ME132 – February 3, 2011

Outline:

- active sensors

- introduction to lab setup (Player/Stage)

- lab assignment

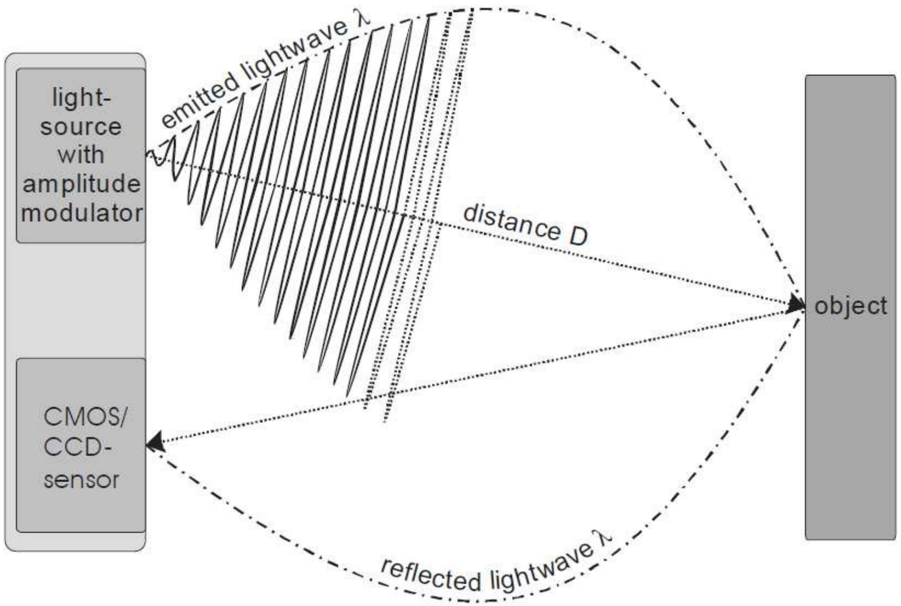
- brief overview of OpenCV

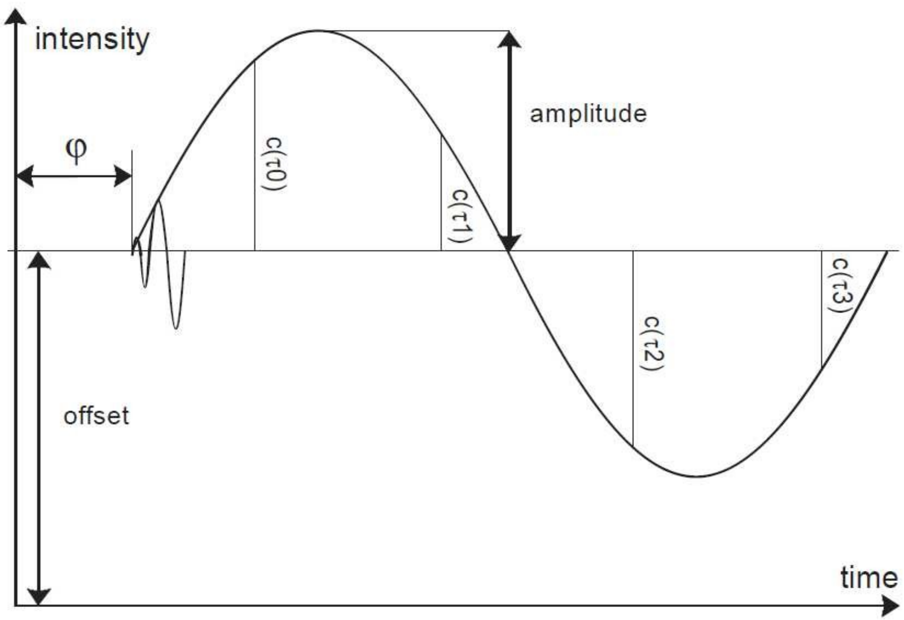
ME132 – February 3, 2011

Outline:

- active sensors
- introduction to lab setup (Player/Stage)
- lab assignment
- brief overview of OpenCV

Active Sensors: Flash Lidar





$$\Delta t = \frac{\phi}{2\pi f_m}$$

$$\begin{aligned} d &= \frac{1}{2} \cdot c\Delta t \\ &= \frac{1}{2} \cdot c\frac{\phi}{2\pi f_m} \end{aligned}$$

Active Sensors: Flash Lidar

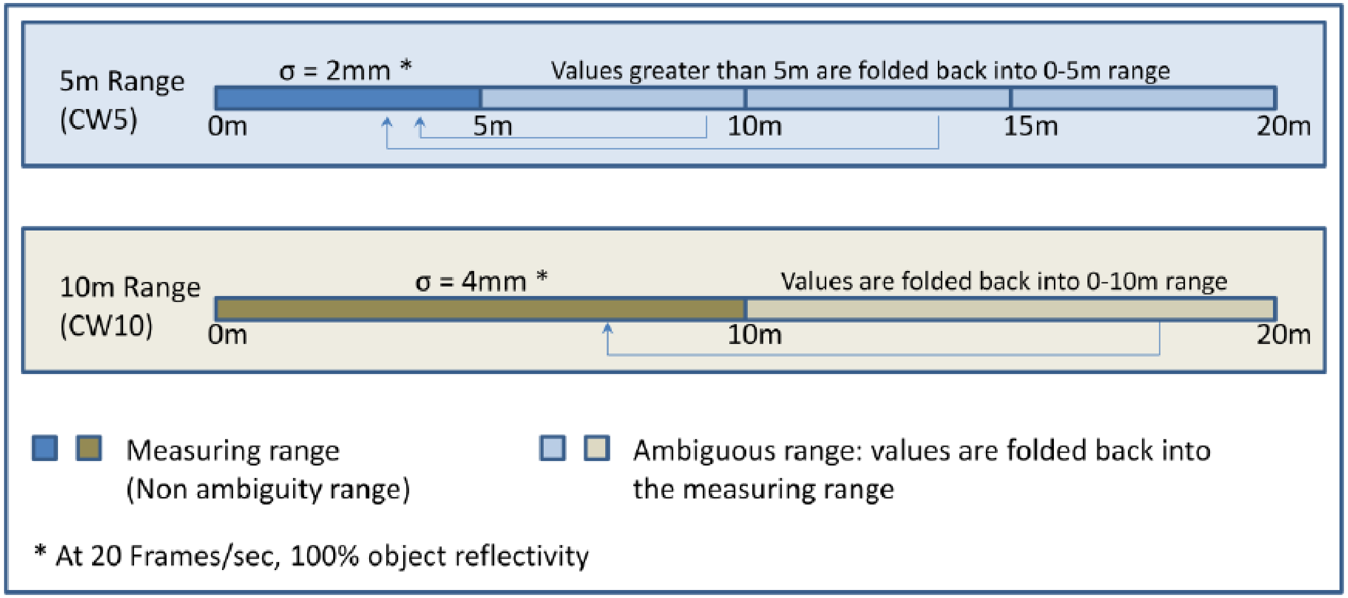


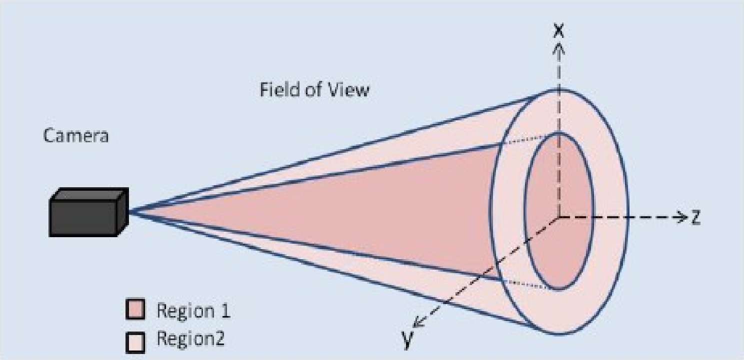
Figure 9: Illustration of the Non-ambiguity range

