Feature descriptors and matching
The SIFT descriptor

SIFT – Lowe IJCV 2004
Scale Invariant Feature Transform

- DoG for scale-space feature detection
- Take 16x16 square window around detected feature at appropriate scale
  - Compute gradient orientation for each pixel
  - Throw out weak edges (threshold gradient magnitude)
  - Create histogram of surviving edge orientations: note: each pixel contributes vote proportional to gradient magnitude
  - Find mode of histogram and rotate patch so that mode is 0

Adapted from slide by David Lowe
SIFT descriptor

Create histogram

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor

Adapted from slide by David Lowe
SIFT vector formation

• Computed on rotated and scaled version of window according to computed orientation & scale
  • resample the window
Reduce effect of illumination

• 128-dim vector normalized to 1: invariance to contrast changes

• Threshold gradient magnitudes to avoid excessive influence of high gradients
  • after normalization, clamp gradients >0.2
  • renormalize
Other tips and tricks

• When identifying dominant orientation, if multiple modes, create multiple keypoints

• Weigh pixels in center of patch more highly (Gaussian weights)

• Trilinear interpolation
  • a given gradient contributes to 8 bins: 4 in space times 2 in orientation
Multiple modes when measuring dominant orientation
Linear interpolation into orientation grid

- Blue arrows are centers of orientation bin
- Pixel with red orientation contributes to:
  - Histogram A with weight q
  - Histogram B with weight p
Bilinear interpolation into spatial grid cells

- Blue dots are centers of histograms
- Red pixel contributes to:
  - Histogram A with weight proportional to $r \cdot s$
  - Histogram B with weight proportional to $p \cdot s$
  - Histogram A with weight proportional to $p \cdot q$
  - Histogram A with weight proportional to $r \cdot q$
Properties of SIFT

Extraordinarily robust matching technique

- Can handle changes in viewpoint
  - Up to about 60 degree out of plane rotation
- Can handle significant changes in illumination
  - Sometimes even day vs. night (below)
- Fast and efficient—can run in real time
- Lots of code available:
Summary

- Keypoint detection: repeatable and distinctive
  - Corners, blobs, stable regions
  - Harris, DoG

- Descriptors: invariant and discriminative
  - spatial histograms of orientation

- Next up: using correspondences for reconstruction
Geometry of Image Formation
The pinhole camera

• Let’s abstract out the details
The pinhole camera

• We don’t care about the other walls of the box, so let’s remove those
The pinhole camera

Let’s look at individual points in the world and not worry about what they are.
The pinhole camera

• Let’s place the origin at the pinhole, with Z axis pointing away from the screen (called camera plane)
The pinhole camera

- Let's remove the wall with the pinhole: all we care about is that all light rays of interest must pass through the pinhole, i.e., the origin.
The pinhole camera

- Question: Where will we see the “image” of point P on the camera plane?
The pinhole camera

\[ Q(\lambda) = O + \lambda(P - O) \]

\[ \lambda = 0 \Rightarrow Q(\lambda) = O \]
\[ \lambda = 1 \Rightarrow Q(\lambda) = P \]

\[ Q(\lambda) = (0 + \lambda(X - 0), 0 + \lambda(Y - 0), 0 + \lambda(Z - 0)) \]
\[ = (\lambda X, \lambda Y, \lambda Z) \]
The pinhole camera

- Pinhole camera collapses ray $OP$ to point $p$
- Any point on ray $OP = O + \lambda(P - O) = (\lambda X, \lambda Y, \lambda Z)$
- For this point to lie on $Z=-1$ plane:
  \[ \lambda^* Z = -1 \]
  \[ \Rightarrow \lambda^* = \frac{-1}{Z} \]
- Coordinates of point $p$:
  \[ (\lambda^* X, \lambda^* Y, \lambda^* Z) = \left( \frac{-X}{Z}, \frac{-Y}{Z}, -1 \right) \]
The projection equation

