# HW5: Feature Detection and Matching

Assigned: Tuesday, November 4

Due: Tuesday, November 18

img1 img3 img5

# Your task

- Write Harris detector **ComputeHarris()** in features.cpp
- Write Feature descriptor **ExtractDescriptor()** in features.cpp
- Run your program to get required numerical and image results
- Compare results from your features and SIFT features

# Interest Point Detection

1. Convert the RGB image to a gray-scale image. (Save the RGB image for later.)
2. Apply a 9 x 9 Gaussian filter we give you.
3. Construct the Harris matrix M using a 5 x 5 neighborhood around each pixel.
4. Compute corner response $R = \det(M) - k*Tr(M)^2$ at each pixel.
5. If R is above a threshold and is a local maximum in a 3 x 3 neighborhood, select it as an interest point.
6. 600 to 3000 points are expected to be detected for each image

# Feature Description

1. Go back to RGB space.
2. Take a 45 x 45 window around each feature point.
3. Divide that window into 9 x 9 squares, each of them size 5 x 5.
4. Applying a (given) 5 x 5 mask to each of these squares in each of R, G, and B, gives 3 results per square x 81 squares = a vector V of 243 values.
5. Compute $V' = V / \|V\|_2$ (V divided by its L2-norm).
6. V′ is the descriptor.

# Evaluation

- Working implementation of all the required parts: 11 points
  - Harris operator: 4 points
  - Feature descriptor: 5 points
  - Feature matching: 2 points
- Quality of code including code structure, comments and documentation: 4 points
- Completion and quality of the report: 5 points
- Quality of results: 5 points
  - Harris operator: 2 points
  - Simple descriptor: 3 points

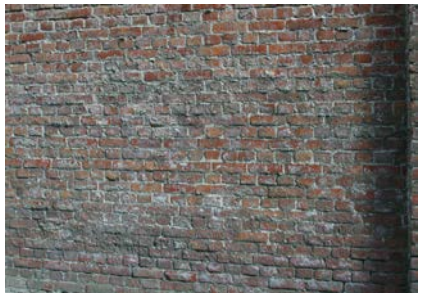
# Evaluation

- Extra credit:
  - Advanced way of match finding: 3 points
  - Advanced descriptor: Rotation invariant descriptor only: 7 points
  - Advanced descriptor: Rotation and scale invariant descriptor: 10 points

# Data sets

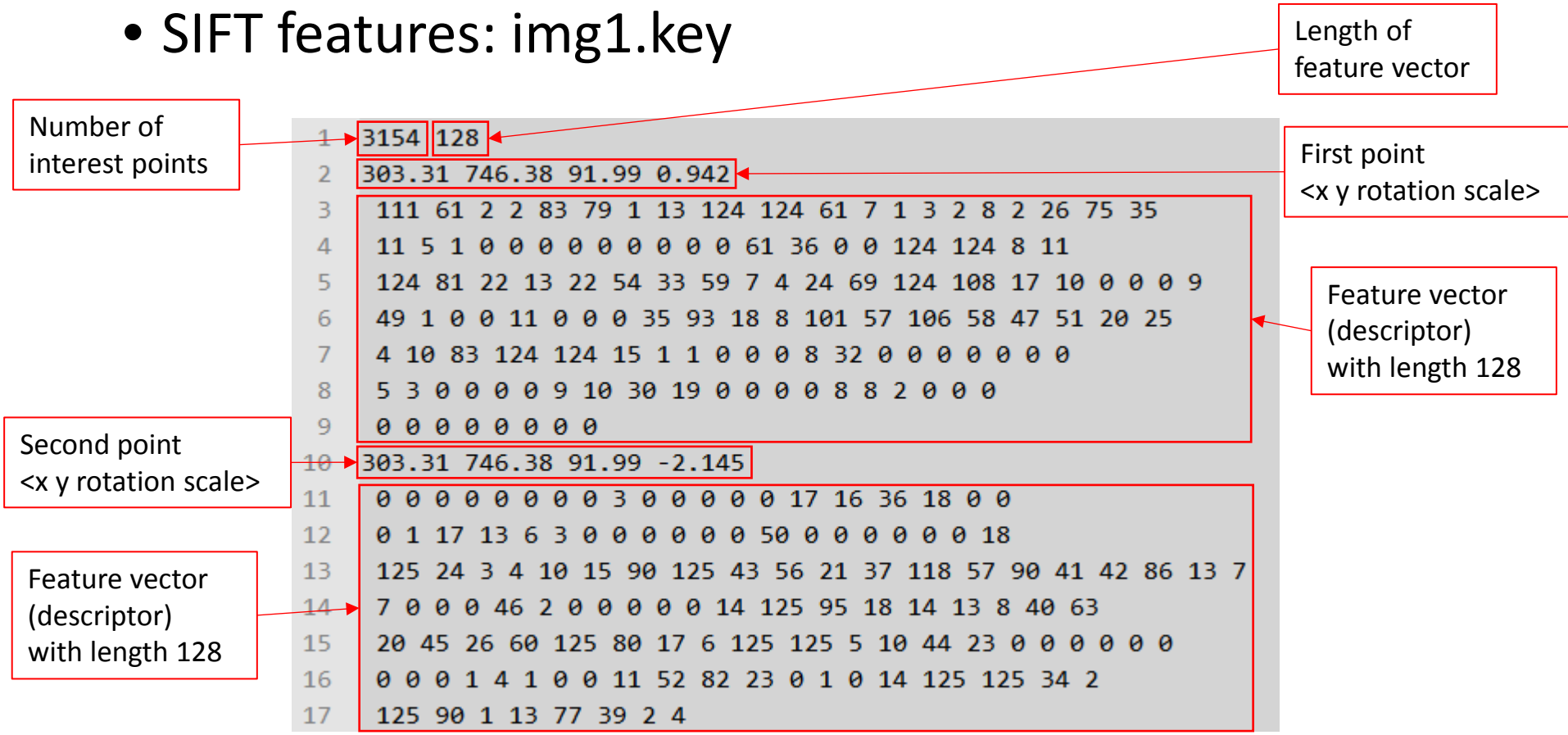
- Required: *bikes* and *leuven*

- Extra credit: *graf* and *wall*



# Data sets

- SIFT features: img1.key



# Data sets

- Image database:
  img.kdb (SIFT)  imgsimple.kdb (Your Harris)

```
1  img1.ppm img1.key
2  img2.ppm img2.key
3  img3.ppm img3.key
4  img4.ppm img4.key
5  img5.ppm img5.key
6  img6.ppm img6.key
```

```
1  img1.ppm feature1.f
2  img2.ppm feature2.f
3  img3.ppm feature3.f
4  img4.ppm feature4.f
5  img5.ppm feature5.f
6  img6.ppm feature6.f
```

- Homography files: H1to2p
  (transformation matrix from image 1 to image 2)

```
1    1.0107879e+00   8.2814684e-03   1.8576800e+01
2   -4.9128885e-03   1.0148779e+00  -2.8851517e+01
3   -1.9166087e-06   8.1537620e-06   1.0000000e+00
```

