TA Section 6

Tools for the Project

- OpenCV
- Point Cloud Library (PCL)

OpenCV

- Began at Intel Russia in 1999
- Useful for almost all 2D vision tasks
- Especially designed for real-time vision
- C++, but full bindings for Python, Java and Matlab
- Free for commercial and non-commercial use

Basic Data Structures

- The `Mat()` class is used to store many different types of data in OpenCV.

- Most commonly used for images

- Can store transformation matrices or lists of points like a Matlab matrix.

- Memory management is handled by the class

- Always 2D! Use MatND for matrices in arbitrary dimensions

Basic Data Structures

Typical constructor calls:

```
Mat(int width, int height, int type, void* data)
Mat(int width, int height, int type, Scalar value)
```

Types are described by constants in the form:

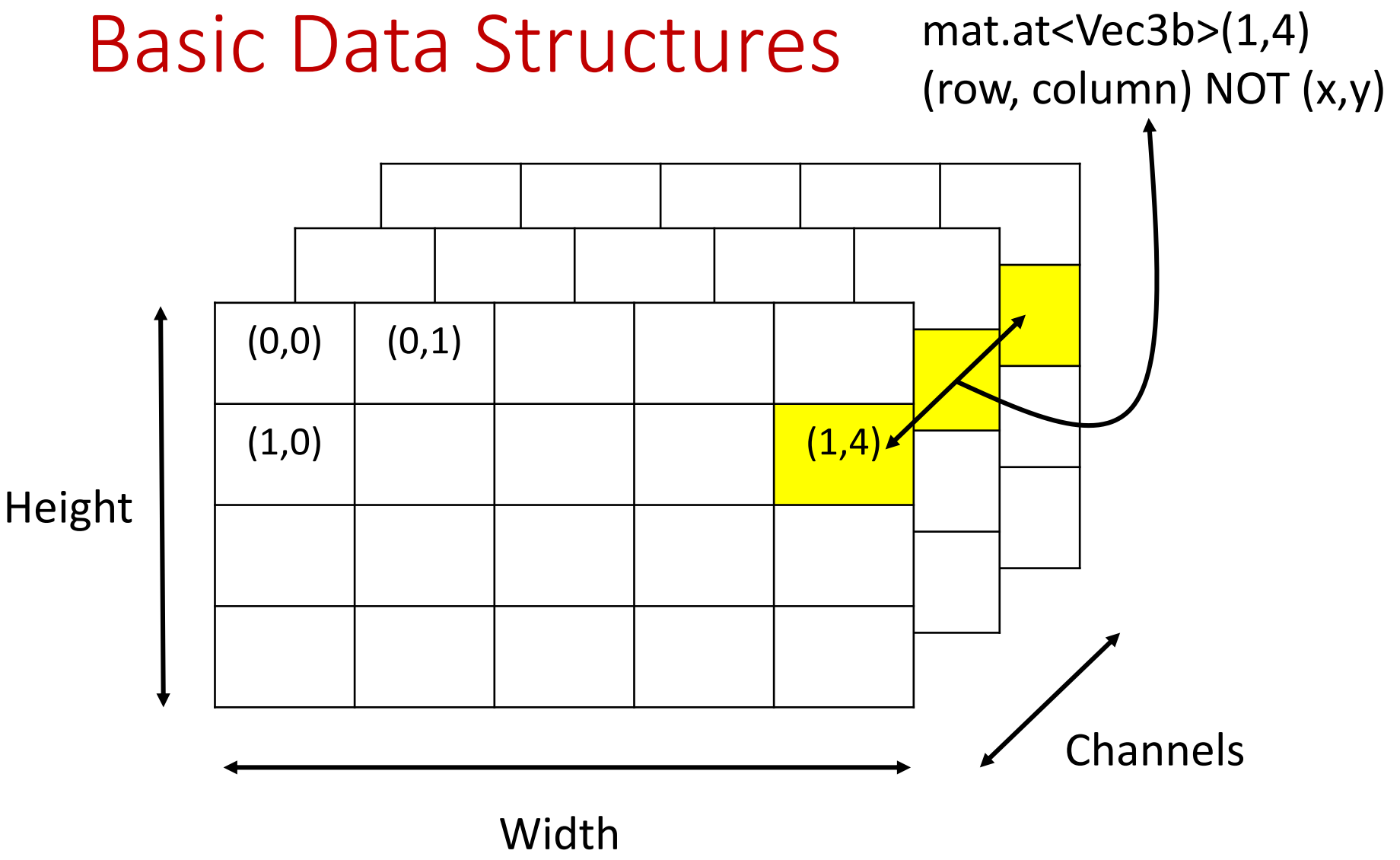
CV_16SC2 – 2 channels of 16 bit signed integers