Kodak grey card (bright)	~ 107%	Rough clean wood palette	~ 25%
White paper	80 - 100%	Smooth concrete	~ 25%
White masonry	~ 85%	Kodak grey card (dark)	~ 33%
Newspaper	~ 70%	Black rubber tire	~ 2%
PVC (grey)	~ 40%		

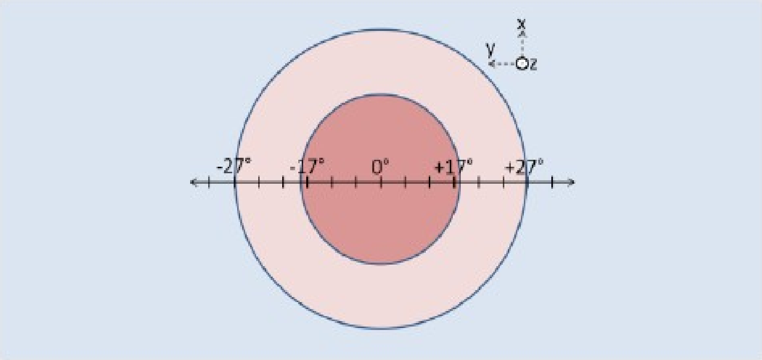
Figure 5: typical reflectivity values for diffusely reflecting materials at a wavelength of 850 nm and at 90° incidence

Active Sensors: Flash Lidar

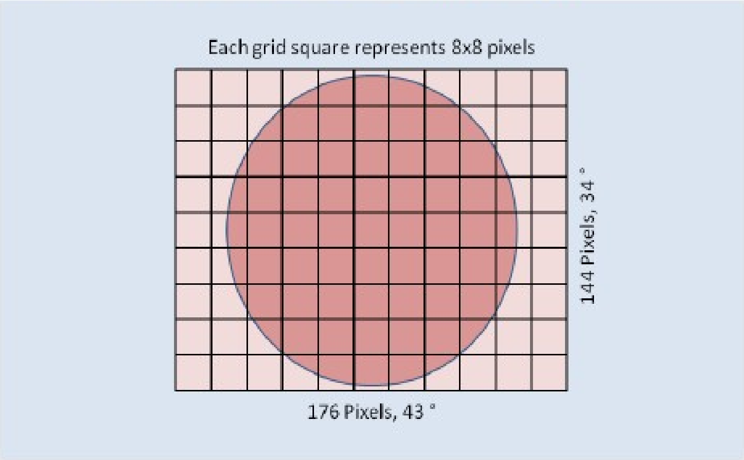
Measurement regions: definition
Region 1: Dark red   Region 2: Bright red



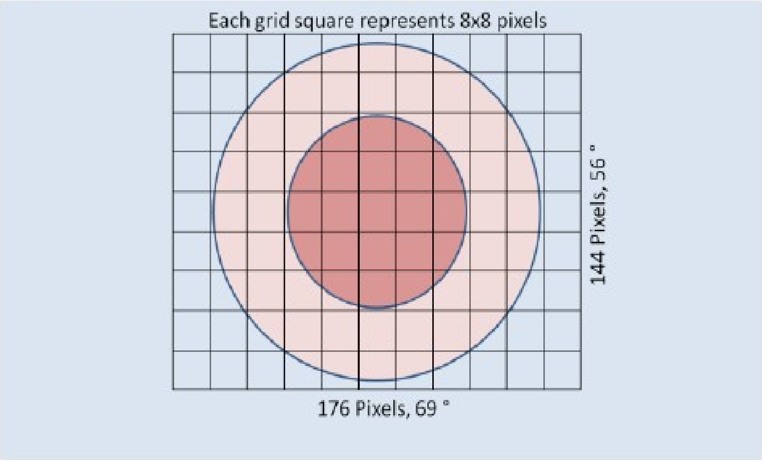
Measurement regions: polar dimensions
Region 1: ±17°   Region 2: ±27°



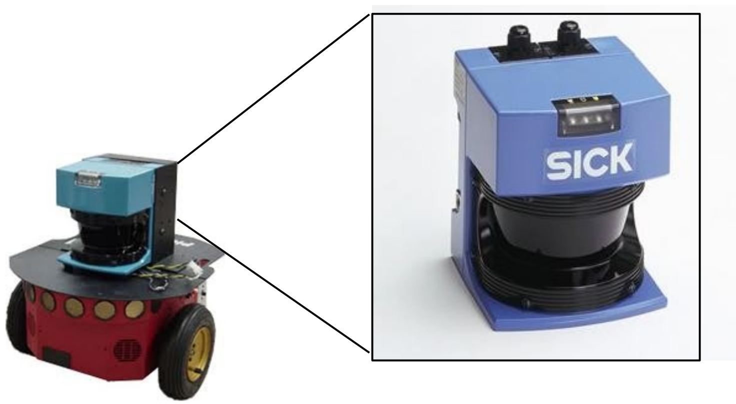
Measurement regions: representation over pixel field
Standard field of view cameras (43° (h) x 34° (v))