# Software

- Command line
  - **`./CSE455 computeFeatures ../bikes/img1.ppm ../bikes/feature1.f`**
  - **`./CSE455 testMatch ../bikes/feature1.f ./bikes/feature2.f ../bikes/H1to2p`**
  - **`./CSE455 testSIFTMatch ../bikes/img1.key ../bikes/img2.key ../bikes/H1to2p`**
  - **`./CSE455 benchmark ../bikes`**

useful for debugging

- Graphical User Interface (GUI): visualization

# Computing features

**./CSE455 computeFeatures ../bikes/img1.ppm ../bikes/feature1.f**

features.cpp

(Global Scope)

```
 7  bool computeFeatures(CFloatImage &image, FeatureSet &features, int featureType) {
 8      // Instead of calling dummyComputeFeatures, you can write your own
 9      // feature computation routines and call them here.
10      switch (featureType) {
11      case 1:
12          dummyComputeFeatures(image, features);
13          break;
14      case 2:
15          dummyComputeFeatures(image, features);
16          break;
17      default:
18          return false;
19      }
20
21      // This is just to make sure the IDs are assigned in order, because
22      // the ID gets used to index into the feature array.
23      for (unsigned int i=0; i<features.size(); i++) {
24          features[i].id = i+1;
25      }
26
27      return true;
28  }
```

# Computing features

- It executes your code!

features.cpp*

(Global Scope)

```
111
112 void dummyComputeFeatures(CFloatImage &image, FeatureSet &features)
113 {
114     // Compute the interest function
115
116     CShape shape0 = image.Shape();
117     CShape shape  = shape0;
118     shape.nBands  = 1;
119
120     CFloatImage harris(shape);
121
122     // TODO: write your interest point detector in ComputeHarris()
123     ComputeHarris(image, harris);
```

```
149             if (isMax){
150                 // TODO: Write your feature descriptor in ExtractDescriptor()
151                 ExtractDescriptor(x, y, blurImage, features);
152             } // end if isMax
```

# Matching features

```
./CSE455 testMatch ../bikes/feature1.f ../bikes/feature2.f
        ../bikes/H1to2p
```

features.cpp

(Global Scope)

```
55  // Match one feature set with another.
56  bool matchFeatures(const FeatureSet &f1, const FeatureSet &f2, vector<FeatureMatch> &matches, double
57      // Instead of calling dummyMatchFeatures, you can write your own
58      // feature matching routines and call them here.
59      switch (matchType) {
60      case 1:
61          dummyMatchFeatures(f1, f2, matches, totalScore);
62          return true;
63      case 2:
64          dummyMatchFeatures(f1, f2, matches, totalScore);
65          return true;
66      default:
67          return false;
68      }
69  }
```

# Matching features

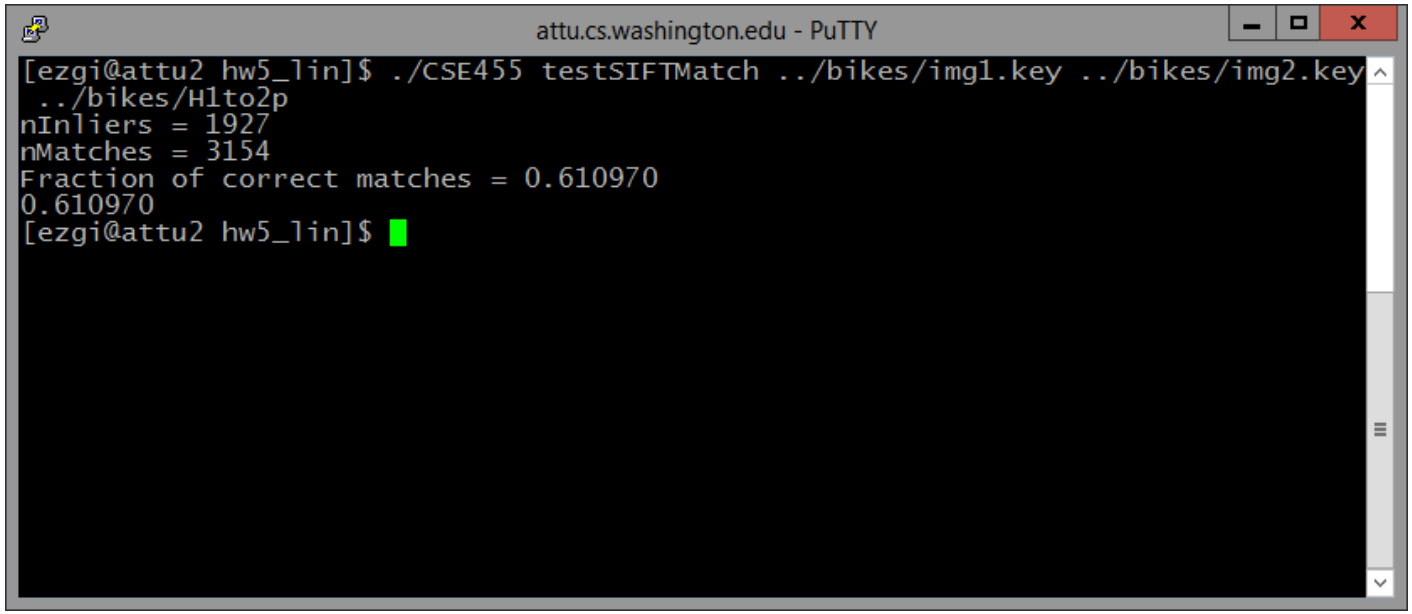
features.cpp

(Global Scope) dummyM

```
163 void dummyMatchFeatures(const FeatureSet &f1, const FeatureSet &f2, vector<FeatureMatch> &matches, double
164     int m = f1.size();
165     int n = f2.size();
166
167     matches.resize(m);
168     totalScore = 0;
169
170     double d;
171     double dBest;
172     int idBest;
173
174     for (int i=0; i<m; i++) {
175         dBest = 1e100;
176         idBest = 0;
177
178         for (int j=0; j<n; j++) {
179             d = distanceEuclidean(f1[i].data, f2[j].data);
180
181             if (d < dBest) {
182                 dBest = d;
183                 idBest = f2[j].id;
184             }
185         }
186
187         matches[i].id = idBest;
188         matches[i].score = exp(-dBest);
189         totalScore += matches[i].score;
```

