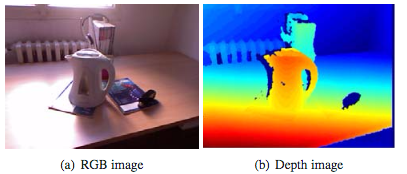
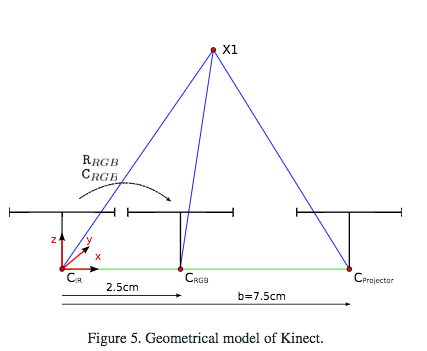
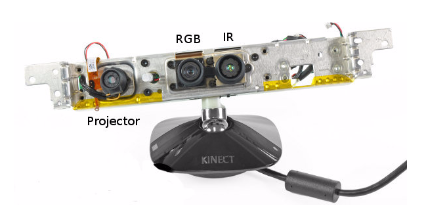
# RGB+D Cameras: Image, 3D point clouds and camera model

"Kinect" (depth) cameras provide 2 type of images: RGB and DEPTH



Q: What coordinate is the "depth" ? How is it obtained ?



Tasks:

1. See RGB and Depth images
   1. display and interpret images
   2. display R,G,B in separate figures
2. Display depth
   1. depth as image
   2. depth as a surface
   3. point cloud, mesh
   4. depth image vs. point cloud

Dataset:

Load depth image from the lab (link)

read image from file (link)

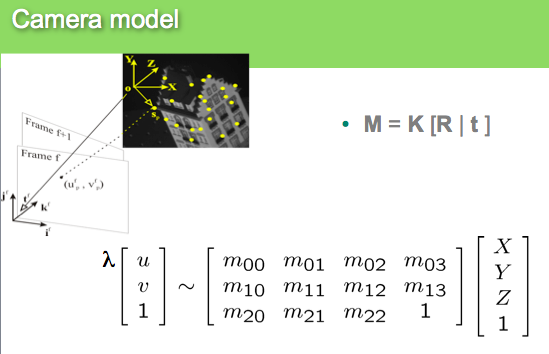
load point cloud (link)

http://printart.isr.ist.utl.pt/piv/lab/pivlab1/

Remember what you learned in class about the pin-hole camera model:

Camera model -

* How to generate an image from 3D points?
* How to compute 3D positions from images/depth

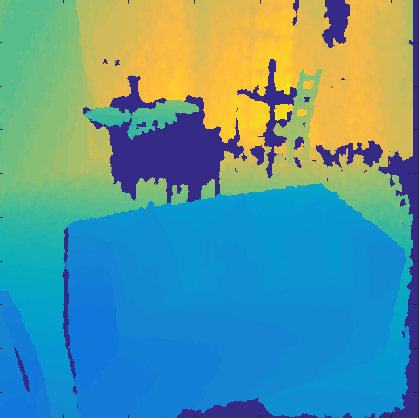


TODO (yourself):

