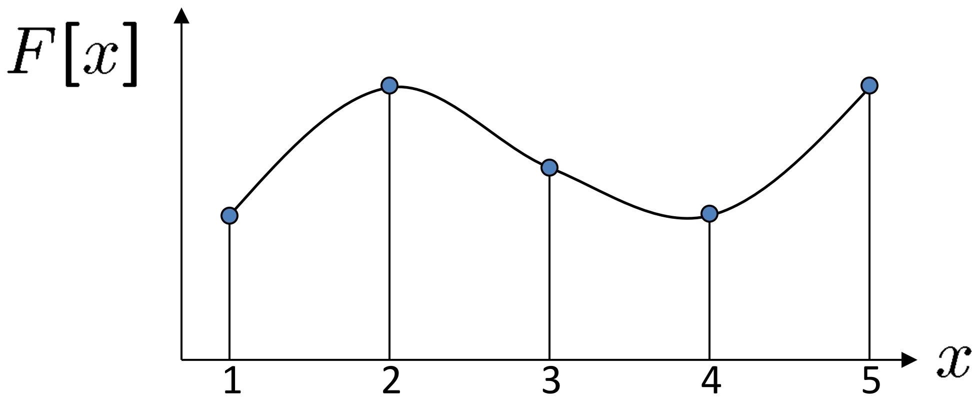
Features - Overview



Upsampling

- This image is too small for this screen: 
- How can we make it 10 times as big?
- Simplest approach: repeat each row and column 10 times
- ("Nearest neighbor interpolation")

Image interpolation



d = 1 in this example

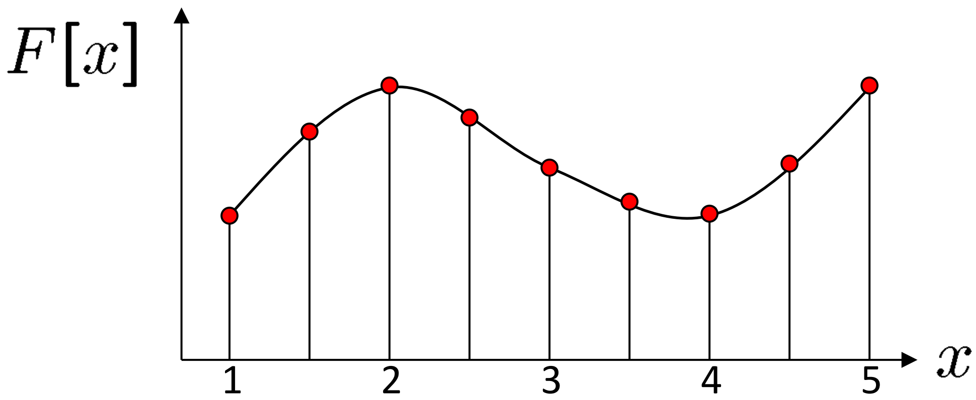
Recall that a digital images is formed as follows:

$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

- It is a discrete point-sampling of a continuous function
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale

Adapted from: S. Seitz

Image interpolation



d = 1 in this example

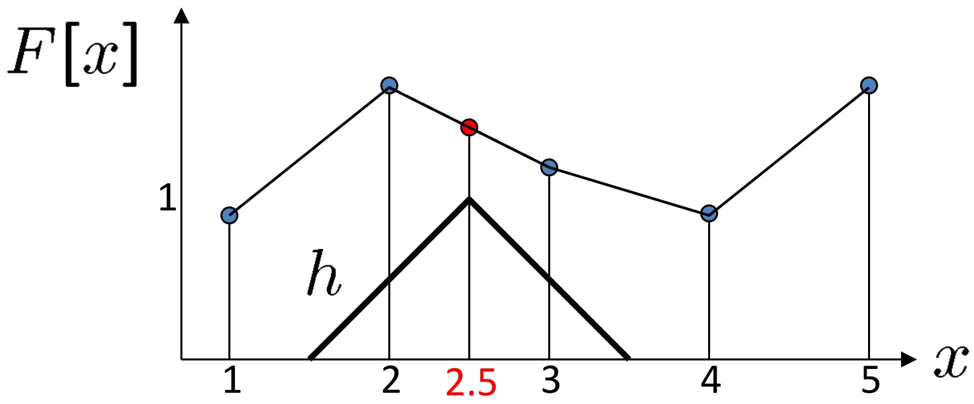
Recall that a digital images is formed as follows:

$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

- It is a discrete point-sampling of a continuous function
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale

Adapted from: S. Seitz

Image interpolation



d = 1 in this example

- What if we don't know f ?
  - Guess an approximation: $\tilde{f}$
  - Can be done in a principled way: filtering
  - Convert F to a continuous function:

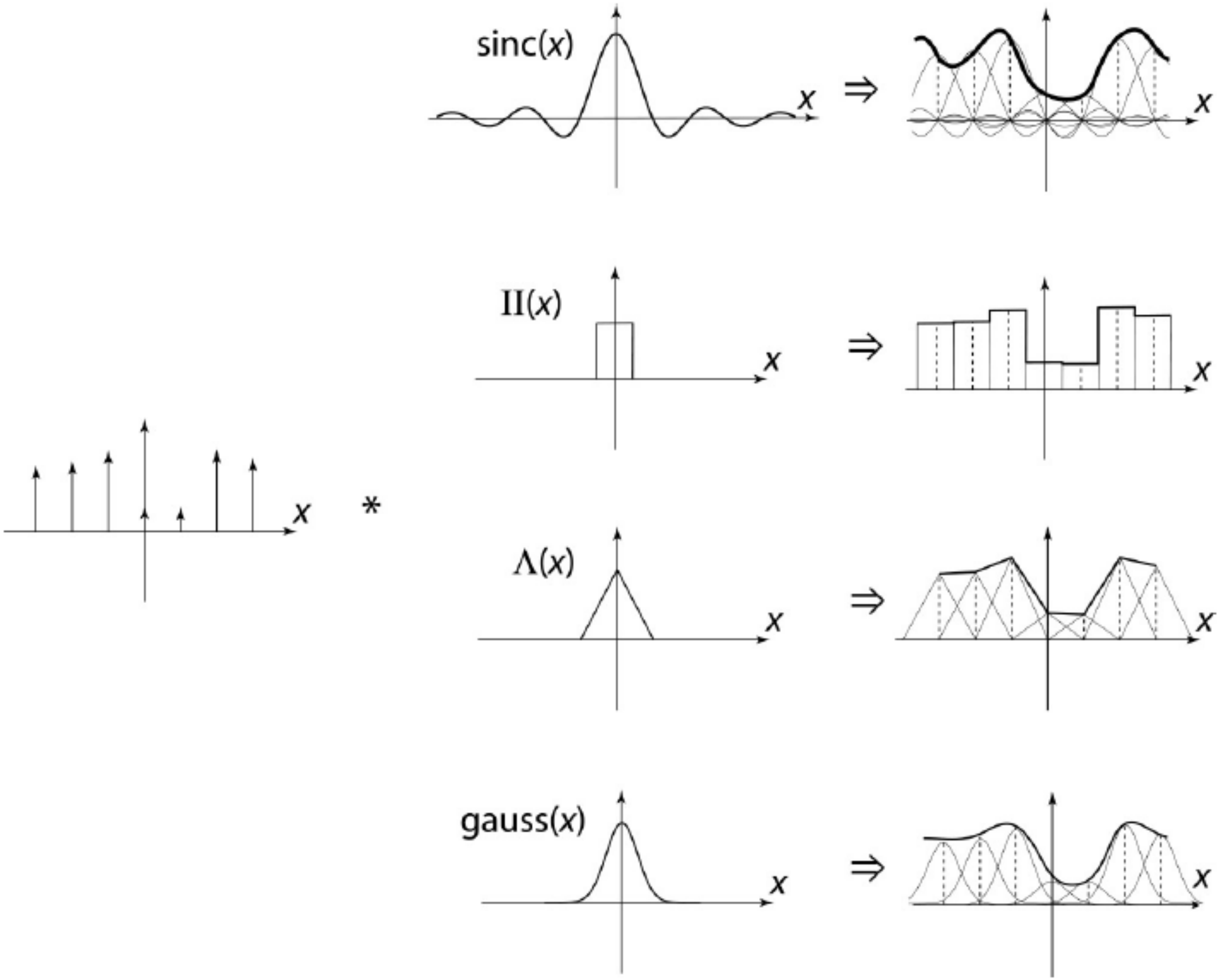
$$f_F(x) = F(\tfrac{x}{d}) \text{ when } \tfrac{x}{d} \text{ is an integer, 0 otherwise}$$

  - Reconstruct by convolution with a *reconstruction filter, h*

$$\tilde{f} = h * f_F$$

Adapted from: S. Seitz

Image interpolation



"Ideal" reconstruction

Nearest-neighbor interpolation

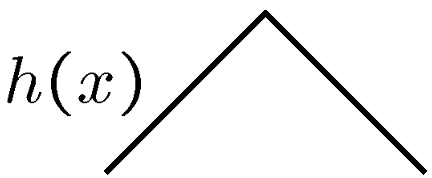
Linear interpolation

Gaussian reconstruction

Source: B. Curless

Reconstruction filters

- What does the 2D version of this hat function look like?

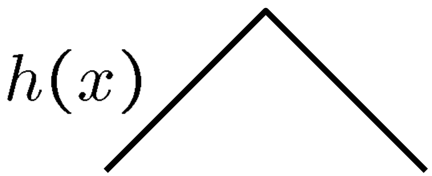


performs
linear interpolation

0	0	0
1	2	1
0	0	0

Reconstruction filters

- What does the 2D version of this hat function look like?



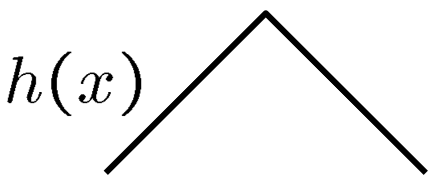
performs
linear interpolation

Hint: try the following convolution:

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 2 & 1 \\ 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Reconstruction filters

- What does the 2D version of this hat function look like?



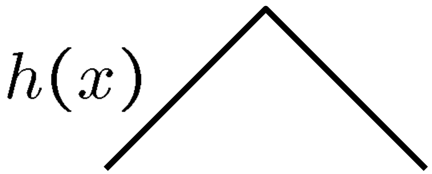
performs
linear interpolation

Hint: try the following convolution:

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

Reconstruction filters

- What does the 2D version of this hat function look like?



performs
linear interpolation

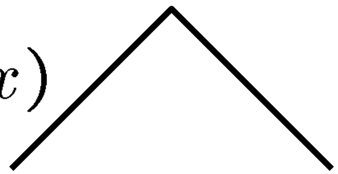
Hint: try the following convolution:

$$\begin{bmatrix}1 & 2 & 1\end{bmatrix} * \begin{bmatrix}1\\2\\1\end{bmatrix} = \begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix}$$

Reconstruction filters

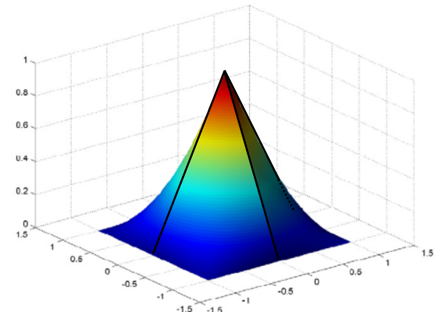
- What does the 2D version of this hat function look like?

