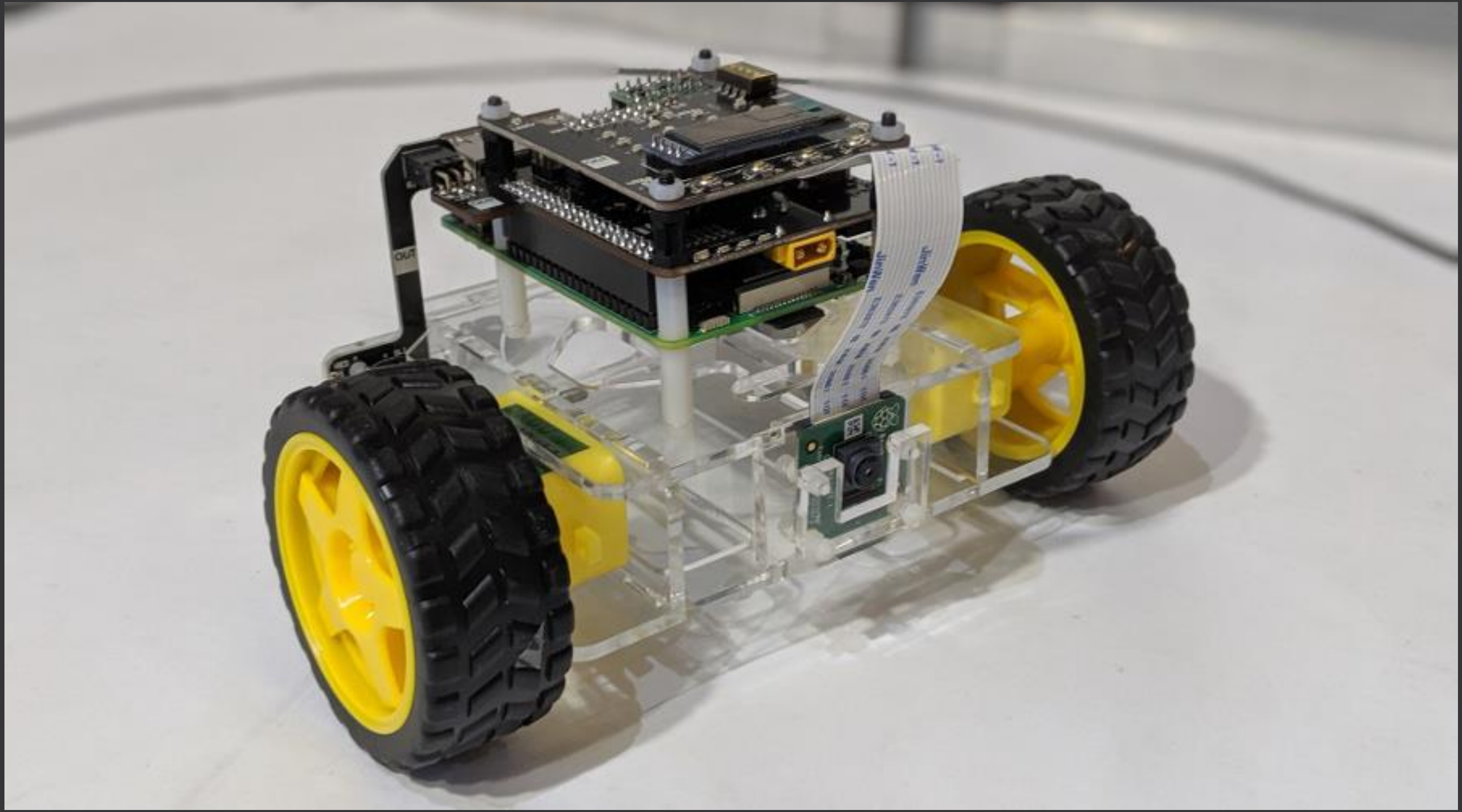
ECE4078 Intelligent robotics



Lab 3-2: Kal Backman & Shujie Zhou

ECE4078 Intelligent robotics

Week	Objectives	Milestones
~~2: M1-1~~	~~Introduction and setup~~	
~~3: M2-1~~	~~Calibration, ARUCO markers~~	~~M1 due~~
~~4: M2-2~~	~~SLAM~~	
~~5: M2-3~~	~~SLAM~~	
~~6: M3-1~~	~~Object recognition & localisation~~	~~M2 due~~
* 7: M3-2	Object recognition & localisation	
8: M4-1	Navigation & Planning	M3 due
9: M4-2	Navigation & Planning	
10: M5-1	Integration	M4 due
11: Final	Improvement	M5 due
12: Final	Final demo & competition	Final demo

Milestone 3

Milestone is focused on object recognition and localisation

Your task is to classify and estimate the position of various fruits given a single image

Task 1: Gather training data (Week 6)

Use the provided scripts to generate data for the default segmentation network
or create your own data generator for YOLO

Task 2: Training the network (Week 6)