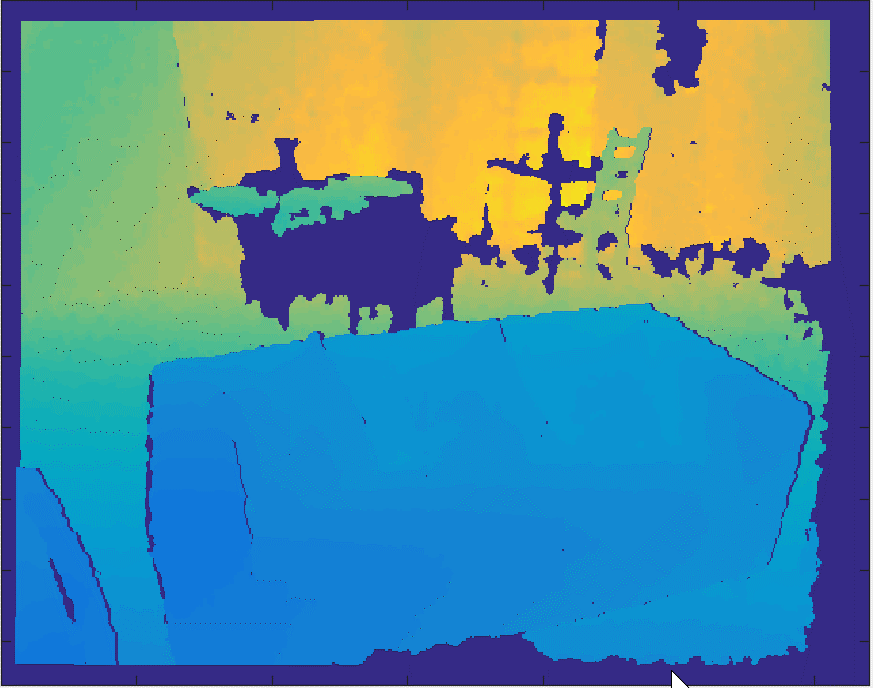
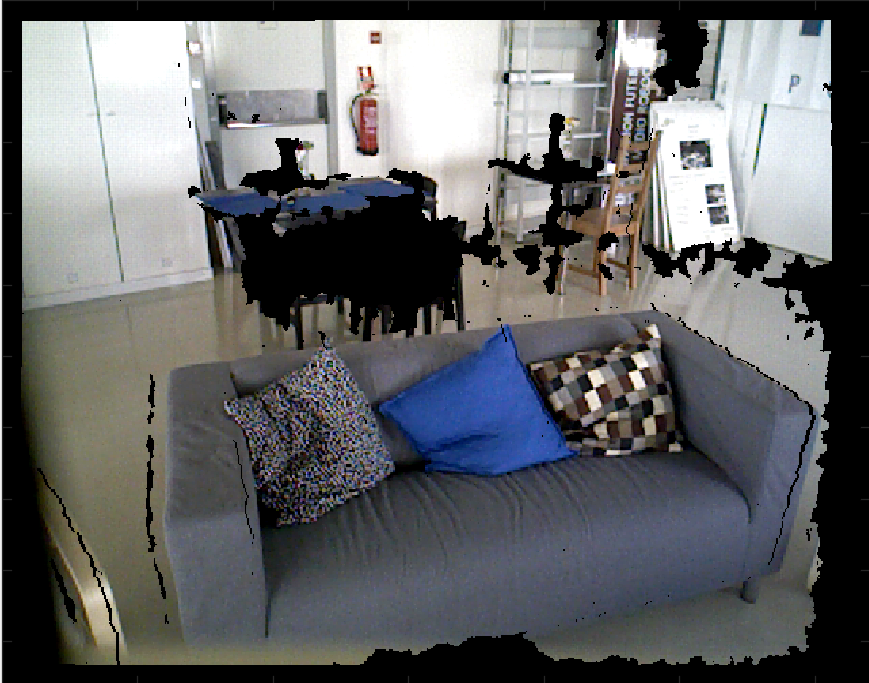
1. given the depth image compute XYZ
2. Rotate and Translate point cloud and represent XYZ in the rgb camera frame
3. Create a “virtual” depth image in the RGB camera frame
4. Associate xyz/Depth values to RGB pixel

