Automated Tie Point Generation

Motivation

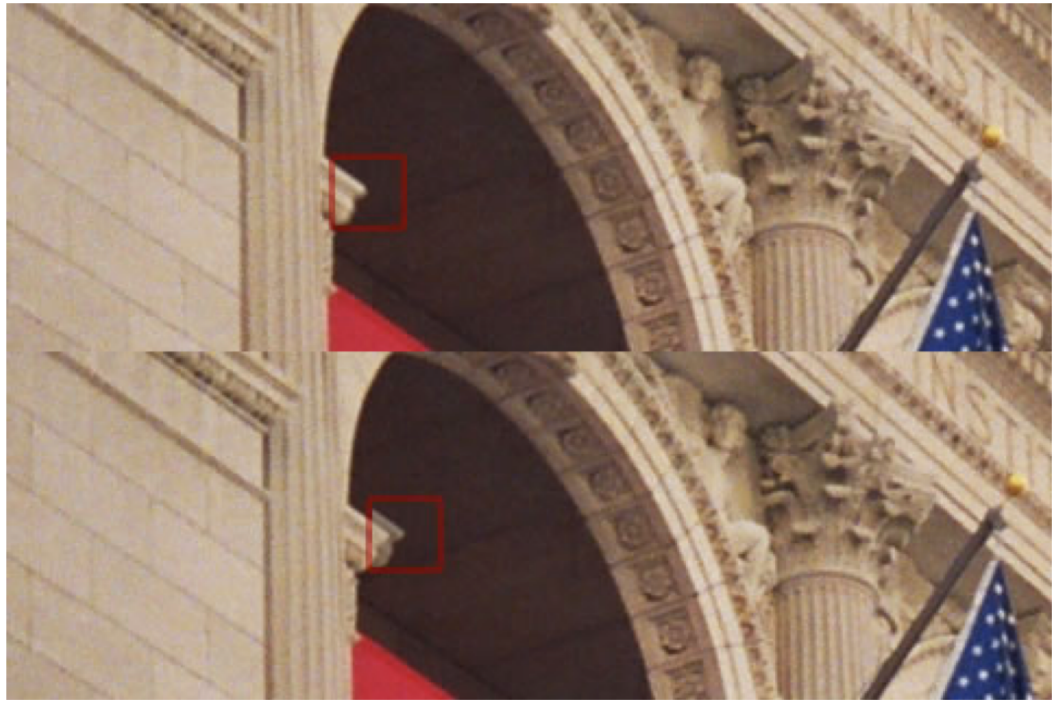
- Automatic generation of tie points for a bundle adjustment or relative orientation is a key requirement for large scale photogrammetry projects

- Matching of features across different images is also a common problem in computer vision (e.g. automatic target detection (ATR) or face matching.

- Efficient and automated methods of matching provide the basis for structure from motion applications

Corner Detectors

- Corners are a stable feature that is relatively simple to match between images.
- Harris Corner Detector is a mathematical operator used to find unique features in an image.
- It is relatively rotation, scale and illumination invariant



Steps for Sift Algorithm

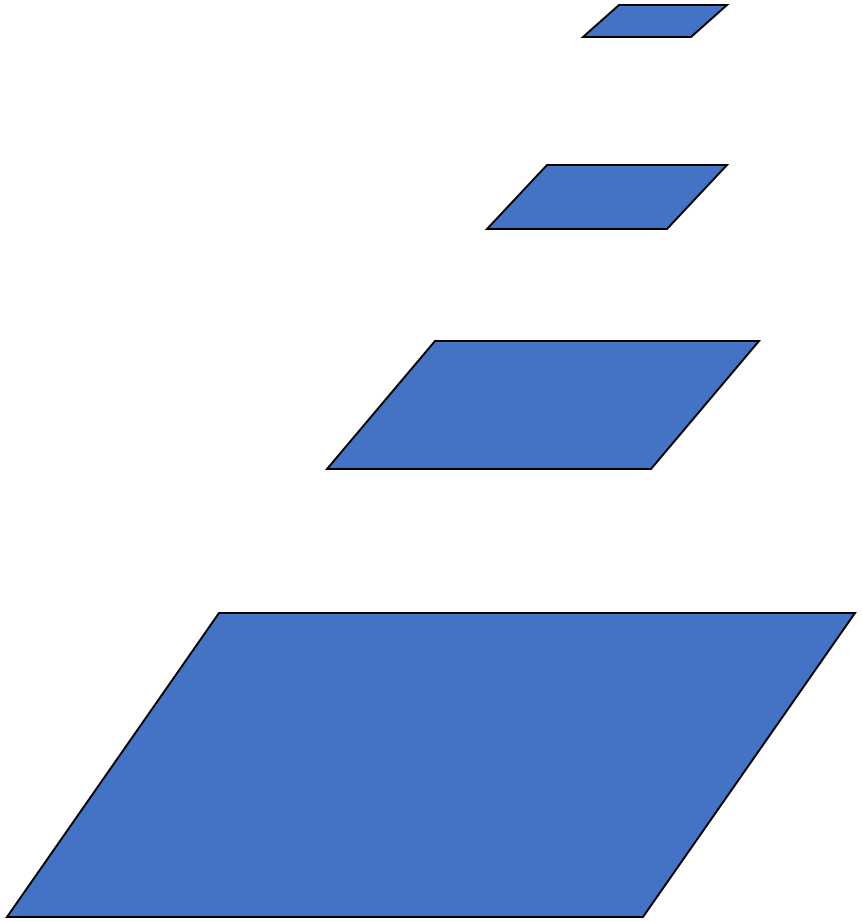
- OUTLINE OF STEPS IN SIFT:

