Project 2

Feature Detection and Matching
Due: Sunday Feb 17
Part 1. Feature Detection

- Use Harris corner detection

- For each image point
  - Use a window about the point
  - Compute the Harris matrix $M$
  - Use $R(H) = \det(M) - k(\text{trace } M)^2$ as the corner strength function

- Choose points where $R$ is above a threshold and is a local maximum
Part 2. Feature Description

1. **Simple descriptor**: use a small square window about the feature point (say 5 X 5). This can be the baseline for matching. Feel free to try variants of this basic descriptor, too, like normalizing the gray tones.

2. **Advanced descriptor**: make your descriptor invariant to rotation, using the dominant orientation idea. You will find the dominant orientation in a window about an interest point and use it to rotate the window so that dominant orientation is up, as shown on Matt’s slides.
Computing Dominant Orientation

• Find the gradient magnitude and direction of each pixel in a square window about the interest point

• Create a histogram of gradient directions, using the magnitudes as weights (instead of just adding 1 to bin counts)

• Find the direction $\theta$ with the highest bin value.
Computing the Rotated Window

- Now that you have $\theta$, you can fill an empty descriptor with the values you sample from a counterclockwise rotation by $\theta$.

- When you want the value for a pixel $(x,y)$ in the descriptor, you have to sample it from the “rotated” window in the image. This requires a rotation followed by a translation.

- Follow the directions in the project handout to compute the floating point coordinates, and use interpolation of the 4 closest pixel values to get the value for $(x,y)$. 
• When doing the rotation, assume the interest point is at (0,0)
• After rotation, translate the rotated points by the interest point location

(a-1,-1) -> rotate by 315 -> a'(-1.4,0) -> translate by (6,5) -> a''(4.6,5)
b(0,-1)  -> rotate by 315 -> b'(-0.7, -0.7), translate by (6,5) -> b''(5.3, 4.3)
c(1,-1)  -> rotate by 315 -> c'(-0, -1.4) translate by (6,5) -> c''(6,3.6)
d(-1,0)  -> rotate by 315 -> d'(-0.7,0.7) translate by (6,5) -> d''(5.3, 5.7)
p(0,0)   -> rotate by 315 -> p(0,0) translate by (6,5) -> p(6,5)
e(1,0)   -> rotate by 315 -> e'(0.7,-0.7) translate by (6,5) -> e''(6.7,4.3)
f(-1,1)  -> rotate by 315 -> f'(0, 1.4) translate by (6,5) -> f''(6,6.4)
g(0,1)   -> rotate by 315 -> g'(0.7,0.7) translate by (6,5) -> g''(6.7,5.7)
h(1,1)   -> rotate by 315 -> h'(1.4,0) translate by (6,5) -> h''(7.4,5)
Part 3. Feature Matching

- You will match your descriptors across a pair of images $I_1$ and $I_2$.

- For each feature detected in $I_1$, find the best corresponding feature in $I_2$ or null if there is no good match. The skeleton code provides the SSD to measure the goodness of a match.

- To decide if a match exists, threshold on $(\text{score of best feature match})/(\text{score of second best feature match})$.

- Test on provided data sets. The fundamental matrix giving the exact transformation from one image to another is given. The function applyHomography is given in the C++ code.

- Compare your two feature descriptors and SIFT. Use testMatch for your own features and testSIFTMatch for SIFT features. EvaluateMatch does the evaluation.
Bikes
Leuven
Wall