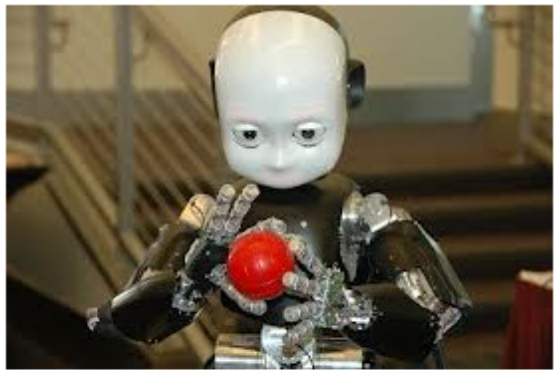
Social Robotics using NAO and Kinect

Arash Akbarinia, Corina Barbalata, Raquel Gil, Eduardo Tusa

December 1, 2012

Robots and Daily Life




Introduction

Social Robot

An autonomous robot that interacts and communicates with humans or other autonomous physical agents by following social behaviours and rules attached to its role

Introduction

Problem statement [Gaschler et al., 2012]

Head pose is an important cue that is used by customers and bartenders in all steps of the ordering sequence.

Introduction

Objective

The objective is to estimate the social scene in a bar/pub/cafe style interaction with multiple users by using Kinect controller.

- To detect multiple humans when they ask for attention.
- To perform face tracking and gaze estimation

Intuitive Idea

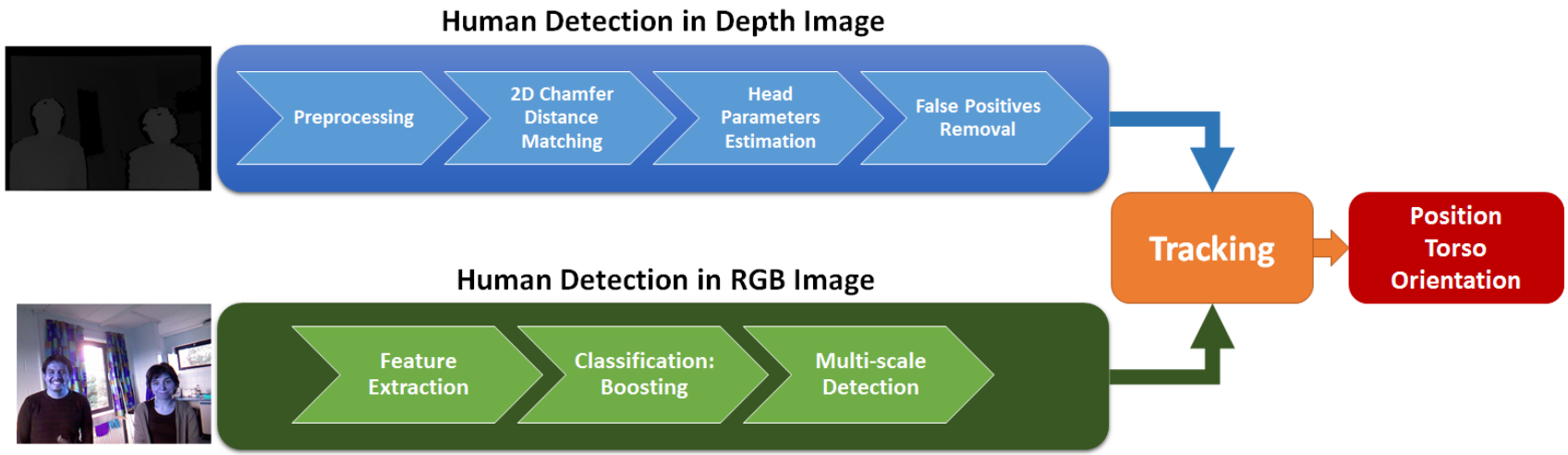
Corina Raquel Eduardo Arash

Intuitive Idea

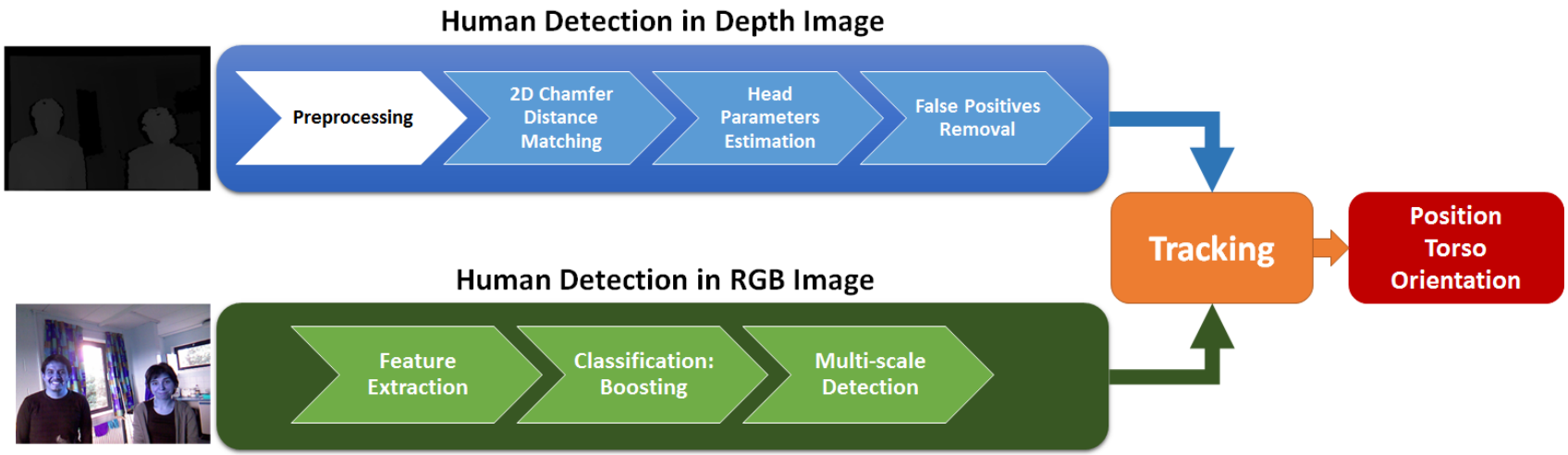
Intuitive Idea

Intuitive Idea

Scheme


Human Detection in Depth Image
Preprocessing
2D Chamfer Distance Matching
Head Parameters Estimation
False Positives Removal
Human Detection in RGB Image
Feature Extraction
Classification: Boosting
Multi-scale Detection
Tracking
Position Torso Orientation