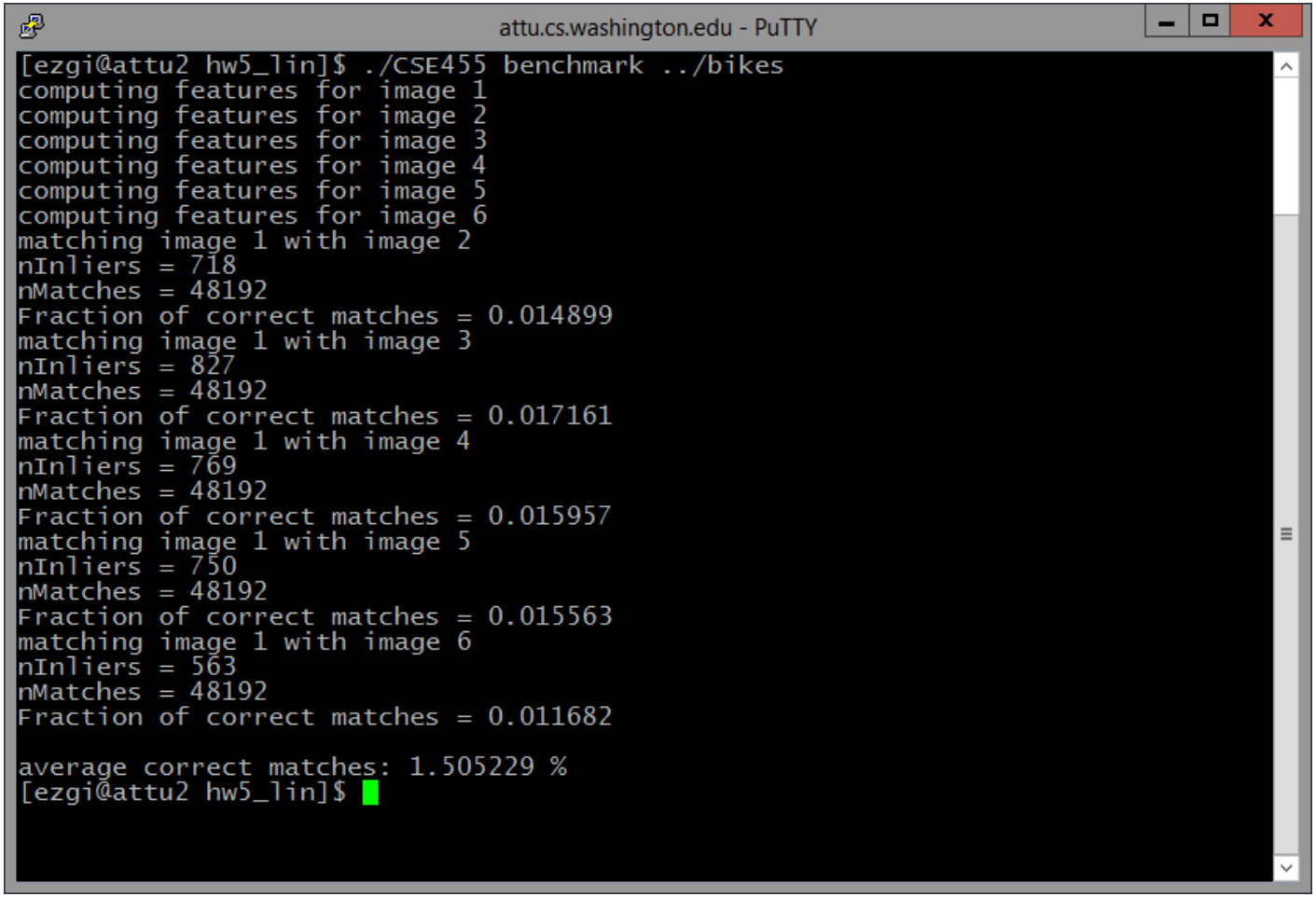
# Matching SIFT features

```
./CSE455 testSIFTMatch ../bikes/img1.key ../bikes/img2.key
        ../bikes/H1to2p
```



# Benchmark

**./CSE455 benchmark ../bikes**



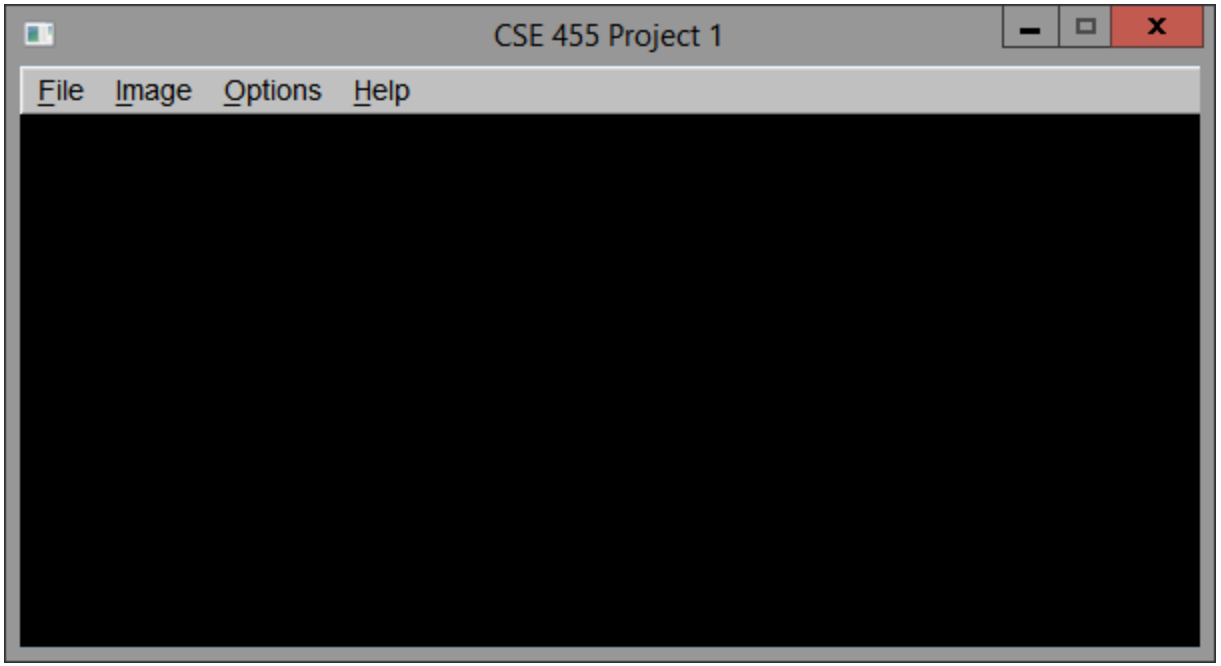
# Benchmark

```
./CSE455 benchmark ../bikes
```

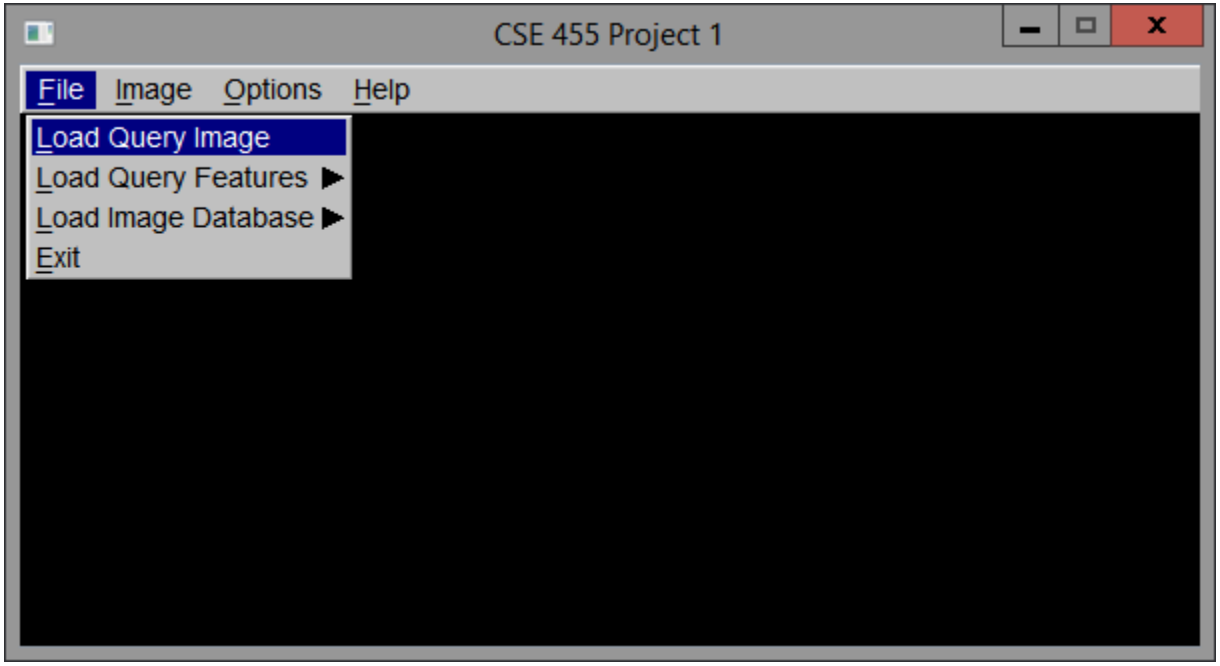
It computes Harris features and match them using SSD for all the images in the given directory.

