SIFT – Scale Invariant Feature Transform

Bart van Blokland
25.09.2015
What is SIFT?

Distinctive image features from scale-invariant keypoints
DG Lowe - International journal of computer vision, 2004 - Springer
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...
Cited by 31525  Related articles  All 254 versions  Web of Science: 11174  Cite  Save  More
What is SIFT?

• Algorithm for automatic image feature extraction

• What features?
  – Depends on their definition
  – As long as they match properly
What is SIFT?

- Algorithm for automatic image feature extraction

Image source: http://robwhess.github.io/opensift/
What is SIFT?

- Algorithm for automatic image feature extraction

Image source: http://robwhess.github.io/opensift/
What is SIFT?

• Algorithm for automatic image feature extraction

• Extracted features are:
  – Scale invariant
  – Rotation invariant
The Algorithm

1. Find keypoints of interest
2. Refine keypoint set
3. Orientation assignment
4. Generate keypoint descriptor
5. Compare descriptor with database
The Algorithm

1. Find keypoints of interest
2. Refine keypoint set
3. Orientation assignment
4. Generate keypoint descriptor
5. Compare descriptor with database
1. Find points of interest

0. Handling difference in scale
1. Calculate scale-space images
2. Find minima and maxima across different scales
1. Find points of interest

0. Handling difference in scale
   1. Calculate scale-space images
   2. Find minima and maxima across different scales
The issue of scale

Comparing scale is difficult

Image credit: Andrew P. Witkin, «Scale-space filtering»
The issue of scale

- Comparing scale is difficult
- Use scale-space

Image credit: Andrew P. Witkin, «Scale-space filtering»
The issue of scale

- Comparing scale is difficult
- Use scale-space
- Smoothing allows comparison at a larger scale!

Image credit: Andrew P. Witkin, «Scale-space filtering»
1. Find points of interest

1. Calculate scale-space images
2. Find minima and maxima across different scales
1.1 Calculate scale-space images
1.1 Calculate scale-space images

Original 1 iteration 2 iterations 10 iterations
1.1 Calculate scale-space images

- Acts as a fast derivative filter
- After each octave, image is rescaled and reused

![Diagram of scale-space images](image-url)
1. Find points of interest

1. Calculate scale-space images
2. Find minima and maxima across different scales
1.2 Find minima and maxima

- Keep a pixel if it is the lowest or highest value in its neighbourhood
- Not all pixels need to be sampled
1. Find points of interest

1. Use Difference of Gaussian
2. Find minima and maxima across different scales

→ We now have candidate keypoints
The Algorithm

1. Find keypoints of interest
2. **Refine keypoint set**
3. Orientation assignment
4. Generate keypoint descriptor
5. Compare descriptor with database
2. Refine the keypoints set

- The algorithm removes:
  - Points with low contrast
  - Points on edges
2.1 Remove points with low contrast

- Sensitive to noise
- Solution:
  - Calculate actual position of maximum
  - Deduce gradient
  - Gradient threshold
- Improves good keypoints
2.2 Remove points on edges

- Side-effect of Difference of Gaussian
- Edge points are not stable features
- Exploit direction
Recap up to this point

- We now got a nice set of keypoints.
- Each keypoint has:
  - A location
  - A scale
  - An orientation (soon!)

→ Keypoints are scale invariant!
The Algorithm

1. Find keypoints of interest
2. Refine keypoint set
3. **Orientation assignment**
4. Generate keypoint descriptor
5. Compare descriptor with database
3. Orientation Assignment

- Used to find out how the object is oriented
3. Orientation Assignment

- Used to find out how the object is oriented
  - Find gradient of each pixel
3. Orientation Assignment

- Used to find out how the object is oriented
  → Find gradient of each pixel

- How:

  \[
  \text{Magnitude} = \sqrt{((a-b)^2 + (c-d)^2)}
  \]

  \[
  \text{Direction} = \tan^{-1}\left(\frac{c-d}{a-b}\right)
  \]
3. Orientation Assignment

- Used to find out how the object is oriented
  
  → Find gradient of each pixel
3. Orientation Assignment

- We have gradients and magnitudes.
- Make a histogram of them

```python
histogram[angle / 10] += distance * magnitude
```
3. Orientation Assignment

- Create keypoints for every other maximum whose magnitude is 80% of the highest one.
We now know:

- Scale
- Location
- Rotation

Let's turn it into a descriptor!
The Algorithm

1. Find keypoints of interest
2. Refine keypoint set
3. Orientation assignment
4. **Generate keypoint descriptor**
5. Compare descriptor with database
4. Generate Keypoint Descriptor

- Make histograms from different pixel areas
- Rotate coordinates based on keypoint orientation
4. Generate Keypoint Descriptor

- Make histograms from different pixel areas
  
  Histogram count: \((4 \times 4\) histograms) \(*\) 8 directions  
  = feature vector with 128 elements
The Algorithm

1. Find keypoints of interest
2. Refine keypoint set
3. Orientation assignment
4. Generate keypoint descriptor
5. Compare descriptor with database
5. Compare descriptor with database

- Run SIFT on training images
- Compare new images against database
  Match with nearest-neighbour
  Linear search per keypoint
5. Compare descriptor with database

- Run SIFT on training images
- Compare new images against database
  Match with nearest-neighbour
  Linear search per keypoint
- Throw away matches with > 80% correlation
5. Compare descriptor with database

- Finding one image in another:
  - Use RANSAC
5. Compare descriptor with database
To conclude:

- Extracts useful features:
  - Scale invariance
  - Rotation invariance
- Efficient: realtime results
- Can solve relatively difficult recognition tasks
Questions?

1. Find keypoints of interest
2. Refine keypoint set
3. Orientation assignment
4. Generate keypoint descriptor
5. Compare descriptor with database