- **Constructing a scale space** This is the initial preparation. You create internal representations of the original image to ensure scale invariance. This is done by generating a "scale space".
- **LoG Approximation** The Laplacian of Gaussian is great for finding interesting points (or key points) in an image. But it's computationally expensive. So we cheat and approximate it using the scale space.
- **Finding keypoints** These are maxima and minima in the Difference of Gaussian image we calculate in the second step
- **Get rid of bad key points** Edges and low contrast regions are bad keypoints. Eliminating these makes the algorithm efficient and robust. A technique similar to the Harris Corner Detector is used here.
- **Assigning an orientation to the keypoints** An orientation is calculated for each key point. Any further calculations are done relative to this orientation. This effectively cancels out the effect of orientation, making it rotation invariant.
- **Generate SIFT features** Finally, with scale and rotation invariance in place, one more representation is generated. This helps uniquely identify features. Lets say you have 50,000 features. With this representation, you can easily identify the feature you're looking for (say, a particular eye, or a sign board).

Scale Space

- Goal: Identify locations and scales that can be repeatably assigned under different views of the same scene or object.
- Method: search for stable features across multiple scales using a continuous function of scale.
- Prior work has shown that under a variety of assumptions, the best function is a Gaussian function.
- The scale space of an image is a function $L(x,y,\sigma)$ that is produced from the convolution of a Gaussian kernel (at different scales) with the input image.

Aside: Image Pyramids



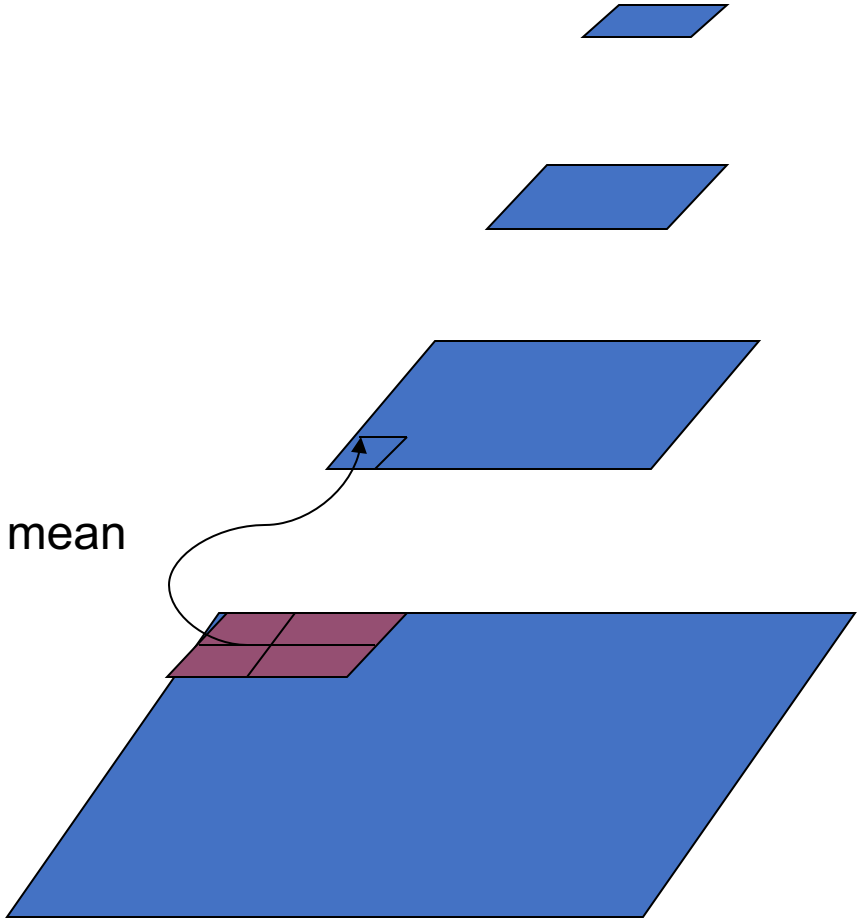
And so on.

3rd level is derived from the 2nd level according to the same funtion

2nd level is derived from the original image according to some function

Bottom level is the original image.