$h(x)$



performs
linear interpolation

$h(x, y)$



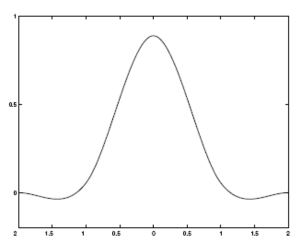
(tent function) performs
bilinear interpolation

Often implemented without cross-correlation

- E.g., http://en.wikipedia.org/wiki/Bilinear_interpolation

Better filters give better resampled images

- **Bicubic** is common choice



Cubic reconstruction filter

$$r(x) = \frac{1}{6}\begin{cases} (12-9B-6C)|x|^3 + (-18+12B+6C)|x|^2 + (6-2B) & |x| < 1 \\ ((-B-6C)|x|^3 + (6B+30C)|x|^2 + (-12B-48C)|x| + (8B+24C) & 1 \le |x| < 2 \\ 0 & \textit{otherwise} \end{cases}$$

Upsampling images

Step 1: blow up to original size with 0's in between

Upsampling images

Step 2: Convolve with upsampling filter (here: Gaussian)

Image interpolation

Original image:  x 10

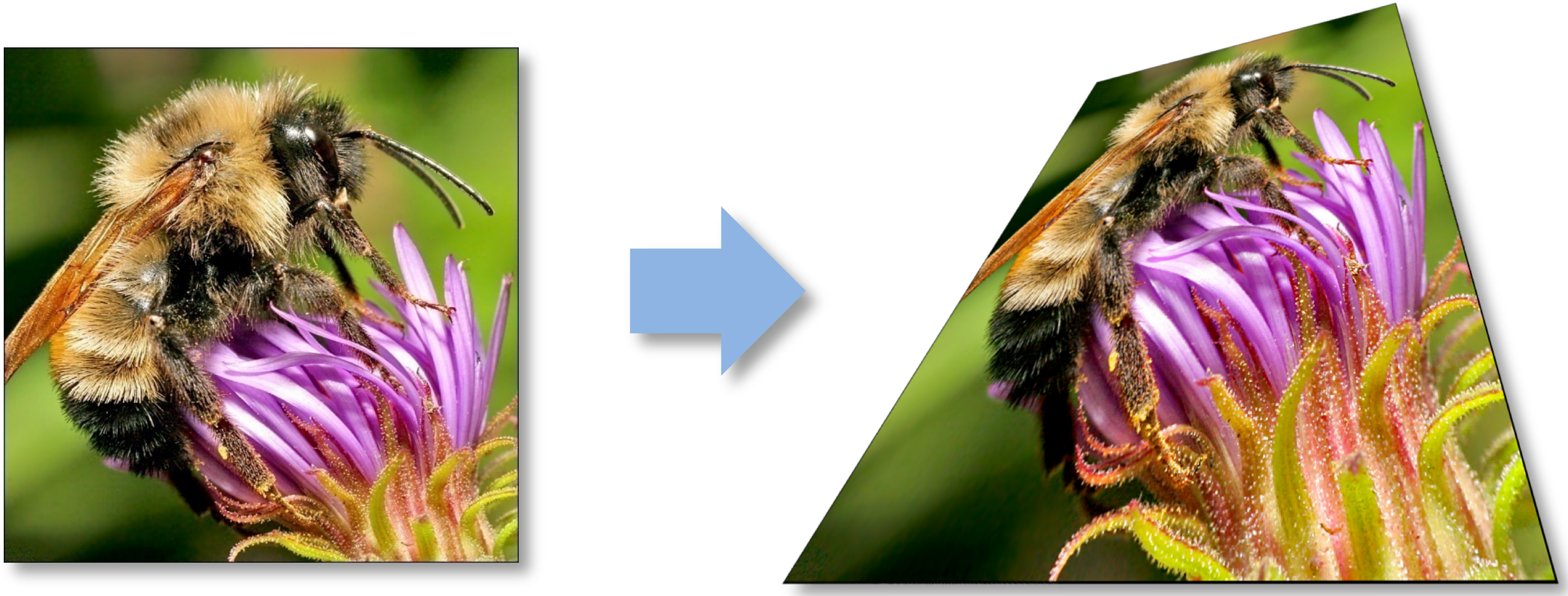
Nearest-neighbor interpolation

Bilinear interpolation

Bicubic interpolation

Image interpolation

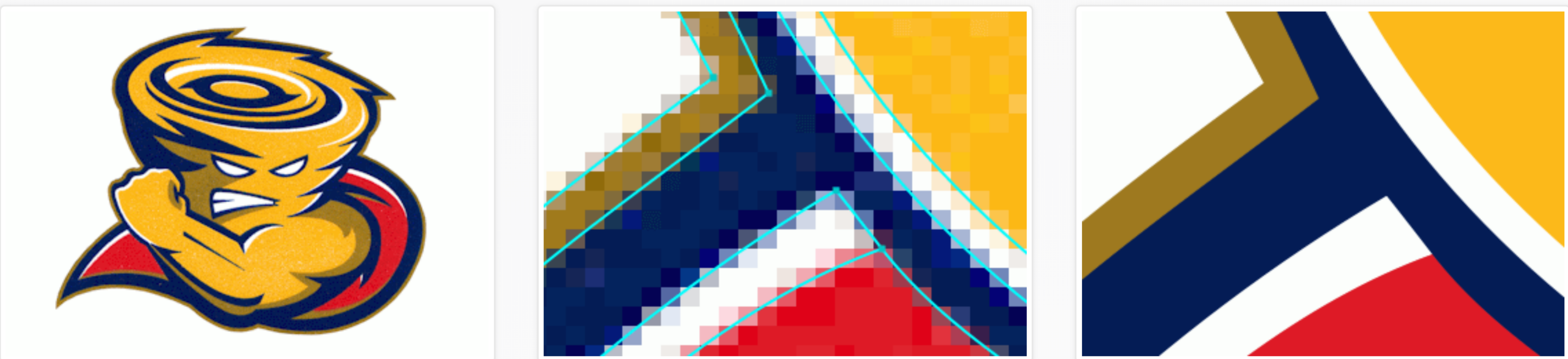
Also used for *resampling*



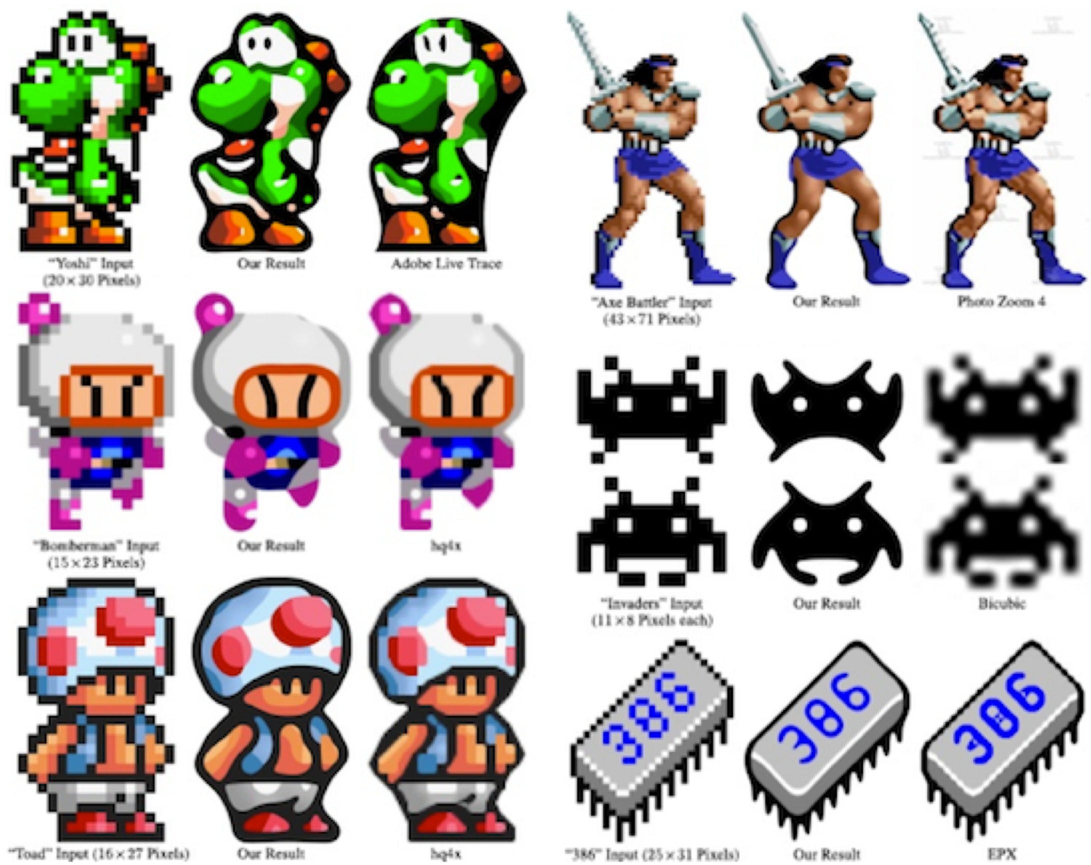
Raster-to-vector graphics

Vector Magic

Simply the Best Auto-Tracer in the World



Depixelating Pixel Art



Modern methods

(a) Bicubic (b) SRCNN (c) A+ (d) RAISR

(e) Bicubic (f) SRCNN (g) A+ (h) RAISR

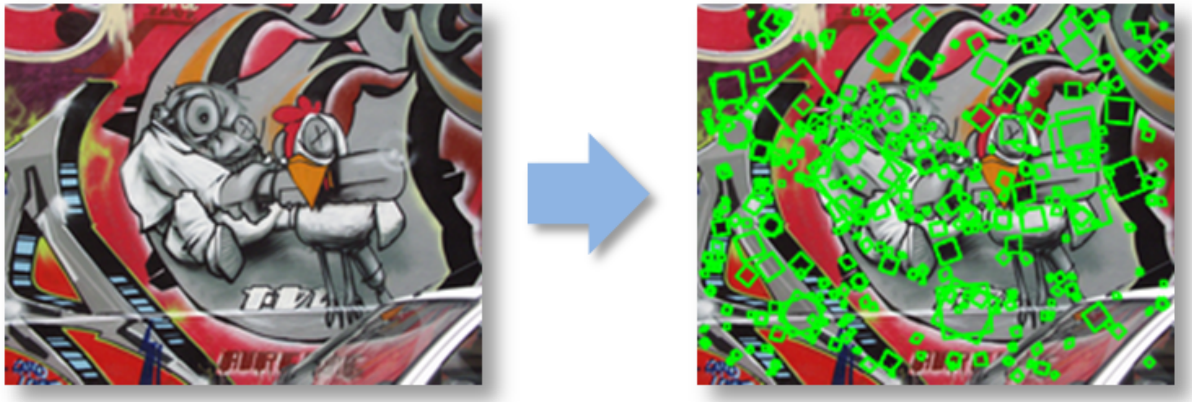
From Romano, et al: RAISR: Rapid and Accurate Image Super Resolution, https://arxiv.org/abs/1606.01299

Questions?

