Robot Vision:
Letting Robots See

91.450, Robotics I
Brigit Schroeder
Introduction

• What is **computer vision**?
  ➡ “The science and technology of machines that see”
  ➡ Deriving (extracting) contextual, quantitative information about the world from **imagery**.

Pioneer robot with computer vision system: stereo cameras and laser range finders.
Introduction

• What is **computer vision**? (con’t)
  - Cameras and other sensors become the physical **eyes** of a robot system.
  - Artificial intelligence becomes the **brain** of a robot system.
How is computer vision more than a reason to tinker with robots and cool sensors? (e.g. Which applications make computer vision a valuable technological discipline?)

- Planetary Exploration
  - Localization for the Mars Exploration Rover.
- Autonomous Vehicle Navigation
  - Google Smart Car
- Facial Recognition/Tracking
  - Tracking head position & eye movement in research.
- Human-Computer Interaction
  - Improved wheelchair control & navigation.
- Traffic Management
  - License plate detection.
OpenCV

- OpenCV is a set of open source computer vision libraries originally developed by the Intel Corporation.

- Aimed at “real-time” computer vision (e.g. live video and face tracking) and designed for performance optimization.

- OpenCV is 4 libraries in 1:
  - CV: Computer vision algorithms
  - CVAux: “Auxillary” or experimental/beta CV algorithms
  - Core: Linear algebra, matrix support
  - HighGUI: Media and window handling (creating avis, GUIs, etc).

- Written in C/C++ with Python bindings available.
OpenCV Features

- **Image Processing**
  - Gradients, Edges and Corners
  - Filters and Color Conversion
  - Connected Components and Contour Detection
  - Histograms
  - Blob Detection

- **Motion Analysis and Object Tracking**
  - Optical Flow
  - Feature Detection for Tracking
  - Accumulation of Background Statistics
  - 2D/3D Object Tracking

- **Camera Calibration and 3D Reconstruction**
  - Camera Calibration (intrinsic and extrinsic parameters)
  - Checkerboard Calibration Method
  - Stereo Processing

* Orange indicates features possibly applicable topics to RoverHawk CV system

* Some material borrowed from Mark Heslep.
RGB Color Space

• Lighting impacts color values!
HSV Color Space

• Hue, the “true” color (such as red, blue, or yellow);
  – Measured in values of 0-360 by the central tendency of its wavelength

• Saturation, the 'intensity' of the color (or how much grayness is present),
  – Measured in values of 0-100% by the amplitude of the wavelength

• Value, the brightness of the color.
  – Measured in values of 0-100% by the spread of the wavelength
HSV Color Space

• Other pictorial representations:
Image Processing Pipeline

① Grab image
② Filter to smooth image
③ Process for some property
  ① Intensity changes for edges
  ② Blobbing to find an area of a particular color
④ Act on the results

Example: Image processing pipeline for robot line detection
Image Processing Methods

• Color filtering via histograms and thresholding
• Segmentation
• Blob Detection
• Image Noise Removal and Smoothing
  – Median Filter
  – Mean Filter
  – Gaussian Filter
• Edge Detection
  – Sobel
  – Canny
Looking for Colors

• Can train on colors in a region of the image, then track that color
• Best to track colors in HSV color space (RGB is too lighting dependent)
• To account for bright conditions, dynamic image contrast adjustment can be applied (also called histogram equalization).
Looking for Colors

Example: Convert robot line follow image to HSV to find lines and barrels.

Peaks in histogram correspond to dominant colors in image
Color Filtering

Threshold image based upon HSV histograms and create a color mask (also called object ‘segmentation’).
Color Blobbing

Group objects by specific color (or groups of color) and mark in image for detection.
Gaussian filter

This filter $H$ is a good approximation to $h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2 + v^2}{\sigma^2}}$.

Properties of Gaussian:
- more weight to the center
- good model of blurring in optical systems
- $\sigma$ corresponds to width of the Gaussian

Prof. Yanco
Gaussian Filter
Mean Blur

- Blurs the image by changing the color of the pixel being looked at to the mean value of the pixels surrounding it. The number of surrounding pixels being looked at is defined by the kernel parameter. If kernel is 3, then the pixel being looked at is the center of a 3x3 box, shown in the diagram.