Preprocessing



Preprocessing

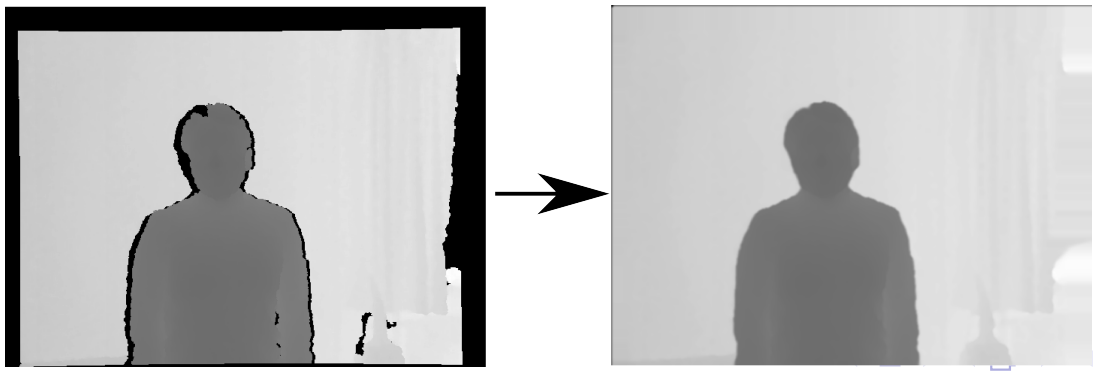
Kinect problems

- Low resolution,
- Limited range of about 4 meters,
- Noise, shadows, etc.

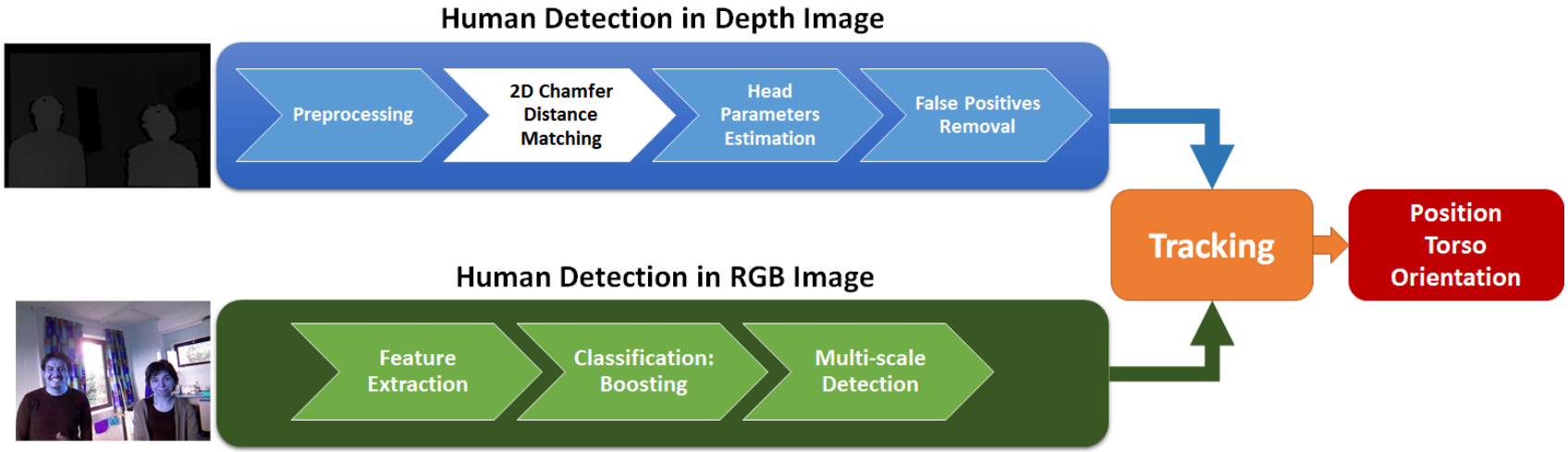
Solution

Filling the holes by inpainting

- Propagate the grey levels,
- Propagate gradients,
- Propagate details.



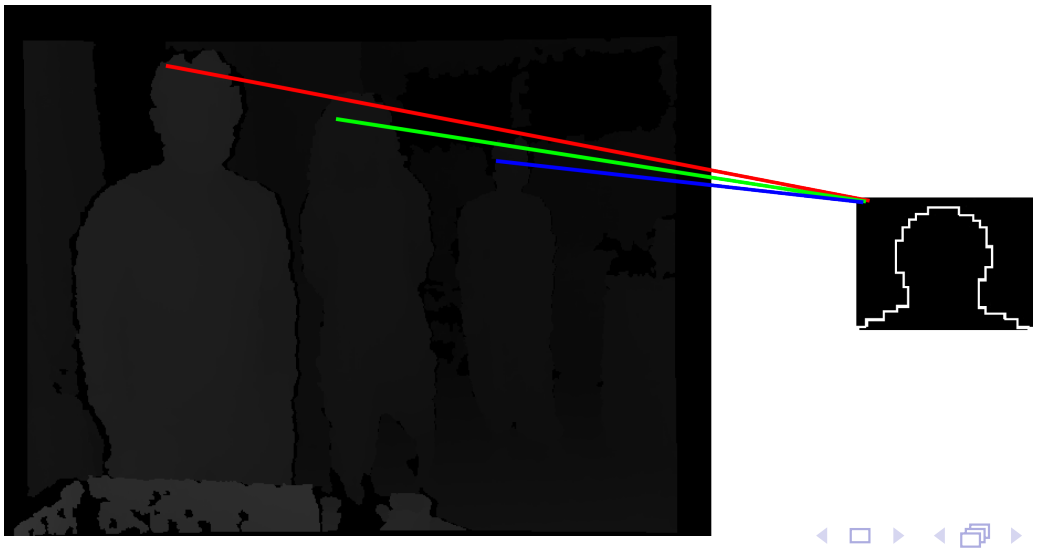
Template matching



Template Matching Algorithm

2D Chamfer Distance Matching

- Used to match the head template to disparity image.
- Advantages: computationally fast, scale invariant.



Template Matching Algorithm

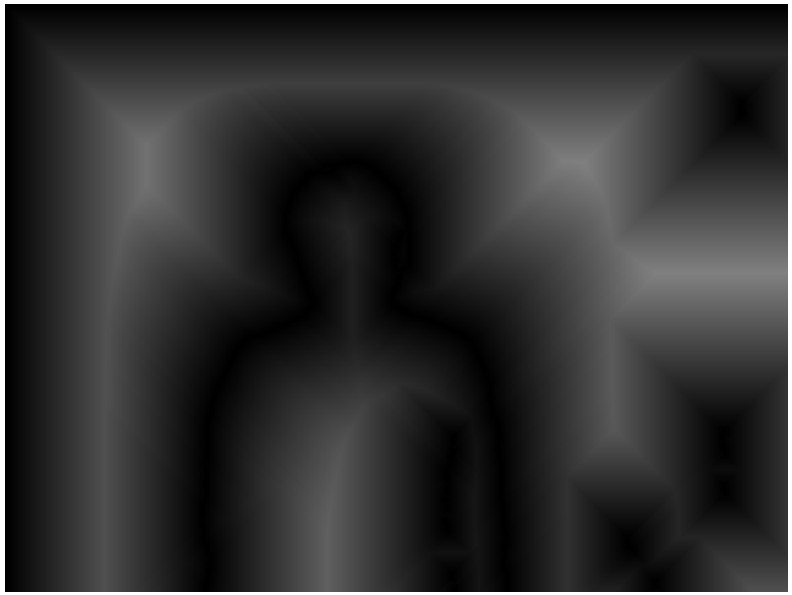
STEPS

1. Edge image using Canny detector.
2. Compute distances to the closest edge.
3. Compute different scales for distances.
4. Match the template with all the scales.
5. A threshold is used to get the best matches.

