Image Primitives and Correspondence
3D Object Recognition

- Extract outlines with background subtraction
3D Object Recognition

- Only 3 keys are needed for recognition, so extra keys provide robustness
- Affine model is no longer as accurate
Recognition under occlusion
Example of keypoint detection

(a) 233x189 image
(b) 832 DOG extrema
(c) 729 above threshold
Test of illumination invariance

- Same image under differing illumination

273 keys verified in final match
Examples of view interpolation
Location recognition
SIFT

- Invariances:
  - Scaling: Yes
  - Rotation: Yes
  - Illumination: Yes
  - Perspective Projection: Maybe

- Provides
  - Good localization: Yes
SOFTWARE for Matlab (at UCLA, Oxford)
www.VLFeat.org

SIFT
An open implementation of SIFT

This is a MATLAB/C implementation of SIFT detector and descriptor. It is fairly customizable and features a decomposition of the algorithm in several reusable M and MEX files. This implementation produces interest points and descriptors which are very similar to David Lowe’s implementation.

Remark. This code is well suited to study, understand and modify SIFT, but it is not particularly fast. If you need to compute lots of features, you might be interested in this lightweight C++ version, which does not require MATLAB and comes with a flexible command line interface.
SIFT demos

Run

sift_compile

sift_demo2
Motivation: panorama stitching

- We have two images – how do we combine them?

Step 1: extract features

Step 2: match features
Edge Detection

Canny edge detector
- Compute image derivatives
- if gradient magnitude > $\tau$ and the value is a local maximum along gradient direction – pixel is an edge candidate
• Edge detection, non-maximum suppression
  (traditionally Hough Transform – issues of resolution, threshold selection and search for peaks in Hough space)
• Connected components on edge pixels with similar orientation
  - group pixels with common orientation
Line fitting

\[ A = \begin{bmatrix} \sum x_i^2 & \sum x_i y_i \\ \sum x_i y_i & \sum y_i^2 \end{bmatrix} \]

second moment matrix associated with each connected component

\[ v_1 \text{ - eigenvector of } A \]

\[ v_1 = [\cos(\theta), \sin(\theta)]^T \]

\[ \theta = \arctan(v_1(2)/v_1(1)) \]

\[ \rho = \bar{x} \sin(\theta) - \bar{y} \cos(\theta) \]

- Line fitting lines determined from eigenvalues and eigenvectors of A
- Candidate line segments - associated line quality
Correspondence Problem:

• How to find corresponding areas of two camera images (points, line segments, curves, regions)

• In feature-based matching, the idea is to pick a feature type (e.g. edges), define a matching criteria (e.g. orientation and contrast sign), and then look for matches within a disparity range

• Feature Matching later
Results - Reconstruction
Color Vision

- With the advent of inexpensive color imagery and processing, color information can be used effectively for machine vision.
- Color provides multiple information per pixel, often enabling complex classification.
- Perception of Color depends on three factors:
  - The spectrum of energy in various wavelengths illuminating the object surface,
  - The spectral reflectance of the object surface, which determines how the surface changes the received spectrum into the radiated spectrum,
  - The spectral sensitivity of the sensor irradiated by the object’s surface.
Color Image

Full color

Red

Green

Blue
Color Consider the problem of locating/segmenting faces from images using color.

First we need to identify the range of colors that could be associated with a face.

The lighting conditions would play a significant role.

Even under uniform illumination, other objects could fall into that color space. In this case we could use shape information for the purpose of segmentation.
T4 – primary face color, t-5 and t-6 secondary face clusters
Three major steps are involved in the face segmentation procedure.

First, we need to create a labeled image based on the training data for identifying the color space that would represent the face.

Connected component is used to merge regions that would be part of the face.

The face is identified as the largest component and areas close to the components are merged.
Color Tracking Sensors

- Motion estimation of ball and robot for soccer playing using color tracking
Adaptive Human-Motion Tracking

**Acquisition**
- Decimation by factor 5

**Motion detector**
- Grayscale convers.
- Image differencing
- Motion history im.
- Segmentation

**Validation**
- Skin color presence
- Big contour presence
- Average travelled distance

**Adaptation**
- Continuous adaptation
- Motion initialization

**Skin color detector**
- RGB to HSV convers.
- Hue-saturat. Limiter
- Skin color binary im.
- Image closing
- Segmentation

**Tracking**
- Distance scoring
- Contour to target assignment
- Event creation
- Narrative-level output
Image Primitives and Correspondence

Given an image point in left image, what is the \textit{(corresponding)} point in the right image, which is the projection of the same 3-D point
Difficulties – ambiguities, large changes of appearance, due to change of viewpoint, non-uniquess
Matching - Correspondence

Lambertian assumption

\[ I_1(x_1) = \mathcal{R}(\rho) = I_2(x_2) \]

Rigid body motion

\[ x_2 = h(x_1) = \frac{1}{\lambda_2(X)}(R\lambda_1(X)x_1 + T) \]

Correspondence

\[ I_1(x_1) = I_2(h(x_1)) \]
Local Deformation Models

- Translational model

\[ h(x) = x + d \]

- Affine model

\[ h(x) = Ax + d \]

- Transformation of the intensity values taking into account occlusions and noise

\[ I_1(x_1) = f_0(X, g)I_2(h(x_1)) + n(h(x_1)) \]
Feature Tracking and Optical Flow

- Translational model

\[ I_1(x_1) = I_2(x_1 + \Delta x) \]

- Small baseline

\[ I(x(t), t) = I(x(t) + ud, t + dt) \]

- RHS approximation by the first two terms of Taylor series

\[ \nabla I(x(t), t)^T u + I_t(x(t), t) = 0 \]