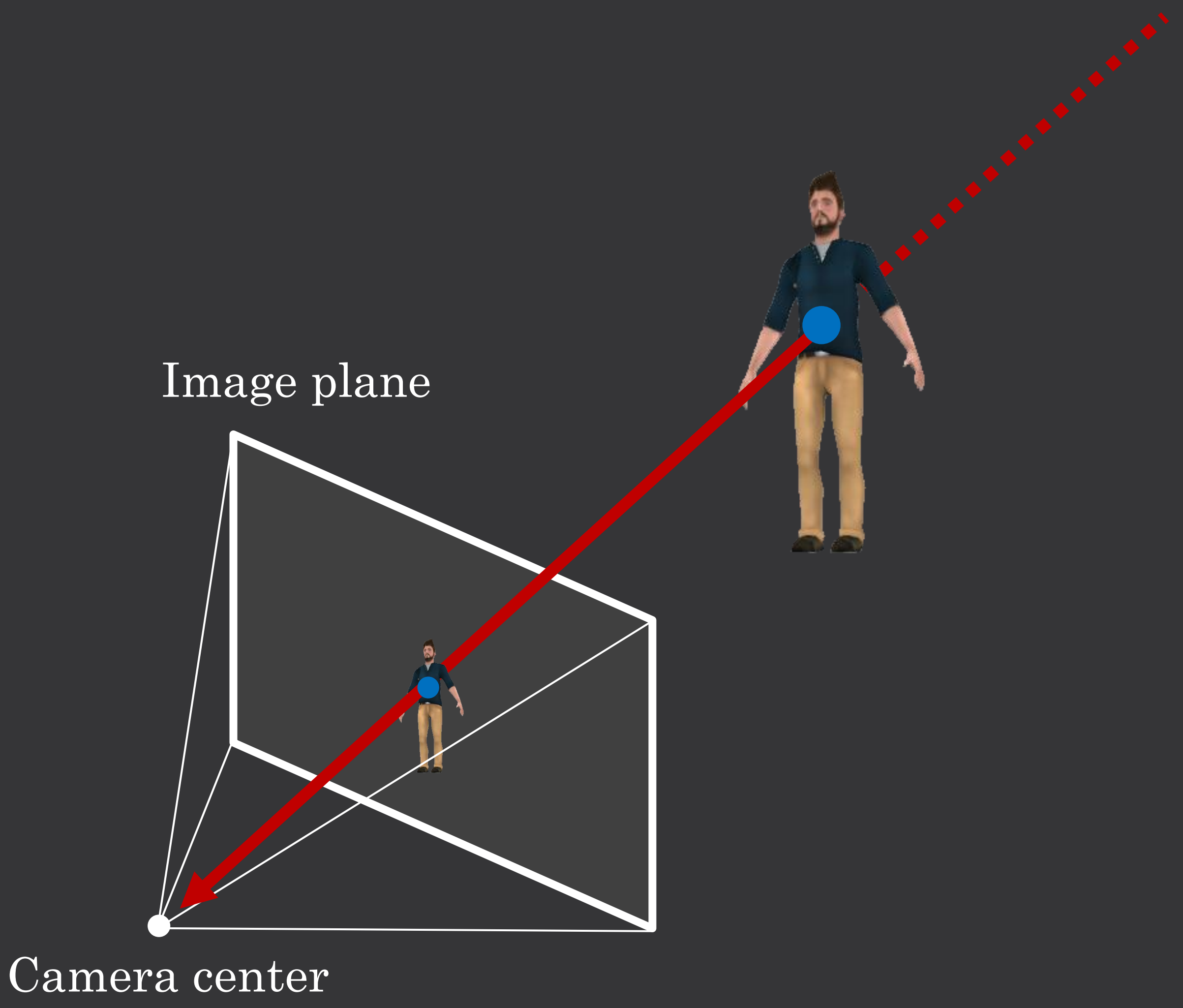
Train the default network using the “main.py” script in the “network/scripts/” folder
or your own YOLO network from your chosen GitHub repository

Task 3: Estimating object poses (Week 7)

Given your trained network’s output, determine the physical position of the observed object