Template Matching Algorithm

STEPS

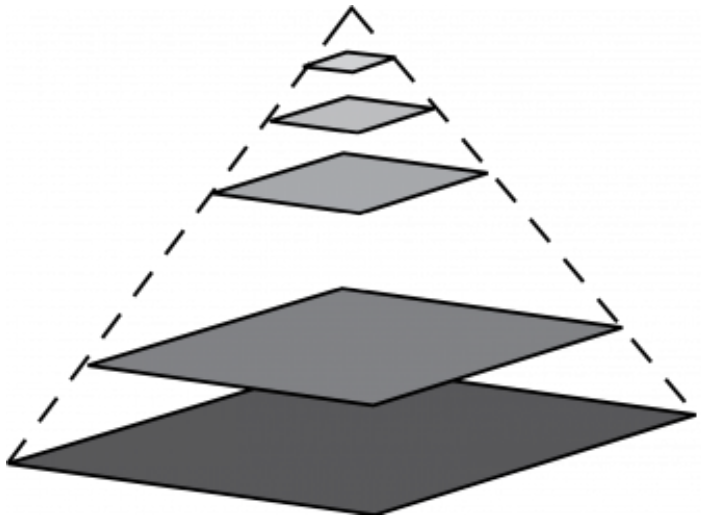
1. Edge image using Canny detector.
2. Compute distances to the closest edge.
3. Compute different scales for distances.
4. Match the template with all the scales.
5. A threshold is used to get the best matches.



Template Matching Algorithm

STEPS

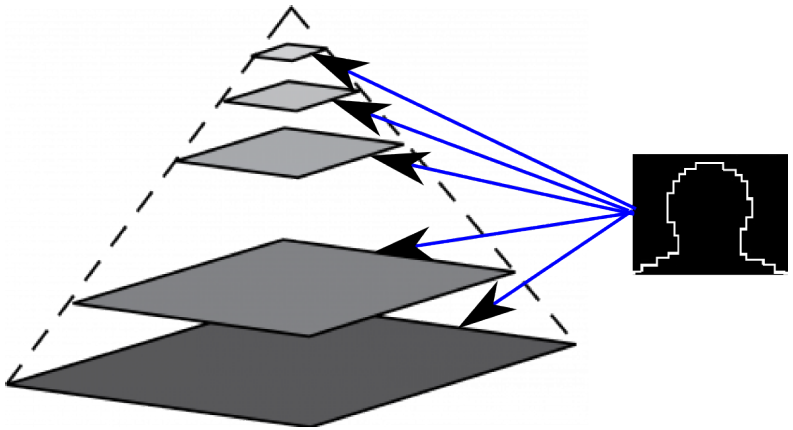
1. Edge image using Canny detector.
2. Compute distances to the closest edge.
3. Compute different scales for distances.
4. Match the template with all the scales.
5. A threshold is used to get the best matches.



Template Matching Algorithm

STEPS

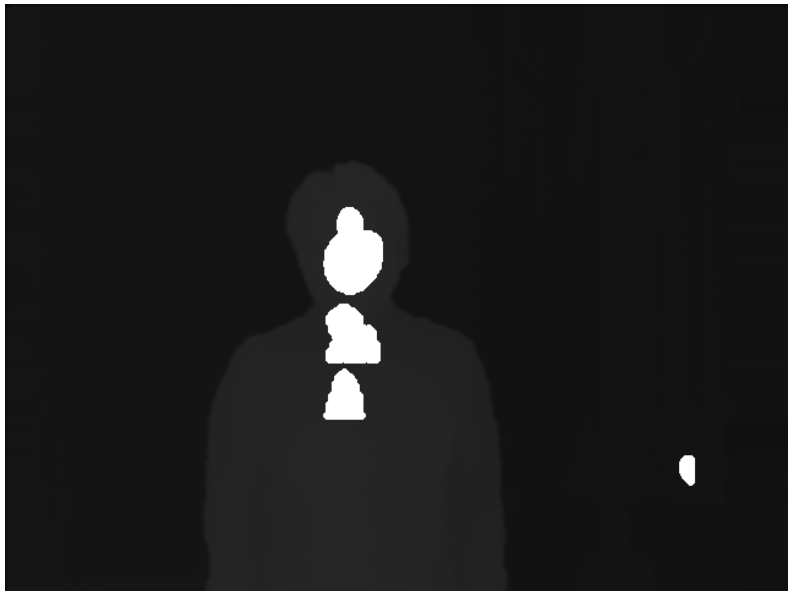
1. Edge image using Canny detector.
2. Compute distances to the closest edge.
3. Compute different scales for distances.
4. Match the template with all the scales.
5. A threshold is used to get the best matches.