In other words …

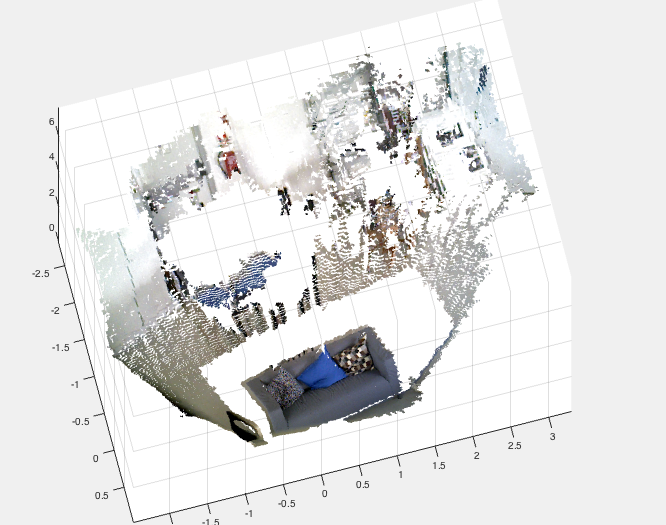
From these two images

and the intrinsic and extrinsic camera parameters you should be able to "superimpose" the depth over the rgb:

Note that now each pixel of this image is in line with the depth! So, you can compute XYZ for each pixel and create this point cloud



Check this code.

%Depth Images Stored in variable depth\_array in mat files

im=imread('rgb\_image\_10.png');

figure(1);

imagesc(im);

load depth\_10.mat

figure(2);

imagesc(depth\_array);

%%Generate XYZ from depth

I=(1:480)'\*ones(1,640);

J=ones(480,1)\*(1:640);

d=double(depth\_array(:)')/1000; %convert to meters

xyz=inv(Depth\_cam.K)\*[d.\*J(:)';d.\*I(:)';d];

%%%Let's look at it;

pc=pointCloud(xyz');

figure(3);

showPointCloud(pc);

%Rigid transformation - RGB camera

xyz\_rgb= R\_d\_to\_rgb\*xyz+repmat(T\_d\_to\_rgb,[1 size(xyz,2)]);

%Project in the RGB image plane (homog. coordinates)

omega=RGB\_cam.K\*xyz\_rgb;

%Euclidean coordinates

u=round(omega(1.,:)./omega(3,:));

v=round(omega(2,:)./omega(3,:));

%just clean up wrong coordinates

v(v>480)=1;v(v<1)=1;u(u>640)=1;u(u<1)=1;

%convert to linear index

ind=sub2ind(size(depth\_array),v,u);

%Generate "virtual" depth image in RGB

depth\_new=zeros(size(depth\_array));

depth\_new(ind)=xyz\_rgb(3,:);

figure(4); imagesc(depth\_new);

%Get fully "colored" point cloud

imnew=reshape(im,[640\* 480 3]);

imnew(depth\_new(:)==0,:)=0;

pc2=pointCloud(xyz\_rgb','color',imnew);

figure(5);

showPointCloud(pc2);

figure(6);

imagesc(reshape(imnew,[480 640 3]))

**Can not understand why? Please go to office hours ! This should be absolutely clear.**

# Represent everything in the Depth Camera frame…another way of doing the same!

**Auxiliary code to implement these steps**

load calib\_asus % Load Calibration Data for ASUS camera (similar to kinect)

load depth\_10 % Load depth image

imagesc(depth\_array)% Look at it...just a depth image

%Compute XYZ from depth image (u,v) and depth z(u,v)- CHECK FILE

xyz=get\_xyzasus(depth\_array(:),[480 640],1:640\*480,Depth\_cam.K,1,0);

figure

%Display point cloud

p=pointCloud(xyz)

showPointCloud(p)

%Read RGB image

im=imread('rgb\_image\_10.png');

figure;

imagesc(im);

%Compute "virtual image" aligned with depth

rgbd=get\_rgbd(xyz,im,R\_d\_to\_rgb,T\_d\_to\_rgb,RGB\_cam.K);

figure

imagesc([im; rgbd])

cl=reshape(rgbd,480\*640,3);

p=pointCloud(xyz,'Color',cl);

figure

% Point cloud with colour per pixel

showPointCloud(p)