CSCI 497/597P: Computer Vision

Scott Wehrwein

Features - Overview



Reading

- Szeliski: 4.1

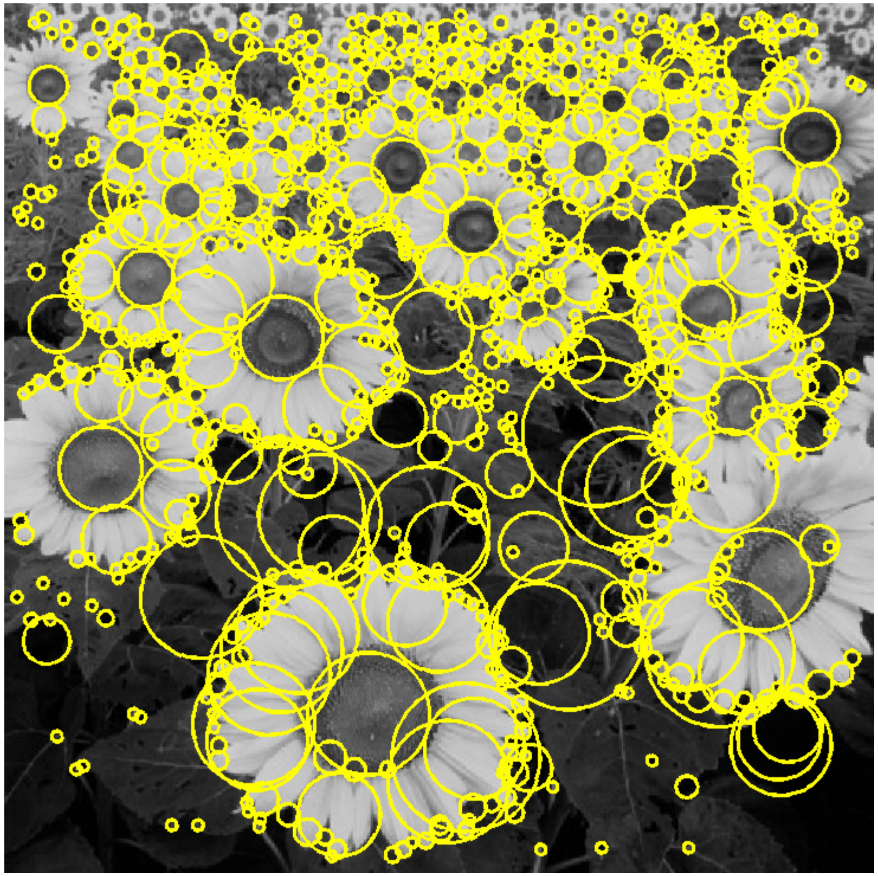
Announcements

- Email me if you’re not enrolled on Piazza
- Please post questions to Piazza so others can benefit from the answers.

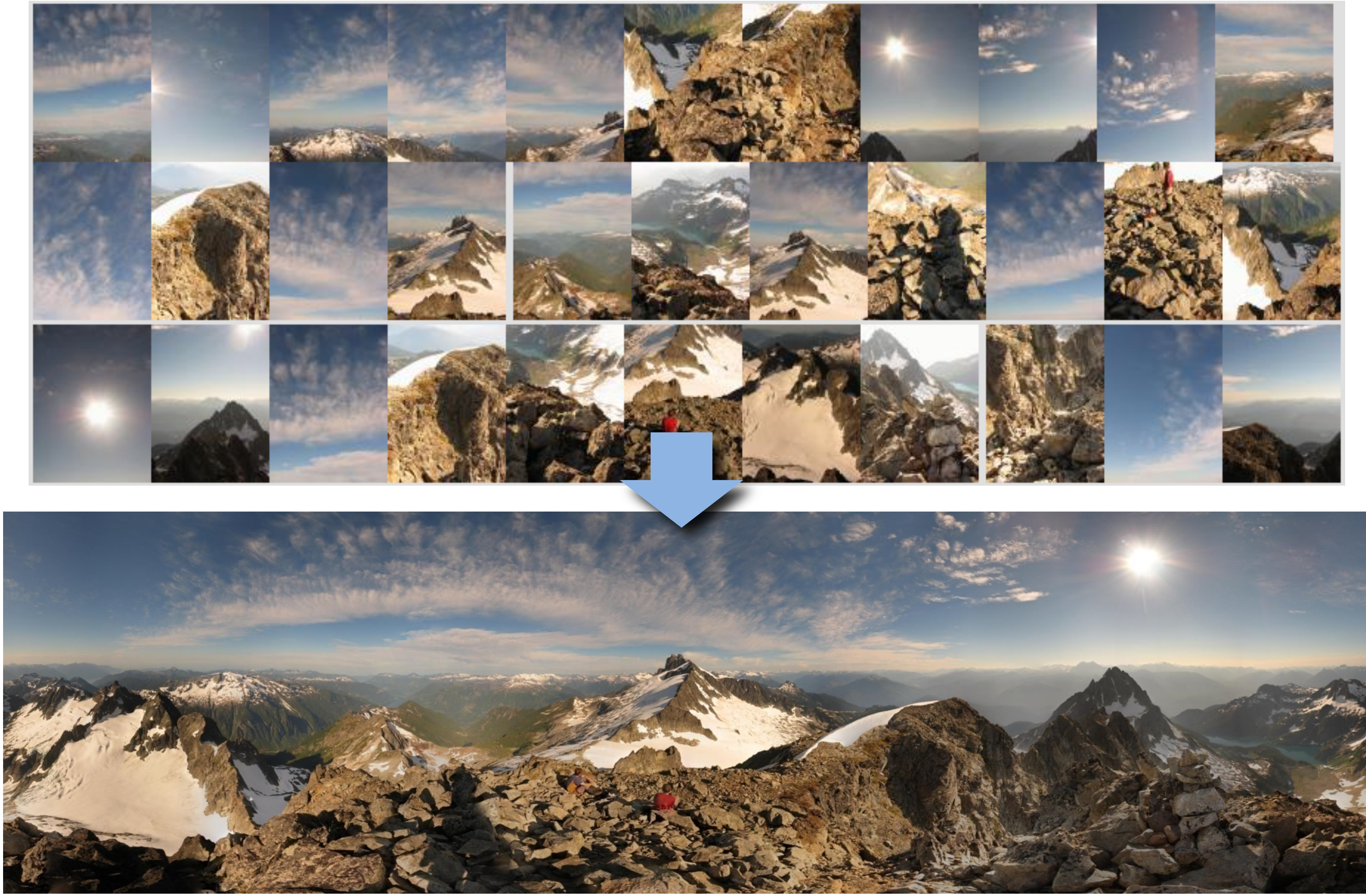
Goals

- Understand the motivation for detecting, describing, and matching local image features.
- Understand the desirable properties of local image features and their descriptors:
  - Uniqueness
  - Invariance
- Gain intuition for corners as image features and why they make good features