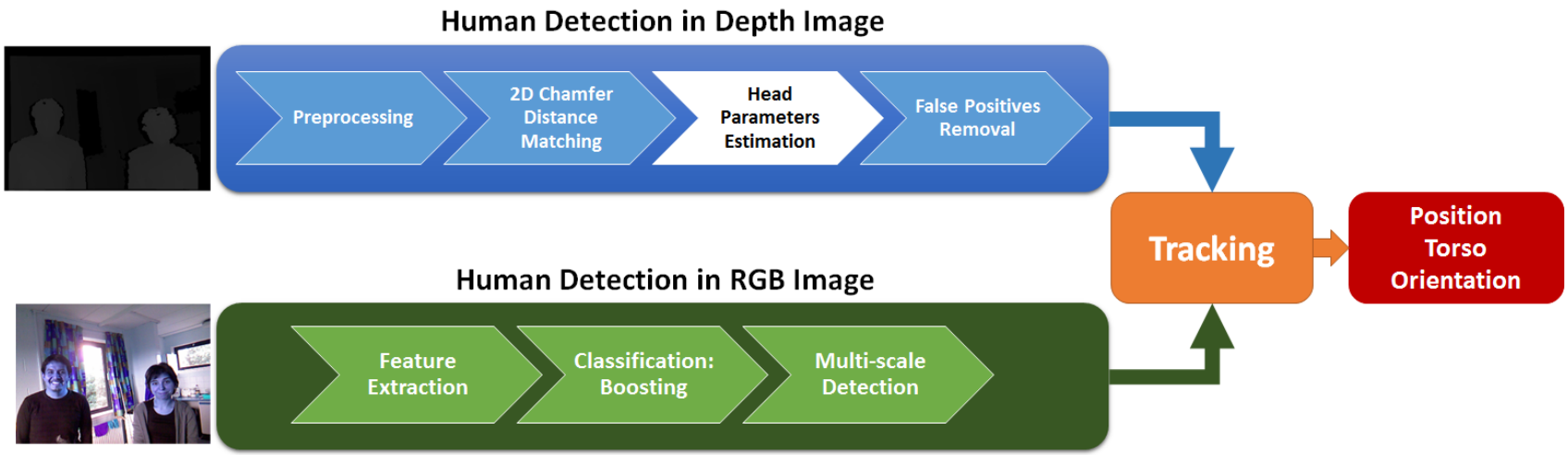
Template Matching Algorithm

STEPS

1. Edge image using Canny detector.
2. Compute distances to the closest edge.
3. Compute different scales for distances.
4. Match the template with all the scales.
5. A threshold is used to get the best matches.



Head Parameters Estimation



Head Parameters Estimation

Polynomial Regression

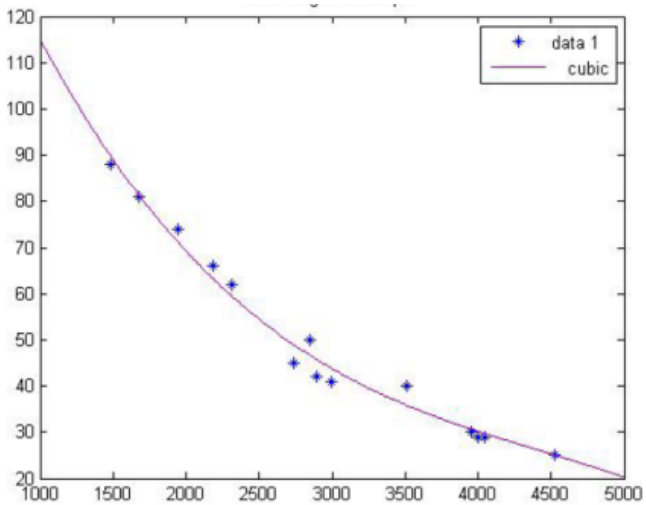
$$h = p_3 x^3 + p_2 x^2 + p_1 x + p_0$$

Millimeters to Pixels

$$R = \frac{1.33 \cdot h}{2} \quad \text{in } mm$$

$$R = \lfloor R/1.3 \rfloor \quad \text{in } pixels$$

Depth vs Height of Head



Head Parameters Estimation

Polynomial Regression

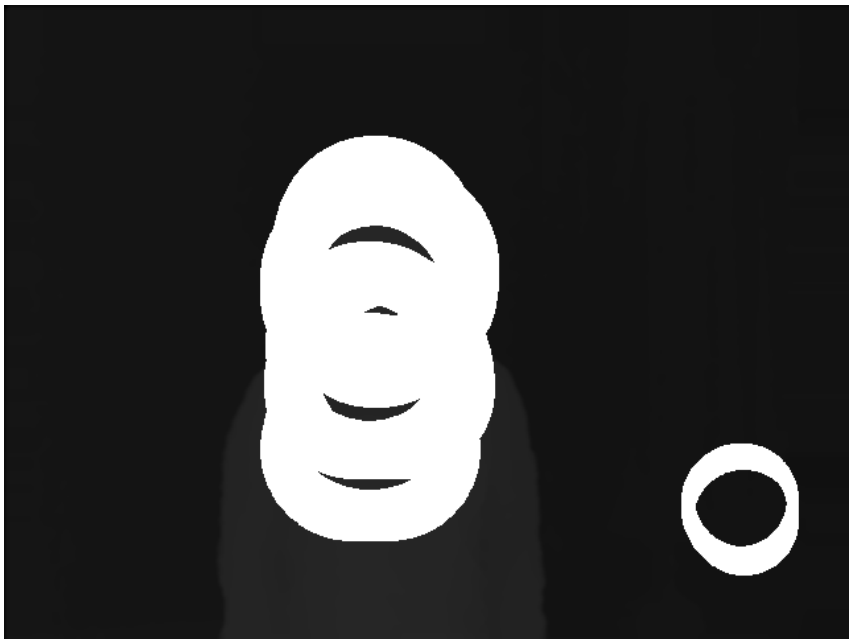
$$h = p_3 x^3 + p_2 x^2 + p_1 x + p_0$$

Millimeters to Pixels

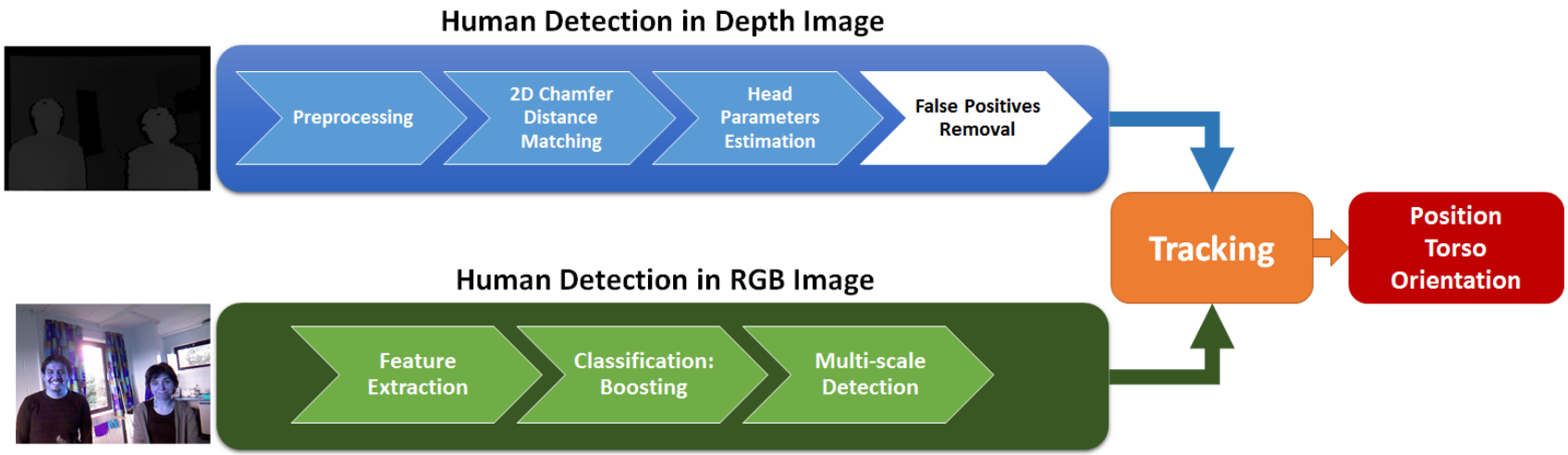
$$R = \frac{1.33 \cdot h}{2} \text{ in } mm$$