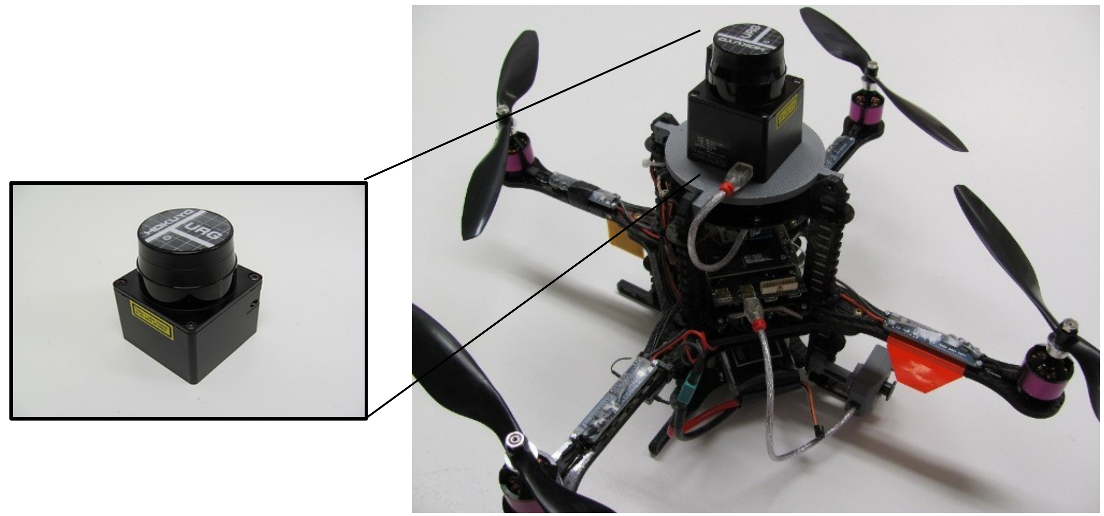


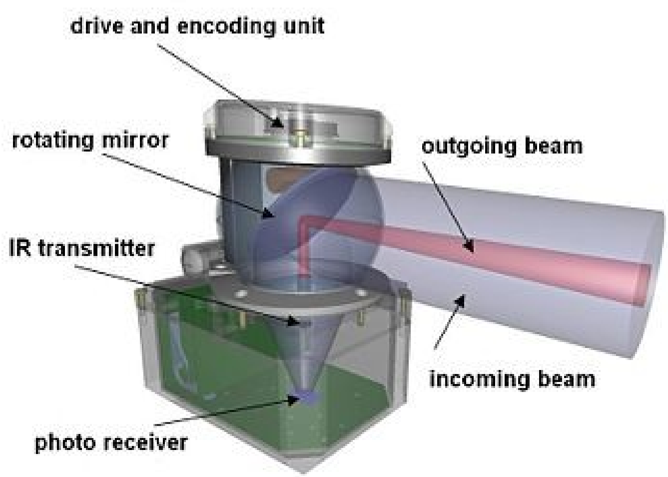
Measurement regions: representation over pixel field
Wide field of view cameras (69° (h) x 56° (v))

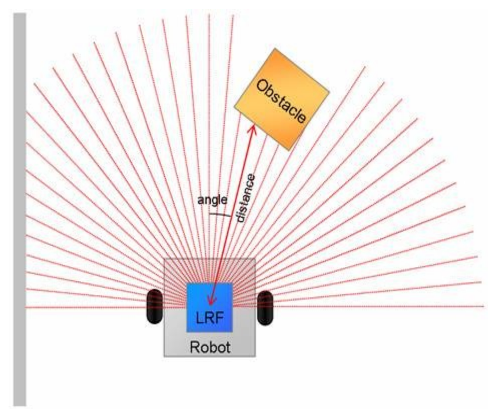


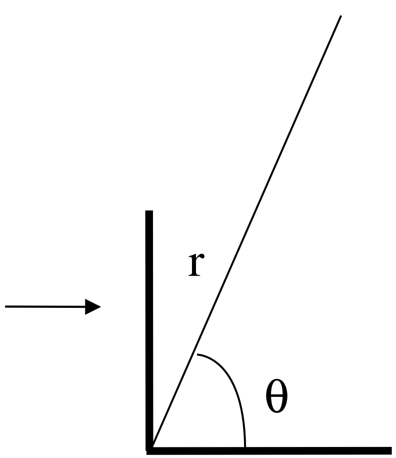
Active Sensors: Laser Scanner











$x = r\cos(\theta)$

$y = r\sin(\theta)$

Active Sensors: Laser Scanner

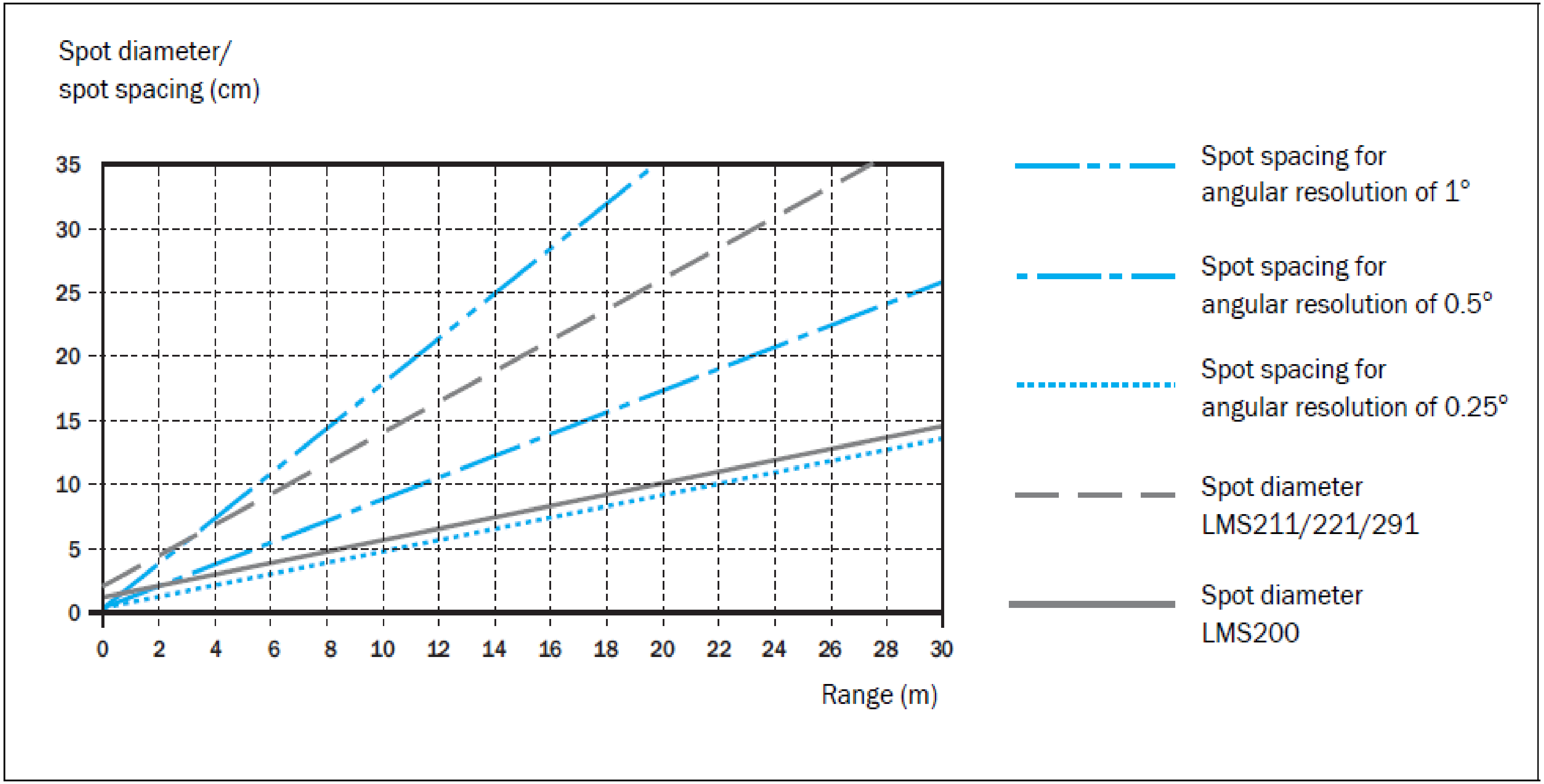
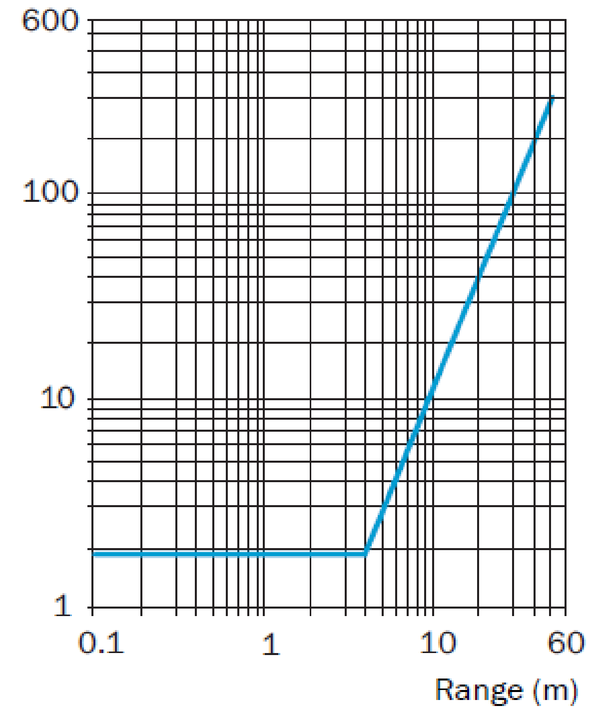


