Feature Detection and Matching (1 of 2)

Lluís Garrido – lluis.garrido@ub.edu

Department of Analysis and Applied Mathematics
University of Barcelona

Computational Vision, October 2015

Local features




What kind of local features might one use to establish a matching between the images ?

Local features

What kind of local features can be extracted from the left images to encounter the object on the images on the right ?

Local features

Local features for object recognition or matching

- The problem: obtain a representation that allows us to find a particular object we've encountered before (i.e. "find Paco's mug" as opposed to "find a mug").
- Local features based on the appearance of the object at particular interest points.
- Features should be reasonably invariant to illumination changes, and ideally, also to scaling, rotation, and minor changes in viewing direction.

Local Features

Key properties of a good local feature:

- Must be highly distinctive, a good feature should allow for correct object identification with low probability of mismatch.
  Question: How to identify image locations that are distinctive enough ?
- Should be easy to extract and sparse (low number of features).
- A good local feature should be tolerant to
  - image noise
  - changes in illumination
  - scaling of the image
  - rotation
  - minor changes in viewing direction

  Question: how to describe (i.e. construct) the local feature to achieve invariance to the above ?
- Should be easy to match against a (large) database of local features.

Local features

Applications of local features

- Object recognition
- Object tracking
- Panoramic image creation
- Camera pose

Local features

We will focus in these (two) lectures mainly on the Scale Invariant Feature Transform (SIFT). It is an approach for detecting and extracting local feature descriptors that are reasonably invariant to changes in illumination, image noise, rotation, scaling and small changes in viewpoint.

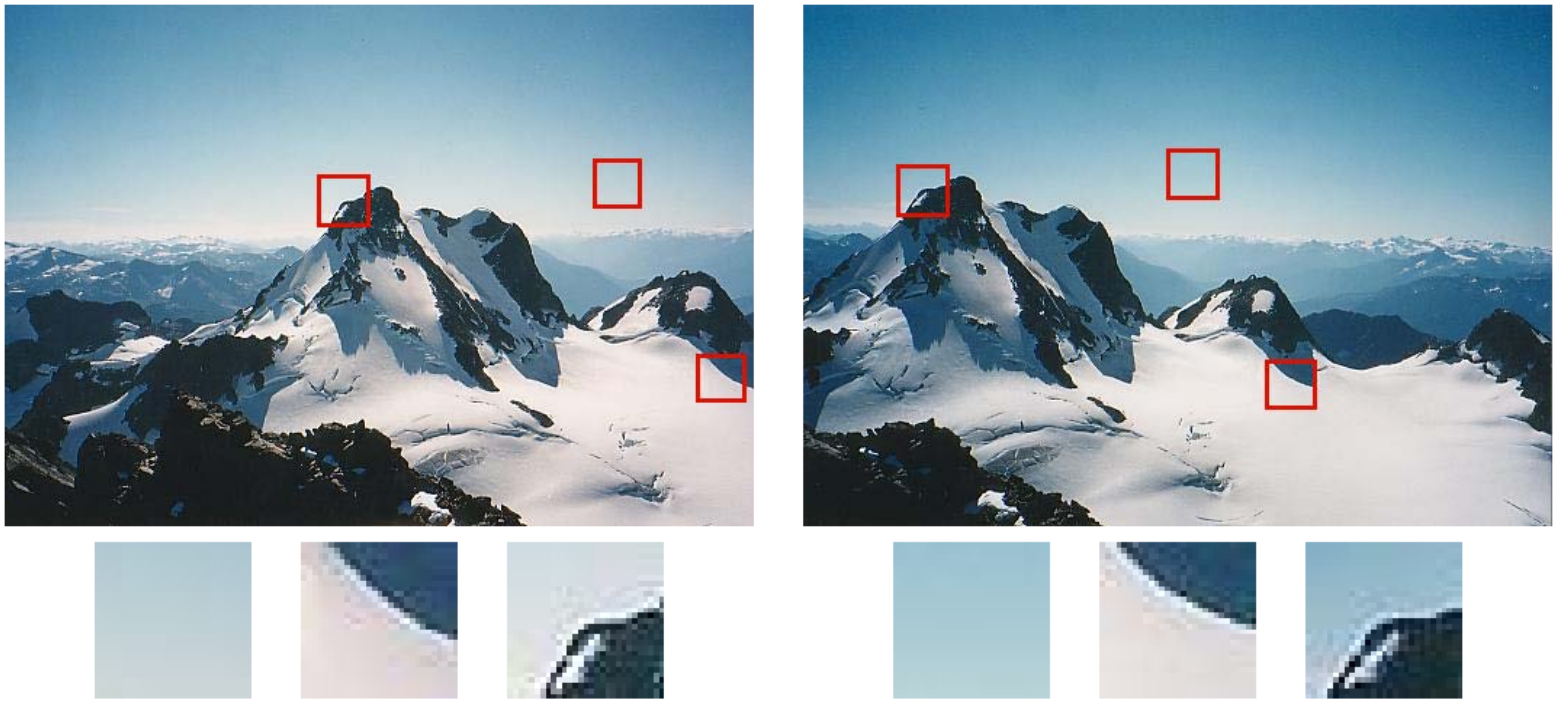
Local features

A feature detection and matching application is composed of the following stages:

1. Feature detection (extraction)
   - Search for interest patches in the image
2. Feature description
   - Describe the interest patches in a compact and stable way
3. Feature matching
   - Search for matching candidates in other images or a database

These slides are mainly focused on stage 1. Some ideas on stage 2 are also given. Stage 3 is left for the next lecture.

Stage 1: Feature Detectors



Which are good patches for candidates to describe ?

Stage 1: Feature Detectors

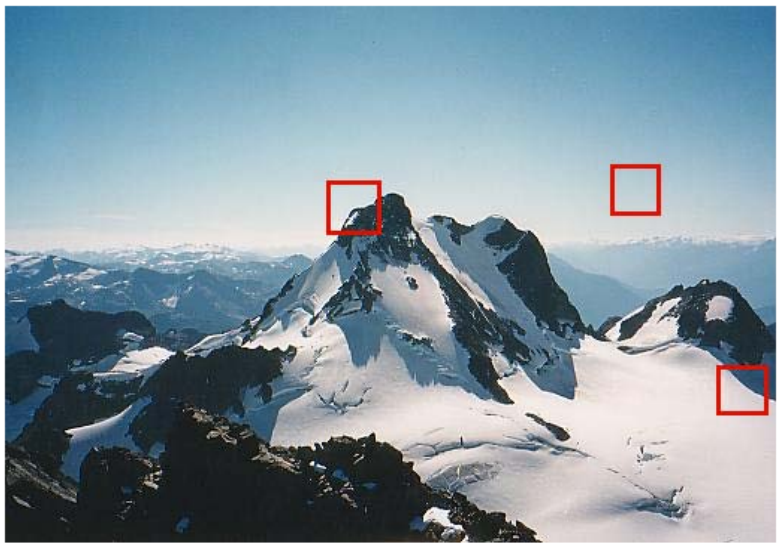
Let us compare the patch R with the patches around it

$$E_{AC}(d_x, d_y) = \sum_{(x,y)\in R} (I(x+d_x, y+d_y) - I(x,y))^2$$

More compactly, $\mathbf{p} = (x,y)^T$ and $\mathbf{d} = (d_x, d_y)^T$

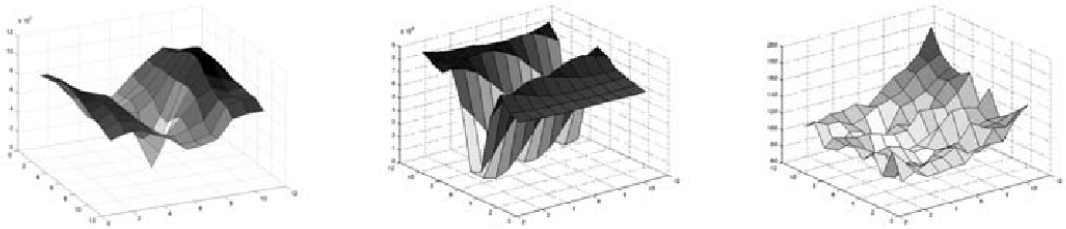
$$E_{AC}(\mathbf{d}) = \sum_{\mathbf{p}\in R} (I(\mathbf{p}+\mathbf{d}) - I(\mathbf{p}))^2$$

This function is called autocorrelation.



Stage 1: Feature Detectors

Example of autocorrelation applied to three locations:

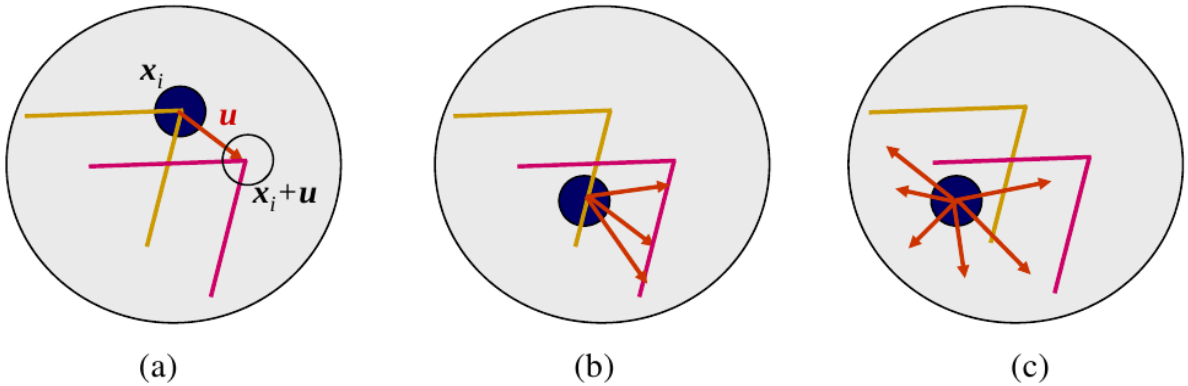


To which point does each autocorrelation surface correspond ?