$$R = \lfloor R/1.3 \rfloor \text{ in } pixels$$

Potential Heads



False Positives Removal

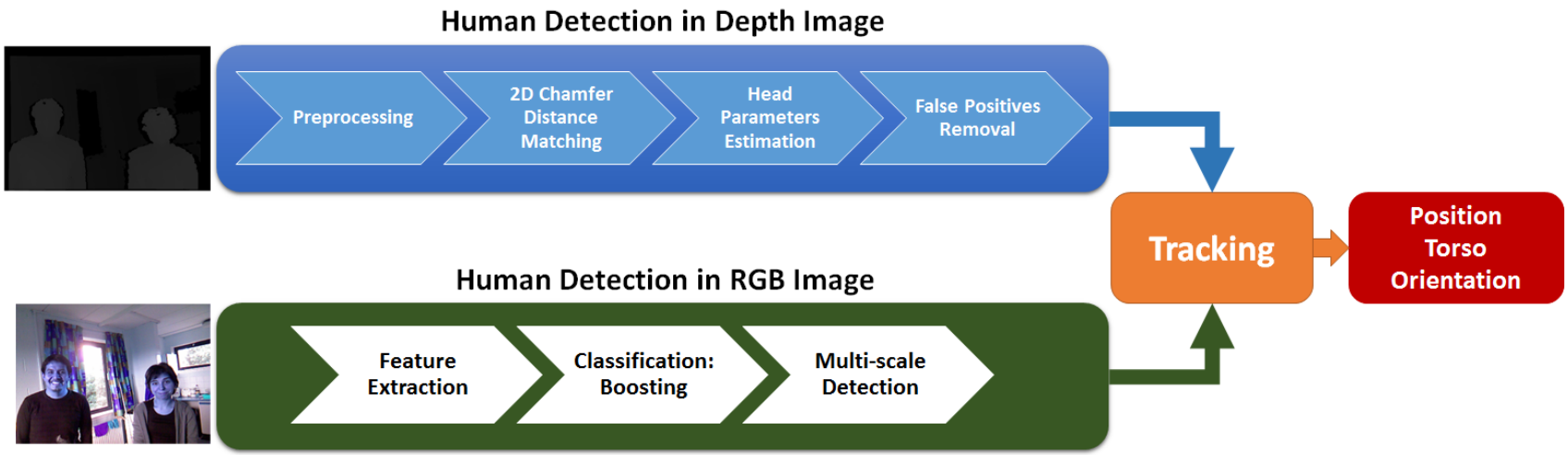


False Positives Removal

Algorithm

1. Approximate the contours by polygons to discard those ones that are not circles or semicircles.
2. Merge the heads, based on the Euclidean Distance, given a threshold. The one with better similarity in template matching is chosen.
3. Matching the ROI in the disparity image with a $3D$ template.

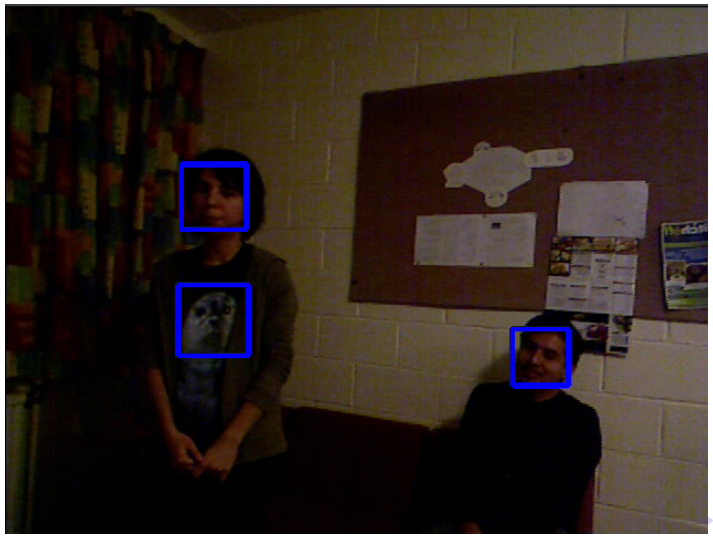
Human detection in RGB image



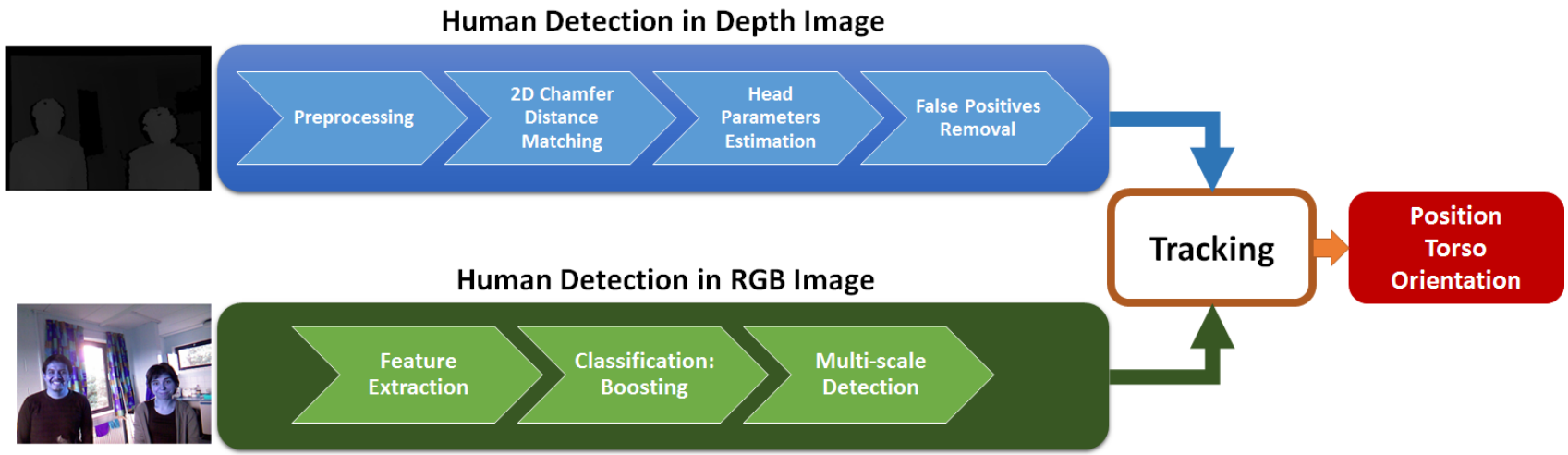
Multicascade

Phases of the algorithm

- Feature extraction.
- Classification using boosting.
- Multi-scale detection algorithm.