Fig. 3-2: Spot sizes/spot spacing

Active Sensors: Laser Scanner



Material	Reflectivity
Cardboard, matt black	10 %
Cardboard, grey	20 %
Wood (raw pine, dirty)	40 %
PVC, grey	50 %
Paper, matt white	80 %
Aluminium, anodised, black	110 to 150 %
Steel, rust-free shiny	120 to 150 %
Steel, very shiny	140 to 200 %
Reflectors	>2,000 %

Table 4-2: Reflectivity values

Fig. 4-1: LMS200: Range in relation to object reflectivity

Active Sensors: Laser Scanner

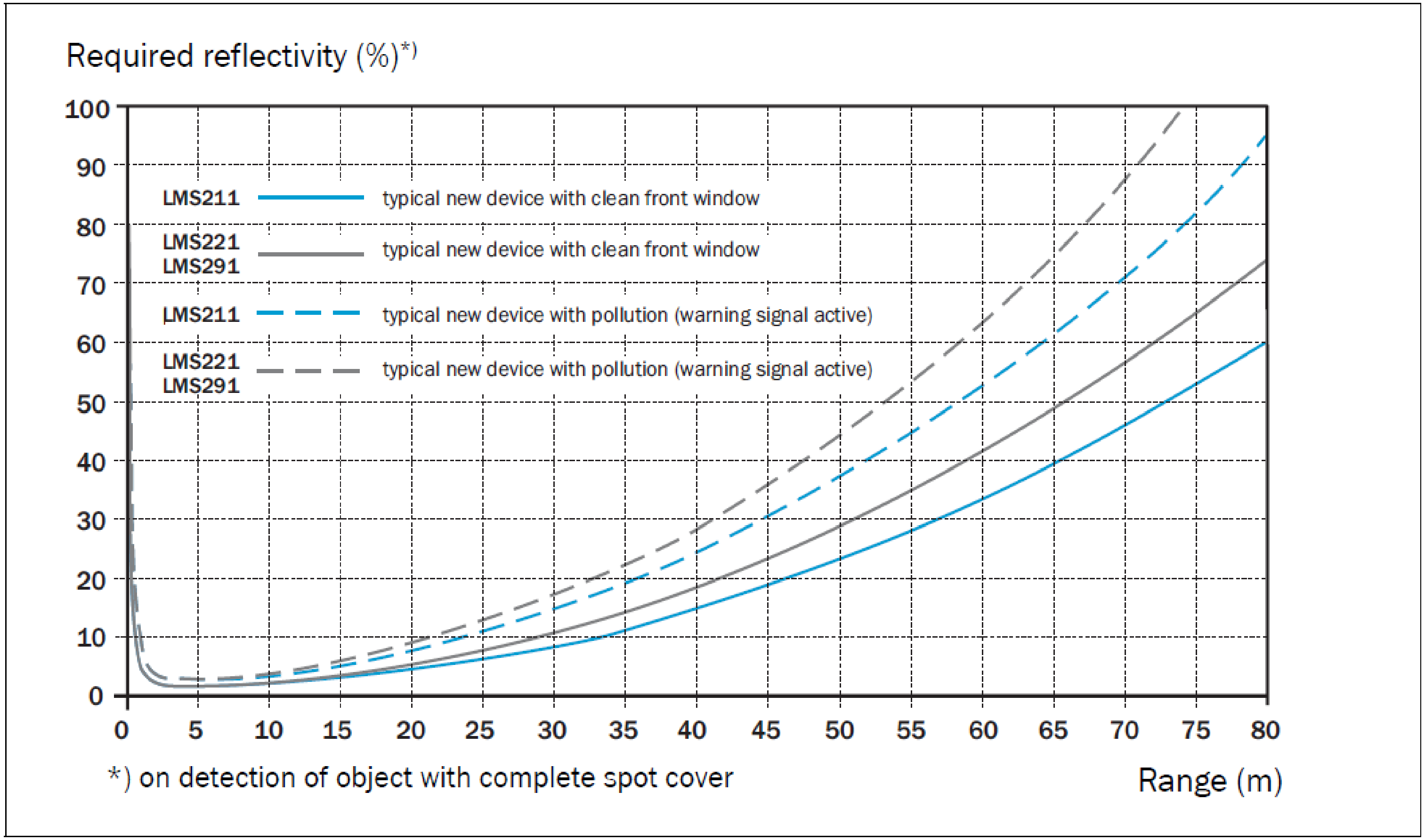
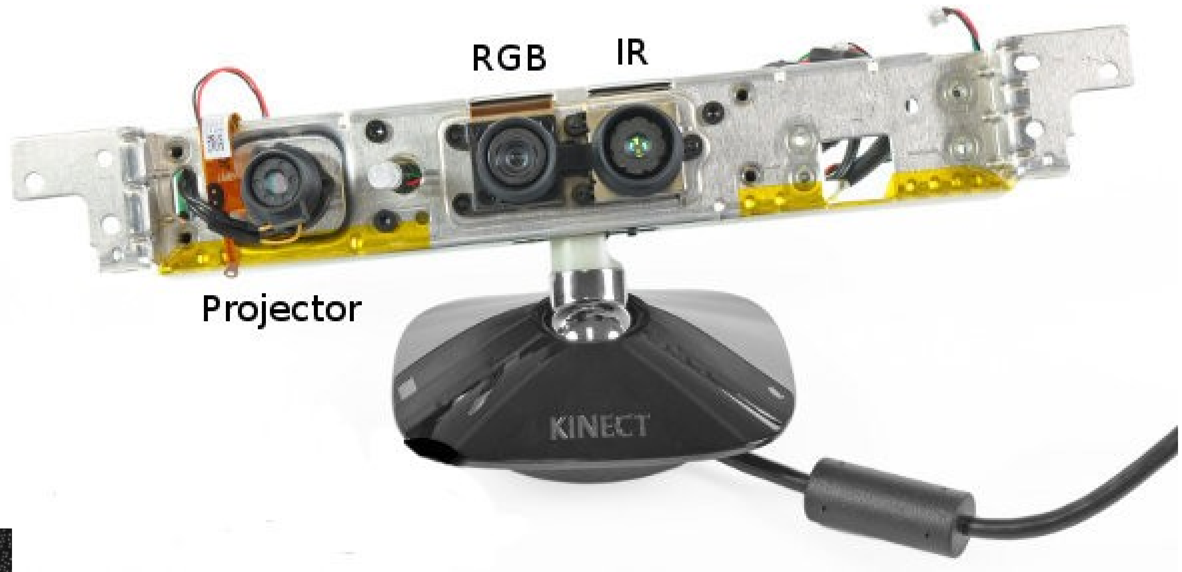
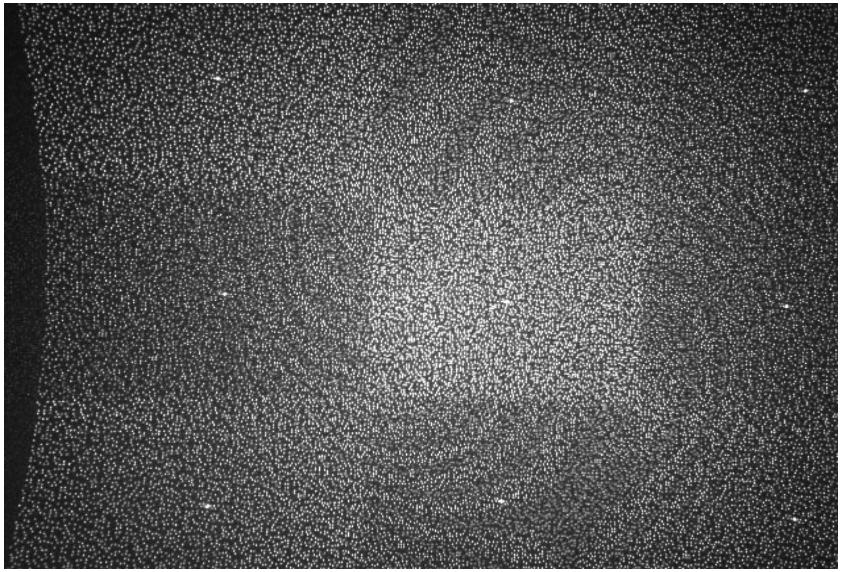