CV_32FC1 – 1 channel containing a 32 bit float

CV_8UC3 – 3 channels containing 8-bit unsigned ints

Can store RGB, BGR, HSV etc

Basic Data Structures



Depth is type and size of cell (ie CV_8U)

Other Classes

`Vec3b, Vec2d, Vec2i` etc
Vector classes for different types of data

`Point`
Vector type specifically for referring to points in images

`Rect`
Rectangle in an image, useful for limiting the Region of Interest (ROI) of an algorithm

`Scalar`
Vector type used to describe pixel values (arbitrary type and number of channels)

GUI Functionality (highgui)

- Basic, easy GUI – good for debugging
- Limited support for user interaction
- Solid IO support
- Not suitable for professional applications

Windows

`namedWindow(String name)`

Opens a window with the given name. This name is used to identify the window in future GUI calls.

`imshow(String windowName, Mat img)`

Opens a window if one with the given name doesn't exisit. Displays the image. One channel images are displayed greyscale, 3 channel images in BGR. For floating point types the max pixel value should be 1.

Window IO

`waitKey(int millis = 0)`

Runs the GUI for a specified amount of time (or until a key press for millis = 0), processing drawing, mouse and keyboard events.

`setMouseCallback`

Used to register a callback to handle mouse presses and movements.

`destroyWindow`

Removes the named window.

File IO

`Mat img = imread(String filename)`

Reads an image from file (many types supported)

`VideoCapture cap`

Class used to read a stream of data from video or device (webcam, Kinect etc). Use cap >> img to read an image from the stream.

`imwrite(String filename, Mat img)`

Writes an image to file, many file formats supported.

OpenCV Modules

improc
Image processing module. Filtering, geometric transformations, color transformations, drawing functions, histograms, simple edge and corner detectors.

calib3D

Camera calibration – put your new knowledge to work! Great calibration tutorial on OpenCV website.

features2D

Implements loads of feature descriptors (SIFT, ORB, BRISK etc), descriptor matching and BOW models.

OpenCV Modules

core

Command line parsing, operations on arrays, optimization methods, least squares, matrix operations, kmeans.

ml
Implementations of many common machine learning tools: SVMs, boosting, random forests, neural nets, naïve Bayes, logistic regression etc.