![Diagram showing mean blur calculation](image)
Mean Blur
Median Blur

- Blurs the image by changing the color of the pixel being looked at to the median value of the pixels surrounding it. The number of surrounding pixels being looked at is defined by the kernel parameter. If kernel is 3, then the pixel being looked at is the center of a 3x3 box, shown in the diagram.
Edge Detection: Sobel

\[ |G| = \sqrt{G_x^2 + G_y^2} \]
Edge Detection: Sobel
Edge Detection: Canny

1. Apply Gaussian filter
2. Sobel edge detection
3. Find direction of edges
4. Relate edge direction to direction that can be traced in an image
5. Nonmaximum suppression used to trace along the edge in the edge direction to suppress any pixel value that is not considered to be an edge
6. Hysteresis used to eliminate streaking (breaking up of an edge contour)
OpenCV Example: *Canny Edge Detection*

Edge detection calculated finds the edges on the input image using the `cvCanny()` function call. Canny algorithm is used to link and find initial segments of strong edges.
OpenCV Sample Code for *Canny Edge Detection*

```c
#include <cv.h>
#include <cvaux.h>
#include <highgui.h>

int main(int argc, char *argv[]) {
    //IplImage is the basic Cv image structure
    IplImage input;
    //Load image from disk; variety of formats auto detected
    //Can also read video formats (avi, mpeg) and cameras (Firewire)
    input = cvLoadImage(argv[1], 1)
    //create the destination buffer, with 8 bits, 3 color channels
    IplImage edge_output = cvCreateImage( cvGetSize(input), 8, 3 );

    if(input && edge_output) {
        //do Canny Edge detection
        cvCanny( input, edge_output, .2, .4)

        //GUI setup and output image display (GTK based on linux)
        cvNamedWindow( "Canny Edge Detection", 1 );
        cvShowImage( "Canny Edge Detection", dst );
        cvWaitKey();
    }
    return 0;
}
```

*Some material borrowed from Mark Heslep.*
OpenCV for ROS

- Available in C++ (good! fast!) and Python (bad! slower!)

- **vision_opencv**: stack containing packages for interfacing OpenCV with ROS
  - **cv_bridge**: converts between ROS Image messages and OpenCV images
OpenCV for ROS

• Create an OpenCV ROS node for image processing:
  
  – **Subscribes** to images published from an camera capture node.

  – **Publishes** processed image or notifications based upon image processing (e.g. *I found a rock!* ) such as the location of a bounding box.

  – Use `roscreate-pkg` to create node, typically with the following OpenCV and image transport dependencies:

    ```
sensor_msgs
opencv2
cv_bridge
roscpp
std_msgs
image_transport
```
OpenCV for ROS

• Simple camera and OpenCV ROS processing pipeline for rock detection:

- /uvc_camera node
  - image capture node
  - publish raw image

- /image_raw
  - /camera_info

- /image_color
  - /image_rect_color

- /image_proc node
  - rock detection node
  - send notification of detection

- /image_view2
  - (rock!)
  - send notification (e.g. display detection) node

- /rosout
  - notification (e.g. display detection) node

- /rosout
Accurate Driving of the Mars Exploration Rover

PROBLEM: Localization (where am I ?)

Need an accurate estimate of vehicle’s location (localization) to insure accurate navigation of Martian surface.

SOLUTION: Odometry

2 Approaches: Mechanical and Computer Vision

- **Wheel odometry** (estimates of changes in position based on encoder readings of how much wheels turned) not precise enough due to wheel drift.

- **Visual odometry** (estimates of changes in camera position based upon feature tracking in imagery) performed using high fidelity stereo sensors, tracking and stereo matching.

* Mars Rover driving accuracy was improved over new or mixed-soil terrains.
Navigation and Detection on the Mars Exploration Rover

Early version of Rover Control GUI

Rock detection on lunar surface
Stereo Processing

Finding a 3D point cloud from a pair of 2D stereo images via stereo matching and disparity calculations.
SLAM and Visual Odometry: Tracking Robot Path and Location

White line is *robot path* as determined by visual odometry. In this example, robot was travelling on a 40 ft straight path. *Robot location* and *distanced traveled* are also know from visual odometry calculation.
Examples of Pedestrian Tracking and Detection

**IDEAL:** Testing Targets from INRIA & MIT Pedestrian Datasets

**HARDER PROBLEM:** Surveillance Targets at Multiple Scale
Questions?
OpenCV Docs Available Online

• OpenCV Wiki for docs and downloads: Your 1st stop.

• The OpenCV Sourceforge repository includes the a course from CVPR 2001

• The OpenCV Yahoo Group is a high volume list, high noise list and good question/answer repository; its monitored by developers.

* Some material borrowed from Mark Heslep.