Fig. 4-2: LMS211/LMS221/LMS291: Relationship between reflectivity/range with good visibility

Active Sensors: Kinect





- Projector and IR camera form a stereo pair
- baseline is ~7.5cm
- each pixel in IR camera uses neighboring pixels to form a correlation window
- that window is compared against the list of memorized patterns that are projected
- the best match gives a disparity value and from that, range

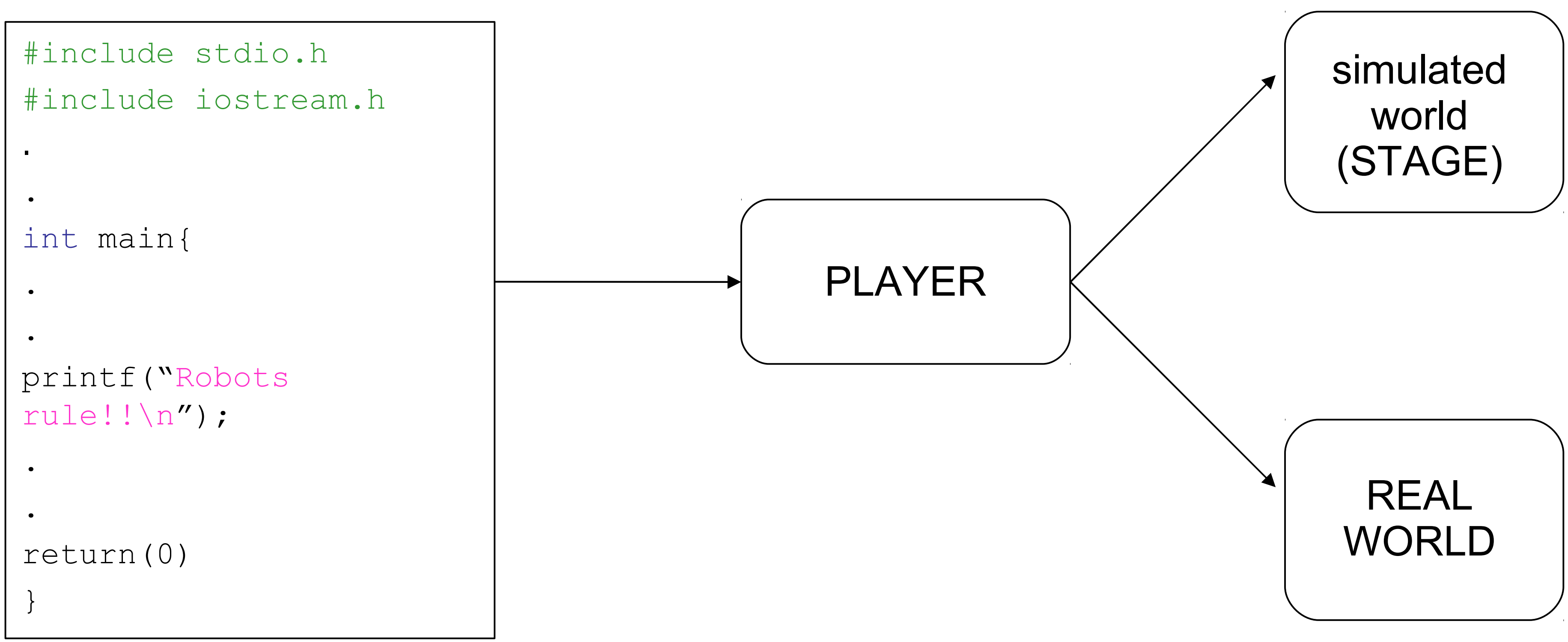
Outline:

- active sensors
- introduction to lab setup (Player/Stage)
- lab assignment
- brief overview of OpenCV

Player: robot device interface

"**Player** provides a network interface to a variety of robot and sensor hardware. **Player**'s client/server model allows robot control programs to be written in any programming language and to run on any computer with a network connection to the robot. **Player** supports multiple concurrent client connections to devices, creating new possibilities for distributed and collaborative sensing and control."