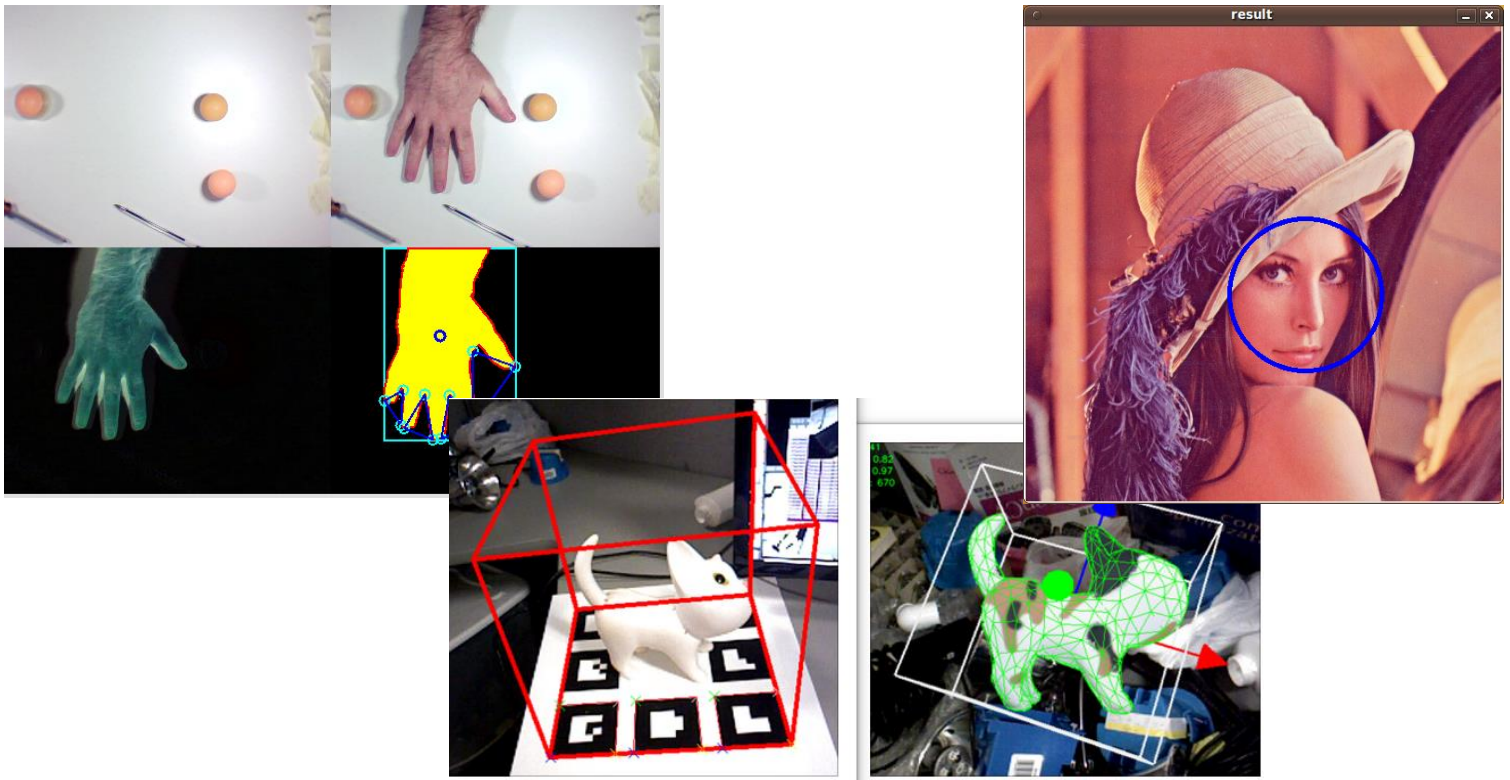
cuda*

A collection of libraries for computer vision on the GPU. Not as mature as the CPU functionality, but very useful for high-performance applications.

OpenCV Questions?

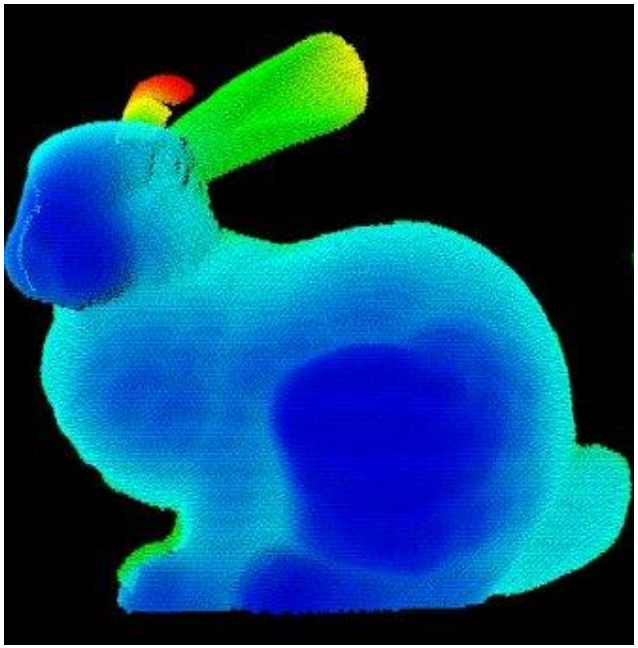
What would you like to do with OpenCV?

The website has lots of great tutorials if you'd like to learn more.

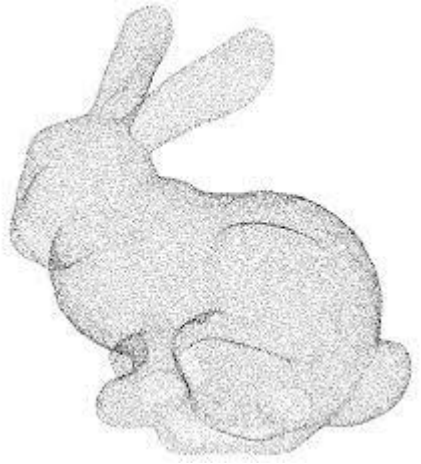


Point Cloud Library

- A library specifically for working with 3D data
- Developed at Willow Garage as the Kinect became hugely popular for vision.
- Not as mature as OpenCV
- Uses Boost and Eigen – two excellent C++ libraries
- C++ only as far as I know, heavily templated



Point Clouds



- Collections of points in space (usually 3D)

- Points can contain data about color, normal, curvature etc as well as their position in space

- Can be generated from a range sensor (ie Kinect, LIDAR etc), mesh, structure from motion algorithm etc etc