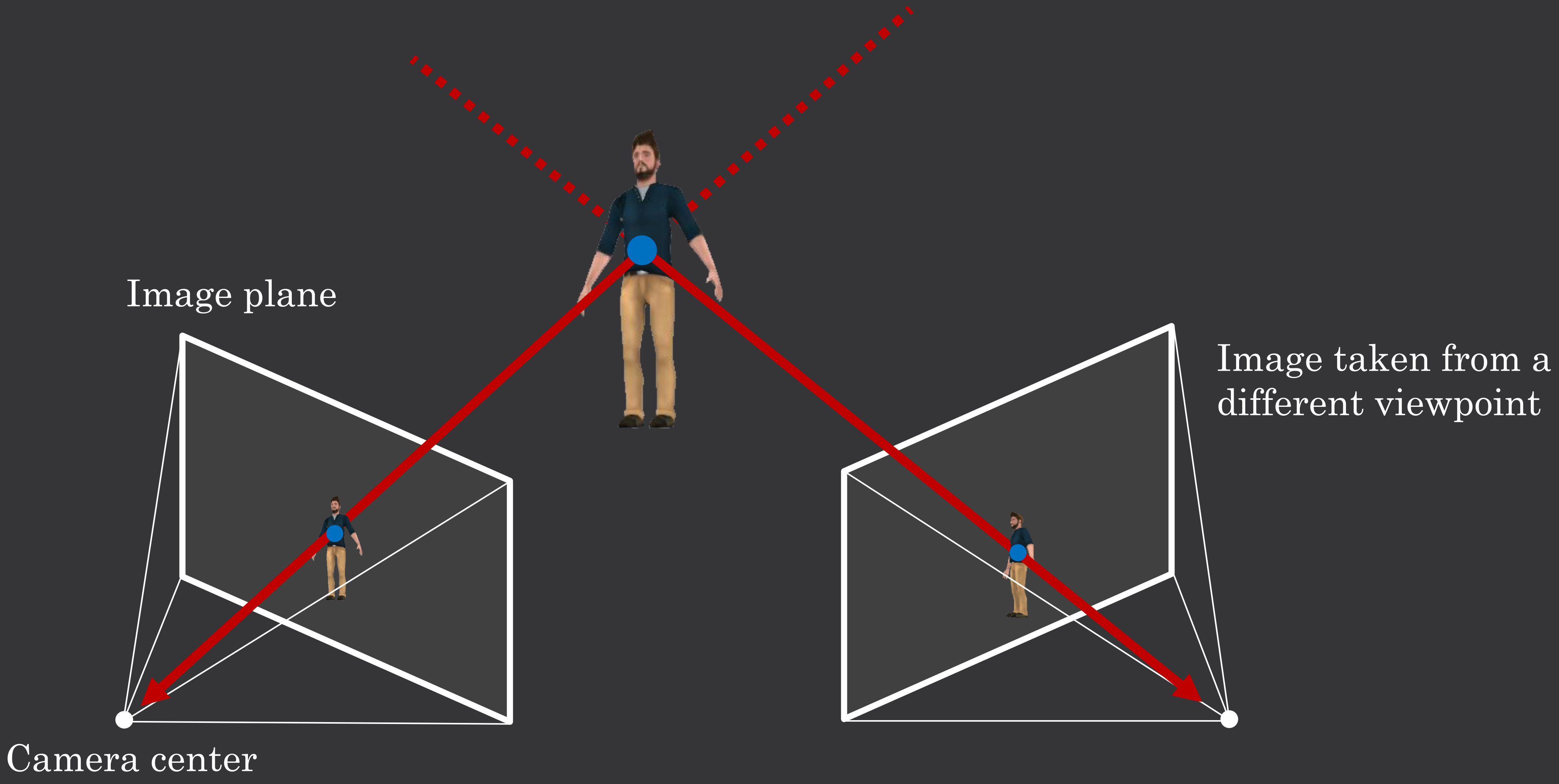
Estimating object pose

Given any standard RGB image, can we calculate the 3D position of any pixel within an image?



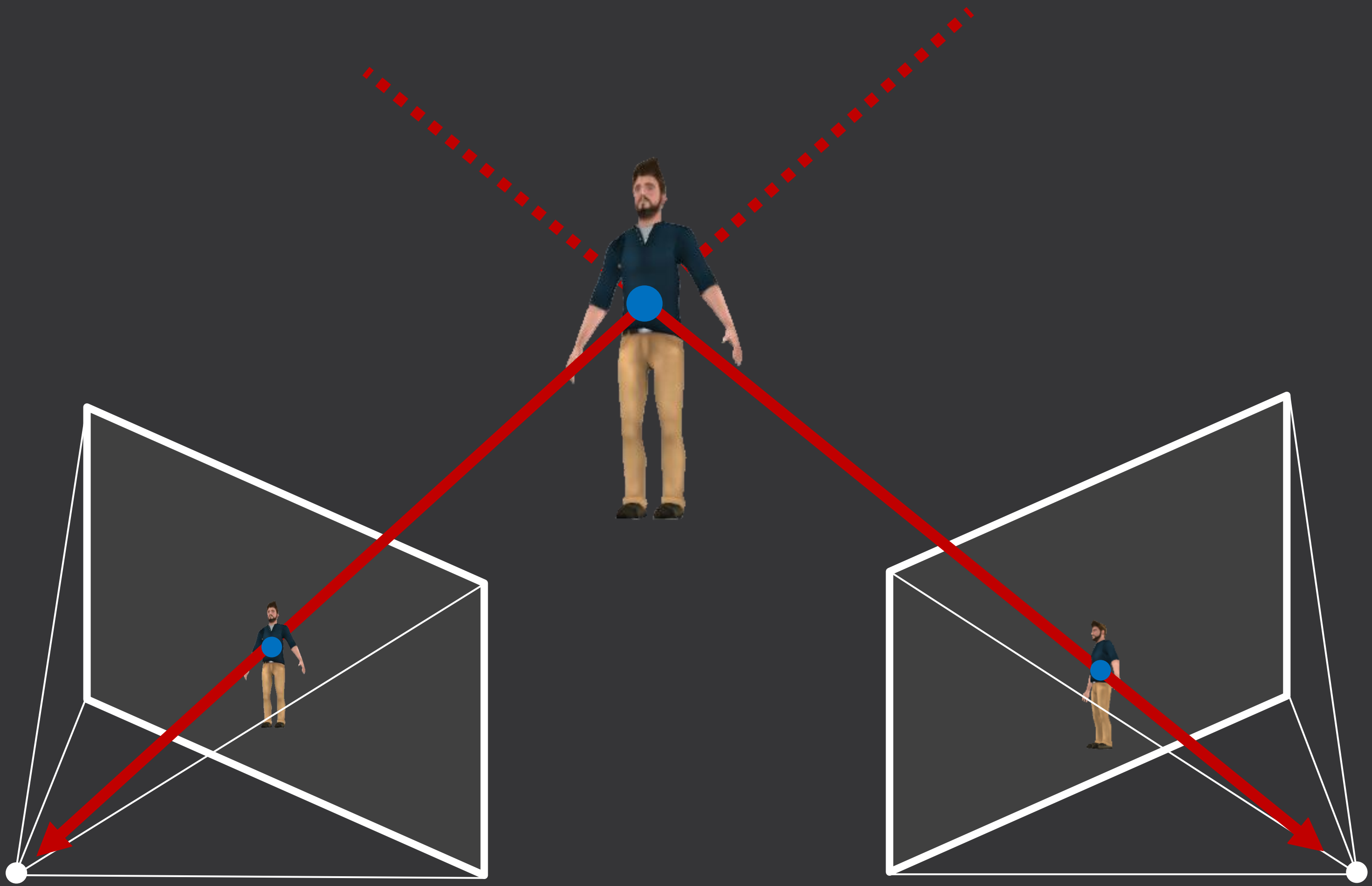
Estimating object pose

What if we had 2 RGB images?
We can triangulate the projected light rays to figure out the 3D position of the corresponding point



