CS 556: Computer Vision

Lecture 5

Prof. Sinisa Todorovic

sinisa@eecs.oregonstate.edu

OSU Oregon State University

MATALB Practice Session 1

MATLAB Computer Vision Toolbox

mathworks.com

Object Detection in a Cluttered Scene Using Point F... | Computer Vision System Toolbox Documentation | Find matching features - MATLAB matchFeatures | popular point descriptors - Google Search

Contact Us How To Buy Log In

MathWorks® Products Solutions Academia Support Community Events Company

Documentation

Search R2015b Documentation Documentation

CONTENTS Close

Trial Software Product Updates

Computer Vision System Toolbox

R2015b

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.

Getting Started
Examples
Functions and Other Reference
Release Notes
PDF Documentation

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, denoising, 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

mathworks.com

Object Detection in a Cluttered Scene Using Point F... | Local Feature Extraction - MATLAB & Simulink | Find matching features - MATLAB matchFeatures | popular point descriptors - Google Search

Contact Us How To Buy Log In

MathWorks® Products Solutions Academia Support Community Events Company

Documentation

Search R2015b Documentation Documentation

CONTENTS Close

< All Products

< Computer Vision System Toolbox

< Feature Detection and Extraction

Local Feature Extraction

Feature Matching

Image Registration

Geometric Transformations

Trial Software Product Updates

Local Feature Extraction

R2015b

Interest point detection and extracting feature descriptors

Functions

detectBRISKFeatures	Detect BRISK features and return BRISKPoints object
detectFASTFeatures	Detect corners using FAST algorithm and return cornerPoints object
detectHarrisFeatures	Detect corners using Harris–Stephens algorithm and return cornerPoints object
detectMinEigenFeatures	Detect corners using minimum eigenvalue algorithm and return cornerPoints object
detectMSERFeatures	Detect MSER features and return MSERRegions object
detectSURFFeatures	Detect SURF features and return SURFPoints object

extractFeatures	Extract interest point descriptors
extractLBPFeatures	Extract local binary pattern (LBP) features
extractHOGFeatures	Extract histogram of oriented gradients (HOG) features

matchFeatures	Find matching features
showMatchedFeatures	Display corresponding feature points

Classes

binaryFeatures	Object for storing binary feature vectors
BRISKPoints	Object for storing BRISK interest points
cornerPoints	Object for storing corner points
SURFPoints	Object for storing SURF interest points
MSERRegions	Object for storing MSER regions

> 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

mathworks.com

Object Detection in a Cluttered Scene Using Point F... | Local Feature Detection and Extraction - MATLAB &... | Find matching features - MATLAB matchFeatures | popular point descriptors - Google Search

Documentation

Search R2015b Documentation | Documentation

CONTENTS | Trial Software | Product Updates

Choose a Detection Function Based on Feature Type

Detector	Feature Type	Function	Scale Independent
FAST [1]	Corner	`detectFASTFeatures`	No
Minimum eigenvalue algorithm[4]	Corner	`detectMinEigenFeatures`	No
Corner detector [3]	Corner	`detectHarrisFeatures`	No
SURF [11]	Blob	`detectSURFFeatures`	Yes
BRISK [6]	Corner	`detectBRISKFeatures`	Yes
MSER [8]	Region with uniform intensity	`detectMSERFeatures`	Yes

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

Descriptor	Binary	Function and Method	Invariance		Typical Use	
			Scale	Rotation	Finding Point Correspondences	Classification
HOG	No	`extractHOGFeatures(I, ...)`	No	No	No	Yes
LBP	No	`extractLBPFeatures(I, ...)`	No	Yes	No	Yes
SURF	No	`extractFeatures(I, 'Method','SURF')`	Yes	Yes	Yes	Yes
FREAK	Yes	`extractFeatures(I, 'Method','FREAK')`	Yes	Yes	Yes	No
BRISK	Yes	`extractFeatures(I, 'Method','BRISK')`	Yes	Yes	Yes	No
• Block • Simple pixel neighborhood around a keypoint	No	`extractFeatures(I, 'Method','Block')`	No	No	Yes	Yes