- Can be organized into rows and columns, or can be unorganized (a bag of points)

Point Cloud Types

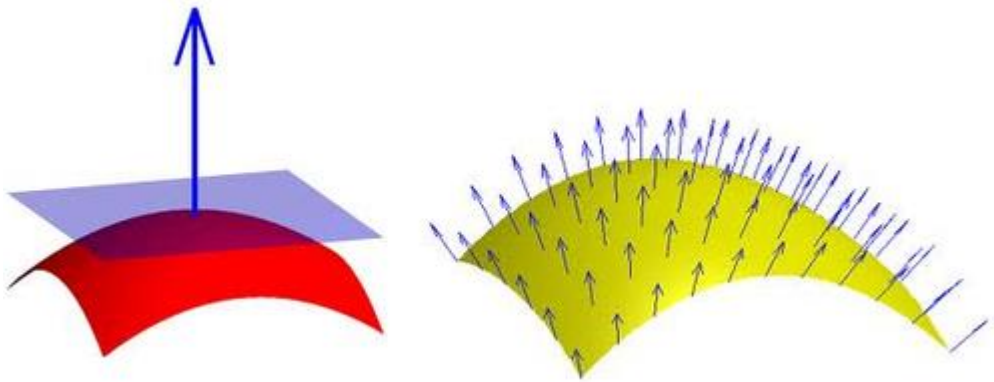
PointCloud<PointXYZ>	(x,y,z)
PointCloud<PointNormal>	(x,y,z,nx,ny,nz)
PointCloud<PointXYZRGB>	(x,y,z,r,g,b)

Everything in PCL is templated.

Different point types encode different information about the points in the cloud, cannot be easily exchanged.

Algorithms – Normal Estimation

```
pcl::NormalEstimation<pcl::PointXYZ, pcl::Normal> ne;
ne.setInputCloud (cloud
pcl::search::KdTree<pcl::PointXYZ>::Ptr tree (
ne.setSearchMethod (new pcl::search::KdTree<pcl::PointXYZ>());
ne.setRadiusSearch (0.03);
ne.compute (*cloud_normals);
```

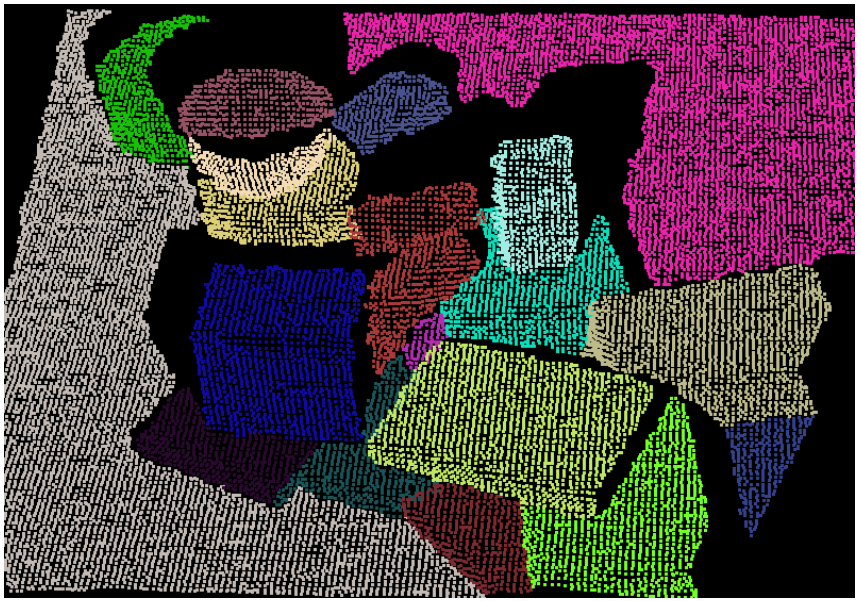


Algorithms – Clustering and Segmentation

PCL supports many different clustering techniques and metrics

Cluster by position, color, normal or a combination of these factors.