Estimating object pose

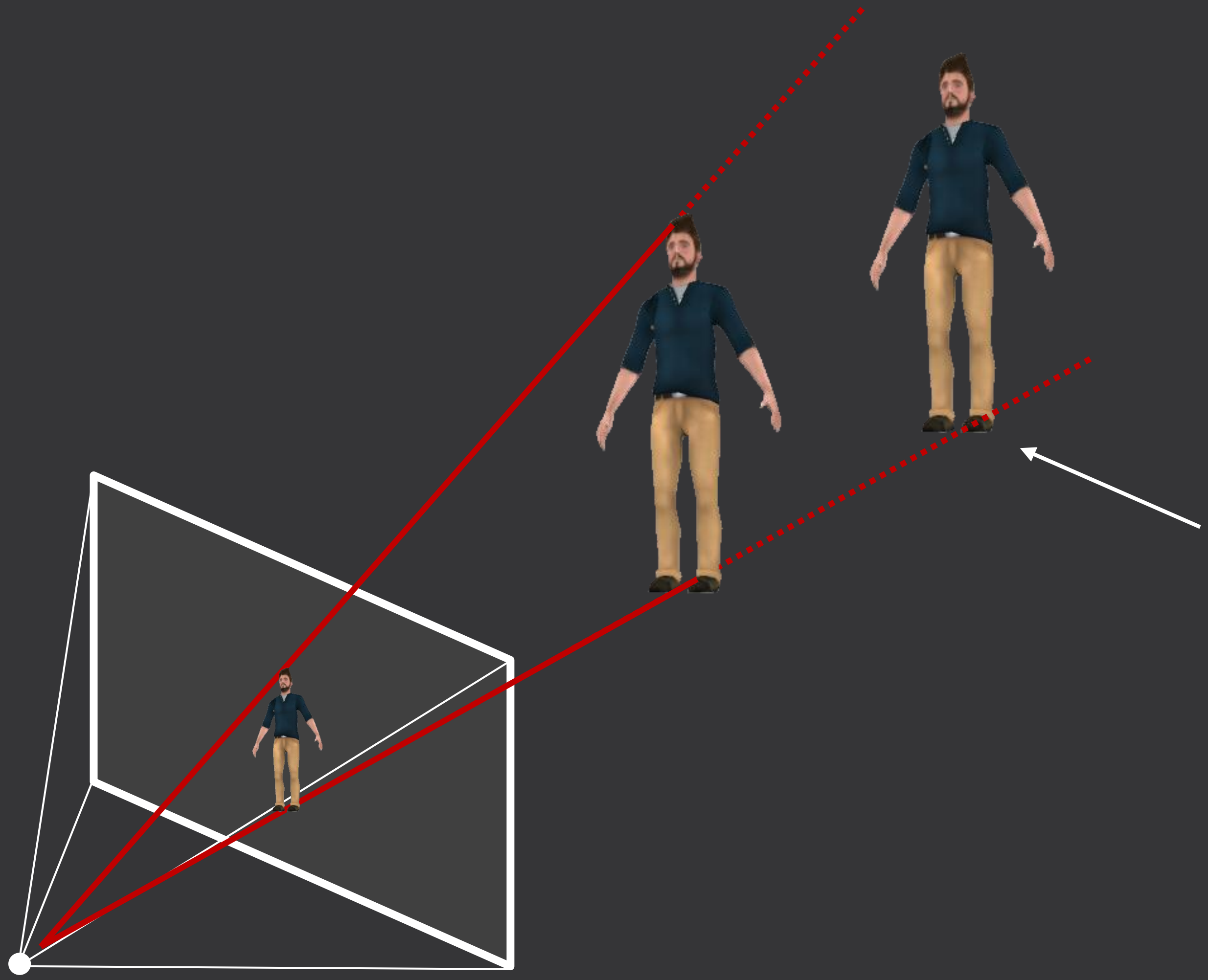
This is difficult to do however as it is reliant on us being confident on the relative poses of both cameras and choosing a pixel in both images that correspond to the exact same spot on the human


First image

Second image

Estimating object pose

What if we project two lines from the camera?