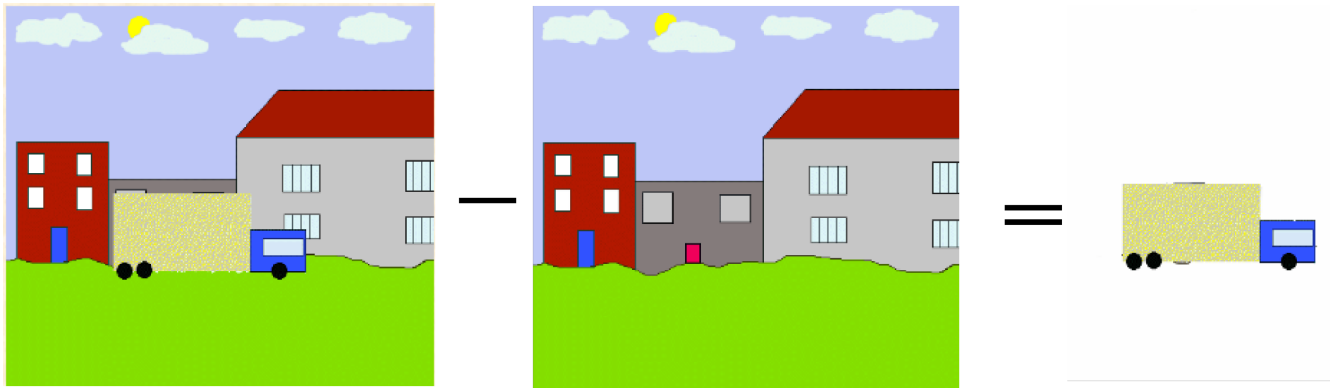
Tracking


Human Detection in Depth Image
Preprocessing
2D Chamfer Distance Matching
Head Parameters Estimation
False Positives Removal
Human Detection in RGB Image
Feature Extraction
Classification: Boosting
Multi-scale Detection
Tracking
Position
Torso
Orientation

Background subtraction

Motivation

- Simple difference of two images shows moving objects.
- Easy to implement.
- Very fast.

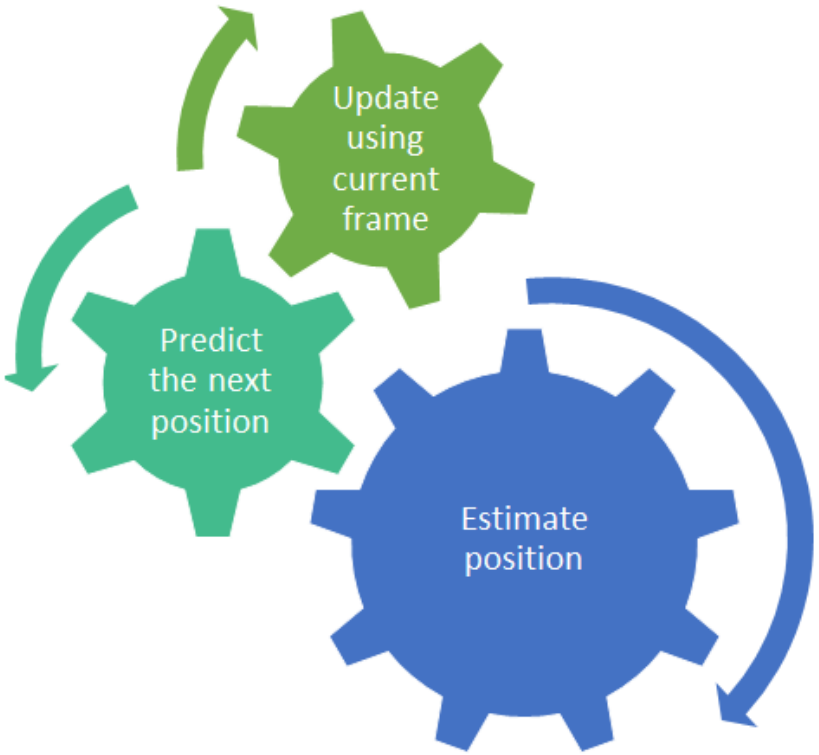


Background subtraction

Why we implemented and did not use it?

- Accuracy of frame differencing depends on object speed and frame rate.
- Background for a bar, is changing too fast.

Particle Filter Framework



Particle Filter Tracking

1. Specify the person to be tracked.
2. Obtain the color-depth model for target person.

Particle Filter Tracking

1. Specify the person to be tracked.
2. Obtain the color-depth model for target person.

Particle Filter Tracking

3. Initialize weighted sample set.

$$s = \{x, y, \dot{x}, \dot{y}, s\}$$

4. Compute color-depth distribution at sample locations.
5. Compare distributions using Bhattacharryya coefficient.

Bhattacharryya coefficient and distance

$$\rho = \sum_{u=1}^{m} \sqrt{p(u) \cdot q(u)}$$

$$dist = \sqrt{1 - \rho}$$

Particle Filter Tracking

3. Initialize weighted sample set.

$$s = \{x, y, \dot{x}, \dot{y}, s\}$$

4. Compute color-depth distribution at sample locations.
5. Compare distributions using Bhattacharryya coefficient.

Bhattacharryya coefficient and distance

$$\rho = \sum_{u=1}^{m} \sqrt{p(u) \cdot q(u)}$$

$$dist = \sqrt{1 - \rho}$$

Particle Filter Tracking

3. Initialize weighted sample set.

$$s = \{x, y, \dot{x}, \dot{y}, s\}$$

4. Compute color-depth distribution at sample locations.
5. Compare distributions using Bhattacharryya coefficient.

Bhattacharryya coefficient and distance

$$\rho = \sum_{u=1}^{m} \sqrt{p(u) \cdot q(u)}$$

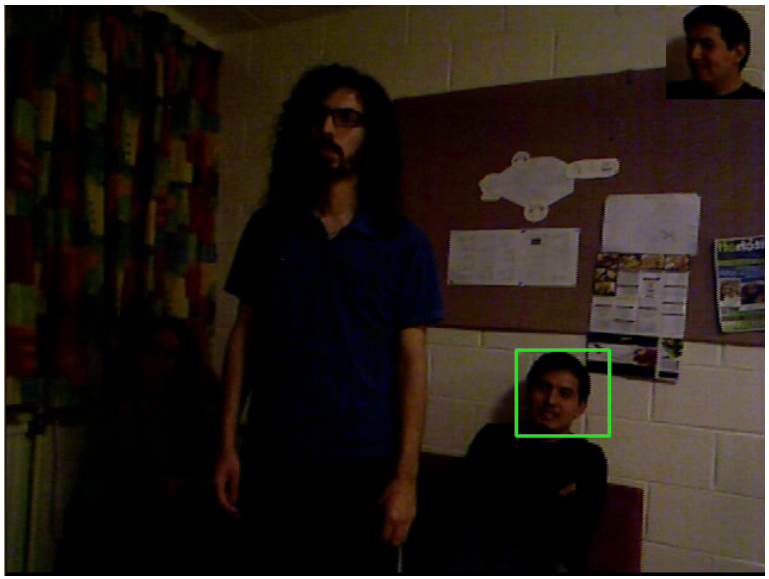
$$dist = \sqrt{1 - \rho}$$

Particle Filter Tracking

6. Use Bhattacharryya distance to assign weight to sample.
7. Position estimation using weighted samples.
8. Samples selected based on re-sampling.