Note:

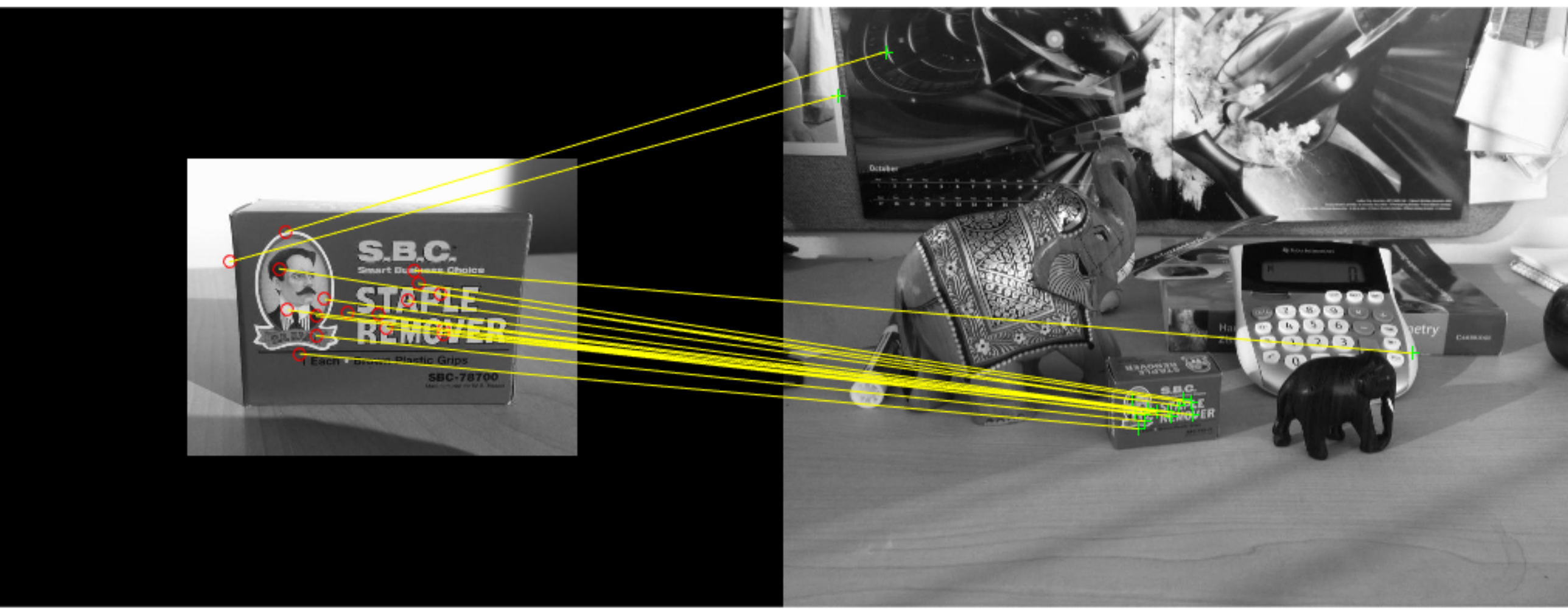
- The `extractFeatures` function provides different extraction methods to best match the requirements of your application. When you do not specify the `'Method'` input for the `extractFeatures` function, the function automatically selects the method based on the type of input point class.
- Binary descriptors 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-distanc based matching metric used in the `matchFeatures` function.