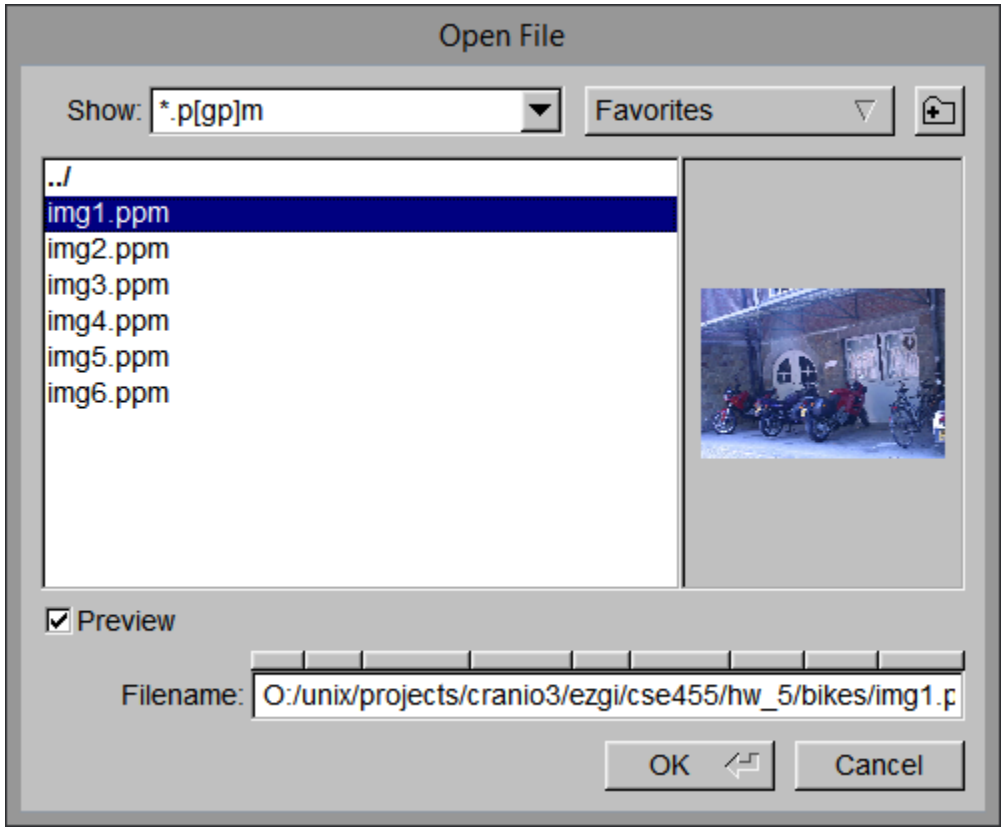
It reports the average match score.