K-means, region growing, Grabcut etc

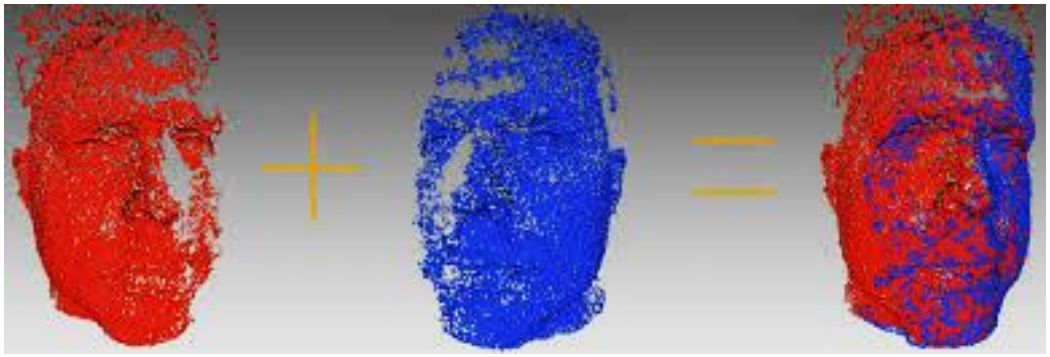


Algorithms – Cloud Alignment

Iterative Closest Point

1) Find closest pairs of points in the two clouds
2) Find the 6DOF transformation that minimizes the distances between correspondences

```
pcl::IterativeClosestPoint<pcl::PointXYZ, pcl::PointXYZ> icp;
icp.setInputCloud(cloud_in);
icp.setInputTarget(cloud_out);
icp.align(cloud_aligned);
std::cout << icp.getFinalTransformation()
```



Algorithms – KinectFusion

Incredible dense SLAM system developed by Microsoft Research

Now has open source PCL implementation!

Uses Iterative Closest Point for tracking, GPU to achieve real time performance

http://youtu.be/quGhaggn3cQ?t=2m45s

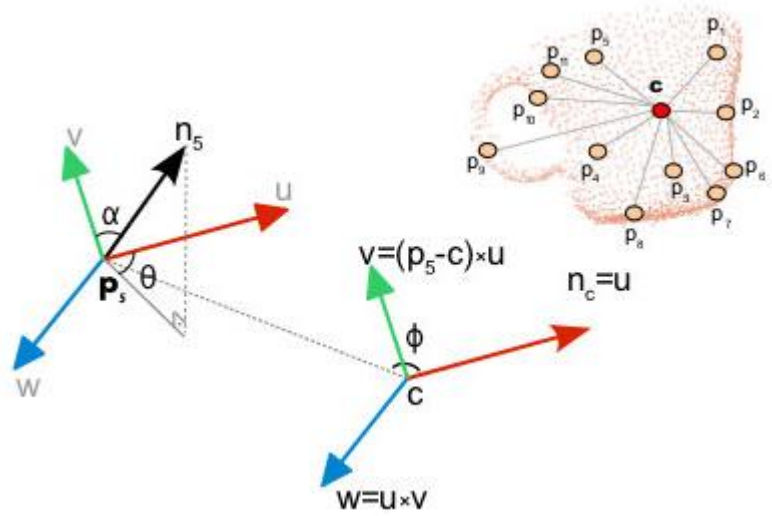
Algorithms – Plane Detection

In many applications the objects you care about lie on a table or ground plane, so you want to detect that plane.

```
pcl::SACSegmentation<pcl::PointXYZ> seg;
seg.setModelType (pcl::SACMODEL_PLANE);
seg.setMethodType (pcl::SAC_RANSAC);
seg.setMaxIterations (1000);
seg.setDistanceThreshold (0.01);
seg.setInputCloud (cloud);
seg.segment (*inliers, *coefficients);
```

Algorithms – 3D Feature Descriptors

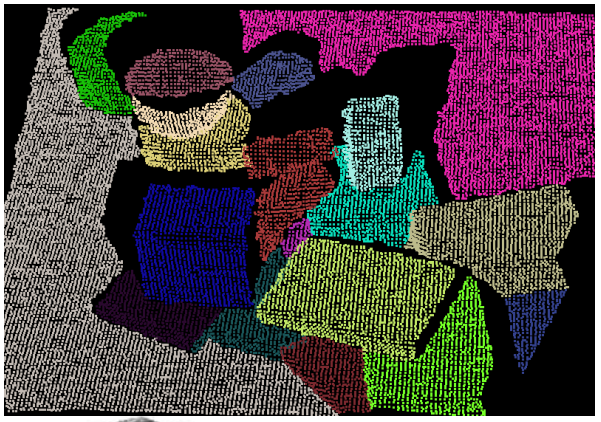
- PCL includes a number of features designed for point cloud data.
- You can use these descriptors learn classifiers for object types or detectors for particular objects.
- Experiment with using 2D and 3D features!



PCL Questions?

What would you like to do with PCL?

The website has lots of great tutorials if you'd like to learn more.



Thanks

Hang around if you'd like to discuss your project!