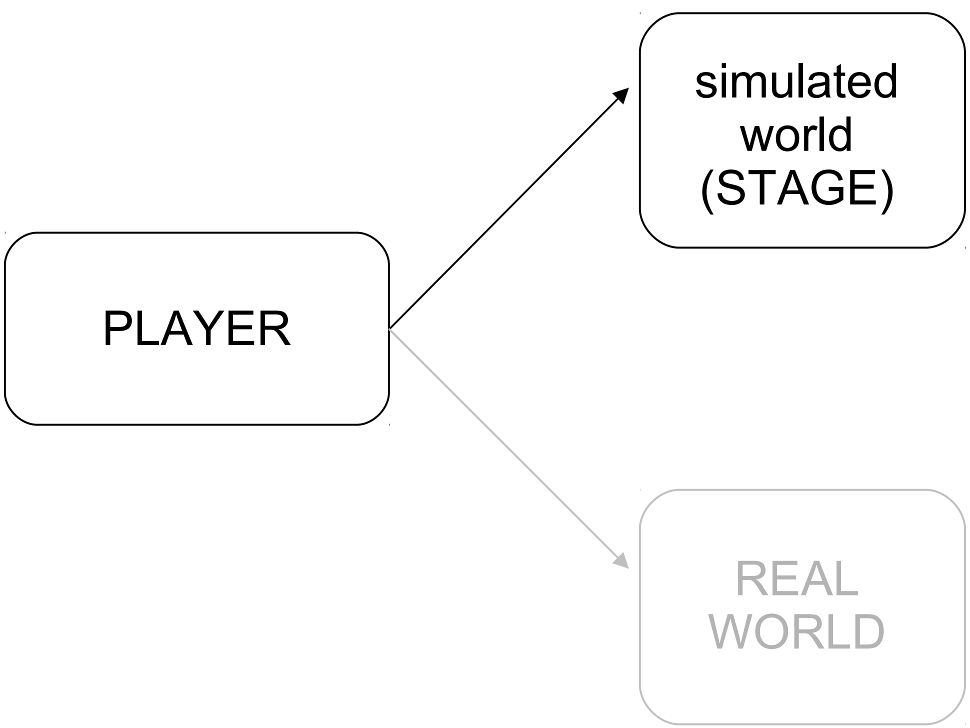
– playerstage.sourceforge.net

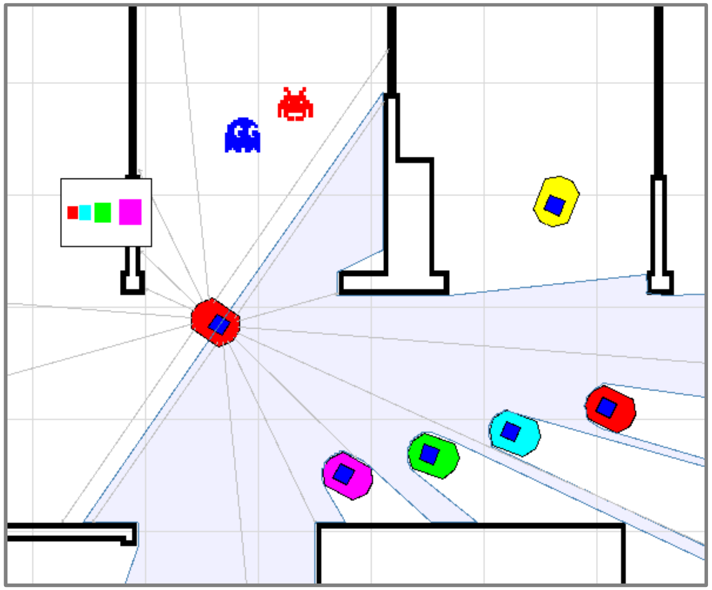


Stage: robot simulator

"**Stage** simulates a population of mobile robots moving in and sensing a two-dimensional bitmapped environment. Various sensor models are provided, including sonar, scanning laser range finder, pan-tilt-zoom camera with color-blob detection and odometry."

– playerstage.sourceforge.net

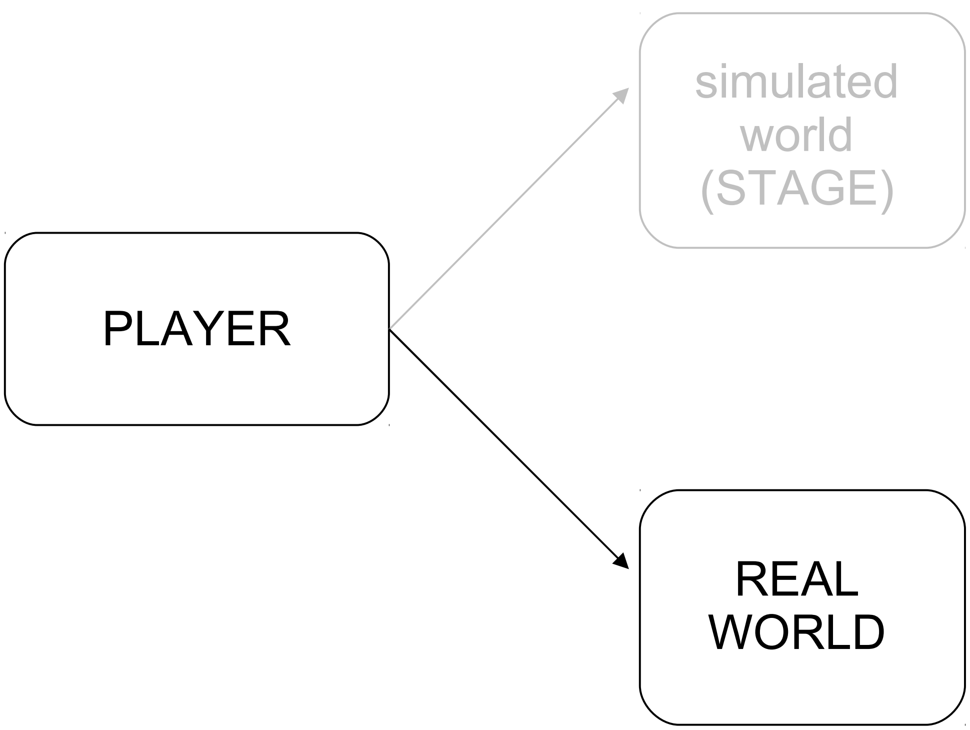




Stage: robot simulator

“**Stage** simulates a population of mobile robots moving in and sensing a two-dimensional bitmapped environment. Various sensor models are provided, including sonar, scanning laser range finder, pan-tilt-zoom camera with color-blob detection and odometry.”