(a) Flower bed (good unique minimum, a stable corner)
(b) Roof edge (aperture problem)
(c) Cloud (texture-less region)

Stage 1: Feature Detectors

Possibilities for a local feature



(a) Stable corner (patch matches one position)

(b) Aperture problem (patch matches multiple positions)

(c) Texture-less region (patch matches multiple positions)

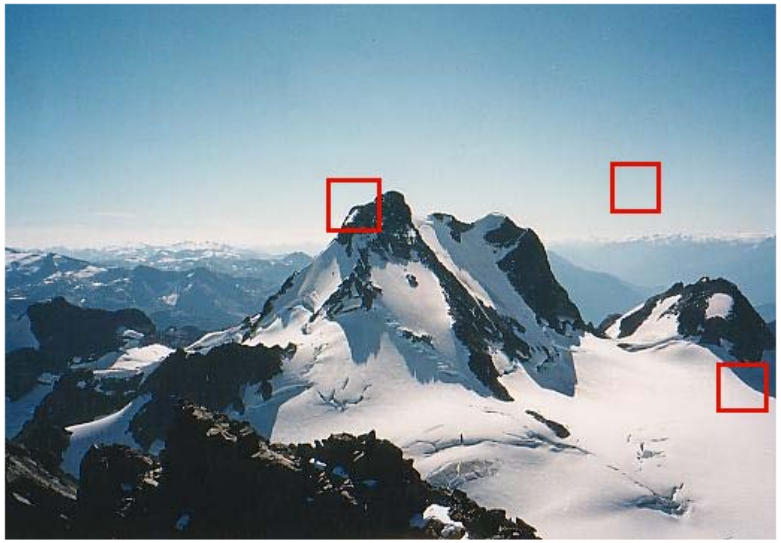
Stage 1: Feature Detectors

How can we easily know, for a given patch R, if it is stable or not ?

$$E_{AC}(\mathbf{d}) = \sum_{\mathbf{p} \in R} \left(I(\mathbf{p}+\mathbf{d}) - I(\mathbf{p}) \right)^2$$

We need to compute the shape of the previous function for each possible patch. This is very time consuming!

Isn't there any other way to tackle the problem ? Maths are here to help! We are going to linearize the previous function using Taylor series.



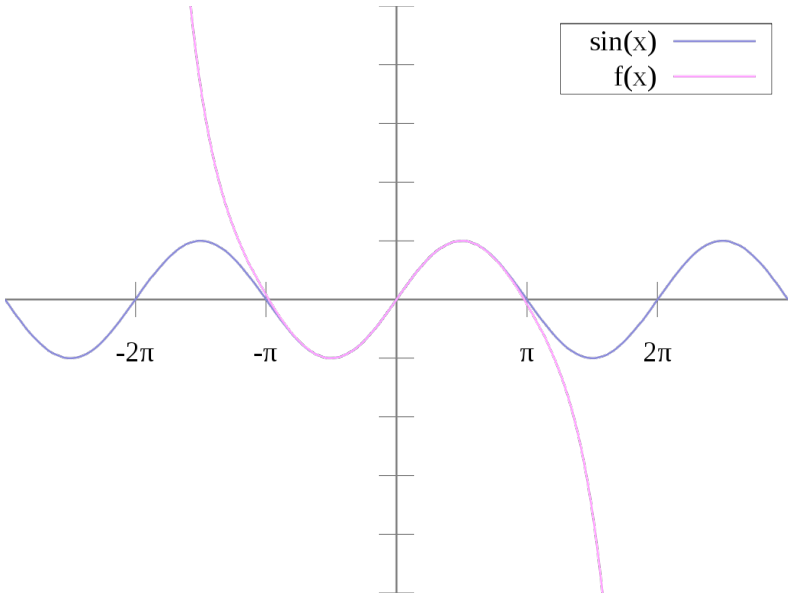
Stage 1: A Reminder on Taylor Series

Just a small reminder on Taylor series. Let $f(x)$ be a one-dimensional real function which has continuous derivatives. Then

$$f(x) = f(x_0) + \frac{1}{1!} f'(x_0)(x - x_0) + \frac{1}{2!} f''(x_0)(x - x_0)^2 + \dots$$

Thus

$$f(x_0 + h) = f(x_0) + \frac{1}{1!} f'(x_0)\, h + \frac{1}{2!} f''(x_0)\, h^2 + \dots$$



- Example (left) showing the approximation of $sin(x)$ using a 7 order polynomial.

Stage 1: A Reminder on Taylor Series

Just a small reminder on Taylor series. Let $I(x, y)$ be a two-dimensional real function with continuous derivatives. Then

$$I(x,y) = I(x_0,y_0) + \frac{\partial I}{\partial x}(x_0,y_0)(x-x_0) + \frac{\partial I}{\partial y}(x_0,y_0)(y-y_0) + \dots$$

Thus

$$I(x_0+d_x,y_0+d_y) = I(x_0,y_0) + \frac{\partial I}{\partial x}(x_0,y_0)d_x + \frac{\partial I}{\partial y}(x_0,y_0)d_y + \dots$$

The previous equation can be compactly written as

$$I(\mathbf{p}+\mathbf{d}) = I(\mathbf{p}) + \frac{\partial I}{\partial x}(\mathbf{p})\,d_x + \frac{\partial I}{\partial y}(\mathbf{p})\,d_y + \dots$$

Let us return to our problem. How can we easily know, for a given patch R, if it is stable or not ?

$$E_{AC}(\mathbf{d}) = \sum_{\mathbf{p} \in R} \left(I(\mathbf{p}+\mathbf{d}) - I(\mathbf{p}) \right)^2$$

We linearize (i.e. we assume small $\mathbf{d} = (d_x, d_y)^T$)

$$I(\mathbf{p}+\mathbf{d}) \approx I(\mathbf{p}) + \frac{\partial I}{\partial x}(\mathbf{p})\, d_x + \frac{\partial I}{\partial y}(\mathbf{p})\, d_y$$

Thus

$$I(\mathbf{p}+\mathbf{d}) - I(\mathbf{p}) \approx \frac{\partial I}{\partial x}(\mathbf{p})\, d_x + \frac{\partial I}{\partial y}(\mathbf{p})\, d_y$$

We have transformed the expression

$$E_{AC}(\mathbf{d}) = \sum_{\mathbf{p} \in R} \left(I(\mathbf{p} + \mathbf{d}) - I(\mathbf{p}) \right)^2$$

into the linear expression

$$E_{AC}(\mathbf{d}) \approx \sum_{\mathbf{p} \in R} \left(\frac{\partial I}{\partial x}(\mathbf{p})\, d_x + \frac{\partial I}{\partial y}(\mathbf{p})\, d_y \right)^2$$

where $\frac{\partial I}{\partial x}(\mathbf{p})$ and $\frac{\partial I}{\partial y}(\mathbf{p})$ are floating point numbers and can be computed from the image I using filters you saw with Petia.

We take this expression

$$E_{AC}(\mathbf{d}) \approx \sum_{\mathbf{p} \in R} \left(\frac{\partial I}{\partial x}(\mathbf{p})\, d_x + \frac{\partial I}{\partial y}(\mathbf{p})\, d_y \right)^2$$

and rewrite it as (to simplify notation)

$$E_{AC}(\mathbf{d}) \approx \sum_{\mathbf{p} \in R} \left(I_x\, d_x + I_y\, d_y \right)^2$$

Note that the inner term can be expressed as a scalar product

$$I_x\, d_x + I_y\, d_y = \begin{pmatrix} I_x \\ I_y \end{pmatrix} \begin{pmatrix} d_x & d_y \end{pmatrix} = (\nabla I)^T \mathbf{d}$$

So, we have that

$$E_{AC}(\mathbf{d}) \approx \sum_{\mathbf{p} \in R} \left((\nabla I)^T \mathbf{d} \right)^2 = \sum_{\mathbf{p} \in R} \mathbf{d}^T \nabla I \, (\nabla I)^T \mathbf{d}$$

Since $\mathbf{d}$ is assumed to be constant for the patch R

$$E_{AC}(\mathbf{d}) \approx \mathbf{d}^T \underbrace{\left(\sum_{\mathbf{p} \in R} \nabla I \, (\nabla I)^T \right)}_{\mathbf{A}} \mathbf{d} = \mathbf{d}^T \mathbf{A} \mathbf{d}$$

where $\mathbf{A}$ is the autocorrelation matrix.

