CS 556: Computer Vision

Lecture 5

Prof. Sinisa Todorovic

sinisa@eecs.oregonstate.edu
MATALB Practice Session 1
Computer Vision System Toolbox

Design and simulate computer vision and video processing systems

Computer Vision System Toolbox™ provides algorithms, functions, and apps for designing and simulating computer vision and video processing systems. You can perform feature detection, extraction, and matching; object detection and tracking; motion estimation; and video processing. For 3-D computer vision, the system toolbox supports camera calibration, stereo vision, 3-D reconstruction, and 3-D point cloud processing. With machine learning based frameworks, you can train object detection, object recognition, and image retrieval systems. Algorithms are available as MATLAB® functions, System objects, and Simulink® blocks.

For rapid prototyping and embedded system design, the system toolbox supports fixed-point arithmetic and C-code generation.

Feature Detection and Extraction
Image registration, interest point detection, extracting feature descriptors, and point feature matching

Object Detection and Recognition
Object detection, recognition, block matching, background estimation, bag of features

Object Tracking and Motion Estimation
Optical flow, activity recognition, motion estimation, and tracking

Camera Calibration
Camera calibration and stereo vision

Multiple View Geometry
3-D information extraction from 2-D images, stereo rectification, depth estimation, 3-D reconstruction, triangulation, and structure from motion

3-D Point Cloud Processing
Downsampling, decimation, transform, visualize, register, and fit geometrical shapes of 3-D point clouds

Analysis and Enhancements
Statistics, FIR filtering, frequency and Hough transforms, morphology, contrast enhancement, and noise removal

Input, Output, and Graphics
Importing, exporting, color space formatting, conversions, display, annotation
## Detecting Interest Points

Interest point detection and extracting feature descriptors

### Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>detectBRISKFeatures</td>
<td>Detect BRISK features and return BRISKPoints object</td>
</tr>
<tr>
<td>detectFASTFeatures</td>
<td>Detect corners using FAST algorithm and return cornerPoints object</td>
</tr>
<tr>
<td>detectHarrisFeatures</td>
<td>Detect corners using Harris–Stephens algorithm and return cornerPoints object</td>
</tr>
<tr>
<td>detectMinEigenFeatures</td>
<td>Detect corners using minimum eigenvalue algorithm and return cornerPoints object</td>
</tr>
<tr>
<td>detectMSERFeatures</td>
<td>Detect MSER features and return MSERRegions object</td>
</tr>
<tr>
<td>detectSURFFeatures</td>
<td>Detect SURF features and return SURFPoints object</td>
</tr>
<tr>
<td>extractFeatures</td>
<td>Extract interest point descriptors</td>
</tr>
<tr>
<td>extractLBPFeatures</td>
<td>Extract local binary pattern (LBP) features</td>
</tr>
<tr>
<td>extractHOGFeatures</td>
<td>Extract histogram of oriented gradients (HOG) features</td>
</tr>
<tr>
<td>matchFeatures</td>
<td>Find matching features</td>
</tr>
<tr>
<td>showMatchedFeatures</td>
<td>Display corresponding feature points</td>
</tr>
</tbody>
</table>

### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>binaryFeatures</td>
<td>Object for storing binary feature vectors</td>
</tr>
<tr>
<td>BRISKPoints</td>
<td>Object for storing BRISK interest points</td>
</tr>
<tr>
<td>cornerPoints</td>
<td>Object for storing corner points</td>
</tr>
<tr>
<td>SURFPoints</td>
<td>Object for storing SURF interest points</td>
</tr>
<tr>
<td>MSERRegions</td>
<td>Object for storing MSER regions</td>
</tr>
</tbody>
</table>

### System Objects

- **blocks**
Detecting Interest Points — Example

```
>> img = imread('filename.jpg');
>> figure;
>> imshow(img);
>> img = rgb2gray(img);
```
Detecting Interest Points — Example

>> SURFPoints = detectSURFFeatures(img);
>> [SURFFeatures, SURFPoints] = extractFeatures(img, SURFPoints);
>> figure;
>> imshow(img);
>> hold on;
>> plot(selectStrongest(SURFPoints, 100));
# Features of Interest Points — Example

## Choose a Detection Function Based on Feature Type

<table>
<thead>
<tr>
<th>Detector</th>
<th>Feature Type</th>
<th>Function</th>
<th>Scale Independent</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAST [1]</td>
<td>Corner</td>
<td>detectFASTFeatures</td>
<td>No</td>
</tr>
<tr>
<td>Minimum eigenvalue algorithm [4]</td>
<td>Corner</td>
<td>detectMinEigenFeatures</td>
<td>No</td>
</tr>
<tr>
<td>Corner detector [3]</td>
<td>Corner</td>
<td>detectHarrisFeatures</td>
<td>No</td>
</tr>
<tr>
<td>BRISK [8]</td>
<td>Corner</td>
<td>detectBRISKFeatures</td>
<td>Yes</td>
</tr>
<tr>
<td>MSER [8]</td>
<td>Region with uniform intensity</td>
<td>detectMSERFeatures</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Note:** Detection functions return objects that contain information about the features. The `extractHOGFeatures` and `extractFeatures` functions use these objects to create descriptors.