Matching Features of Interest Points — Example

Matching Features of Interest Points — Example

```
>> 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

feature vectors

$$d(\mathbf{f_1}, \mathbf{f_2}) = \|\mathbf{f_1} - \mathbf{f_2}\|^2$$

Euclidean distance

Matching Features of Interest Points — MATLAB

```
[indexPairs,matchmetric] = matchFeatures(features1,features2,Name,Value)
```

mathworks.com

szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf | Find matching features - MATLAB matchFeatures

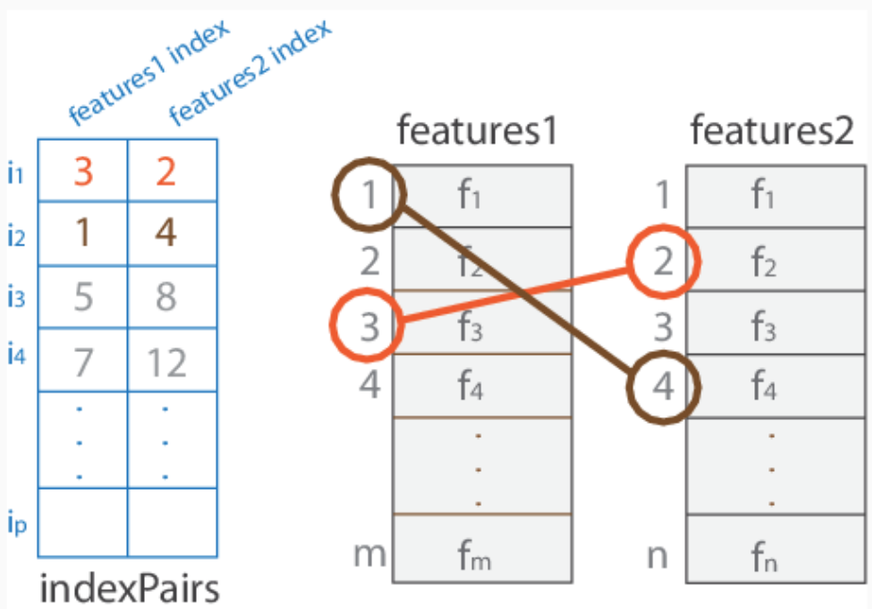
Documentation | Search R2015b Documentation

CONTENTS

`indexPairs` — Indices to corresponding features

P-by-2 matrix

Indices of corresponding features between the two input feature sets, returned as a *P*-by-2 matrix of *P* number of indices. Each index pair corresponds to a matched feature between `features1` and `features2` inputs. The fi
`features1`. The second element indexes the matching feature in `features2`.



`matchmetric` — Distance between matching features

p-by-1 vector

Distance between matching features, returned as a *p*-by-1 vector. The value of the distances are based on the metric selected. Each *i*th element in `matchmetric` corresponds to the *i*th row in the `indexPairs` output matrix. '
metrics, the feature vectors are normalized to unit vectors before computation.

Metric	Range	Perfect Match Value
SAD	[0, 2*sqrt(size(features1, 2))].	0
SSD	[0,4]	0
Hamming	[0, features1.NumBits]	0

Note: You cannot select the `Hamming` metric. It is invoked automatically when `features1` and `features2` inputs are `binaryFeatures`.

Matching Features of Interest Points — MATLAB

mathworks.com

szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf | Find matching features - MATLAB matchFeatures

Documentation | Search R2015b Documentation | Documentation

CONTENTS | Trial Software | Product Updates

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`.
`'Approximate'`	Use an efficient approximate nearest neighbor search. Use this method for large feature sets. [3]

`'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

```
showMatchedFeatures(I1,I2,matchedPoints1,matchedPoints2,method)
```

Match features.

```
indexPairs = matchFeatures(f1, f2) ;
matchedPoints1 = vpts1(indexPairs(1:20, 1));
matchedPoints2 = vpts2(indexPairs(1:20, 2));
```

Visualize candidate matches.

```
figure; ax = axes;
showMatchedFeatures(I1,I2,matchedPoints1,matchedPoints2,'montage','Parent',ax);
title(ax, 'Candidate point matches');
legend(ax, 'Matched points 1','Matched points 2');
```