Stage 1: Feature Detectors

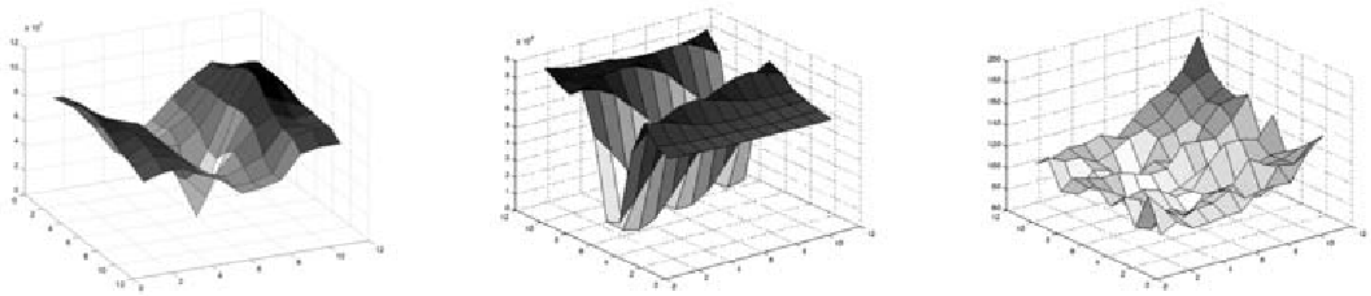
The autocorrelaction function is approximated as

$$E_{AC}(\mathbf{d}) = \sum_{\mathbf{p} \in R} \left(I(\mathbf{p} + \mathbf{d}) - I(\mathbf{p}) \right)^2 \approx \mathbf{d}^T \mathbf{A} \mathbf{d}$$

where the autocorrelation matrix is

$$\mathbf{A} = \begin{pmatrix} \sum_{\mathbf{p} \in R} (I_x)^2 & \sum_{\mathbf{p} \in R} I_x I_y \\ \sum_{\mathbf{p} \in R} I_x I_y & \sum_{\mathbf{p} \in R} (I_y)^2 \end{pmatrix}$$

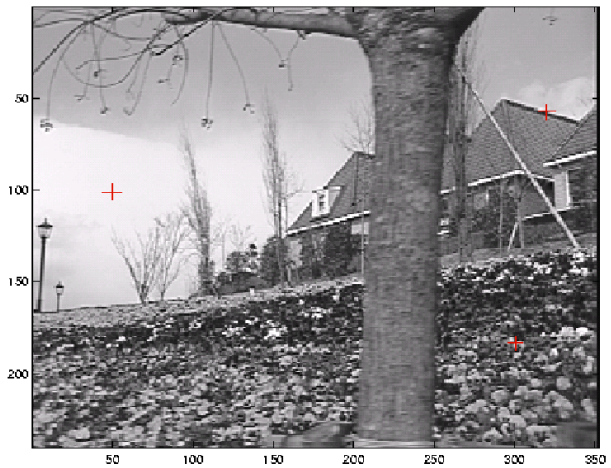
which can be easily computed for each patch R from the original image.

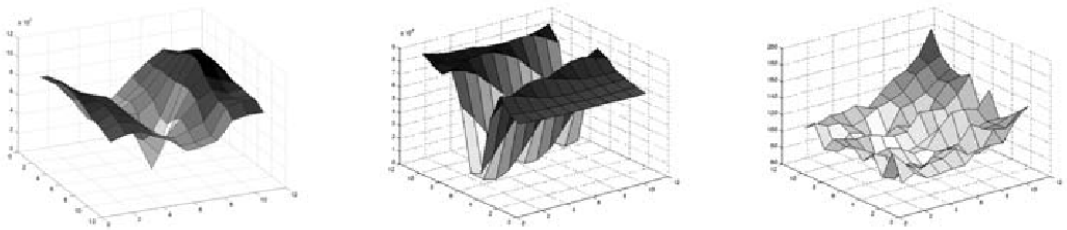


Stage 1: Feature Detectors

Can we infer the shape of $E_{AC}(\mathbf{d})$ using the matrix $\mathbf{A}$? YES!

$$\mathbf{A} = \begin{pmatrix} \sum_{\mathbf{p}\in R} (I_x)^2 & \sum_{\mathbf{p}\in R} I_x I_y \\ \sum_{\mathbf{p}\in R} I_x I_y & \sum_{\mathbf{p}\in R} (I_y)^2 \end{pmatrix}$$





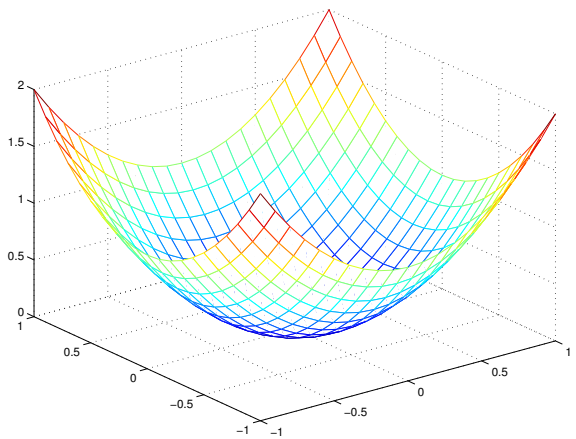
Stage 1: Feature Detectors

Recall we want to know if a feature is stable using the autocorrelation matrix! The function

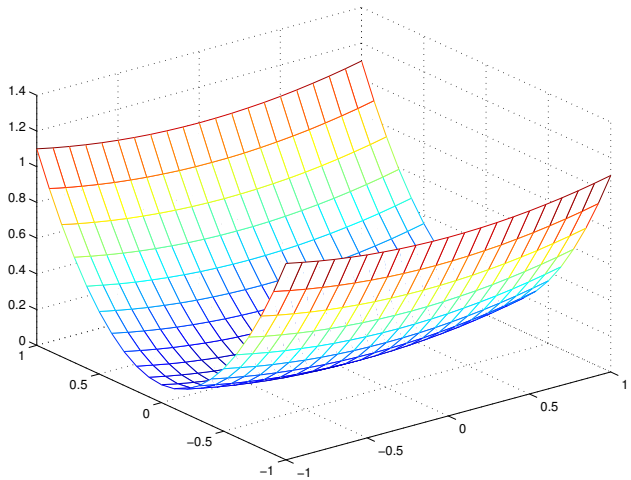
$$E_{AC}(\mathbf{d}) \approx \mathbf{d}^T \mathbf{A} \mathbf{d}$$

is a quadratic function. Intuitively

$$E_{AC}(\mathbf{d}) \approx a\, d_x^2 + b\, d_y^2$$



$a = b = 1$

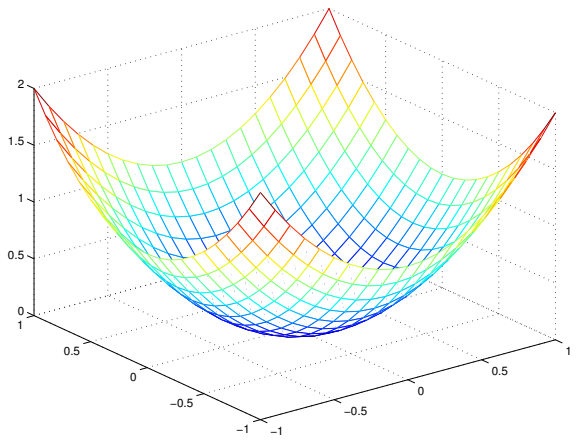


$a = 0.1,\ b = 1$

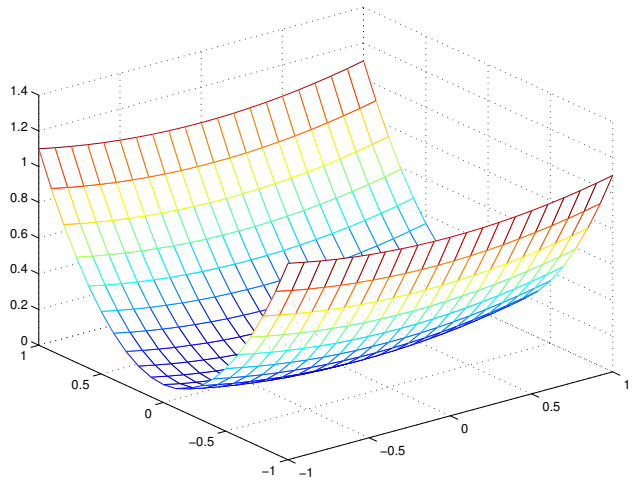
In order to know if a feature is stable we only need to compute the a and b value.

Stage 1: Feature Detectors

The eigenvalues of $\mathbf{A}$, λ_0 and λ_1, are associated to a and b.