Aside: Mean Pyramid



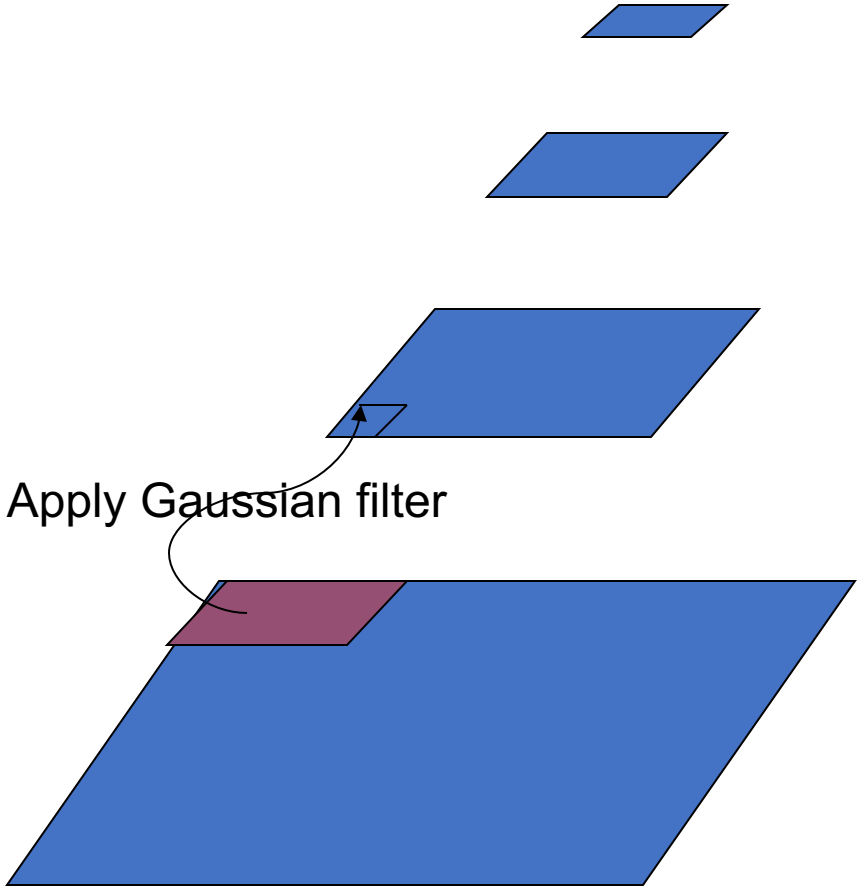
And so on.

At 3rd level, each pixel is the mean of 4 pixels in the 2nd level.

At 2nd level, each pixel is the mean of 4 pixels in the original image.

Bottom level is the original image.

Aside: Gaussian Pyramid
At each level, image is smoothed and reduced in size.



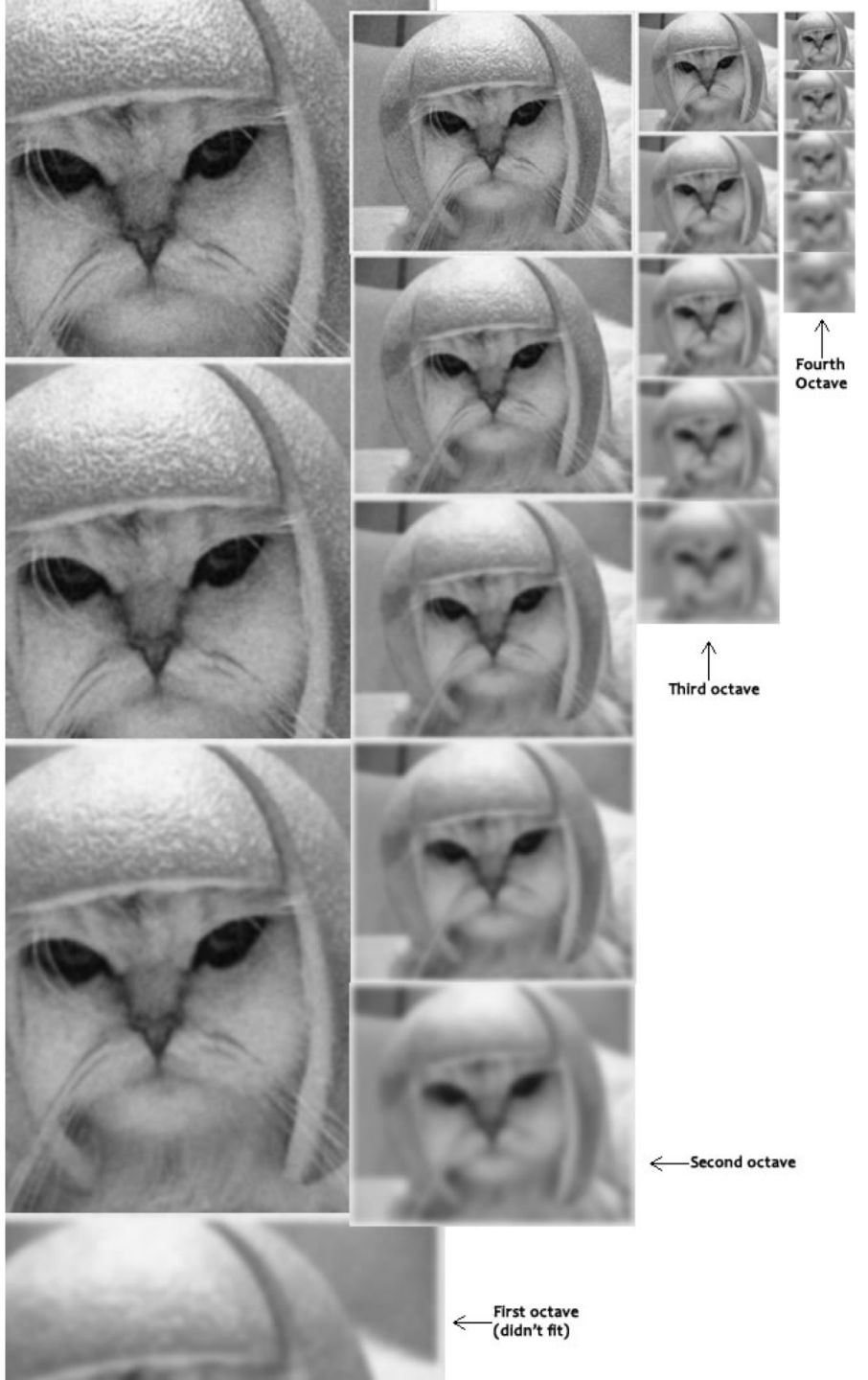
And so on.