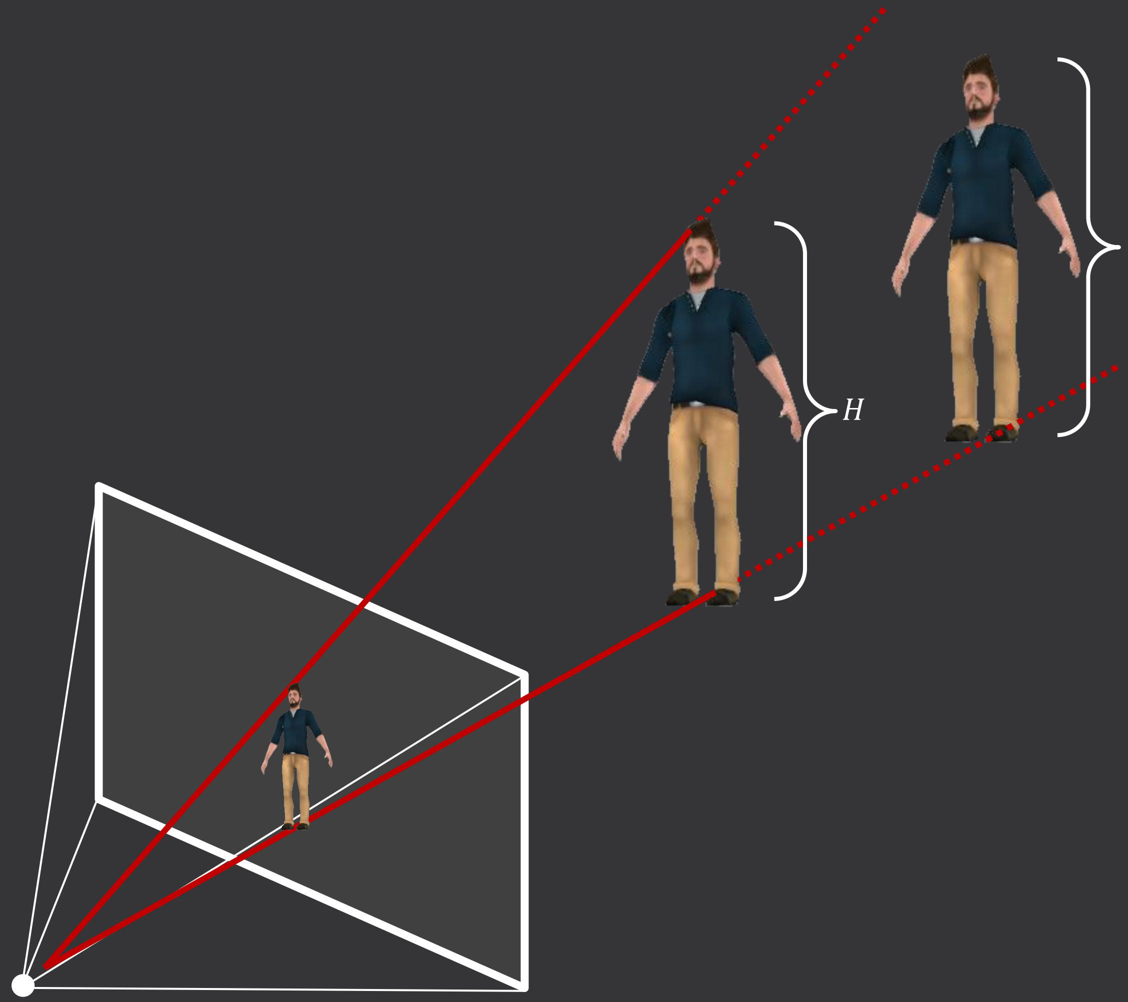


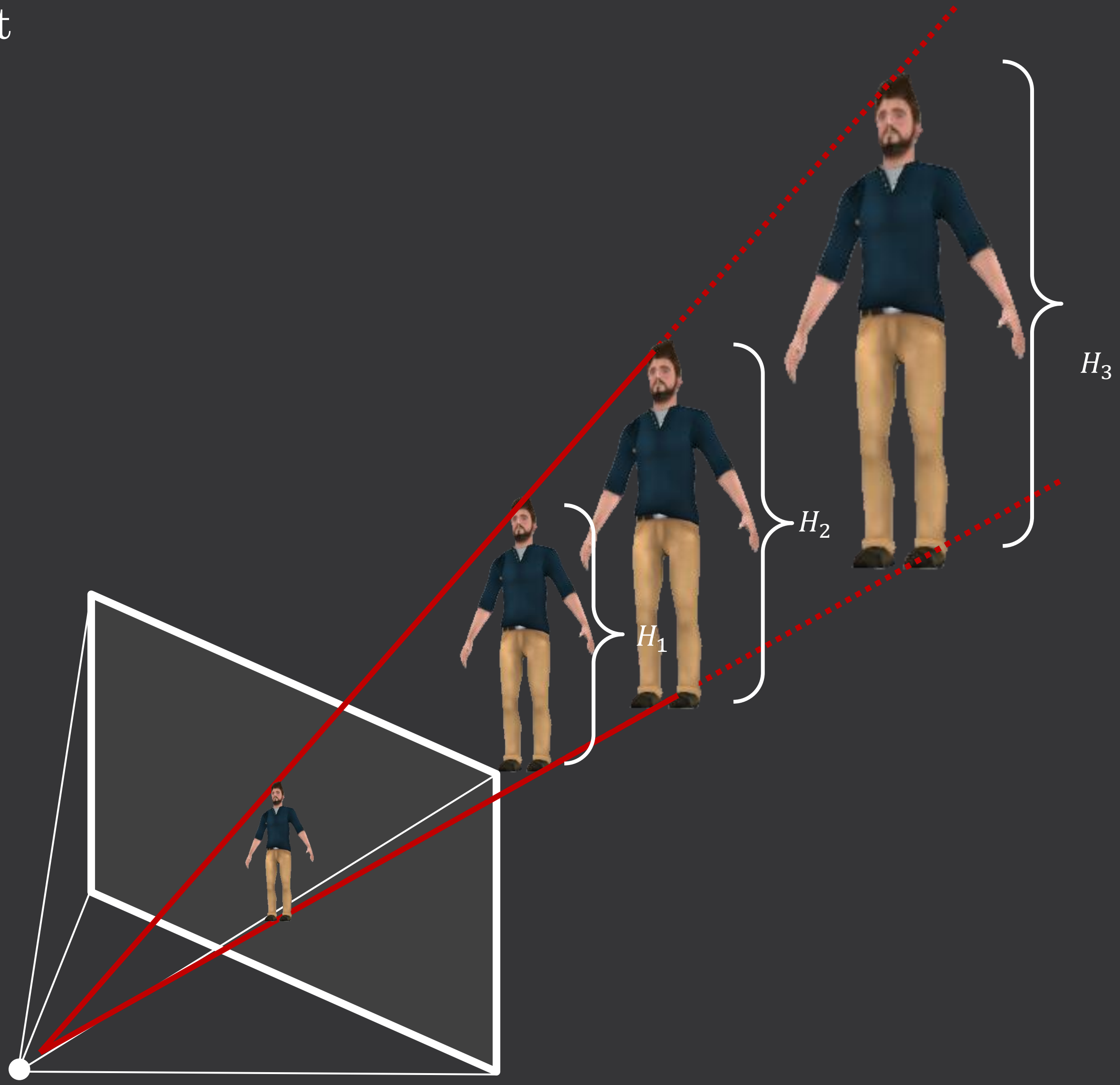
If the human were to walk backwards, away from the camera, they should appear smaller in the image