Feature extraction: Corners and blobs



Motivation: Automatic panoramas



Credit: Matt Brown

Motivation: Automatic panoramas

GigaPan
http://gigapan.com/

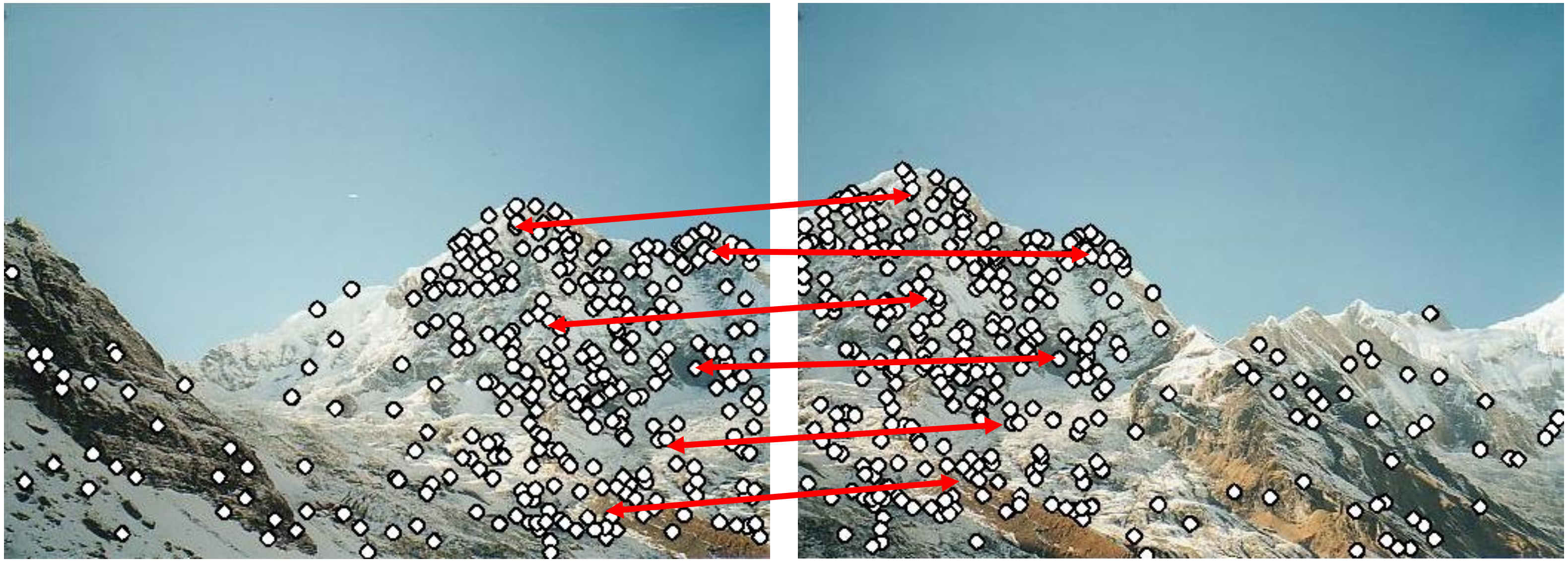
Also see Google Zoom Views:
https://www.google.com/culturalinstitute/beta/project/gigapixels

Why extract features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?

Why extract features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?



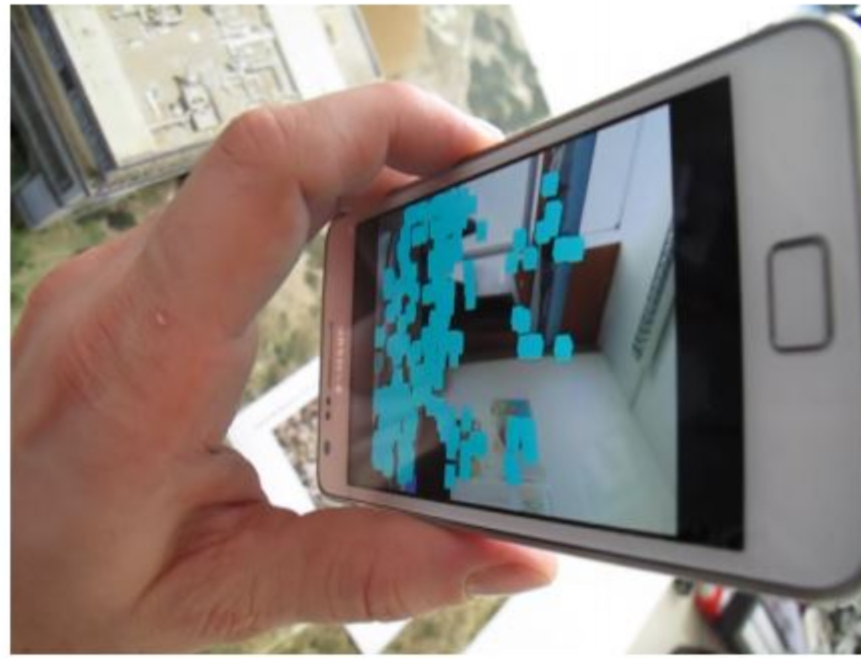
Step 1: extract features
Step 2: match features

Why extract features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?