At 2nd level, each pixel is the result of applying a Gaussian mask to the first level and then subsampling to reduce the size.

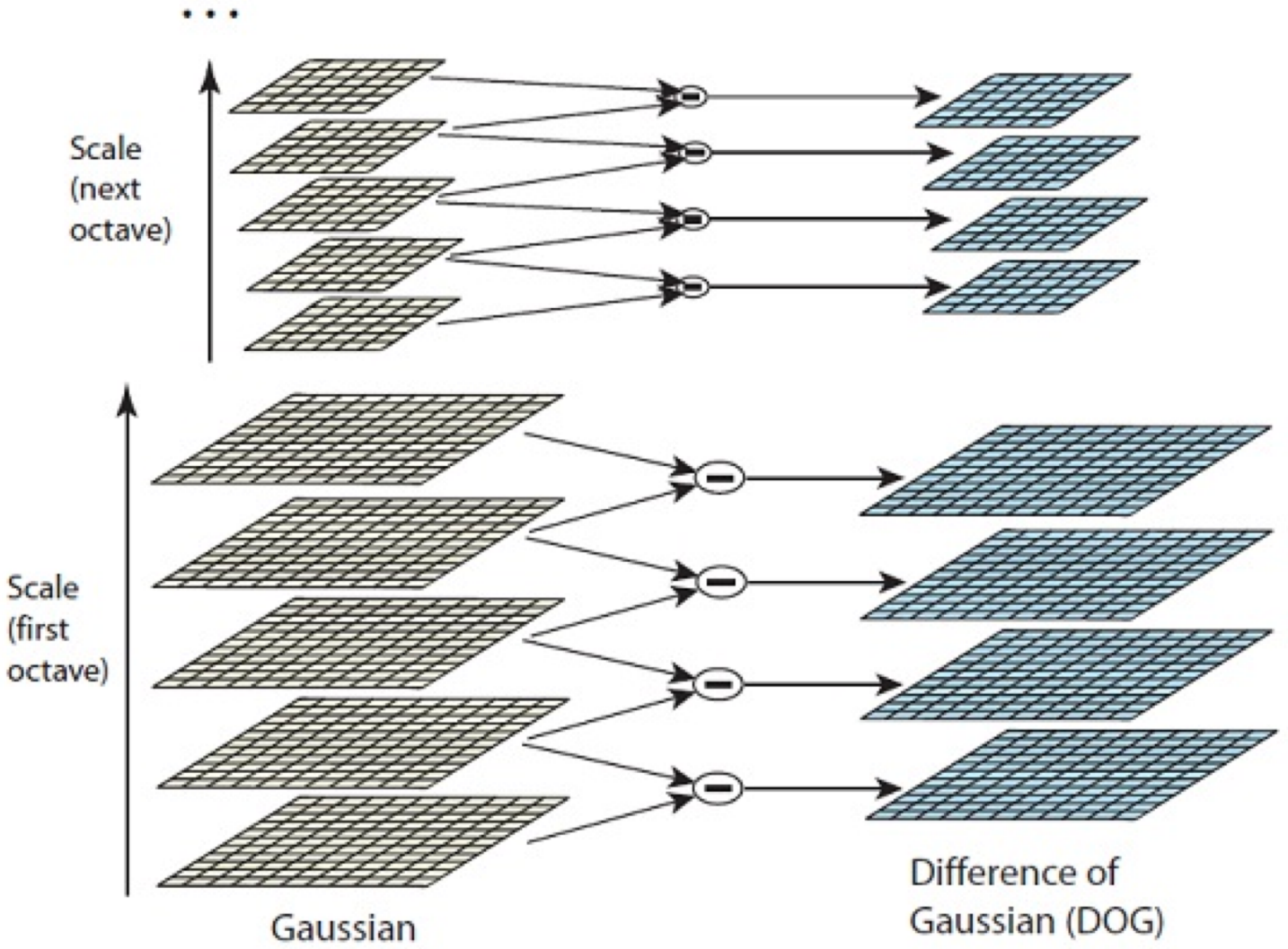
Bottom level is the original image.