$a = b = 1$



$a = 0.1,\ b = 1$

Stable feature points

- To avoid texture-less patches

$$\lambda_0, \lambda_1 > \gamma$$

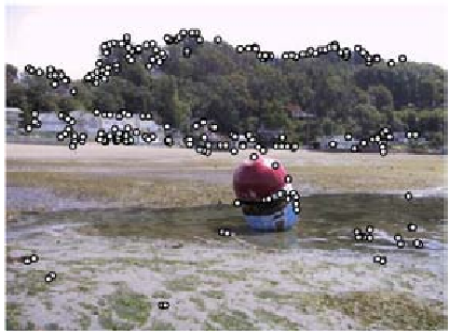
- To avoid aperture problem

$$\frac{\lambda_{max}}{\lambda_{min}} \approx 1$$

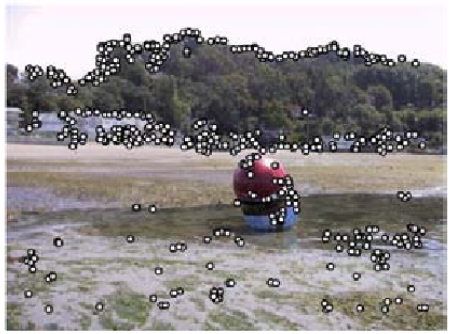
Stage 1: Feature Detectors

There is no need to explicitly compute the eigenvalues of $\mathbf{A}$. One may demonstrate that

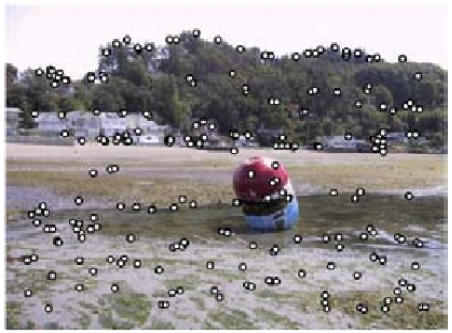
$$\frac{\det \mathbf{A}}{\operatorname{trace} \mathbf{A}} = \frac{\lambda_0 \lambda_1}{\lambda_0 + \lambda_1}$$



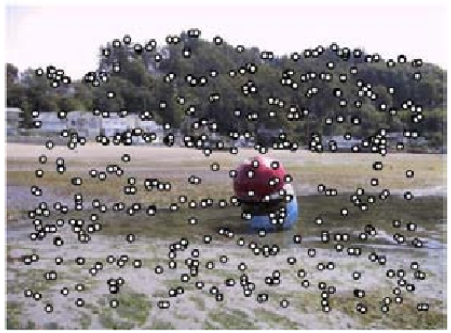
(a) Strongest 250



(b) Strongest 500



(c) ANMS 250, $r = 24$



(d) ANMS 500, $r = 16$

Feature points are located in in textured regions

Significant feature points within a region of radius r

Stage 1: Feature Detectors

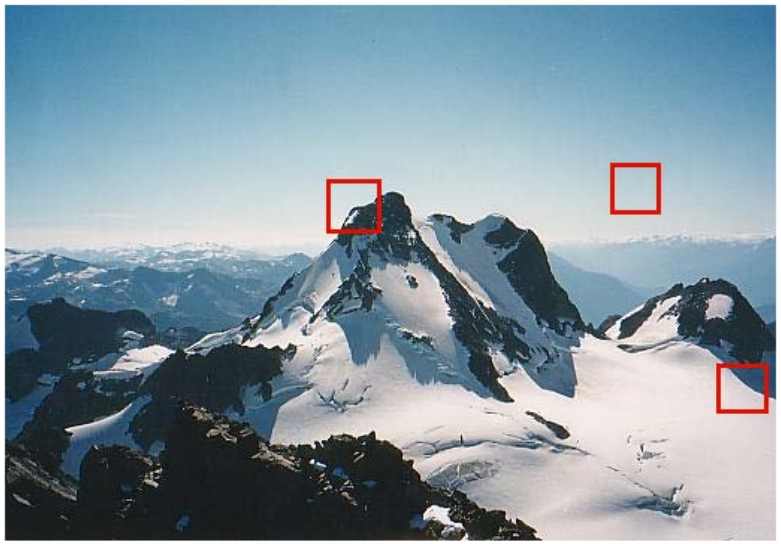
Let us summarize. The autocorrelation function is

$$E_{AC}(\mathbf{d}) \approx \mathbf{d}^T \mathbf{A} \mathbf{d}$$

A measure of significance for the patch R is

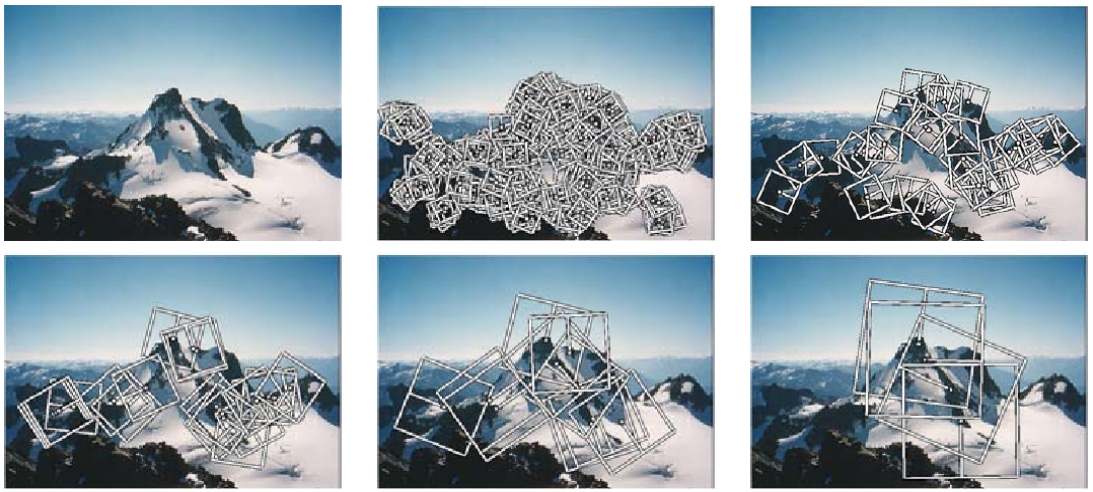
$$\frac{\det \mathbf{A}}{\operatorname{trace} \mathbf{A}} = \frac{\lambda_0 \lambda_1}{\lambda_0 + \lambda_1}$$

where λ_0 and λ_1 are the eigenvalues of $\mathbf{A}$.



Stage 1: Feature Detectors

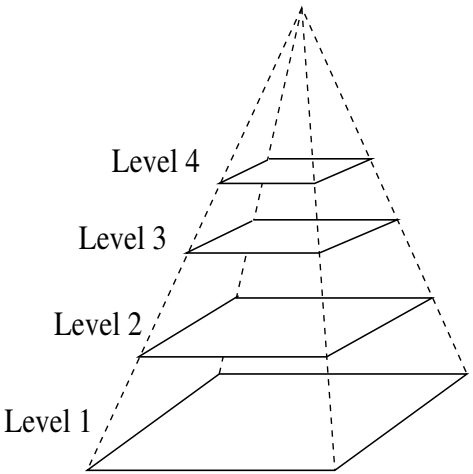
What about scale, that is, the size of the patch R ? Features can be extracted at many different scales!



Rather than extracting features at only one scale a simple way to proceed is to different use sizes of R. This, however, increases computational completely as R increases.

Stage 1: Feature Detectors

Instead of increasing the size of the patch R for a given image, one may reduce the size of the images and fix the size of R through the scales.

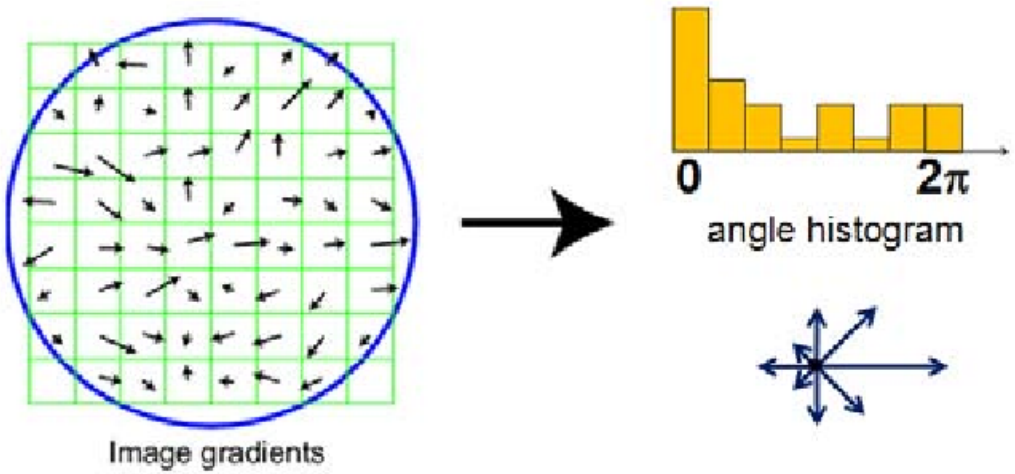


- For each scale significant patches with size R are extracted.
- Level 1 corresponds to original image, level 2 to subsampled image, and so on.
- This technique reduces computational effort (with respect increasing the size of R with a fixed image).

Stage 1: Feature Detectors

In addition to scale invariance, orientation invariance also is useful. How is this done ?

- Estimate dominant orientation at each keypoint
  - Compute the mean of gradient (not always reliable)
  - Use the histogram of orientations (HOG) as done in SIFT).



Stage 1: Feature Detectors

Let us summarize. We have seen

- How to know, for a given patch R, if it is significant.
- The necessity to test for patches at different resolution scales.
- For a given R, its orientation can be estimated using the HOG.

Let us now see how the SIFT descriptor works.

Stage 1: Scale Invariant Feature Transform (SIFT)

Scale Invariant Feature Transform Descriptors (SIFT) are reasonably invariant to scale, rotation, image noise, changes in illumination and small changes in viewpoint. In SIFT the significant patches R are called keypoints.