- Brightness constancy constraint
• Normal flow

\[ \mathbf{u}_n = \frac{\nabla I^T \mathbf{u}}{\|\nabla I\|} \cdot \frac{\nabla I}{\|\nabla I\|} = -\frac{I_t}{\|\nabla I\|} \cdot \frac{\nabla I}{\|\nabla I\|} \]

Given brightness constancy constraint at single point – all we can recover is normal flow
Optical Flow

- Integrate around over image patch

\[ E_b(u) = \sum W(x,y) \left[ \nabla I^T(x, y, t) u(x, y) + I_t(x, y, t) \right]^2 \]

- Solve

\[
\begin{align*}
\nabla E_b(u) &= 2 \sum_{W(x,y)} \nabla I(\nabla I^T u + I_t) \\
&= 2 \sum_{W(x,y)} \left( \begin{bmatrix}
I_x^2 & I_x I_y \\
I_x I_y & I_y^2
\end{bmatrix} u + \begin{bmatrix}
I_x I_t \\
I_y I_t
\end{bmatrix} \right)
\end{align*}
\]

\[
\begin{bmatrix}
\sum I_x^2 & \sum I_x I_y \\
\sum I_x I_y & \sum I_y^2
\end{bmatrix} u + \begin{bmatrix}
\sum I_x I_t \\
\sum I_y I_t
\end{bmatrix} = 0
\]

\( Gu + b = 0 \)
Optical Flow, Feature Tracking

\[ u = -G^{-1}b \]

\[ G = \begin{bmatrix}
\sum I_x^2 & \sum I_x I_y \\
\sum I_x I_y & \sum I_y^2
\end{bmatrix} \]

Conceptually:
- \( \text{rank}(G) = 0 \) blank wall problem
- \( \text{rank}(G) = 1 \) aperture problem
- \( \text{rank}(G) = 2 \) enough texture – good feature candidates

In reality: choice of threshold is involved
Optical Flow

- Previous method - assumption locally constant flow

- Alternative regularization techniques (locally smooth flow fields, integration along contours)
- Qualitative properties of the motion fields
Point Feature Extraction

\[ G = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \]

- Compute eigenvalues of G
- If smallest eigenvalue \( \sigma \) of G is bigger than \( \tau \) - mark pixel as candidate feature point

- Alternatively feature quality function (Harris Corner Detector)

\[ C(G) = \text{det}(G) + k \cdot \text{trace}^2(G) \]
Harris Corner Detector - Example
Feature Selection

- Compute Image Gradient \( \nabla I^T = [I_x, I_y] \)
- Compute Feature Quality measure for each pixel
  \[ C(x) = \det(G) + k \cdot \text{trace}^2(G) \]
- Search for local maxima

\[ G = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \]
Feature Tracking

- Translational motion model

\[ E(d) = \min_d \sum_{W(x)} [I_2(\tilde{x} + d) - I_1(\tilde{x})]^2 \]

- Closed form solution

\[ d = -G^{-1}b \]

\[ G = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \]

\[ b = \begin{bmatrix} \sum_{W(x)} I_x I_t \\ \sum_{W(x)} I_y I_t \end{bmatrix} \]

- Build an image pyramid
- Start from coarsest level
- Estimate the displacement at the coarsest level
- Iterate until finest level
Coarse to fine feature tracking

1. compute \( d_k = -Gb \)
2. warp the window \( W(x) \) in the second image by
3. update the displacement \( d \leftarrow d + 2d_k \)
4. go to finer level \( k \leftarrow k - 1 \)
5. At the finest level repeat for several iterations
Affine feature tracking

\[ E(A, d, \lambda_E, \delta_E) = \sum_{\tilde{x} \in \mathcal{W}(x)} w(\tilde{x})[I(\tilde{x}, 0) - (\lambda_E I(A\tilde{x}+d, t) + \delta_E)]^2 \]

where

\[ S = \begin{bmatrix}
    x^2 I_x^2 & xy I_x^2 & x y^2 I_x I_y & xy y^2 I_x I_y & x y I_x I_y & x I_x I_y & x I_x I & x I_x & x I_x \\
    x y I_x^2 & y^2 I_x^2 & xy I_x I_y & x y y^2 I_x I_y & x y I_x I_y & x I_x I_y & x I_x I & y I_x & y I_x \\
    x^2 I_x I_y & x y I_x I_y & x^2 I_y I_x & xy y^2 I_y I_x & xy I_y I_x & y I_x I_y & y I_x I & x I_y I & y I_y \\
    x y I_x I_y & y^2 I_x I_y & xy I_x I_y & y^2 I_y I_x & y x y I_y I_x & y I_x I_y & y I_x I & x I_y I & y I_y \\
    x I_x^2 & y I_x I & x I_y I & y I_x I & I_x I & I_x I & I_x I & I_x & I \\
    x I_x I & y I_x I & x I_y I & y I_x I & I_y I & I_y I & I_y I & I_y & I \\
    x I_x & y I_x & x I_y & y I_x & I x & I x & I x & I x & 1 \\
    x I_x & y I_x & x I_y & y I_x & I y & I y & I y & I y & 1 \\
    1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix} \]

and

\[ z = [a_{11} \ a_{12} \ a_{21} \ a_{22} \ d_x \ d_y \ \lambda \ \delta_E]^T \]

\[ z = S^{-1} c \]
Tracked Features
1. Use multiple image streams to compute the information about camera motion and 3D structure of the scene
2. Tracking image features over time

Original sequence

Tracked Features
Structure and Motion Recovery from Video

Computed model
3D coordinates of the feature points

Original picture