Estimating object pose

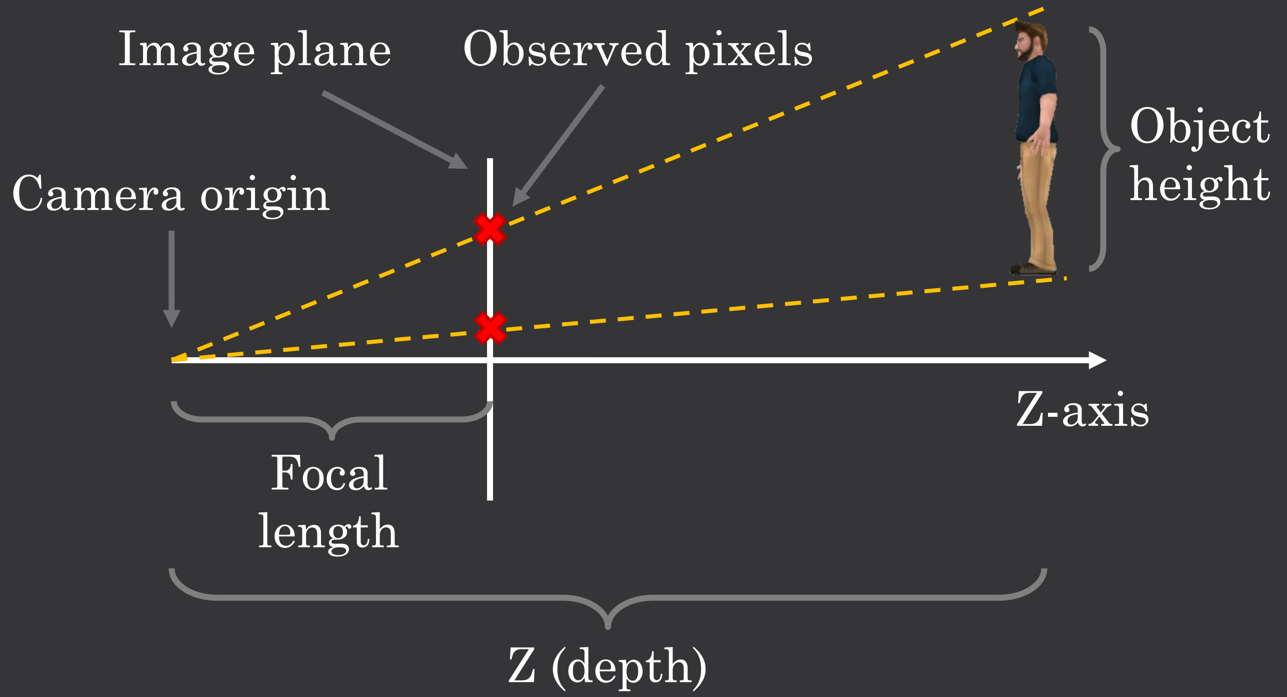
This will ONLY work if:

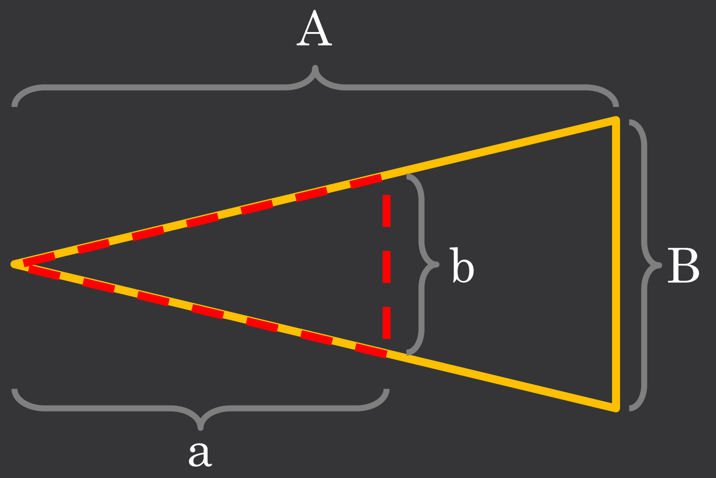
- We know the height of the object we are looking at
- The object's height does not change