Steps to perform to construct the description[1]:

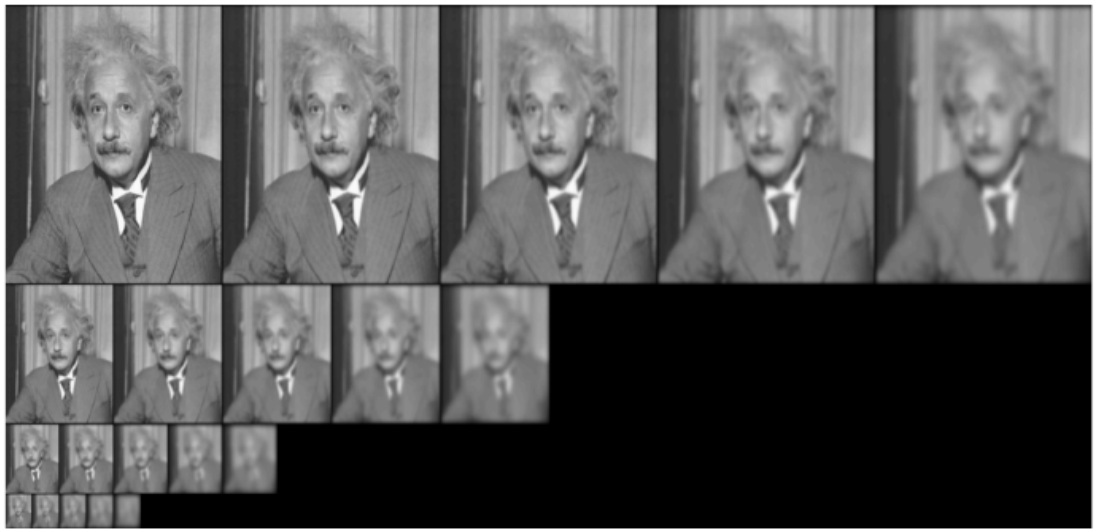
1. Select some candidate keypoints using scale-space extrema detection.
2. Check which of candidate keypoint are significant.
3. Compute the orientation of the keypoint.
4. Generate a compact description of the keypoint robust to image noise, changes in illumination and small changes in viewpoint.

[1] Steps 1 to 3 (resp. step 4) correspond to the first stage (resp. second) of the general problem of feature detection and matching application.

Stage 1: Scale Invariant Feature Transform (SIFT)

Step 1. Select some candidate keypoints using scale-space extrema detection.

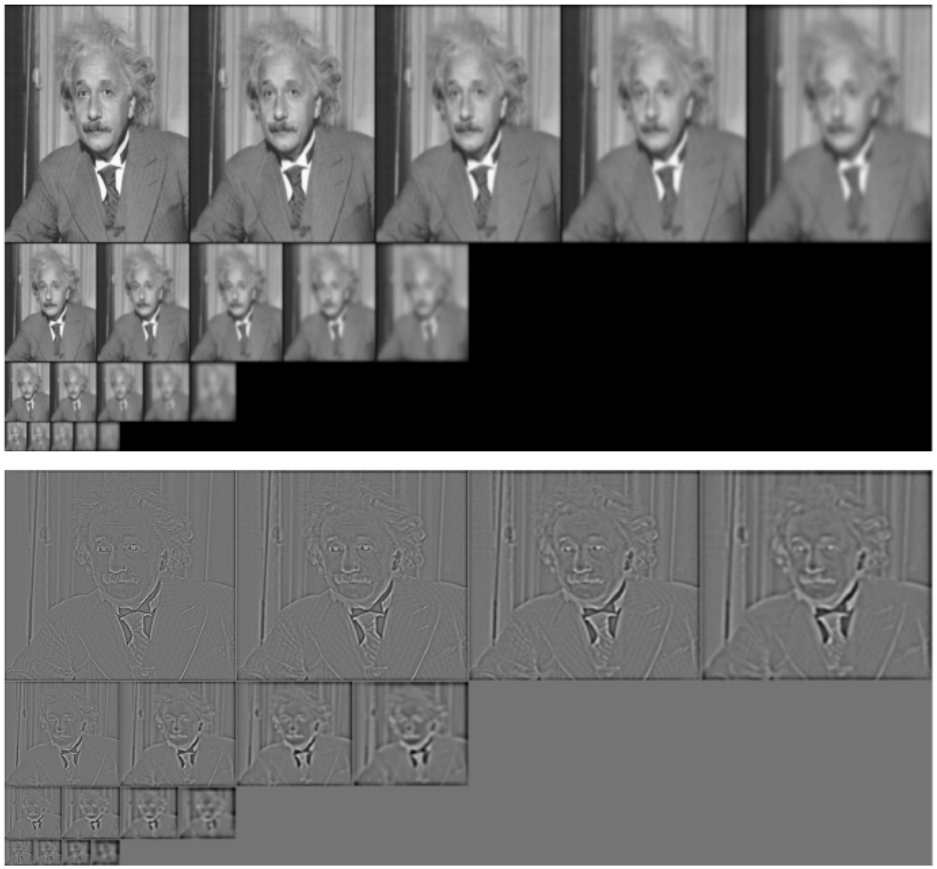
Instead of checking each possible patch R for each possible scale, we select some interest candidate points. For that issue, images are first convolved at each scale with Gaussian filters.

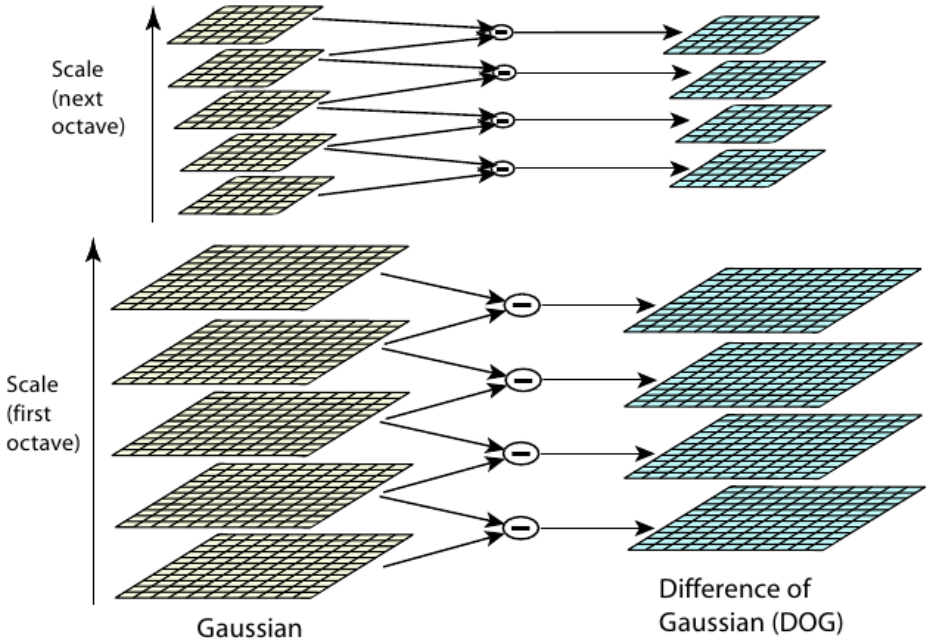


Stage 1: Scale Invariant Feature Transform (SIFT)

Step 1. Select some candidate keypoints using scale-space extrema detection.

The Difference-of-Gaussian images are computed from the difference of blurred images.

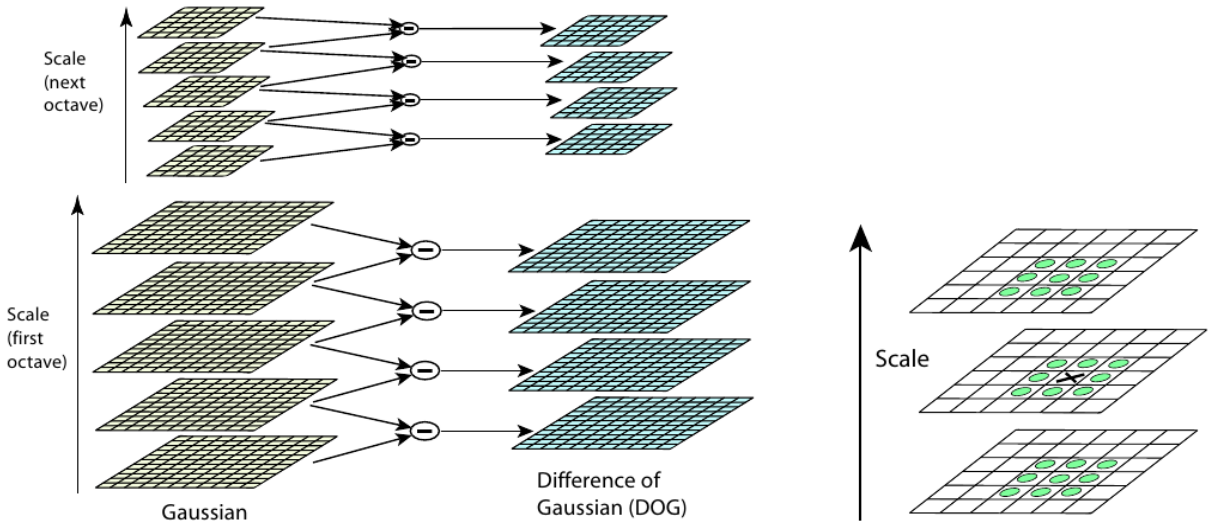




Stage 1: Scale Invariant Feature Transform (SIFT)

Step 1. Select some candidate keypoints using scale-space extrema detection.

Candidate keypoints are identified as local maxima or minima of the Difference-of-Gaussian images across scale.