# GUI - DEMO

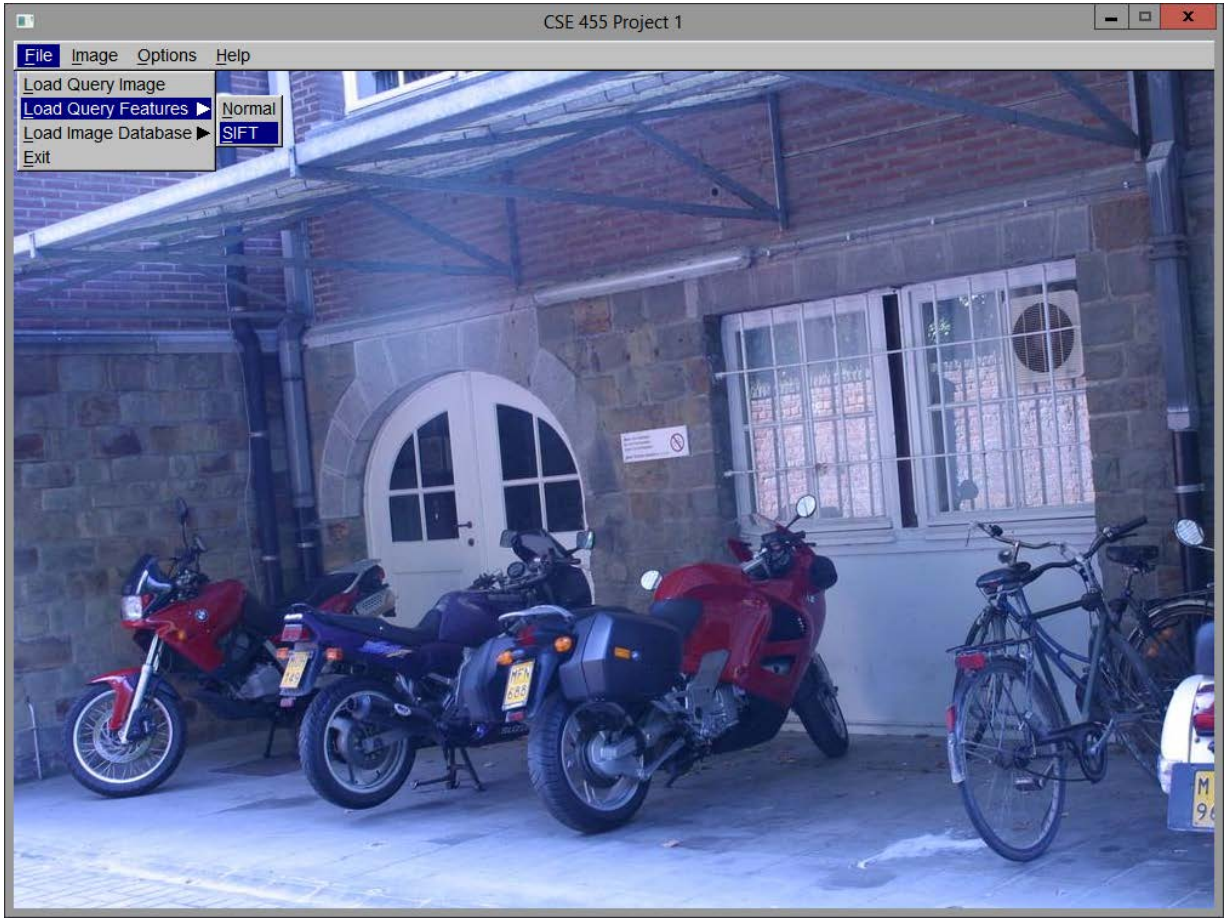


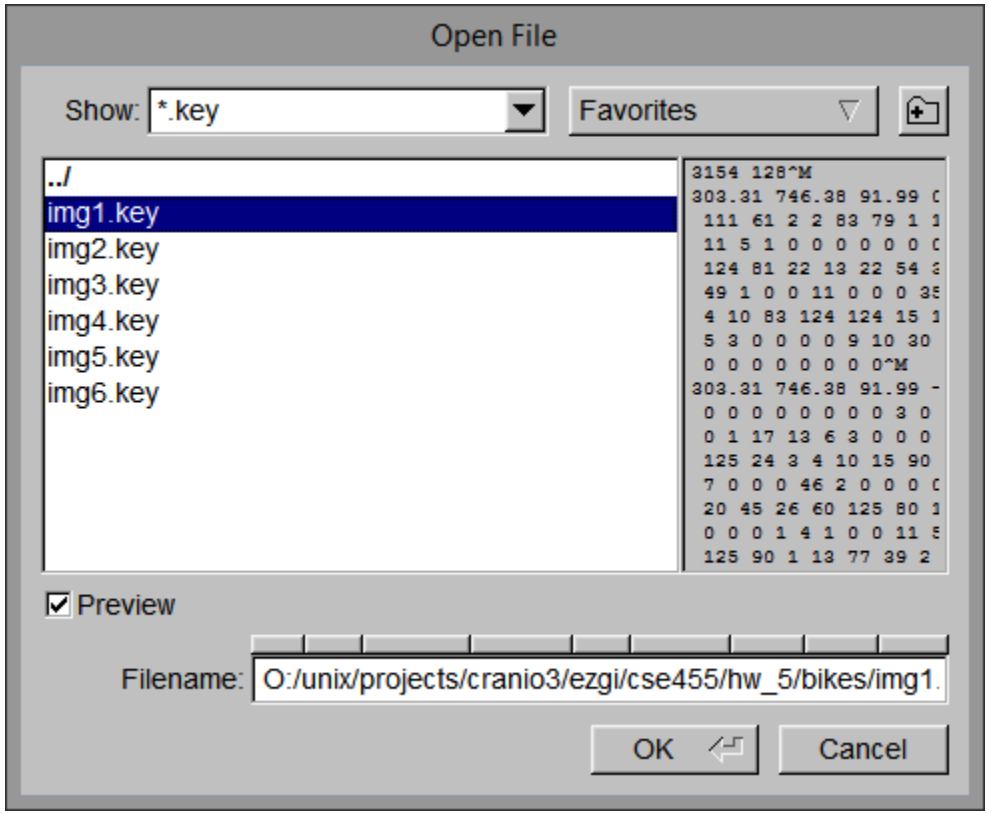
First load the query image
This is the image you will compare with the others.



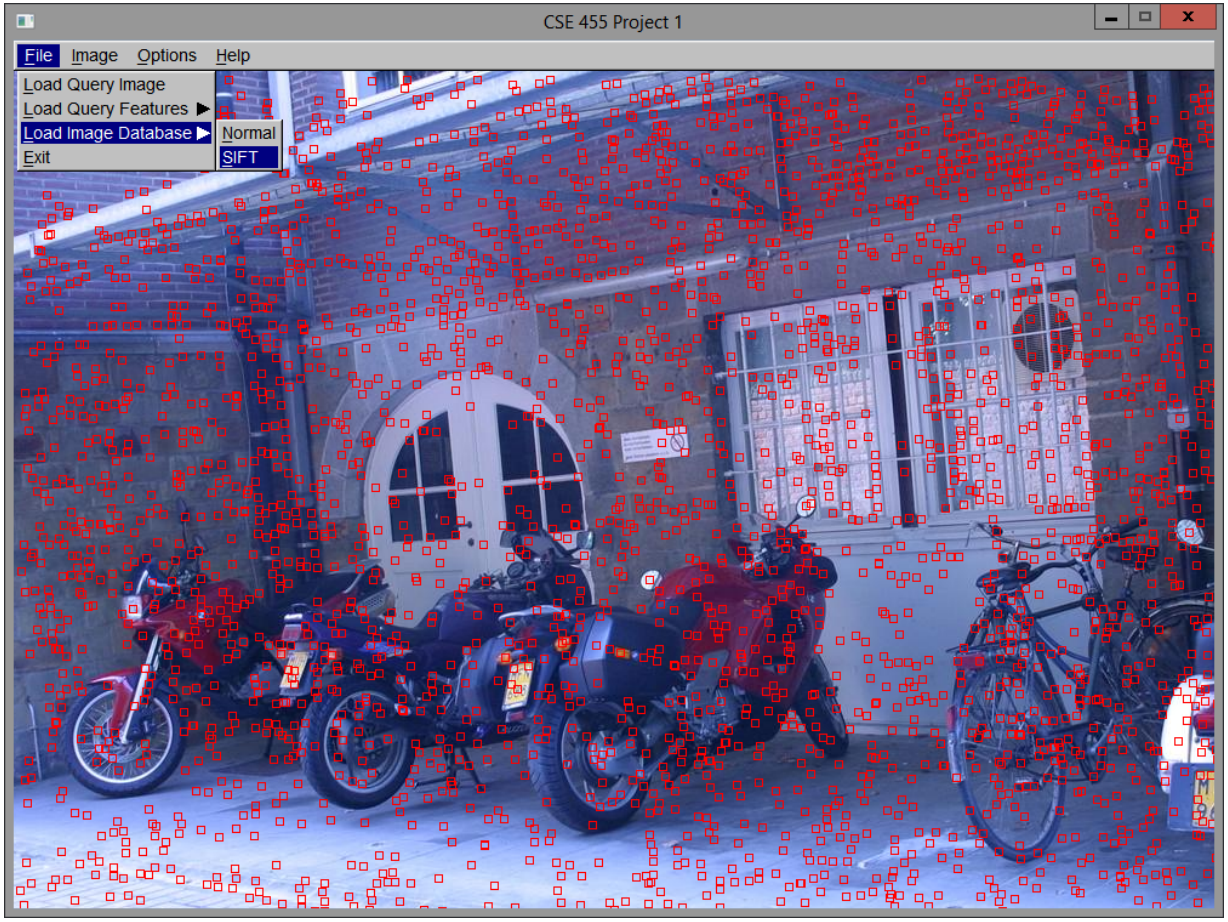

Open File
Show:
*.p[gp]m
Favorites
../
img1.ppm
img2.ppm
img3.ppm
img4.ppm
img5.ppm
img6.ppm
Preview
Filename:
O:/unix/projects/cranio3/ezgi/cse455/hw_5/bikes/img1.p
OK
Cancel