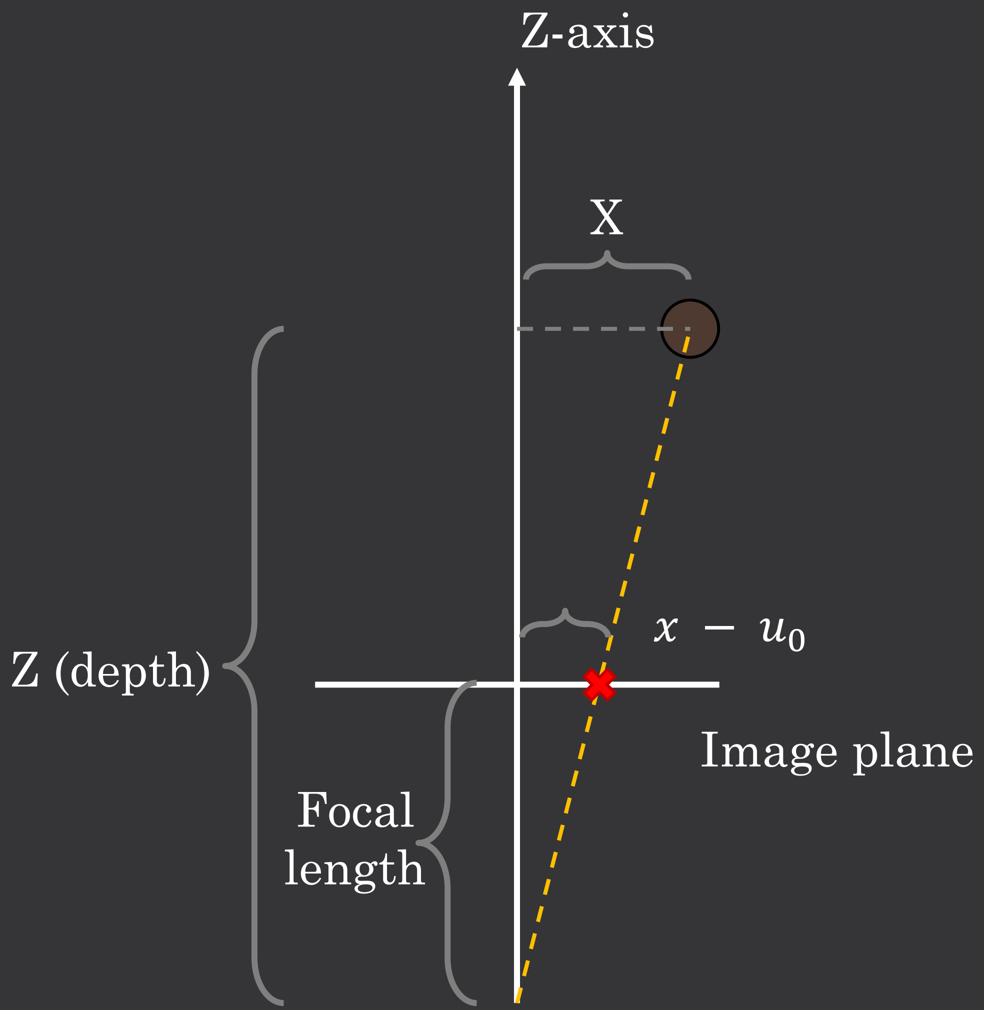


Estimating object pose

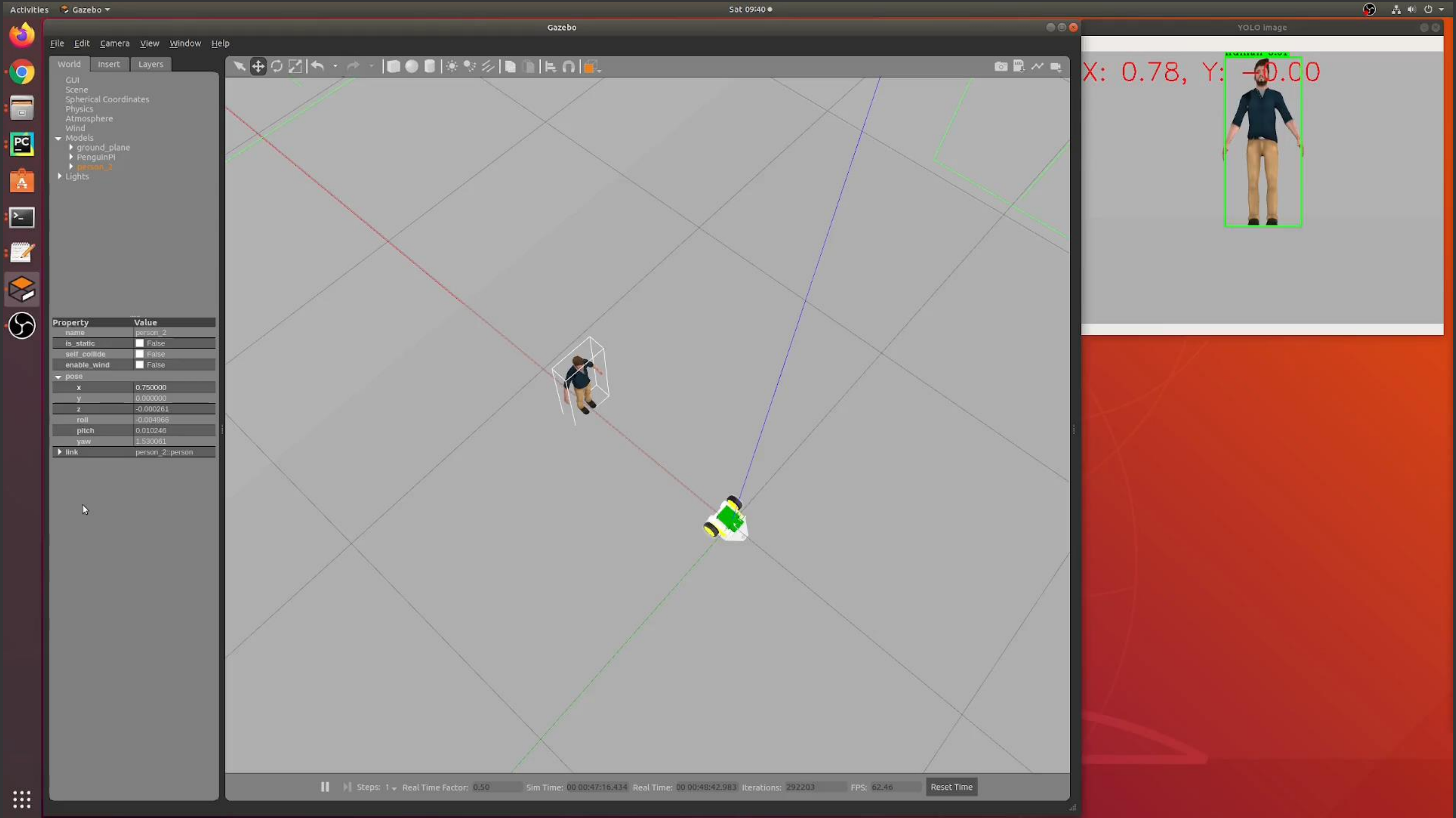




$$\frac{a}{b} = \frac{A}{B}$$



Estimating object pose



Property	Value
name	person_2
is_static	False
self_collide	False
enable_wind	False
pose	
x	0.750000
y	0.000000
z	-0.000261
roll	-0.004966
pitch	0.010246
yaw	1.530061
link	person_2::person