Step 1: extract features
Step 2: match features
Step 3: align images

Application: Visual SLAM



https://youtu.be/gAbhM59N54k?t=26

Image matching

by [Diva Sian]

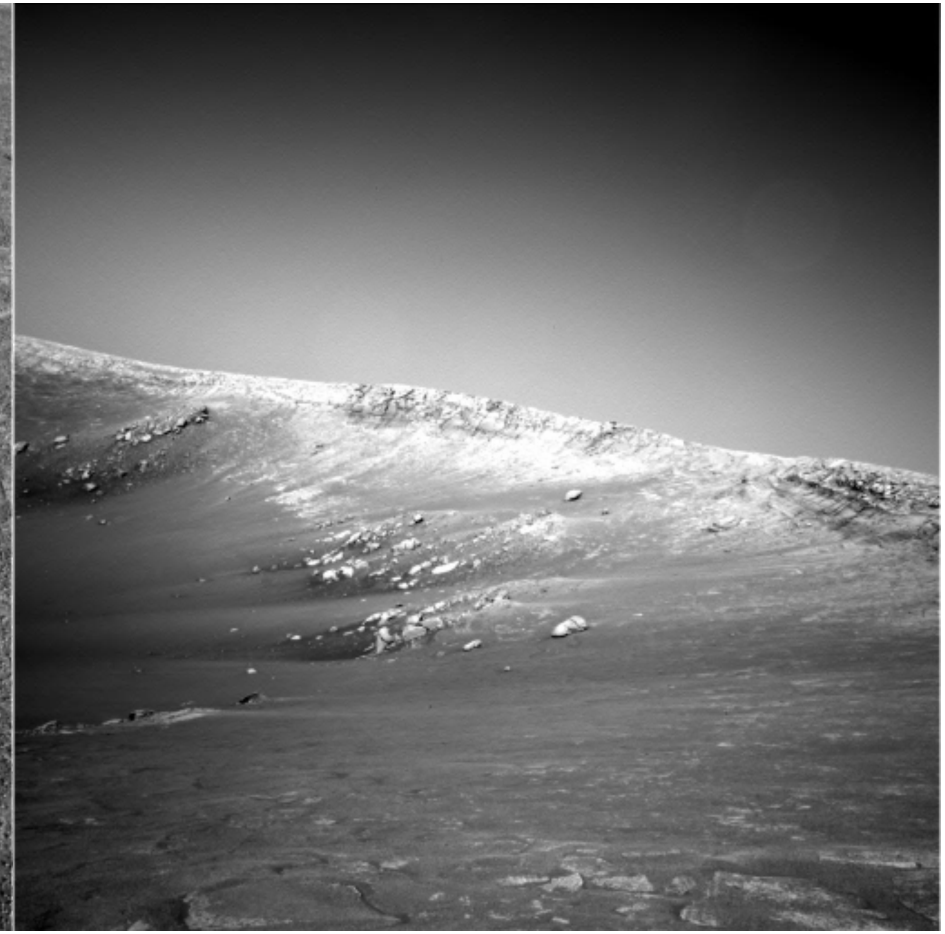
by [swashford]

Harder case

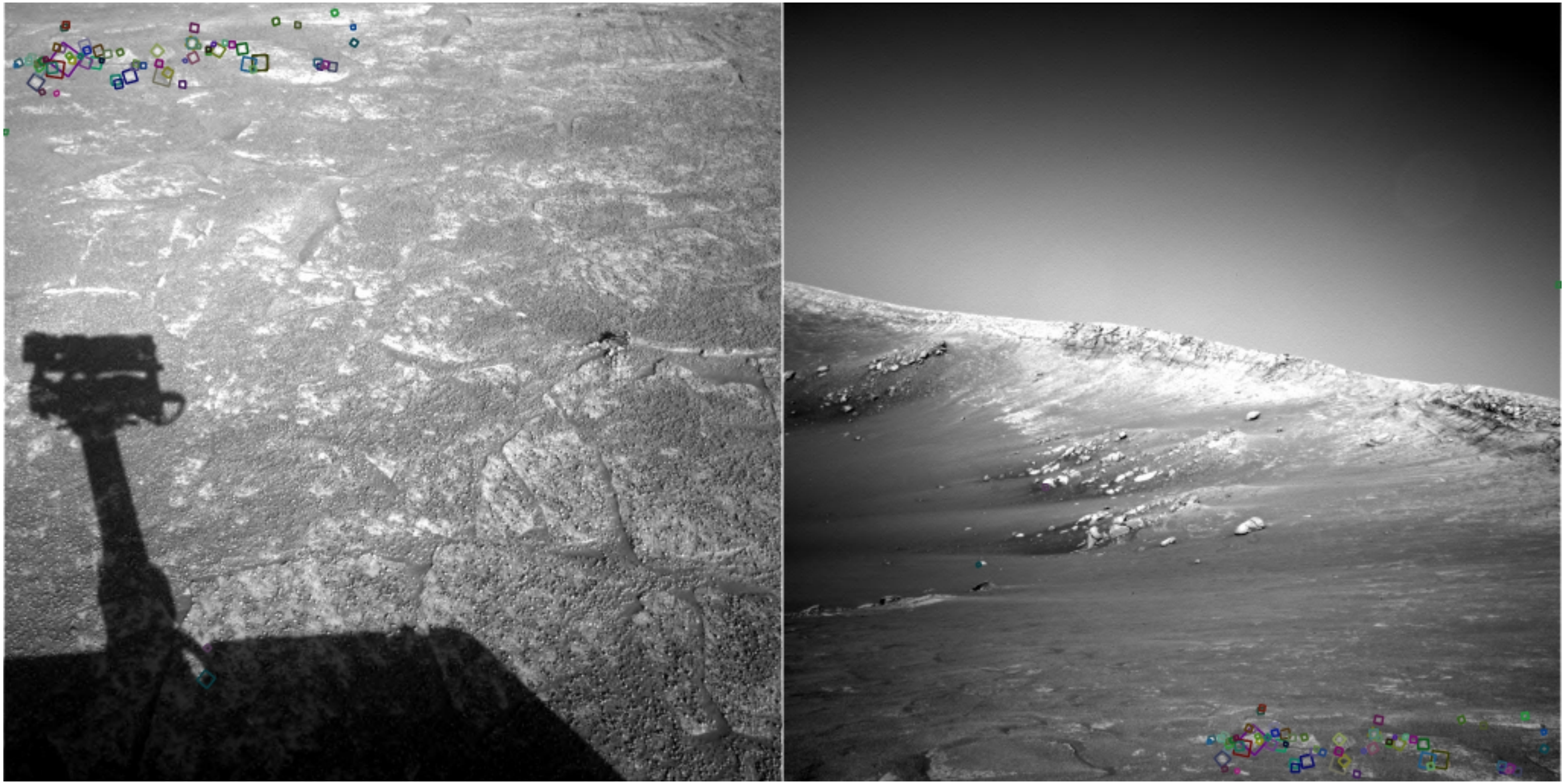
by Diva Sian

by scgbt

Harder still?