Particle Filter Tracking

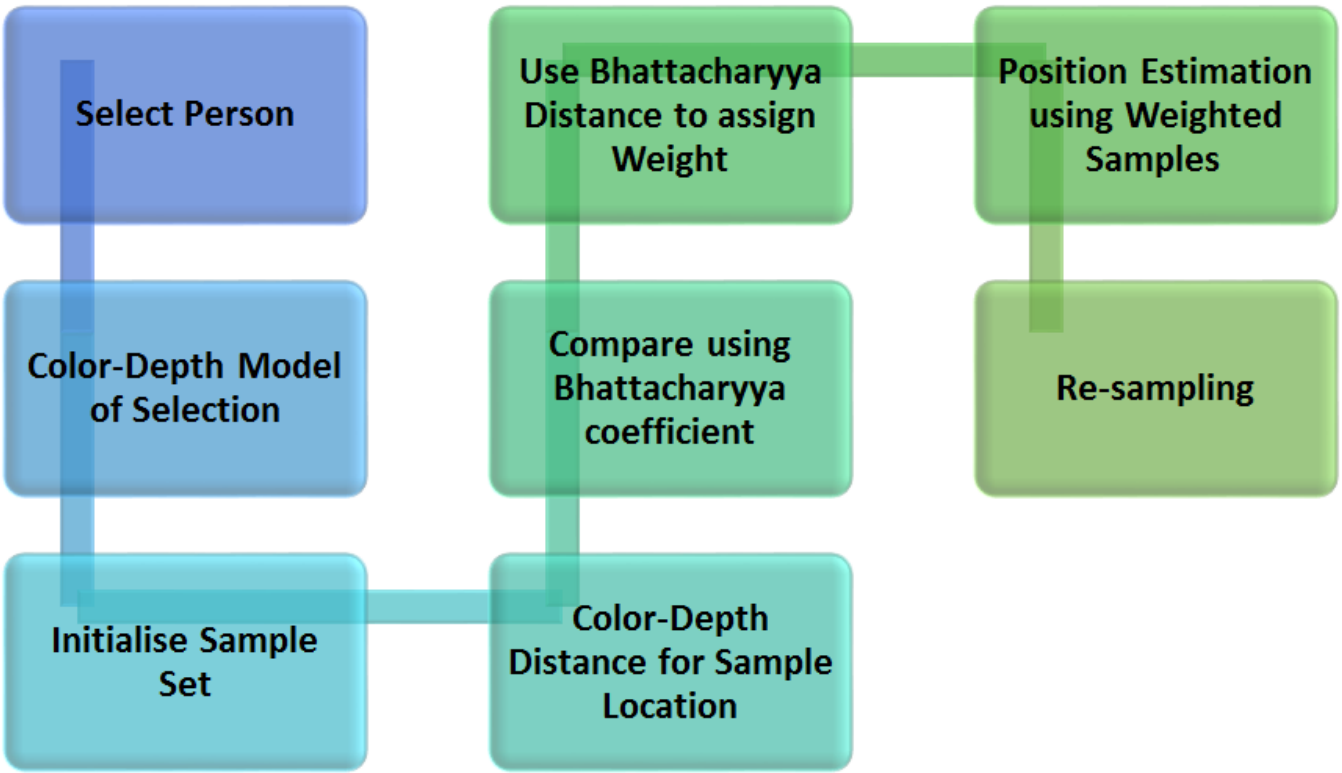
6. Use Bhattacharryya distance to assign weight to sample.
7. Position estimation using weighted samples.
8. Samples selected based on re-sampling.



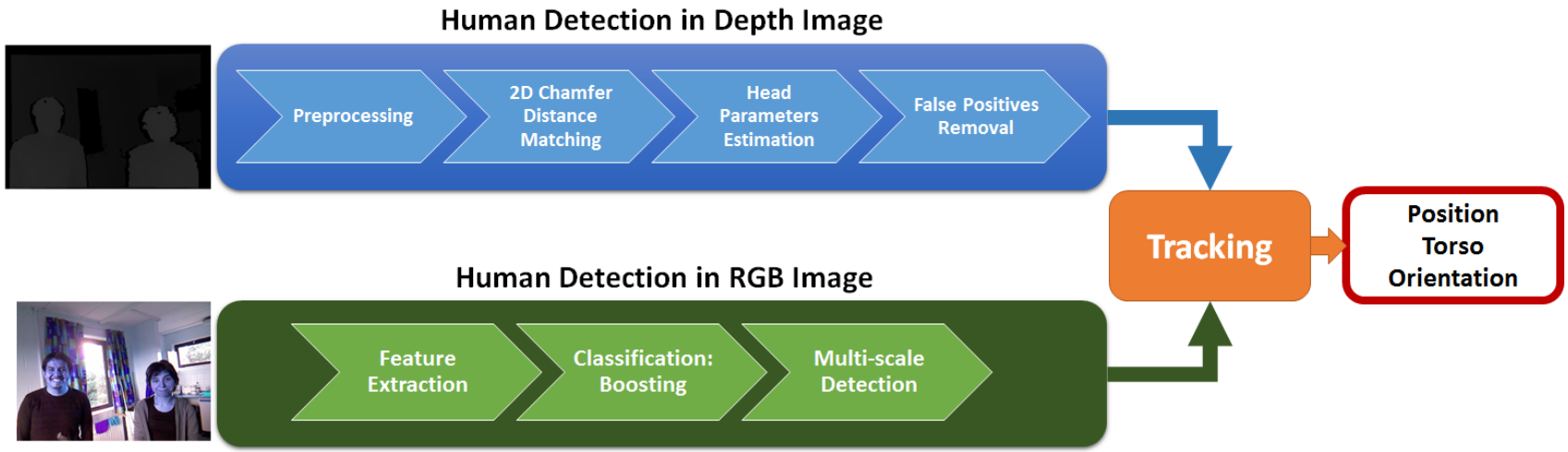
Particle Filter Tracking

6. Use Bhattacharryya distance to assign weight to sample.
7. Position estimation using weighted samples.
8. Samples selected based on re-sampling.