## Choose a Descriptor Method

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Binary</th>
<th>Function and Method</th>
<th>Scale</th>
<th>Rotation</th>
<th>Finding Point Correspondences</th>
<th>Typical Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOG</td>
<td>No</td>
<td><code>extractHOGFeatures(I,...)</code></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>LBP</td>
<td>No</td>
<td><code>extractLBPFeatures(I,...)</code></td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>SURF</td>
<td>No</td>
<td><code>extractFeatures(I., Method;SURF)</code></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FREAK</td>
<td>Yes</td>
<td><code>extractFeatures(I., Method;FREAK)</code></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>BRISK</td>
<td>Yes</td>
<td><code>extractFeatures(I., Method;BRISK)</code></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Block</td>
<td>No</td>
<td><code>extractFeatures(I., Method;Block)</code></td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Note:**
- The `extractFeatures` function provides different extraction methods to best match the requirements of your application. When you do not specify the `Method` input for the `extractFeatures` function, the function automatically selects the method based on the type of input point class.
- Binary descriptors are fast but less precise in terms of localization. They are not suitable for classification tasks. The `extractFeatures` function returns a `binaryFeatures` object. This object enables the Hamming-distance-based matching metric used in the `matchFeatures` function.
Matching Features of Interest Points — Example
Matching Features of Interest Points — Example

```matlab
>> boxImage = rgb2gray(imread('stapleRemover.png'));
>> sceneImage = rgb2gray(imread('clutteredDesk.png'));
>> boxPoints = detectSURFFeatures(boxImage);
>> scenePoints = detectSURFFeatures(sceneImage);
>> [boxFeatures, boxPoints] = extractFeatures(boxImage, boxPoints);
>> [sceneFeatures, scenePoints] = extractFeatures(sceneImage, scenePoints);

>> boxPairs = matchFeatures(boxFeatures, sceneFeatures);
>> showMatchedFeatures(boxImage, sceneImage, boxPoints(boxPairs(:, 1), :), ...
scenePoints(boxPairs(:, 2), :), 'montage');
```
Matching Features of Interest Points — Example

\[ d(f_1, f_2) = \|f_1 - f_2\|^2 \]

Euclidean distance

feature vectors
[\text{indexPairs}, \text{matchmetric}] = \text{matchFeatures}(\text{features1}, \text{features2}, \text{Name}, \text{Value})
Matching Features of Interest Points — MATLAB

### Name-Value Pair Arguments

Specify optional comma-separated pairs of Name, Value arguments. Name is the argument name and Value is the corresponding value. Name must appear inside single quotes ("'"). You can specify several name and value pair arguments in any order as Name1,Value1,...,NameN,ValueN.

Example: 'Metric','SSD' specifies the sum of squared differences for the feature matching metric.

#### 'Method' — Matching method

```
'Exhaustive' (default) | 'Approximate'
```

Matching method, specified as the comma-separated pair consisting of 'Method' and the string 'Exhaustive' or 'Approximate'. The method specifies how nearest neighbors between `features1` and `features2` are found. Two feature vectors match when the distance between them is less than the threshold set by the `MatchThreshold` parameter.

- `'Exhaustive'`: Compute the pairwise distance between feature vectors in `features1` and `features2`. Use this method for small feature sets.
- `'Approximate'`: Use an efficient approximate nearest neighbor search. Use this method for large feature sets.

#### 'MatchThreshold' — Threshold

```
percent value in the range (0, 100) | 10.0 (default for binary feature vectors) | 1.0 (default for nonbinary feature vectors)
```

Threshold, specified as the comma-separated pair consisting of 'MatchThreshold' and a scalar percent value in the range (0, 100). You can use the match threshold for selecting the strongest matches. The threshold represents a percent of the distance from a perfect match. Two feature vectors match when the distance between them is less than the threshold set by `MatchThreshold`. The function rejects a match when the distance between the features is greater than the value of `MatchThreshold`. Increase the value to return more matches.

Inputs that are `binaryFeatures` typically require a larger value for the match threshold. The `binaryFeatures` objects are returned by the `extractFeatures` function when extracting FREAK or BRISK descriptors.

#### 'MaxRatio' — Ratio threshold

```
0.6 (default) | ratio in the range [0,1]
```

Ratio threshold, specified as the comma-separated pair consisting of 'MaxRatio' and a scalar value in the range (0,1]. Use the max ratio for rejecting ambiguous matches. Increase this value to return more matches.

#### 'Metric' — Feature matching metric

```
'SSD' (default) | 'SAD'
```

Feature matching metric, specified as the comma-separated pair consisting of 'Metric' and the string 'SAD' or 'SSD'.

- `'SAD'`: Sum of absolute differences
- `'SSD'`: Sum of squared differences
Visualizing the Matches

$\text{showMatchedFeatures}(I_1, I_2, \text{matchedPoints}_1, \text{matchedPoints}_2, \text{method})$

**Match features.**

```matlab
indexPairs = matchFeatures(f1, f2);
machedPoints1 = vpts1(indexPairs(1:20, 1));
machedPoints2 = vpts2(indexPairs(1:20, 2));
```

**Visualize candidate matches.**

```matlab
figure; ax = axes;
showMatchedFeatures(I1,I2,machedPoints1,machedPoints2, 'montage', 'Parent', ax);
title(ax, 'Candidate point matches');
legend(ax, 'Matched points 1', 'Matched points 2');
```

**Candidate point matches**

![Candidate point matches](image)