SIFT Scale Spaces

- In each octave, original image is progressively blurred
- Then take original and resize to half of original, and blur again. Repeat.
- Images of the same size (vertical) form an octave. Above are four octaves. Each octave has 5 images. The individual images are formed because of the increasing "scale" (the amount of blur).
- 4 octaves and 5 blur levels are suggested by algorithm developer (Lowe)

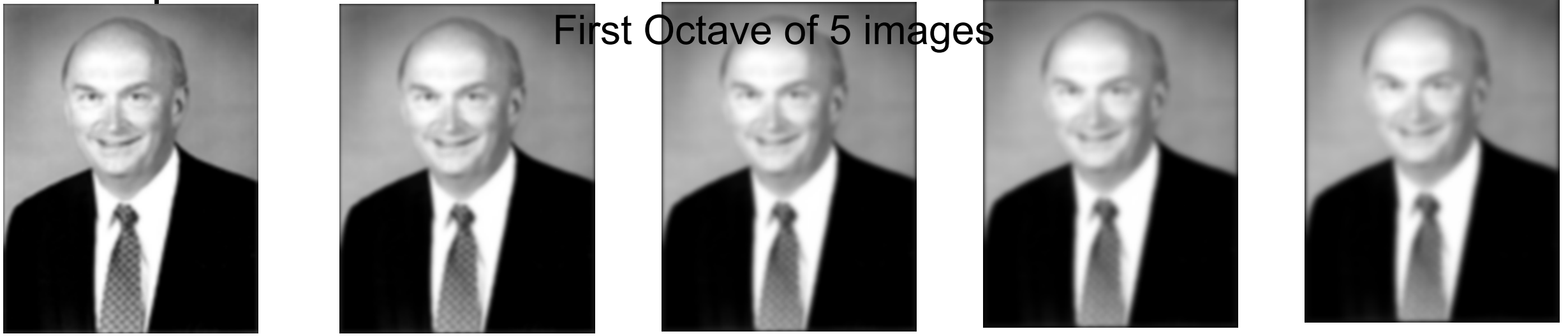


Laplacian of Gaussian



Laplacian of Gaussian

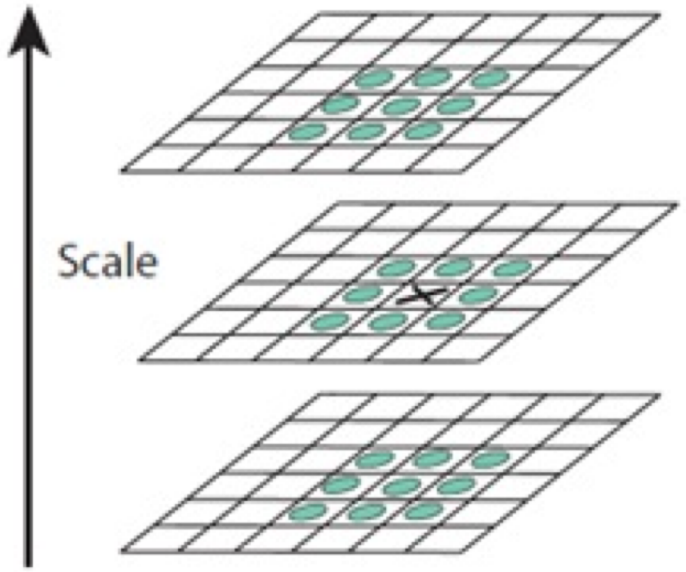
First Octave of 5 images

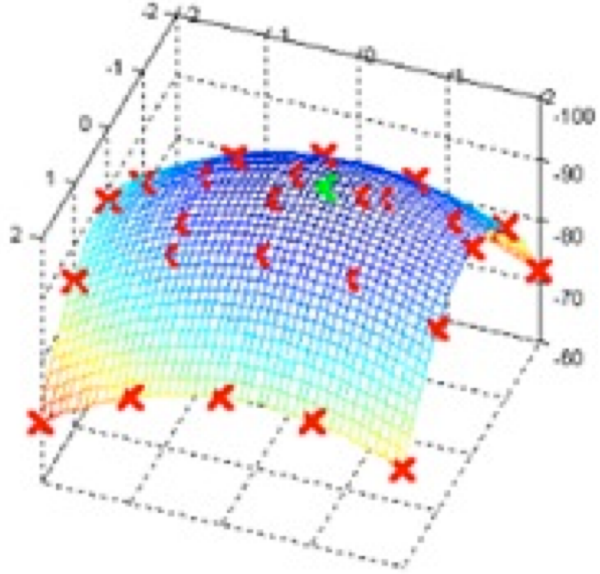


First Octave difference-of-Gaussians

Finding Keypoints

- Two step process:
  1. Locate Max/Min in DoG images
  2. Find subpixel Max/Min





Removing Bad Keypoints

- Reject flats:

$|D(\hat{\mathbf{x}})|$ < 0.03

- Reject edges:

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

Let α be the eigenvalue with larger magnitude and β the smaller.

$$\mathrm{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$

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

Let $r = \alpha/\beta$. So $\alpha = r\beta$

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

$(r+1)^2/r$ is at a min when the 2 eigenvalues are equal.