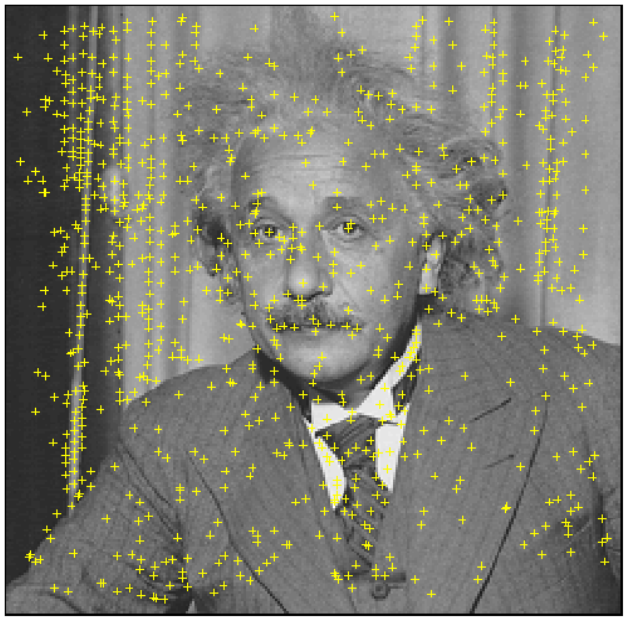


Each pixel in the Difference-of-Gaussian images is compared to its 8 neighbors at the same scale, plus the 9 corresponding neighbors at neighboring scales. If the pixel is a local maximum or minimum, it is selected as candidate keypoint.

Stage 1: Scale Invariant Feature Transform (SIFT)

Step 1. Select some candidate keypoints using scale-space extrema detection.

These are the candidate keypoints (at which we locate the patch R)



We now need to check which of the candidate points are really significant.

Stage 1: Scale Invariant Feature Transform (SIFT)

Step 2. Check which of candidate keypoint are significant.

The candidate keypoints are tested for significance by a two-step procedure

1. Test if the absolute value of the Difference-of-Gaussian image D at the minimum or maximum $\hat{x}$ is greater than a threshold:

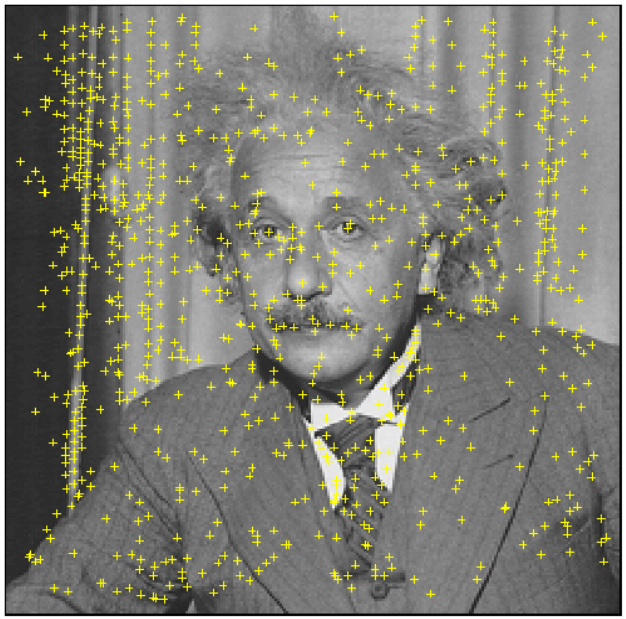
$$|D(\hat{x})| > \gamma$$

   This allows to reduce sensitivity to noise.

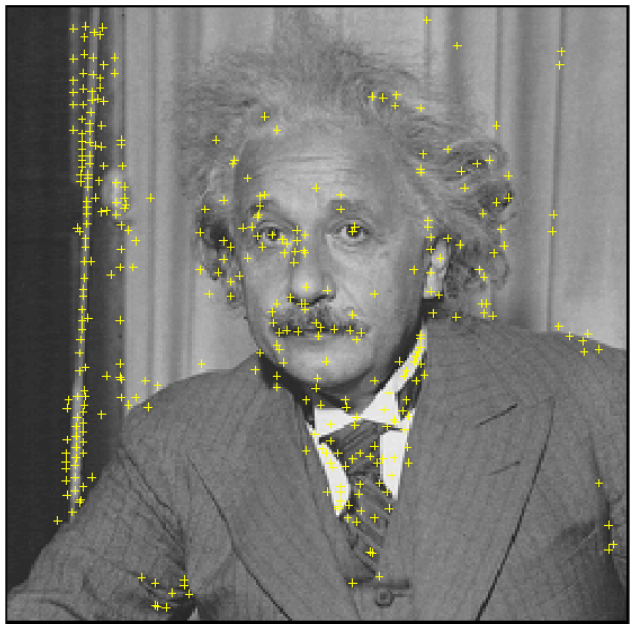
2. Test if the auto-correlation of the Difference-of-Gaussian images D at $\hat{x}$ indicates significance. This step allows to remove badly localized candidates (i.e. at a straight edge).

Stage 1: Scale Invariant Feature Transform (SIFT)

Step 2. Check which of candidate keypoint are significant.



Candidate points



Points with $|D(\hat{x})| > \gamma$

Step 2. Check which of candidate keypoint are significant.

The auto-correlation at each candidate point of the DoG images D is then computed

$$H = \begin{pmatrix} \sum_{\mathbf{p}\in R} \left(\frac{\partial D}{\partial x}\right)^2 & \sum_{\mathbf{p}\in R} \frac{\partial D}{\partial x}\frac{\partial D}{\partial y} \\ \sum_{\mathbf{p}\in R} \frac{\partial D}{\partial x}\frac{\partial D}{\partial y} & \sum_{\mathbf{p}\in R} \left(\frac{\partial D}{\partial y}\right)^2 \end{pmatrix}$$

where D is the considered DoG image.

Assume $\lambda_{max} = r\,\lambda_{min}$, note that

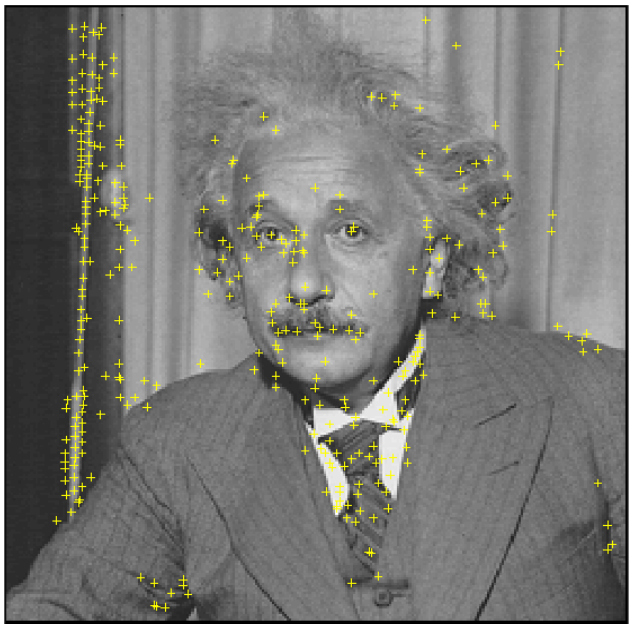
$$\kappa = \frac{\mathrm{Tr(H)}^2}{\mathrm{Det(H)}} = \frac{(\lambda_{max}+\lambda_{min})^2}{\lambda_{max}\lambda_{min}} = \frac{(r+1)^2}{r}$$

only depends on r and is minimum for $r = 1$. Those candidates with $\kappa \leq 12.1$ ($r \leq 10$) may be accepted as significant.

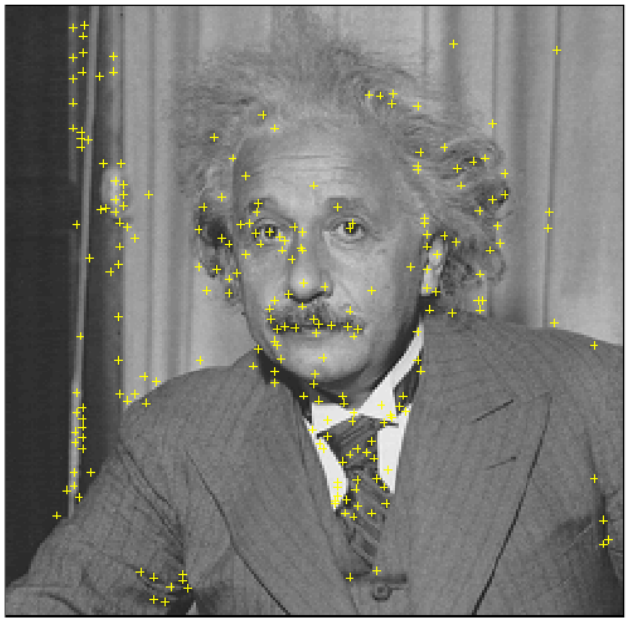
Stage 1: Scale Invariant Feature Transform (SIFT)

Step 2. Check which of candidate keypoint are significant.

The previous condition allows to improve badly localized candidate points.