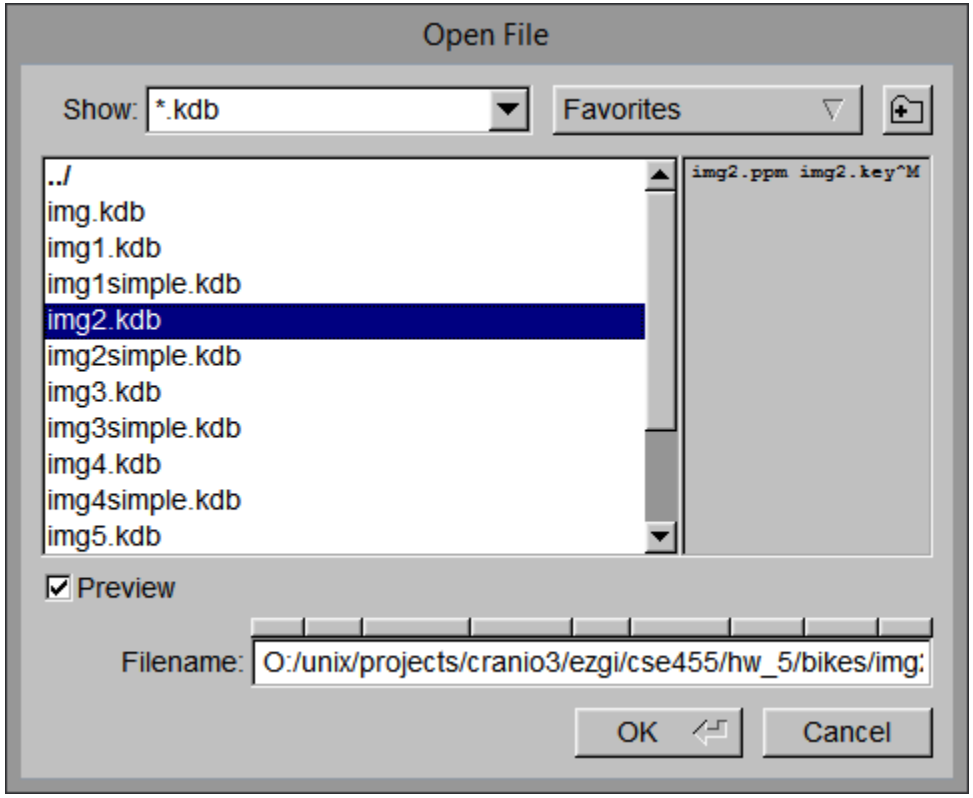
Then load the features for the query image.
Normal = Your implementation, SIFT is provided.



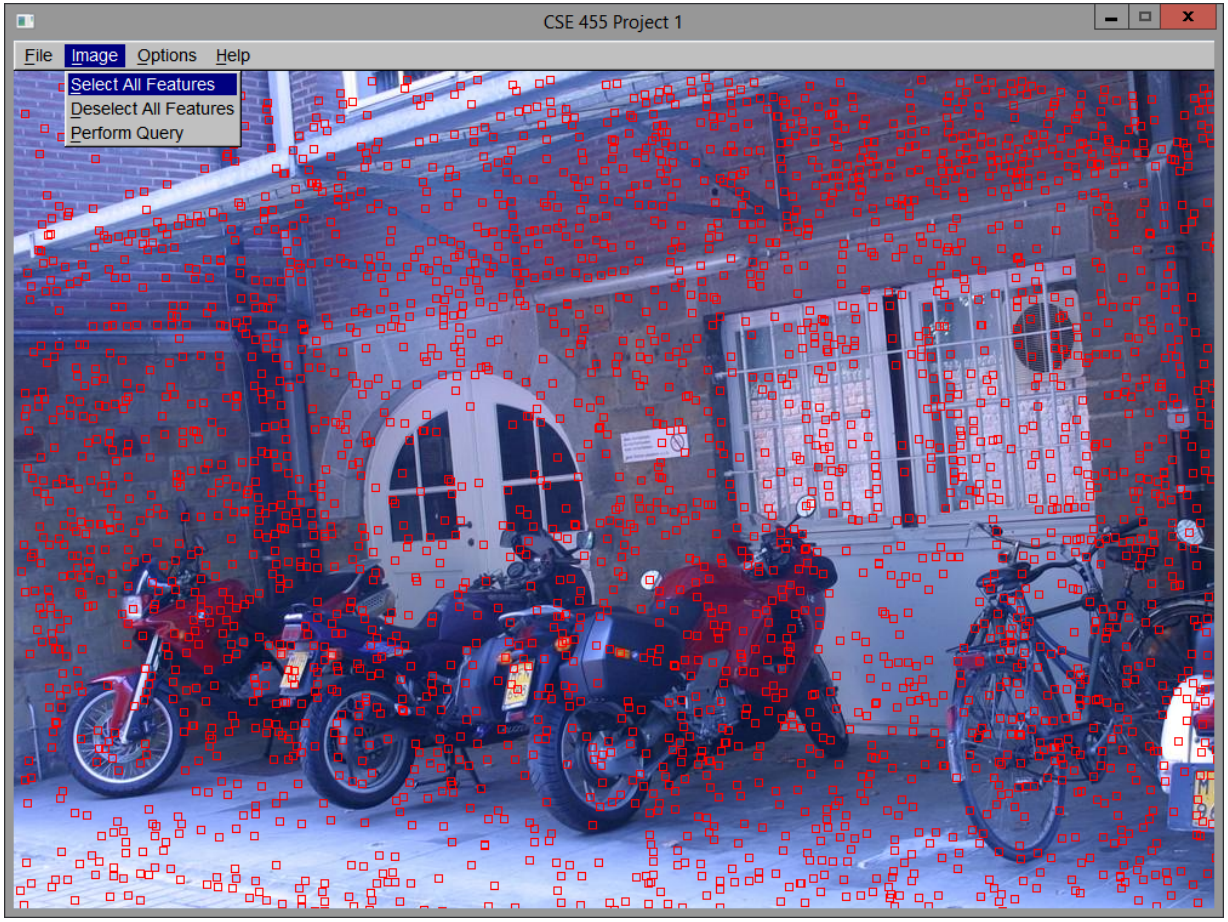

Open File
Show: *.key
Favorites
../
img1.key
img2.key
img3.key
img4.key
img5.key
img6.key
3154 128^M
303.31 746.38 91.99 (
111 61 2 2 83 79 1 1
11 5 1 0 0 0 0 0 0 (
124 81 22 13 22 54 3
49 1 0 0 11 0 0 0 35
4 10 83 124 124 15 1
5 3 0 0 0 0 9 10 30
0 0 0 0 0 0 0 0^M
303.31 746.38 91.99 -
0 0 0 0 0 0 0 0 3 0
0 1 17 13 6 3 0 0 0
125 24 3 4 10 15 90
7 0 0 0 46 2 0 0 0 (
20 45 26 60 125 80 1
0 0 0 1 4 1 0 0 11 5
125 90 1 13 77 39 2
Preview
Filename: O:/unix/projects/cranio3/ezgi/cse455/hw_5/bikes/img1.
OK
Cancel

Then load the image database.
This is the other image you will compare with the query. It's actually a list of filenames.