- $r < 10$

Stages of Keypoint Refinement

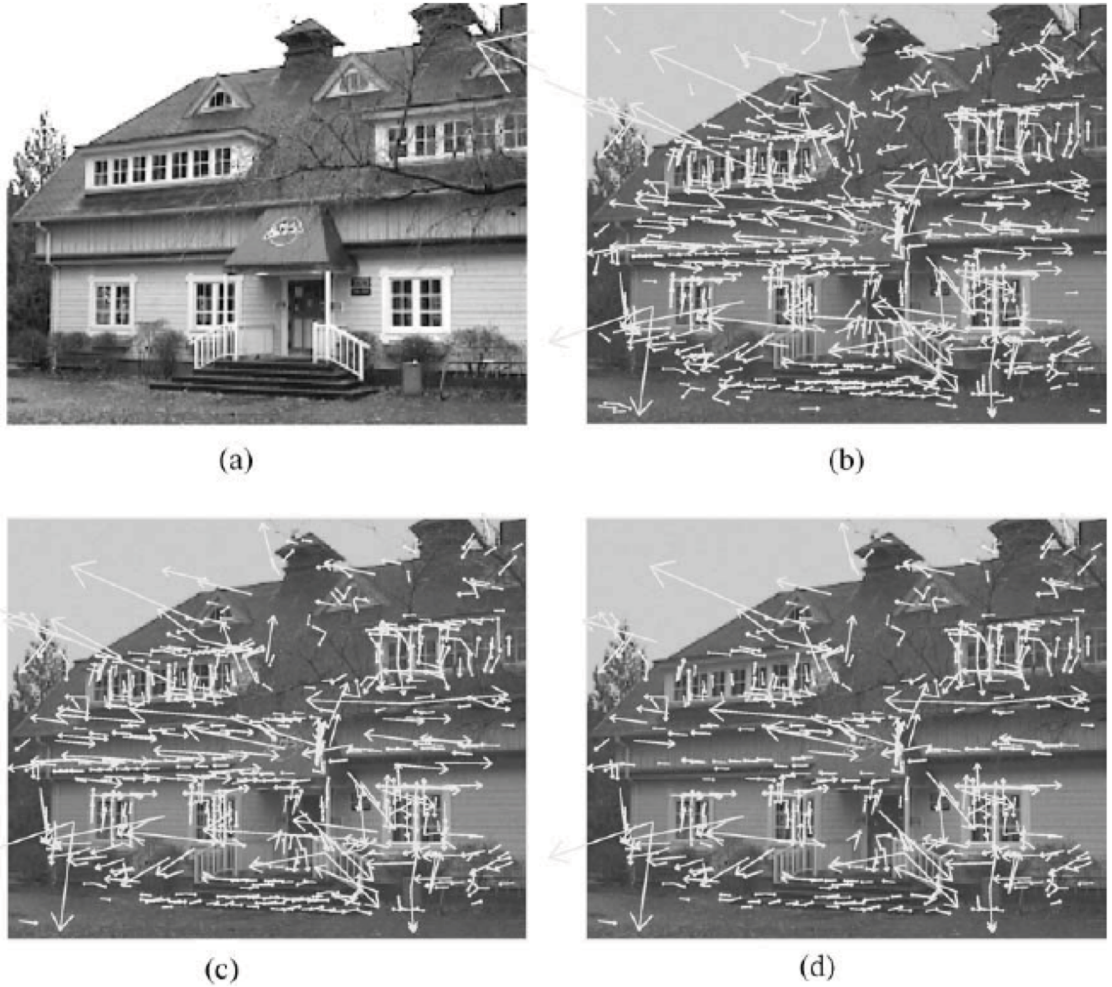
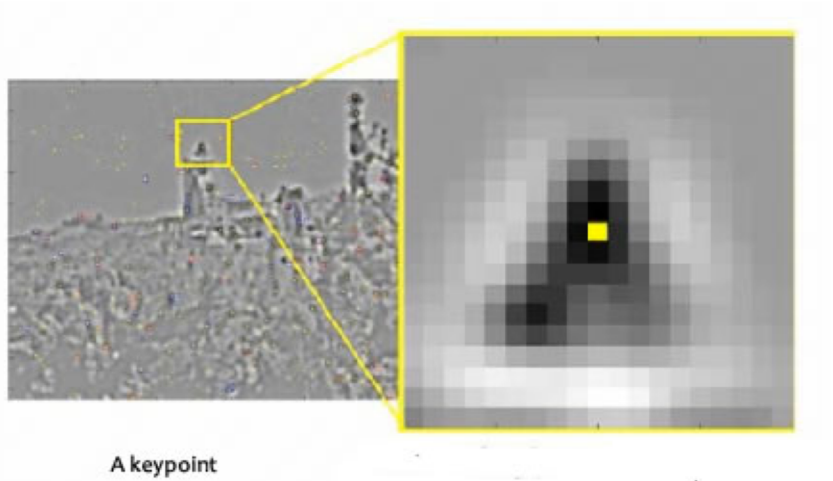
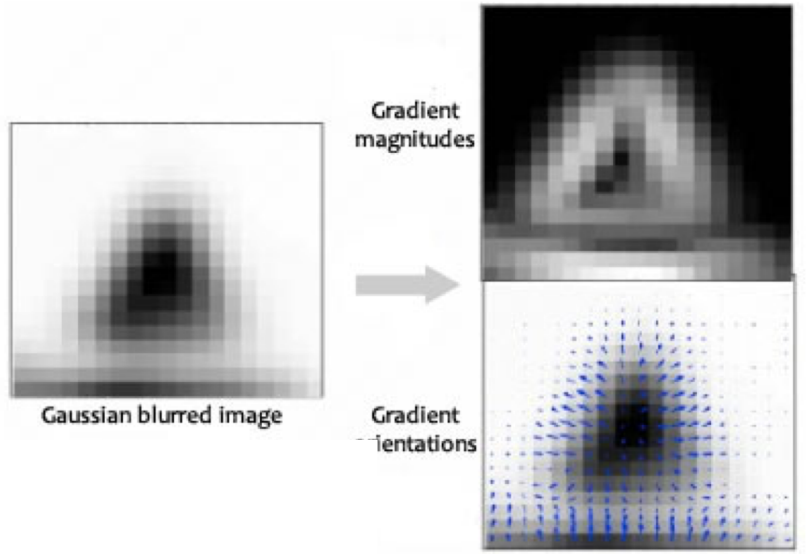