Points with $|D(\hat{x})| > \gamma$

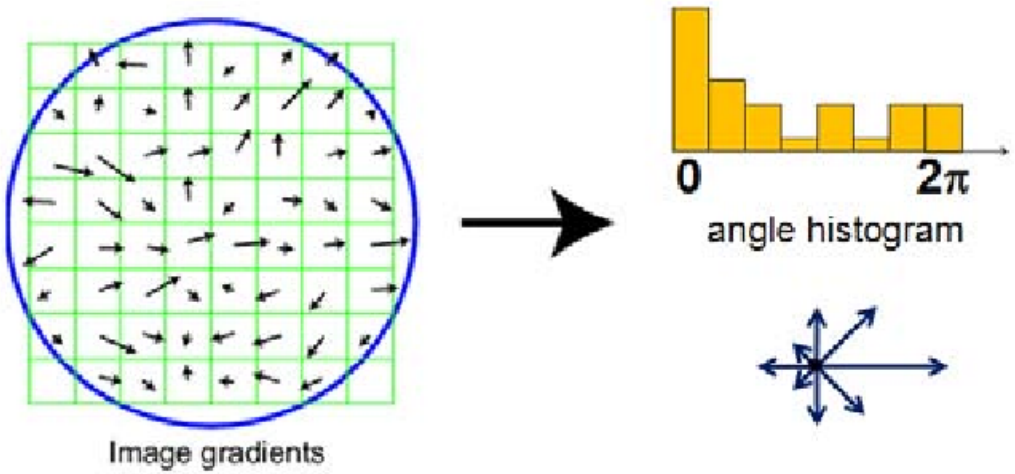


Significant points

Stage 1: Scale Invariant Feature Transform (SIFT)

Step 3. Compute the orientation the keypoint.

To determine the orientation of the keypoint, a gradient orientation histogram is computed in the neighborhood of the keypoint (using the corresponding Gaussian image of the scale-space). Peaks in the histogram correspond to dominant orientations.

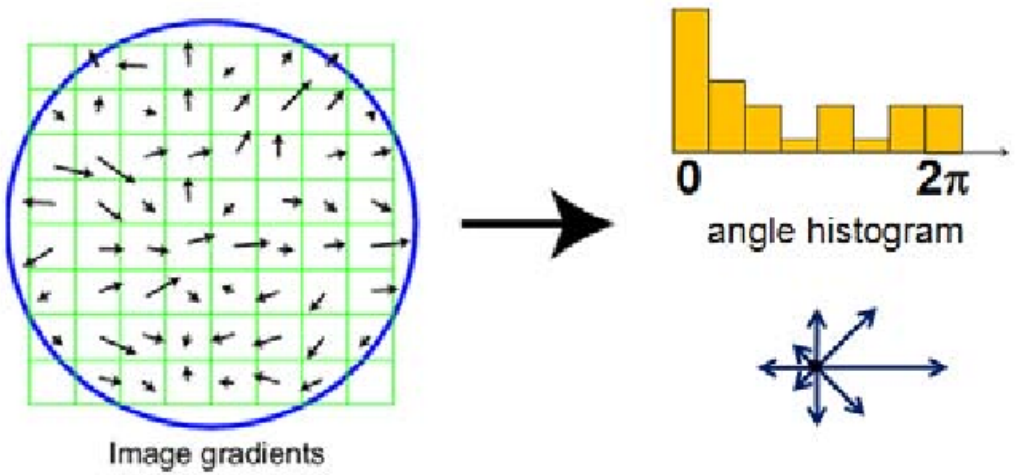


Stage 1: Scale Invariant Feature Transform (SIFT)

Step 3. Compute the orientation the keypoint.

In SIFT

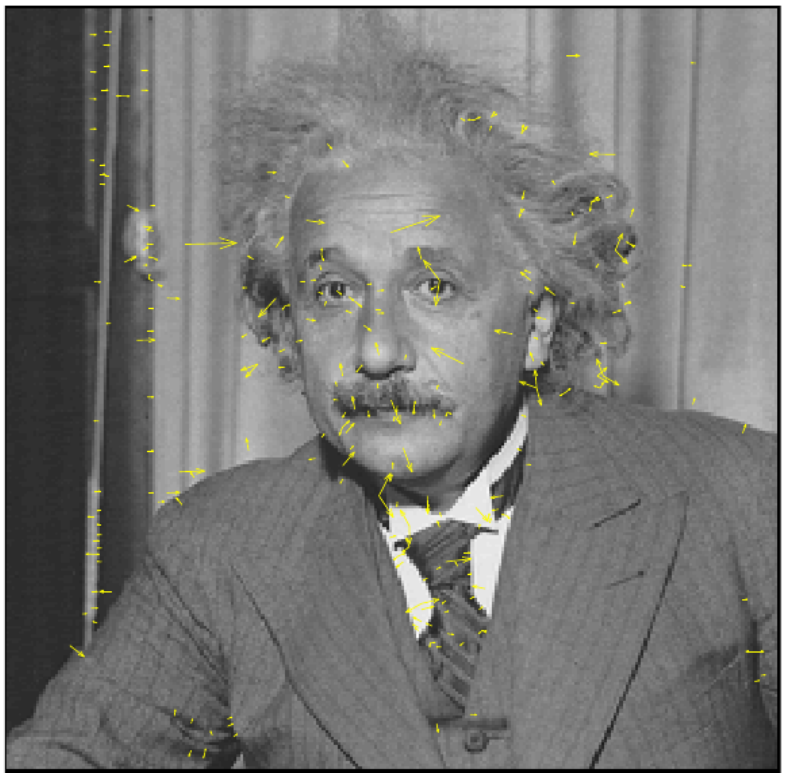
- A separate description is created for the direction corresponding to the histogram maximum, and any other direction within 80% of the maximum value.
- All the properties of the description are measured relative to the keypoint orientation, this provides invariance to rotation.



Stage 1: Scale Invariant Feature Transform (SIFT)

Step 3. Compute the orientation the keypoint.

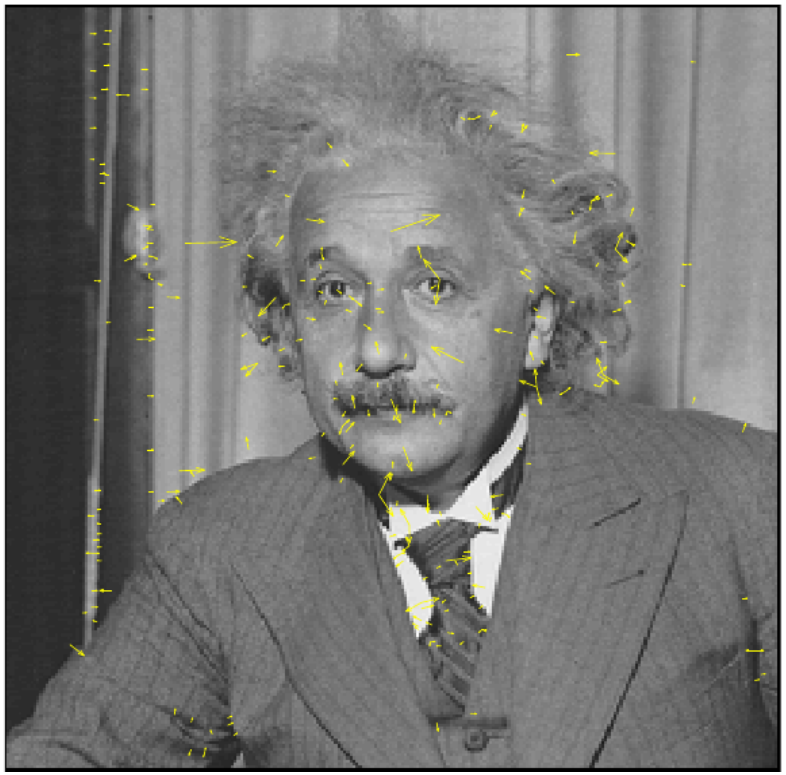
Final keypoints, arrows indicate scale and orientation.

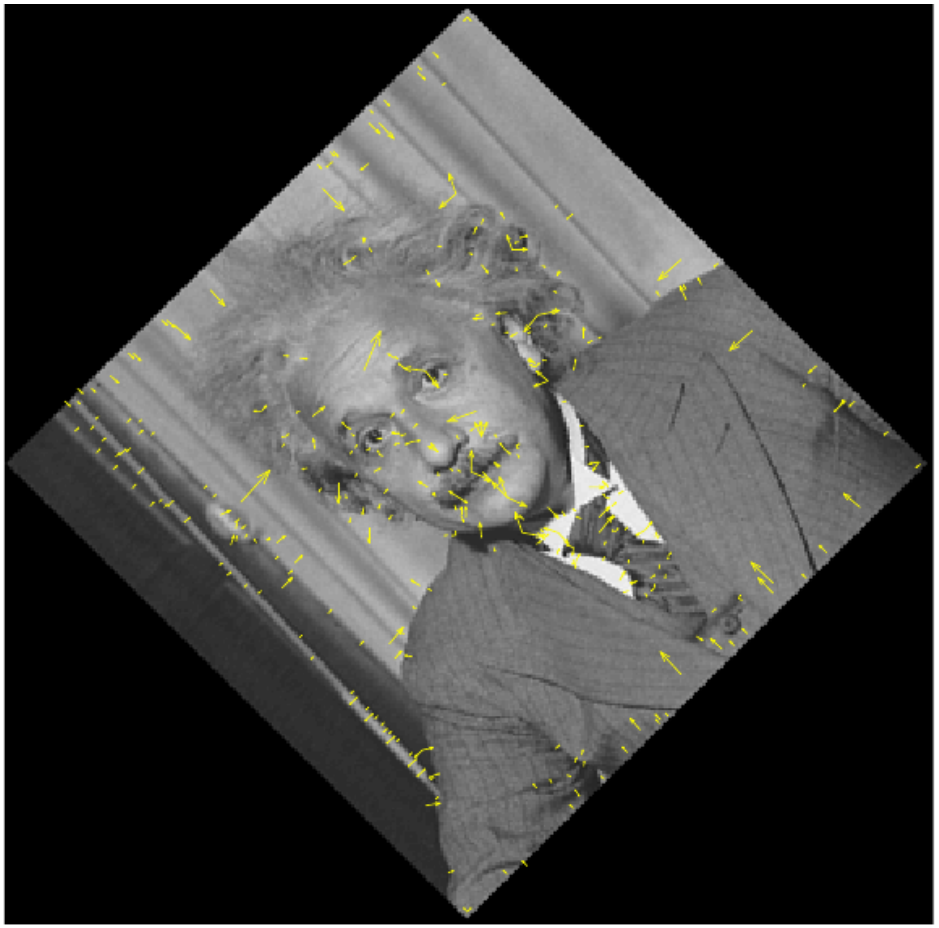


Stage 1: Scale Invariant Feature Transform (SIFT)

Step 3. Compute the orientation the keypoint.