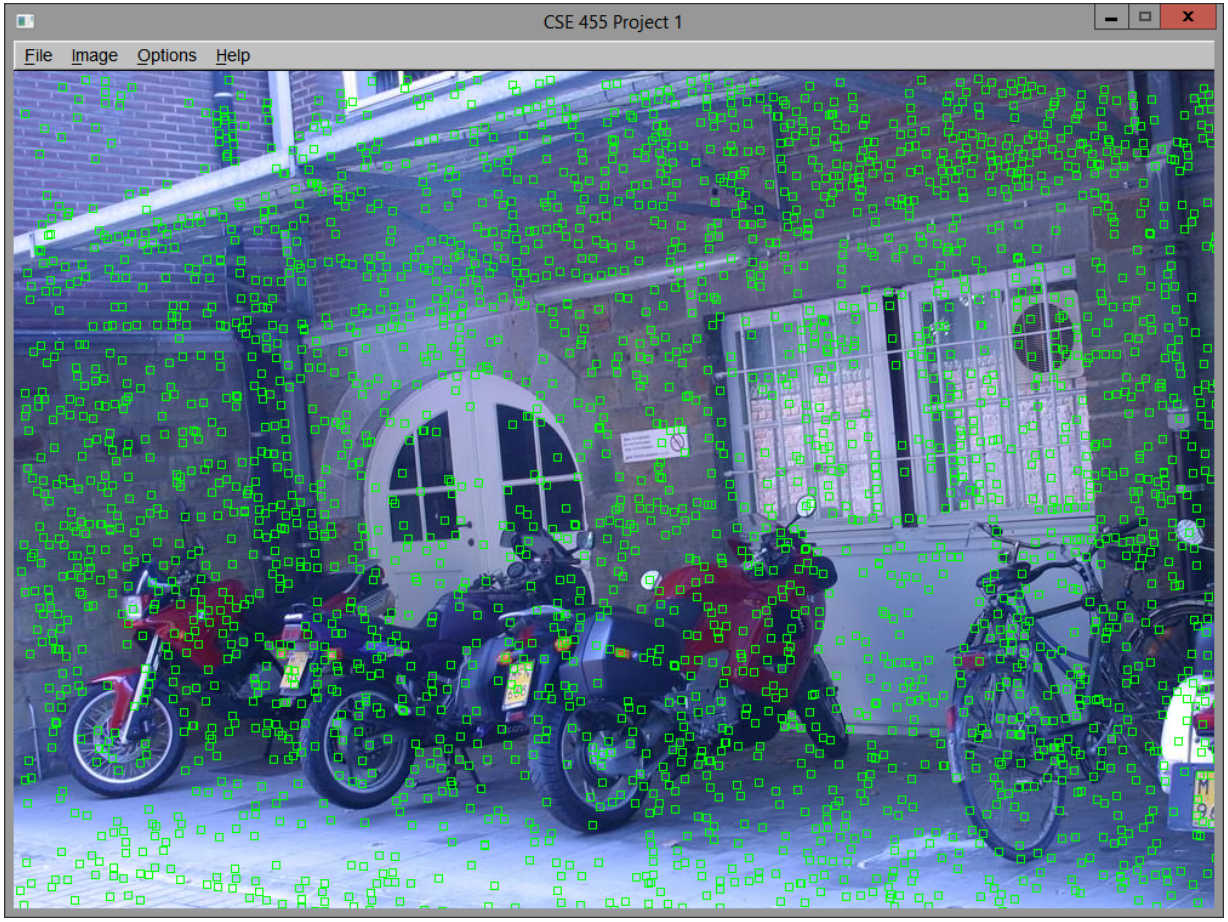



Open File
Show: *.kdb
Favorites
../
img.kdb
img1.kdb
img1simple.kdb
img2.kdb
img2simple.kdb
img3.kdb
img3simple.kdb
img4.kdb
img4simple.kdb
img5.kdb
img2.ppm img2.key^M
Preview
Filename: O:/unix/projects/cranio3/ezgi/cse455/hw_5/bikes/img
OK
Cancel

Then select which interest points will be used in matching. You can select all points.