- A point $P = (X, Y, Z)$ in 3D projects to a point $p = (x, y)$ in the image

\[
x = \frac{-X}{Z}
\]

\[
y = \frac{-Y}{Z}
\]

- But pinhole camera’s image is inverted, invert it back!

\[
x = \frac{X}{Z}
\]

\[
y = \frac{Y}{Z}
\]
Another derivation

\[ P = (X,Y,Z) \]

\[ \frac{Y}{Z} = \frac{y}{1} \]
A virtual image plane

• A pinhole camera produces an inverted image
• Imagine a "virtual image plane" in the front of the camera
The projection equation

\[ x = \frac{X}{Z} \]
\[ y = \frac{Y}{Z} \]
Consequence 1: Farther away objects are smaller

Image of foot: \( \left( \frac{X}{Z}, \frac{Y}{Z} \right) \)
Image of head: \( \left( \frac{X}{Z}, \frac{Y + h}{Z} \right) \)

\[
\frac{Y + h}{Z} - \frac{Y}{Z} = \frac{h}{Z}
\]
Consequence 2: Parallel lines converge at a point

• Point on a line passing through point A with direction D:
  \[ Q(\lambda) = A + \lambda D \]

• Parallel lines have the same direction but pass through different points
  \[ Q(\lambda) = A + \lambda D \]
  \[ R(\lambda) = B + \lambda D \]
Consequence 2: Parallel lines converge at a point

- Parallel lines have the same direction but pass through different points:
  \[ Q(\lambda) = A + \lambda D \]
  \[ R(\lambda) = B + \lambda D \]
- \( A = (A_x, A_y, A_z) \)
- \( B = (B_x, B_y, B_z) \)
- \( D = (D_x, D_y, D_z) \)
Consequence 2: Parallel lines converge at a point

• \( Q(\lambda) = (A_X + \lambda D_X, A_Y + \lambda D_Y, A_Z + \lambda D_Z) \)
• \( R(\lambda) = (B_X + \lambda D_X, B_Y + \lambda D_Y, B_Z + \lambda D_Z) \)
• \( q(\lambda) = \left( \frac{A_X + \lambda D_X}{A_Z + \lambda D_Z}, \frac{A_Y + \lambda D_Y}{A_Z + \lambda D_Z} \right) \)
• \( r(\lambda) = \left( \frac{B_X + \lambda D_X}{B_Z + \lambda D_Z}, \frac{B_Y + \lambda D_Y}{B_Z + \lambda D_Z} \right) \)
• Need to look at these points as \( Z \) goes to infinity
• Same as \( \lambda \to \infty \)
Consequence 2: Parallel lines converge at a point

• \( q(\lambda) = \left( \frac{A_X + \lambda D_X}{A_Z + \lambda D_Z}, \frac{A_Y + \lambda D_Y}{A_Z + \lambda D_Z} \right) \)

• \( r(\lambda) = \left( \frac{B_X + \lambda D_X}{B_Z + \lambda D_Z}, \frac{B_Y + \lambda D_Y}{B_Z + \lambda D_Z} \right) \)

\[
\lim_{\lambda \to \infty} \frac{A_X + \lambda D_X}{A_Z + \lambda D_Z} = \lim_{\lambda \to \infty} \frac{A_X}{A_Z + \lambda D_Z} = \frac{A_X}{A_Z} + D_X = \frac{D_X}{D_Z}
\]

\[
\lim_{\lambda \to \infty} q(\lambda) = \left( \frac{D_X}{D_Z}, \frac{D_Y}{D_Z} \right) \quad \lim_{\lambda \to \infty} r(\lambda) = \left( \frac{D_X}{D_Z}, \frac{D_Y}{D_Z} \right)
\]
Consequence 2: Parallel lines converge at a point

• Parallel lines have the same direction but pass through different points

\[ Q(\lambda) = A + \lambda D \]
\[ R(\lambda) = B + \lambda D \]