Steps: 1 Real Time Factor: 0.50 Sim Time: 00 00:47:16.434 Real Time: 00 00:48:42.983 Iterations: 292203 FPS: 62.46 Reset Time

Estimating object pose

Things you will need to calculate fruit pose:

- Camera's intrinsic parameters, which you would have calculated in M2

$$\begin{bmatrix} F_x & 0 & u_0 \\ 0 & F_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Height in pixels and x-pixel coordinate of the object
  - YOLO provides this information to you with bounding boxes
  - The default network requires you to create bounding boxes from your segmentation map
    Take a look at OpenCV's "findContours" or "SimpleBlobDetector" to do this

- Physical height of the objects
  The physical heights of the simulated fruits can be found in "TargetPoseEst.py" or you can measure them in gazebo. For the physical fruit you can measure them with a ruler.
  You can also use the physical width of the fruit as an additional estimate (make sure the fruit has a constant width from all viewing angles)

Make sure if your network is trained on a different image size compared to the PenguinPi camera resolution of 640x480, then you scale your pixel coordinates

Eg: Network trained on 320x240 images will need all pixel values scaled by 2x

Milestone 3

We will mark you based on the output of CV_eval.py script which compares the true global positions of the fruit with that of your global fruit estimates

You will have to modify TargetPoseEst.py to create your global fruit estimates
TargetPoseEst.py will receive a list of images along with their associated poses they were taken at as input
The input images will be taken as if someone was mapping out the arena like in M2
There are only 2 of each fruit in the marking map, the same fruit may appear in multiple images therefore you will have to perform some sort of filter / clustering algorithm to ONLY output 2 of each fruit

You will be marked on both a simulated dataset and a physical dataset

You can modify TargetPoseEst.py script as much as you like (even completely rewrite it, which I recommend) as long as your script adheres to the same input and output format as the skeleton code

In your submission make sure you submit:

- TargetPoseEst.py - To generate your global fruit map (you can upload 2 versions, sim & physical if you like)
- Network related scripts - scripts used for inference (NOT training scripts)
- Model weights - If weights are too large for moodle you can submit a link to a google drive file
- Read me file - If you are not using the skeleton code please give us detailed step-by-step instructions on how to setup and run your solution. (Tell us how you installed YOLO)

Milestone 3

Before you leave today, it is recommended you take some photos of the physical fruit from known distances so you can validate your pose estimation for the physical robot at home

Remember to submit your M3 before next lab session

Milestone 2 marking

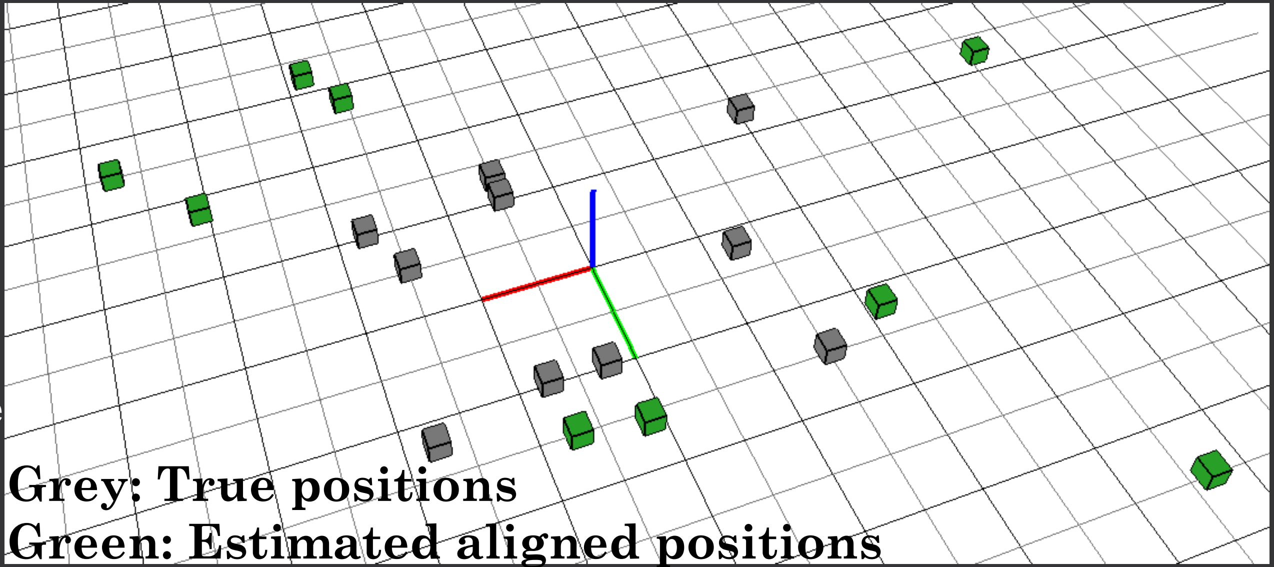
We've looked over some of the results for M2, grades will be released when all groups have finished being marked off and looked over.

Some things we've noticed:

- Some physical maps have the correct shape but incorrect scale

  You should check that your physical camera intrinsic parameters are correct

  You should place an ArUco marker in front of the robot and see if it can correctly estimate the distance of the marker like how we validated the update() function in M2_3 slides



- Think about your use of time and driving strategy, some groups spent the entire 5 minutes on a single run while others completed 3 runs within the same time.
  When the covariance in the markers get small, their estimate wont update that much