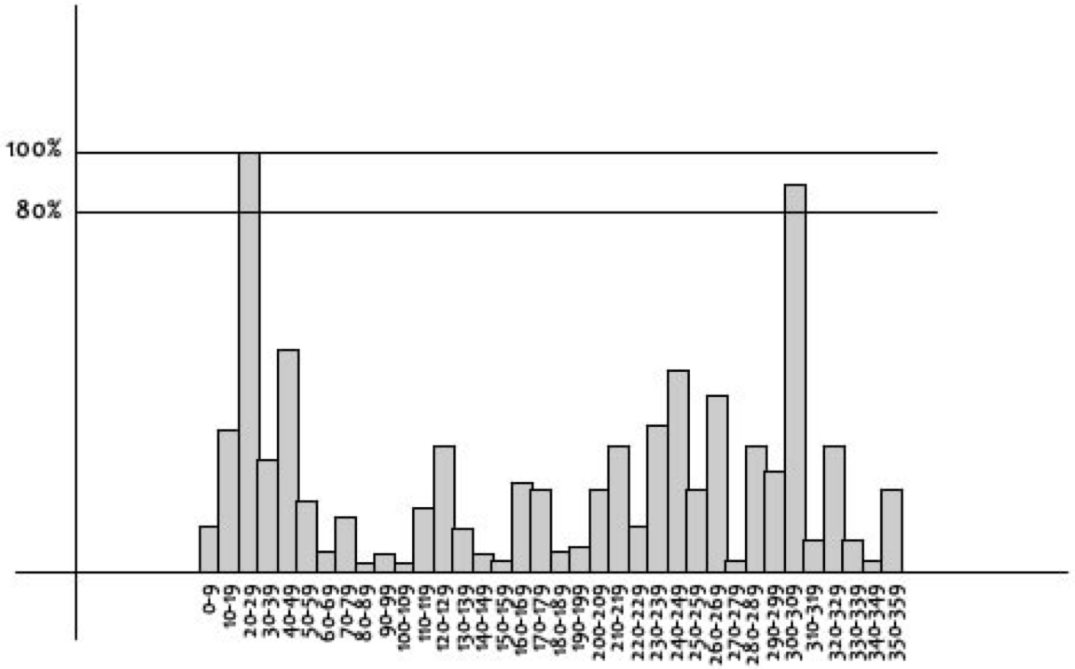
Figure 5. This figure shows the stages of keypoint selection. (a) The 233 × 189 pixel original image. (b) The initial 832 keypoints locations at maxima and minima of the difference-of-Gaussian function. Keypoints are displayed as vectors indicating scale, orientation, and location. (c) After applying a threshold on minimum contrast, 729 keypoints remain. (d) The final 536 keypoints that remain following an additional threshold on ratio of principal curvatures.

