Project 2

• Python, NumPy, SciPy
  • Tutorial link in writeup
• Main parts
  • Harris corner detection
  • MOPS feature descriptor
  • Simple feature matching
Images + NumPy

• Image coordinates: x, y
• Numpy array
  • Access like a matrix
  • Pixel at coordinate (x, y) is image[y, x]
Keypoint orientation

• Harris corner detector
  • Feature orientation angle *approximation*
    • *angle of gradient* computed on blurred image. 0 angle: counter clockwise from the point (1, 0)
  • **Not** given by an eigenvector of the structure tensor
Take 40×40 square window around detected feature

- Prefilter (because we are subsampling)
- Scale to 1/5 size
- Rotate to horizontal
- Sample 8×8 square window centered at feature
- Intensity normalize the window by subtracting the mean, dividing by the standard deviation in the window

You **don’t** have to implement the multiscale part

CSE 576: Computer Vision

Adapted from slide by Matthew Brown
<table>
<thead>
<tr>
<th>Transformation</th>
<th>Matrix</th>
<th># DoF</th>
<th>Preserves</th>
<th>Icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>translation</td>
<td>$\begin{bmatrix} I &amp; t \end{bmatrix}_{2\times3}$</td>
<td>2</td>
<td>orientation</td>
<td></td>
</tr>
<tr>
<td>rigid (Euclidean)</td>
<td>$\begin{bmatrix} R &amp; t \end{bmatrix}_{2\times3}$</td>
<td>3</td>
<td>lengths</td>
<td>🟠</td>
</tr>
<tr>
<td>similarity</td>
<td>$\begin{bmatrix} sR &amp; t \end{bmatrix}_{2\times3}$</td>
<td>4</td>
<td>angles</td>
<td>🟠</td>
</tr>
<tr>
<td>affine</td>
<td>$\begin{bmatrix} A \end{bmatrix}_{2\times3}$</td>
<td>6</td>
<td>parallelism</td>
<td>🟠</td>
</tr>
<tr>
<td>projective</td>
<td>$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3\times3}$</td>
<td>8</td>
<td>straight lines</td>
<td>🟠</td>
</tr>
</tbody>
</table>

**Table 2.1** Hierarchy of 2D coordinate transformations. Each transformation also preserves the properties listed in the rows below it, i.e., similarity preserves not only angles but also parallelism and straight lines. The $2 \times 3$ matrices are extended with a third $[0^T \ 1]$ row to form a full $3 \times 3$ matrix for homogeneous coordinate transformations.
• We provide you a routine, `cv2.warpAffine`, that can perform the resampling, transformation and cropping
• You have to pass a **forward** warping affine transformation matrix (2x3), multiplied from the **left**, the coordinates are represented as a **column vector**
MOPS descriptor

• You can combine transformations together to get the final transformation
• Pass this transformation matrix to `cv2.warpAffine`

\[ T = ? \]
$T = M_{T_1}$
$T = M_R M_{T_1}$
Scale

\[ T = M_S M_R M_{T1} \]
Translate

\[ T = M_{T2}M_SM_RM_{T1} \]
Crop

• `cv2.warpAffine` also takes care of the cropping
Demo