Final keypoints, arrows indicate scale and orientation.





Local features

Recall that a feature detection and matching application is composed of the following stages:

1. Feature detection (extraction)
   - Search for interest points in the image: done!
2. Feature description
   - Describe the interest point in a compact and stable way
3. Feature matching
   - Search for matching candidates in other images

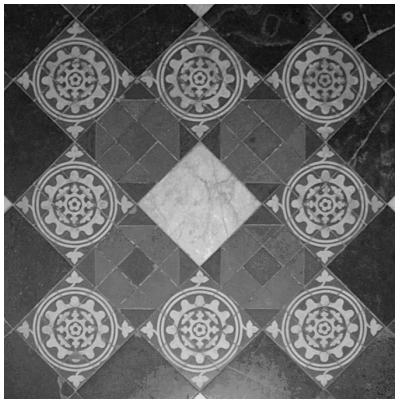
We focus now (very briefly) on stage 2.

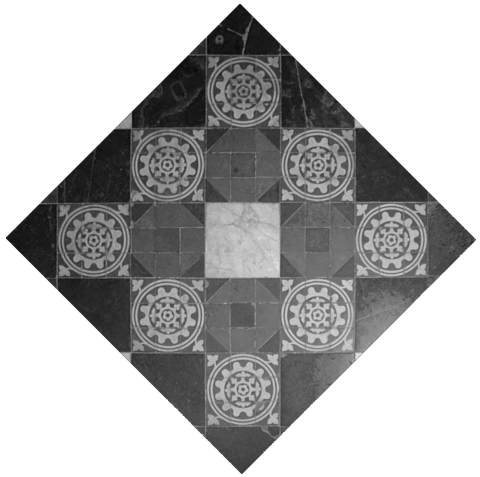
Stage 2: Feature Description

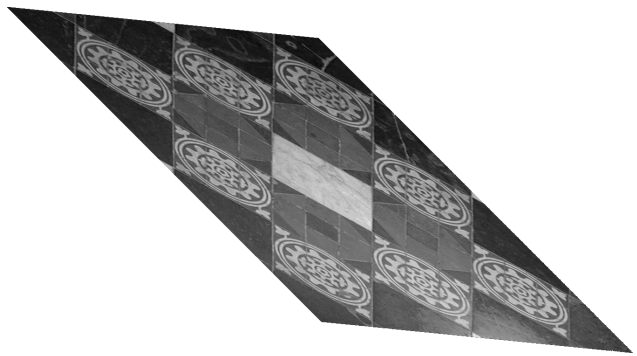
We want to describe the feature points in a consistent manner to match them afterward.

Feature description should be ideally robust to

- Local scale
- Orientation
- Viewpoint changes



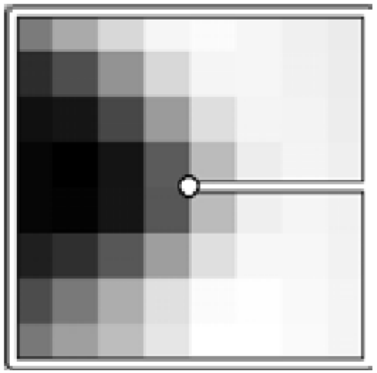




Stage 2: Feature Description

Bias and again normalization (MOPS) descriptor: uses an 8×8 patch with gray-values normalized according to the computed orientation.

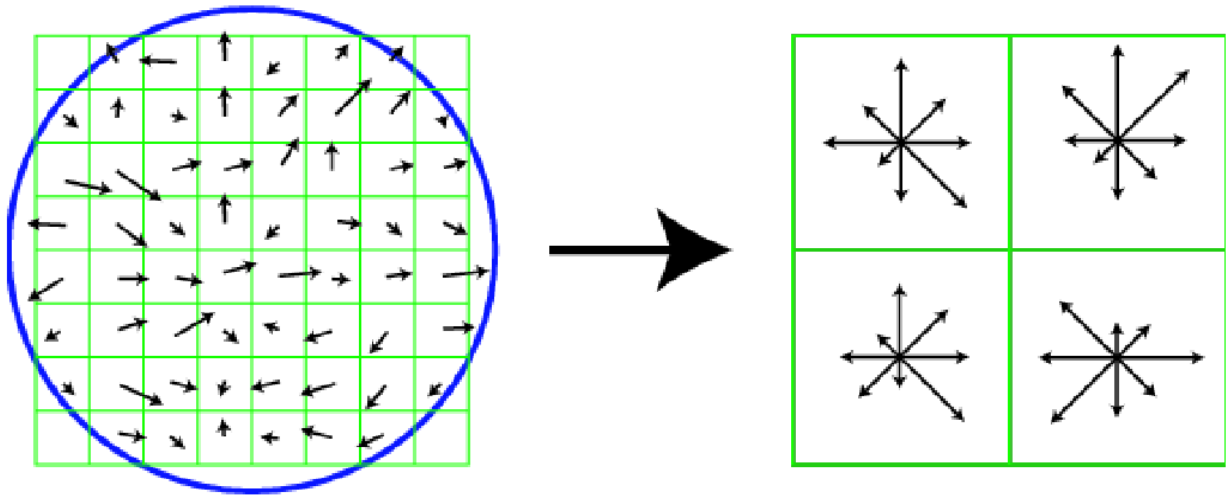
- Easy to implement.
- Not robust to illumination changes.
- Useful when dealing with images that do not get a lot distorted due to rotations or viewpoint changes.



Stage 2: Feature Description

Scale Invariant Feature Transform (SIFT)

- Uses histogram of gradients (relative to computed orientation) to describe pixels around the keypoint
- Gradients are computed in the corresponding spale-space image.



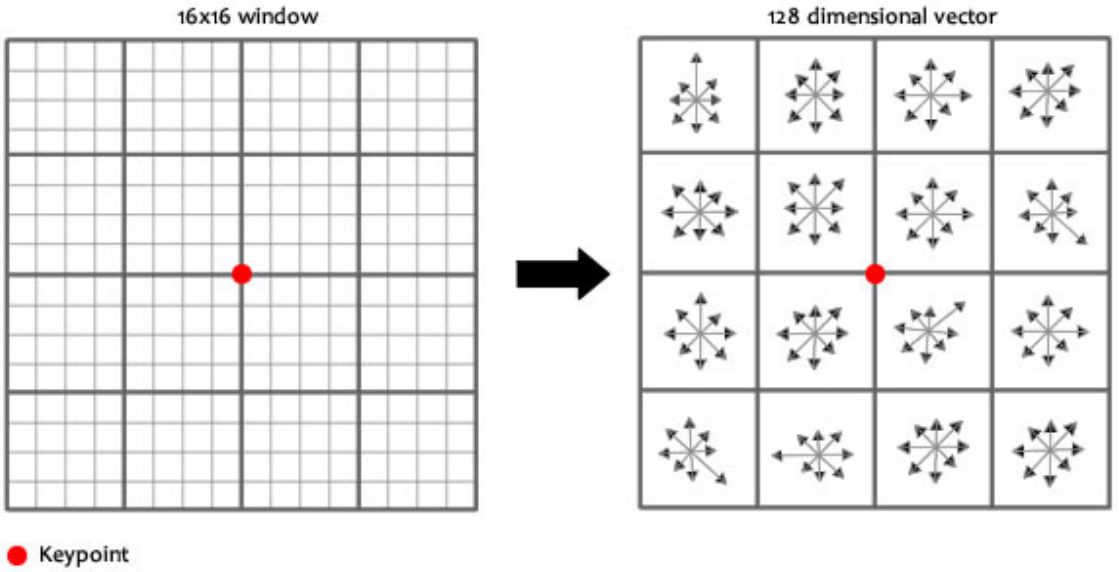
(a) image gradients (b) keypoint descriptor

The figure shows an 8×8 patch and 2×2 descriptor array with 8 bin histograms.

Stage 2: Feature Description

Scale Invariant Feature Transform (SIFT)

- The figure shows an 8×8 patch and 2×2 descriptor array with 8 bin histograms.
- Original SIFT descriptor uses 16×16 patches and 4×4 array of 8 bin histograms. Thus, each SIFT descriptor is made up of $4 \times 4 \times 8 = 128$ non-negative numbers.



Local features

A feature detection and matching application is composed of the following stages:

1. Feature detection (extraction)
   - Search for interest points in the image: done!
2. Feature description
   - Describe the interest point in a compact and stable way: done!
3. Feature matching
   - Search for matching candidates in other images (next slides)

Bibliography

Bibliography

- Szelisky, "Computer Vision: algorithms and applications"
- Lowe, "Distinctive image features from Scale-Invariant Keypoints"

Most of the images shown are have been taken from these two sources.