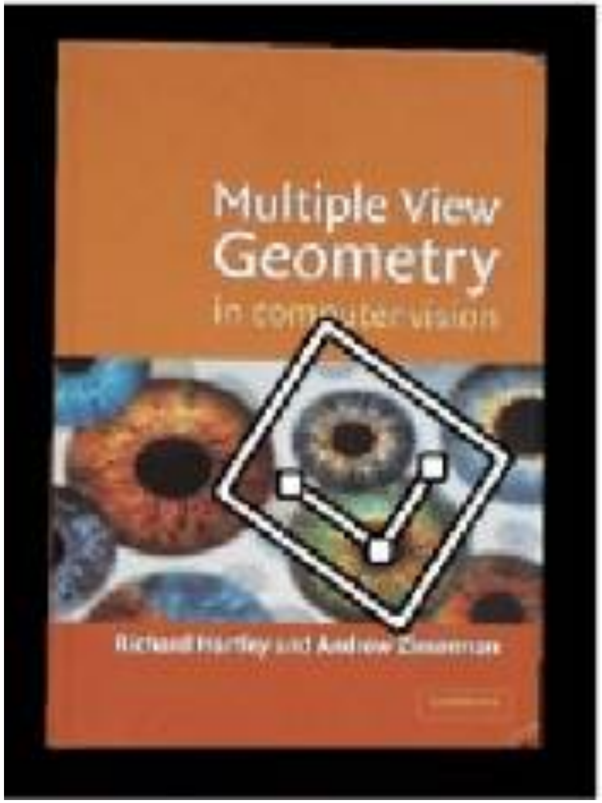


Answer below (look for tiny colored squares…)



NASA Mars Rover images
with SIFT feature matches

Feature Matching



Feature Matching

Advantages of local features

Locality
- features are local, so robust to occlusion and clutter

Quantity
- hundreds or thousands in a single image

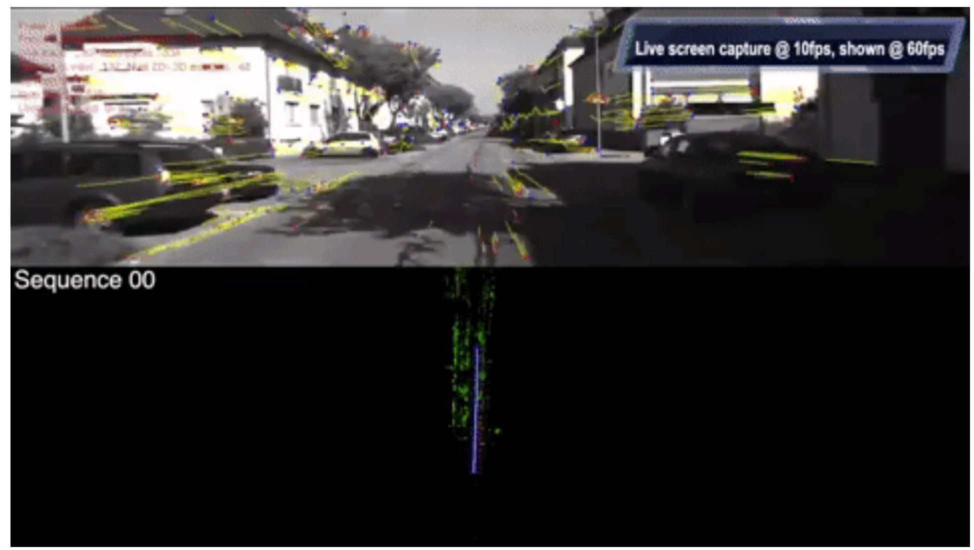
Distinctiveness:
- can differentiate a large database of objects

Efficiency
- real-time performance achievable

More motivation…

Feature points are used for:

- Image alignment
  - (e.g., mosaics)
- 3D reconstruction
- Motion tracking
  - (e.g. for AR)
- Object recognition
- Image retrieval
- Robot navigation
- … other



Approach

1. **Feature detection**: find it

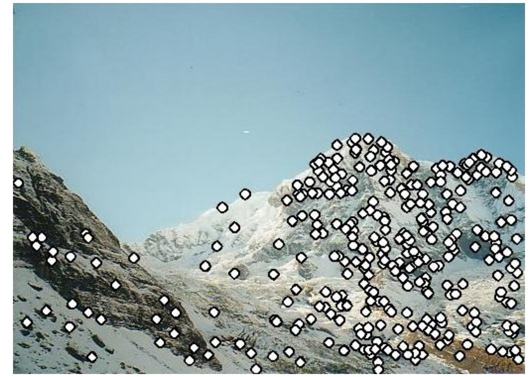
2. **Feature descriptor**: represent it

3. **Feature matching**: match it

Feature tracking: track it, when motion

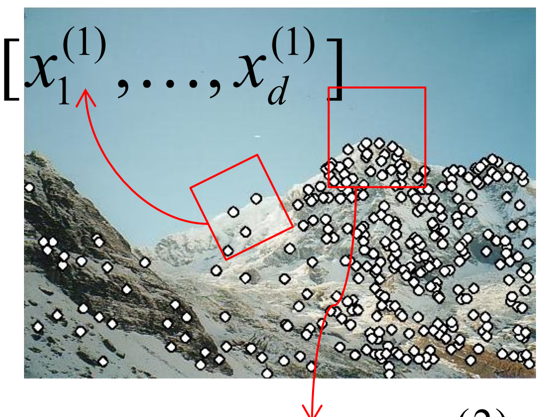
Local features: main components

1) **Detection:** Identify the interest points



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

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



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

3) **Matching:** Determine correspondence between descriptors in two views