• Parallel lines converge at the same point \( \left( \frac{D_X}{D_Z}, \frac{D_Y}{D_Z} \right) \)

• This point of convergence is called the \textit{vanishing point}

• What happens if \( D_Z = 0 \)?
Consequence 2: Parallel lines converge at a point
What about planes?

\[ N_X X + N_Y Y + N_Z Z = d \]

\[ \Rightarrow N_X \frac{X}{Z} + N_Y \frac{Y}{Z} + N_Z = \frac{d}{Z} \]

\[ \Rightarrow N_X x + N_Y y + N_Z = \frac{d}{Z} \]

Take the limit as \( Z \) approaches infinity

\[ N_X x + N_Y y + N_Z = 0 \]

Vanishing line of a plane
What about planes?

Normal: \((N_X, N_Y, N_Z)\)

What do parallel planes look like?

\[
N_X X + N_Y Y + N_Z Z = d
\]

\[
N_X X + N_Y Y + N_Z Z = c
\]

\[
N_X x + N_Y y + N_Z = 0
\]

\[
N_X x + N_Y y + N_Z = 0
\]

Vanishing lines

Parallel planes converge!
Vanishing line

\[ N_X X + N_Y Y + N_Z Z = d \]

• What happens if \( N_X = N_Y = 0 \)?
• Equation of the plane: \( Z = c \)
• Vanishing line?
Changing coordinate systems
Changing coordinate systems
Changing coordinate systems
Changing coordinate systems
Changing coordinate systems
Changing coordinate systems
Rotations and translations

• How do you represent a rotation?
• A point in 3D: (X,Y,Z)
• Rotations can be represented as a matrix multiplication

\[ \mathbf{v}' = R \mathbf{v} \]

• What are the properties of rotation matrices?
Properties of rotation matrices

• Rotation does not change the length of vectors

\[
\begin{align*}
v' &= Rv \\
\|v'\|^2 &= v'^T v' \\
&= v^T R^T R v \\
\|v\|^2 &= v^T v \\
\Rightarrow R^T R &= I
\end{align*}
\]
Properties of rotation matrices

\[ \Rightarrow R^T R = I \]
\[ \Rightarrow \det(R)^2 = 1 \]
\[ \Rightarrow \det(R) = \pm 1 \]

\[ \det(R) = 1 \quad \text{Rotation} \]
\[ \det(R) = -1 \quad \text{Reflection} \]
Rotation matrices

• Rotations in 3D have an axis and an angle
• Axis: vector that does not change when rotated
  \[ Rv = v \]
• Rotation matrix has eigenvector that has eigenvalue 1
Rotation matrices from axis and angle

- Rotation matrix for rotation about axis \( \mathbf{v} \) and \( \theta \)
- First define the following matrix

\[
[\mathbf{v}] \times = \begin{bmatrix}
0 & -v_z & v_y \\
v_z & 0 & -v_x \\
-v_y & v_x & 0
\end{bmatrix}
\]

- Interesting fact: this matrix represents cross product

\[
[\mathbf{v}] \times \mathbf{x} = \mathbf{v} \times \mathbf{x}
\]
Rotation matrices from axis and angle

- Rotation matrix for rotation about axis $\mathbf{v}$ and $\theta$
- Rodrigues’ formula for rotation matrices

$$R = I + (\sin \theta)[\mathbf{v}] \times + (1 - \cos \theta)[\mathbf{v}]^2 \times$$
Translations

\[ x' = x + t \]

• Can this be written as a matrix multiplication?
Putting everything together

• Change coordinate system so that center of the coordinate system is at pinhole and Z axis is along viewing direction

\[ \mathbf{x}'_w = R\mathbf{x}_w + \mathbf{t} \]

• Perspective projection

\[ \mathbf{x}'_w \equiv (X, Y, Z) \]
\[ \mathbf{x}'_{img} \equiv (x, y) \]
\[ x = \frac{X}{Z} \]
\[ y = \frac{Y}{Z} \]