– playerstage.sourceforge.net



Lab Setup: Remote Testing

- a dedicated ME132 server has been setup to host remote testing in preparation for future labs:

`tokyo.cds.caltech.edu`

- all students will have an account on the me132 server

- to do any remote testing in simulation, login to `tokyo` and run the Stage simulator using an assigned port number (the purpose of the port number is to prevent conflicting hosted remote sessions in Stage)

- once the Stage simulator is running, in a separate terminal, run your program and observe robot behavior in the Stage simulator

Lab Setup: Real Testing

- during week-long lab sessions, each student will be required to pair up with one partner (20 students enrolled, 10 groups)

- each group will be given a 1 hour lab session in which they will demonstrate on the robots to the TAs their solution to the assigned lab (which they have prepared during the days/week prior to the scheduled lab session)

- source code / binaries that are to be run on the robots must first be demonstrated in simulation (Stage) to work correctly before running live on the robot

Lab Setup: Hardware

Two Pioneer robot platforms

SICK laser scanner
PGR Bumblebee Stereo Camera

Two laptops running Ubuntu 10.04

Lab Example

Outline:

- active sensors

- introduction to lab setup (Player/Stage)

- lab assignment

- brief overview of OpenCV

Lab Logistics

1) **Find a partner or two and form a team of 2 to 3**

2) **Each student must sign up for an account AND lab session time here (VERY IMPORTANT) by midnight tonight:**

   http://www.cds.caltech.edu/~murray/wiki/index.php/ME/CS_132a,_Winter_2010,_Lab_1_Sign-Up

   You will also receive a Robot Port Identification Number (RPIN) which you will use to access remotely

3) **Once you receive an account, learn about accessing the server remotely and do Exercise #1 and #2 at this site:**

   http://www.cds.caltech.edu/~murray/wiki/index.php/ME/CS_132a,_Winter_2010,_Lab_1

4) **Download the lab assignment from the course website.**

   - The lab is divided into two parts: robot-component, vision-component
   - Robot-component: go through a set of tutorials, familiarizing yourself with the robot hardware
   - Vision-component: learn how to implement an object detector
   - **Both components have on-line and off-line work associated**

   Due date: Tuesday, Feb 15, 2011

Lab: Tutorials

- series of tutorials in C/C++ aimed at familiarizing you (the students) with writing source code for robot control and sensor acquisition
- tutorial source code is already written with compiler instructions as well
- you are encouraged to read through the source code and understand each line

tutorial_0.cc
- learn how to move the robot in simulation and in real-life

tutorial_1.cc
- learn how to acquire data from a laser scanner

tutorial_2.cc
- learn how to run a C library implementation of SIFT extraction on a test image and a reference image

tutorial_3.cc
- learn how to acquire stereo camera data and apply SIFT feature extraction to the stereo imagery

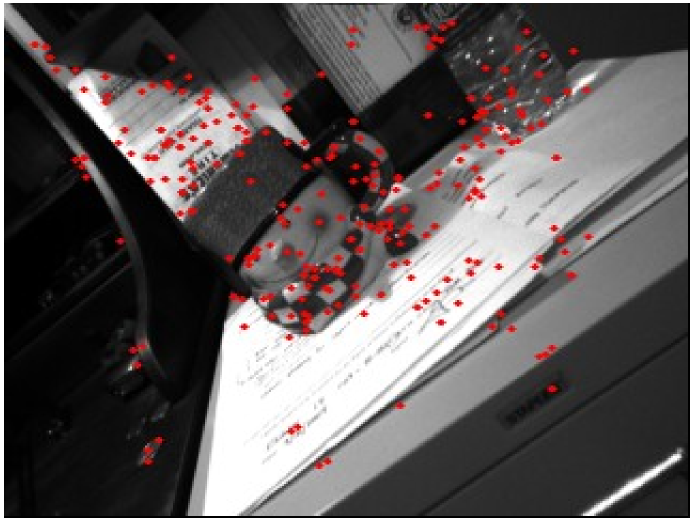
Lab: Object Detection

Given an object model:

Can you detect it in a given scene?

Lab: Object Detection

Traditionally done using feature descriptors:



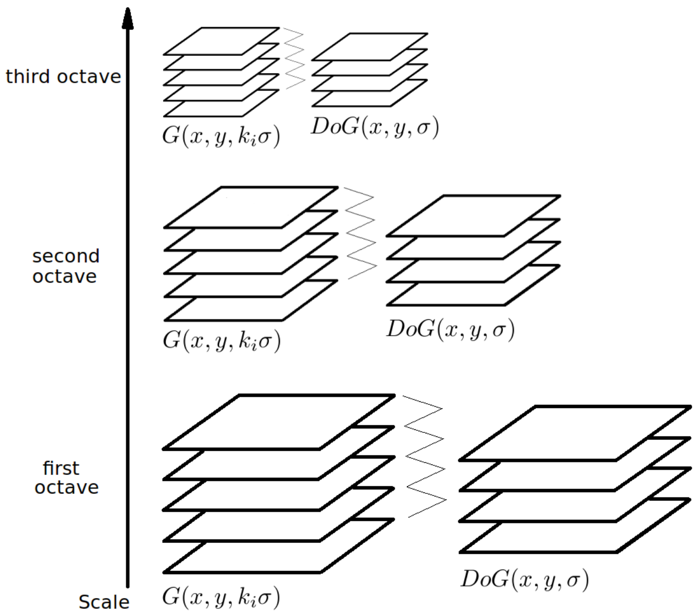
model features (*database*) | current image features

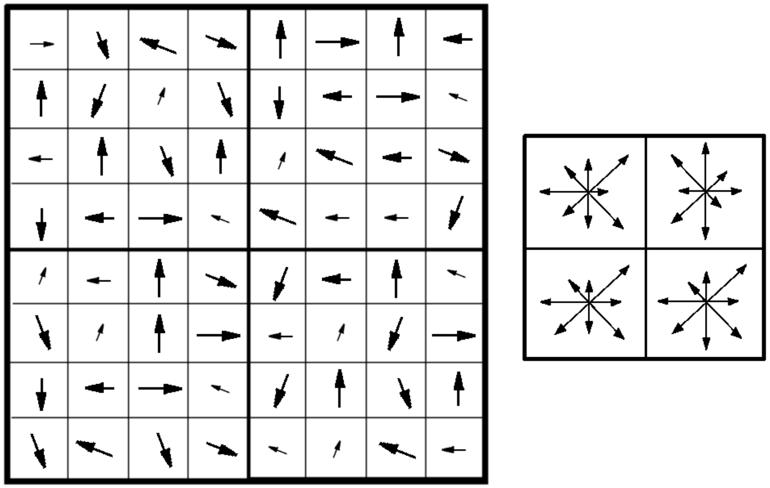
Different types of features can be used:

- SIFT
- Optical Flow
- Corner patches
- Contours
- Edges
- Color Histograms

Lab: Object Detection

Scale Invariant Feature Transform





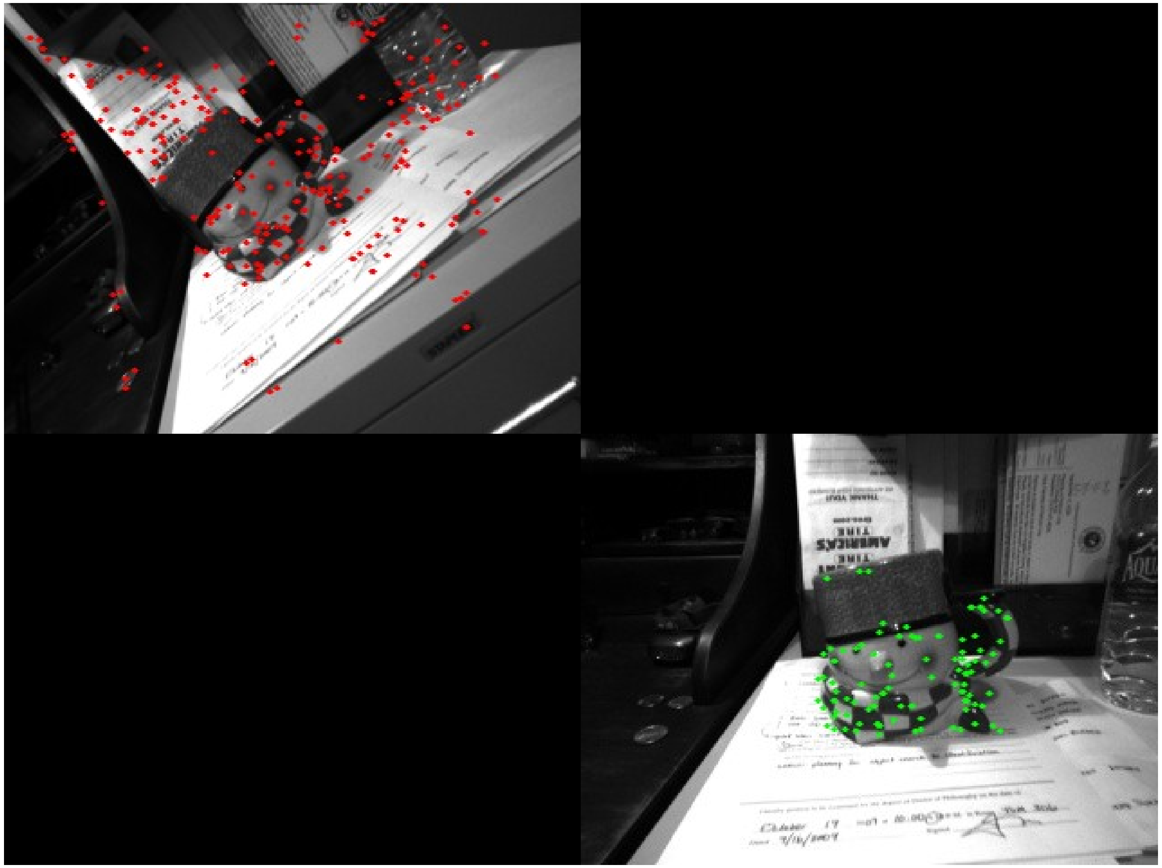
- Interest point detection in scale space
- Interest point selection and localization

- Orientation histogram selection
- Keypoint descriptor assignment

$$\mathbf{d}_b \in \mathbb{R}^{128\times 1}$$

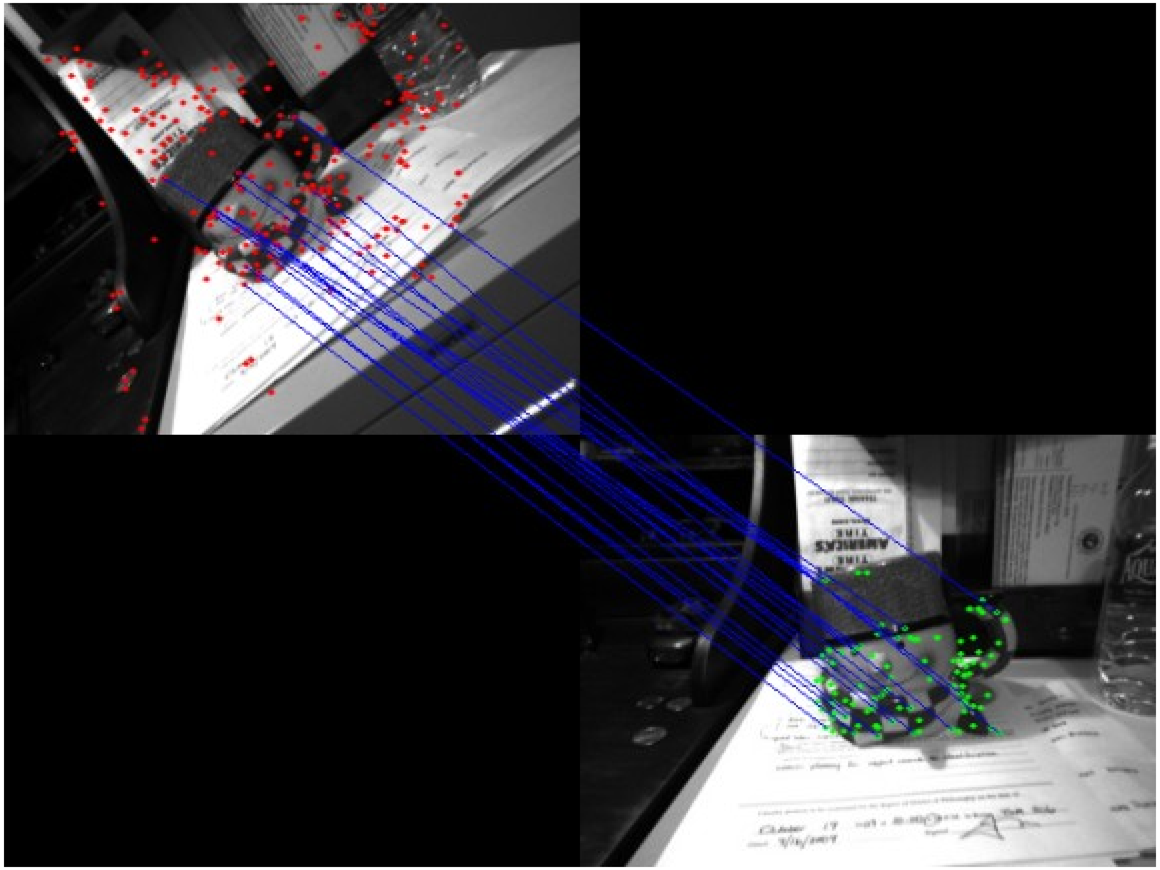
Lab: Object Detection

Feature Correspondence



Lab: Object Detection

Feature Correspondence



Best-Bin-First Search Method
[Beis & Lowe, 1997]

- k-d tree organization
- ratio of nearest neighbor and next-nearest neighbor
- high probability of accurate correspondences

Lab: Object Detection

- You will be given a set of reference images to extract SIFT features on and generate a database of SIFT features (MATLAB functions provided)

- A set of test images will then be provided for you to match against the generated database (which test image matches which reference image?)

- There will be outliers in your matches between test image SIFT features and database of SIFT reference image features.

- Write a homography-based outlier detector that implements RANSAC to estimate the best homography from matched features

- Test your homography outlier detector on a set of test images acquired during your labtime (**tutorial_3.cc**)

Outline:

- active sensors
- introduction to lab setup (Player/Stage)
- lab assignment
- brief overview of OpenCV