Keypoint Orientation

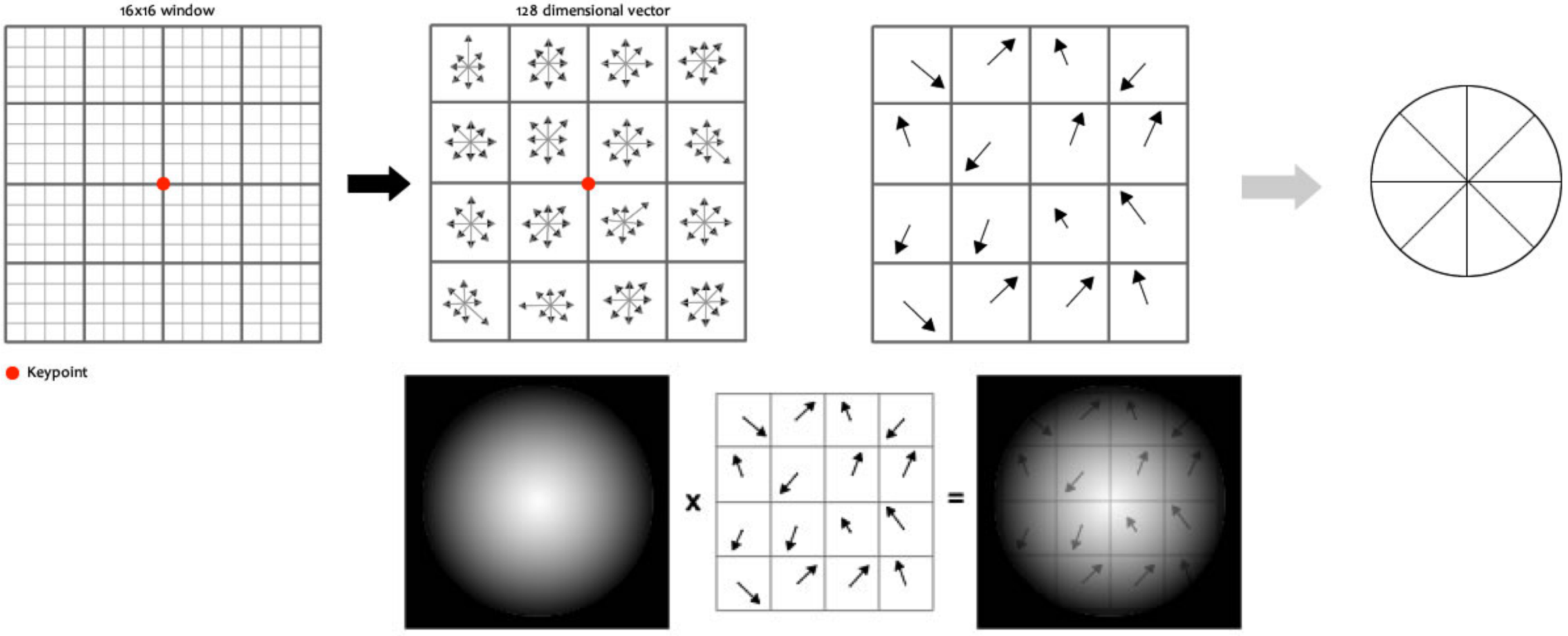


A keypoint





Feature Generation



We now have features

- Up to this point we have:
  - Found rough approximations for features by looking at the difference-of-Gaussians
  - Localized the keypoint more accurately
  - Removed poor keypoints
  - Determined the orientation of a keypoint
  - Calculated a 128 feature vector for each keypoint
- What do we do now?

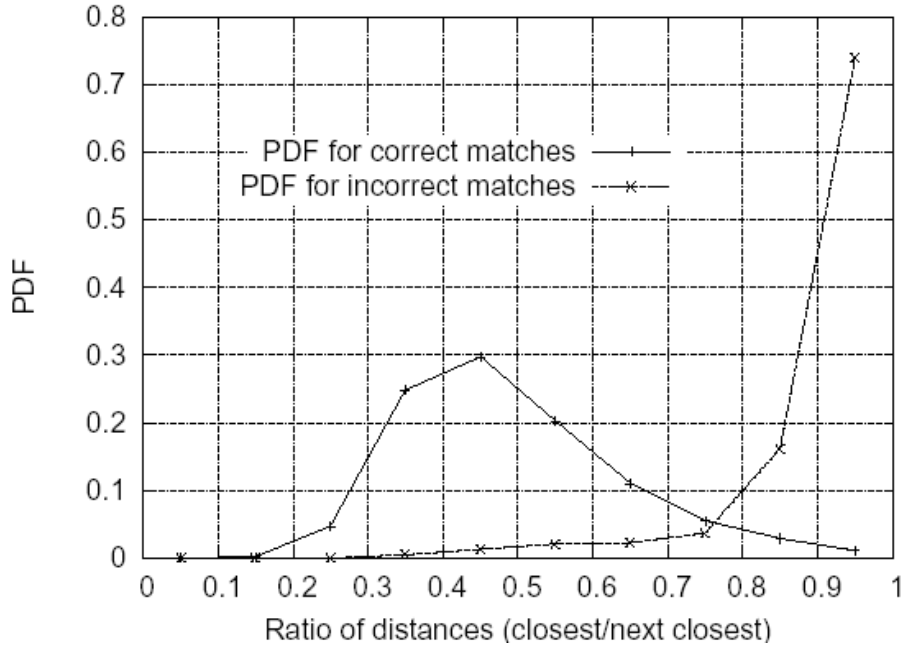
Matching Keypoints between images

- Tale of two images (or more)
  - One image is the training sample of what we are looking for
  - The other image is the world picture that might contain instances of the training sample
- Both images have features associated with them across different octaves
- How do we match features between the two?

Matching Keypoints between images (cont.)

- Nearest Neighbor Algorithm(Euclidean Distance)
  - Independently match all keypoints in all octaves in one image with all keypoints in all octaves in other image
  - How to solve problem of features that have no correct match with opposite image (i.e. background noise, clutter)
    - Global threshold on distance (doesn’t not perform well)
    - Ratio of closest neighbor with second closest neighbor.

Matching Keypoints between images (cont.)



- Threshold at 0.8
  - Eliminates 90% of false matches
  - Eliminates less than 5% of correct matches

Efficient Nearest Neighbor indexing

- Use an approximate algorithm called Best-Bin-First (feature space)
  - Bins are searched in order of their closest distance .
  - Heap-based priority queue.
  - Returns closest neighbor with high probability
  - Search cut off after searching the 200 nearest-neighbor candidates
    - Works well since we only consider matches where nearest neighbor is less than 0.8 times the distance to second nearest neighbor.

- Three good candidates are all that are needed initially to provide approximate transformation

Model Verification-Affine Transformation

- Geometric Verification of clusters

$$\begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix}$$

$$\begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ & & \cdots & \cdots & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u \\ v \\ \vdots \end{bmatrix}$$

$$x = [A^T A]^{-1} A^T b$$