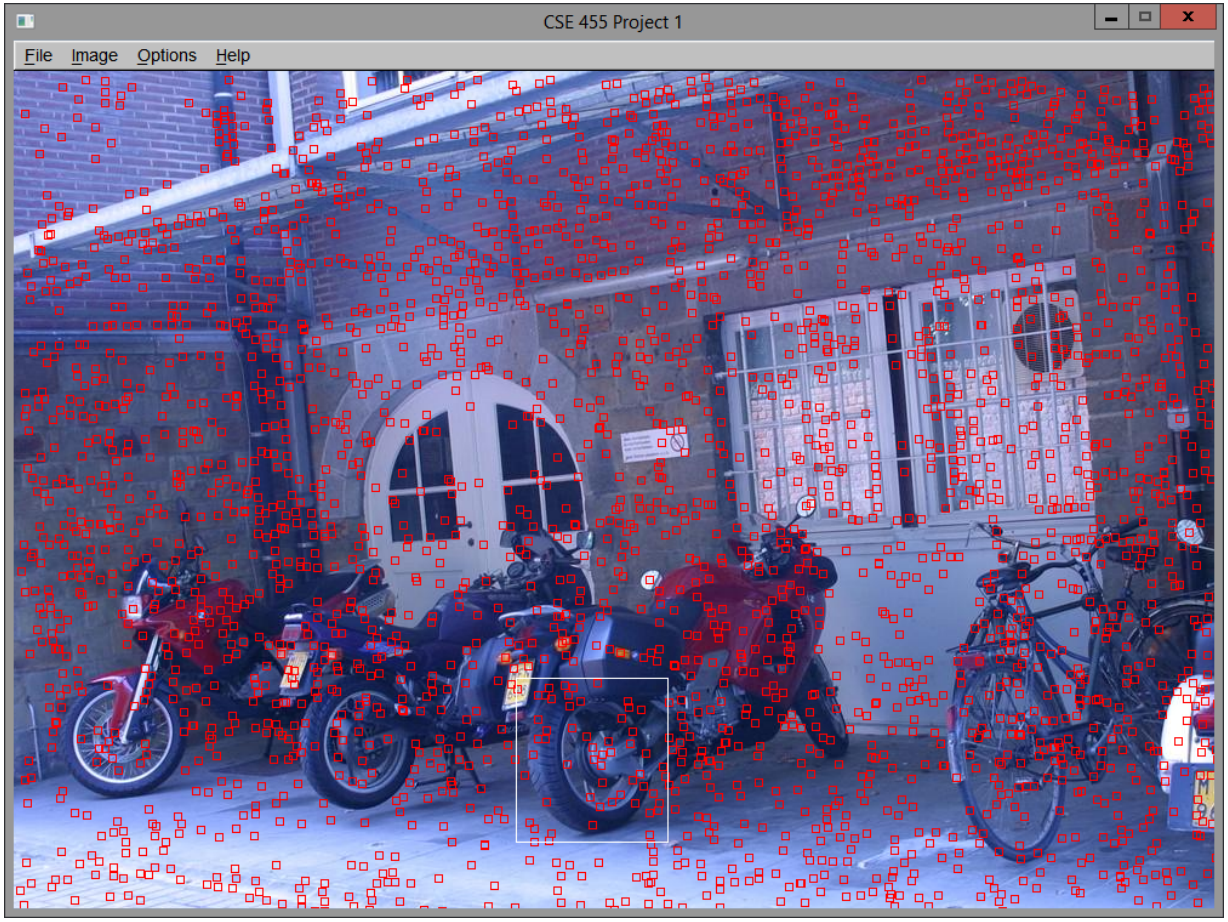


# Selected points are colored green.

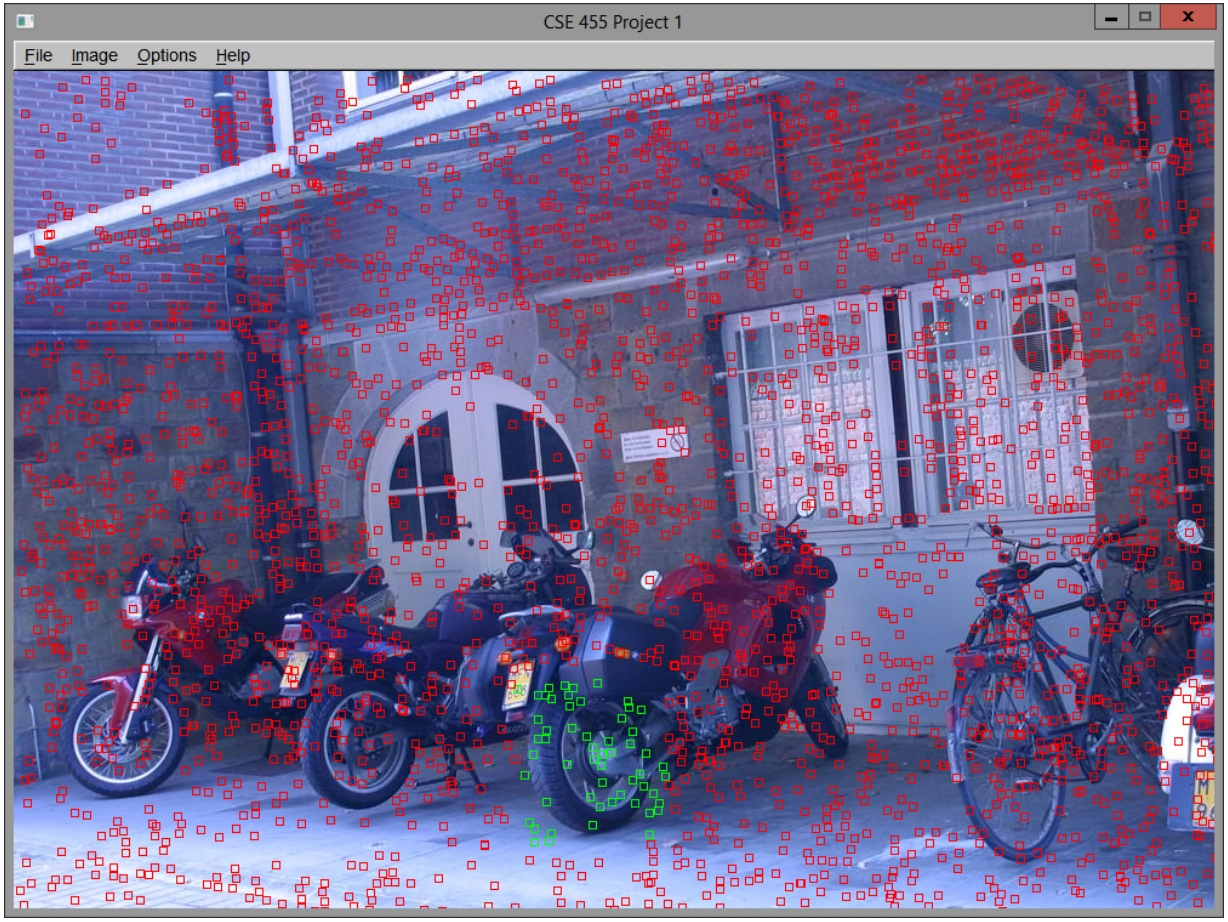


OR

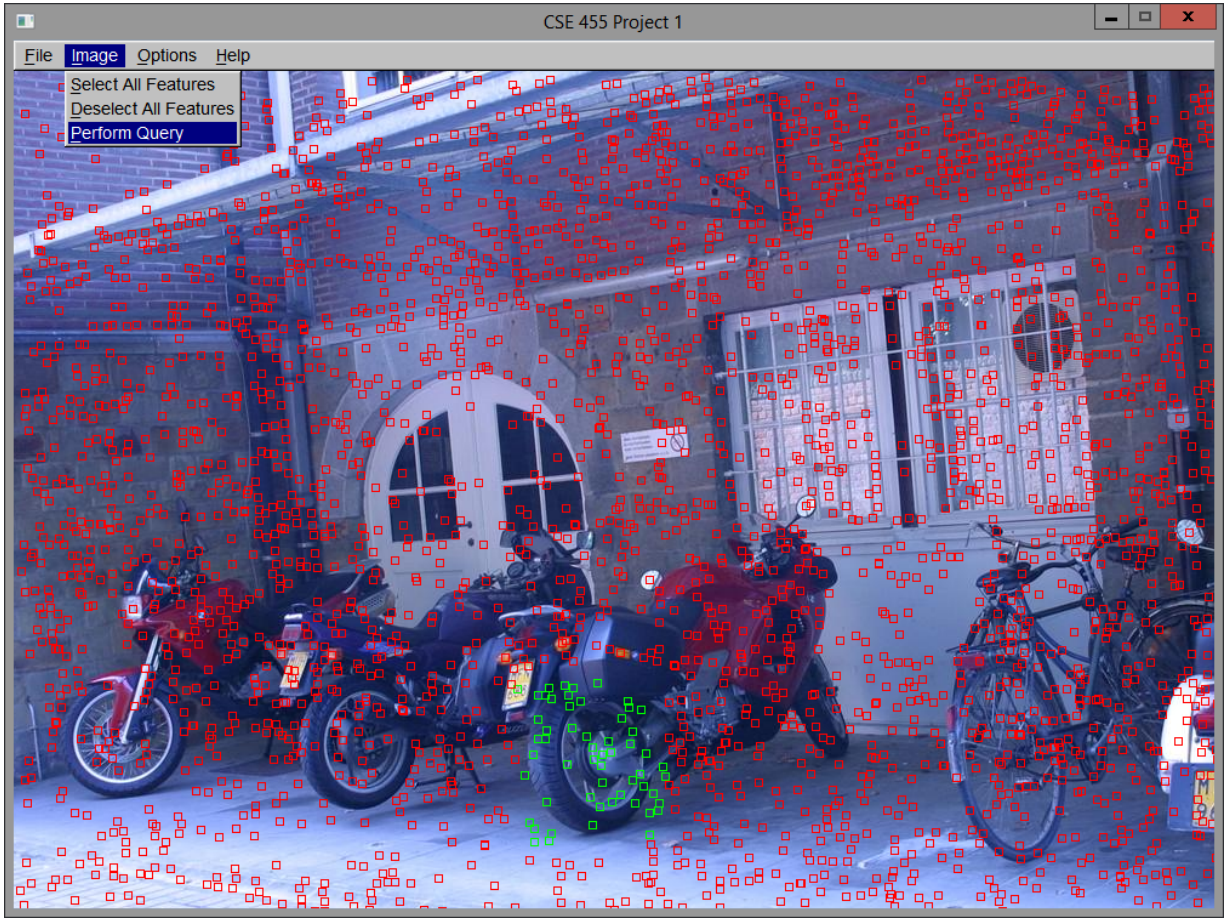
You can select a bunch of points by using the mouse pointer, right clicking and dragging.



# Selected points are colored green.



# Finally, perform the query. This may take a while.



# Matches are returned.