Summary: Particle Filter Tracking



Torso Orientation



Torso Orientation Algorithm

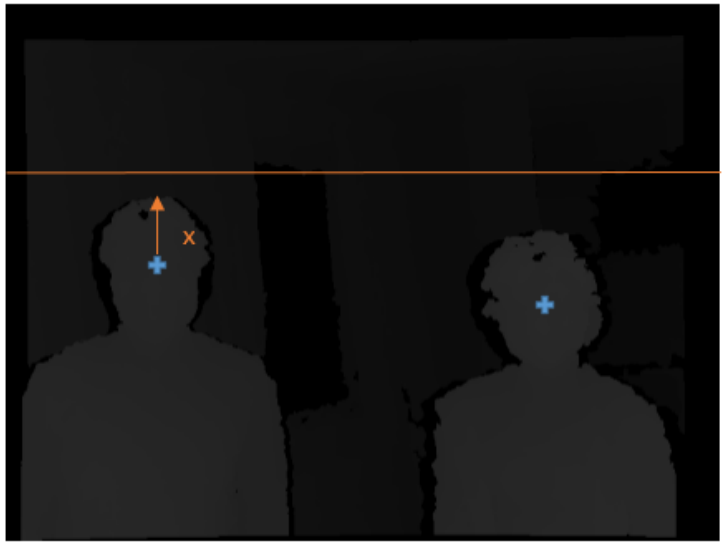
Inputs

2D coordinates of the centroid (x,y).

Torso Orientation Algorithm

Step 1

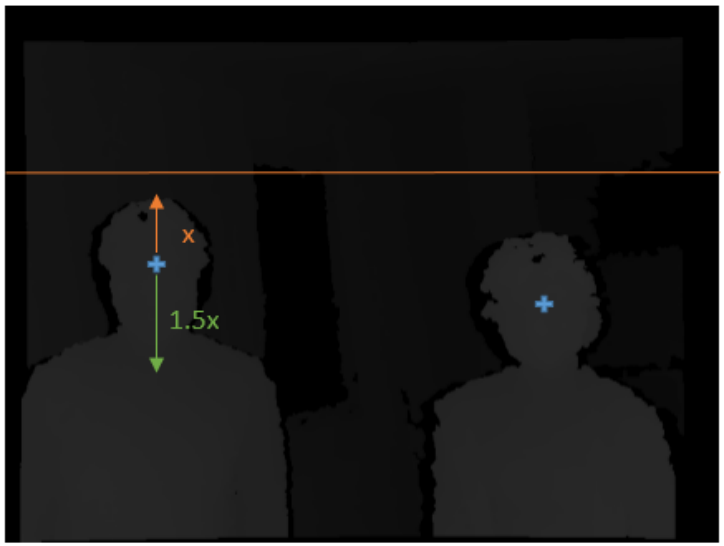
Look for the maximum variation of pixel intensity values.

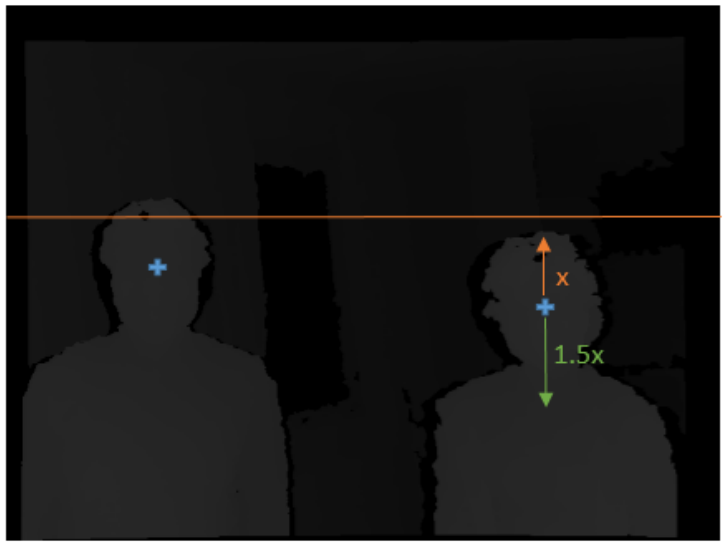


Torso Orientation Algorithm

Step 2:

Look for a point in the torso region.

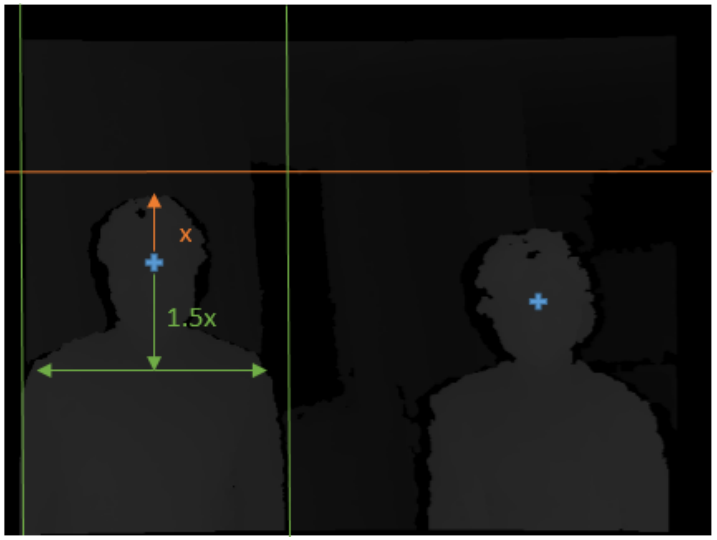


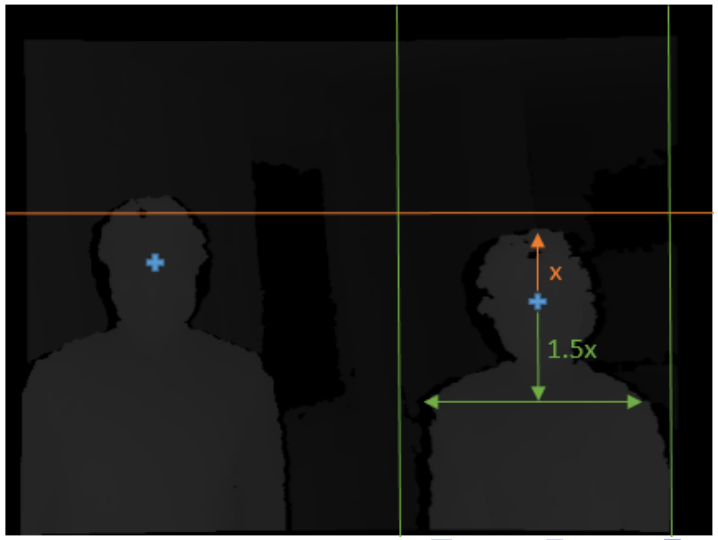


Torso Orientation Algorithm

Step 3:

Find the side boundaries by moving left and right in a horizontal direction.

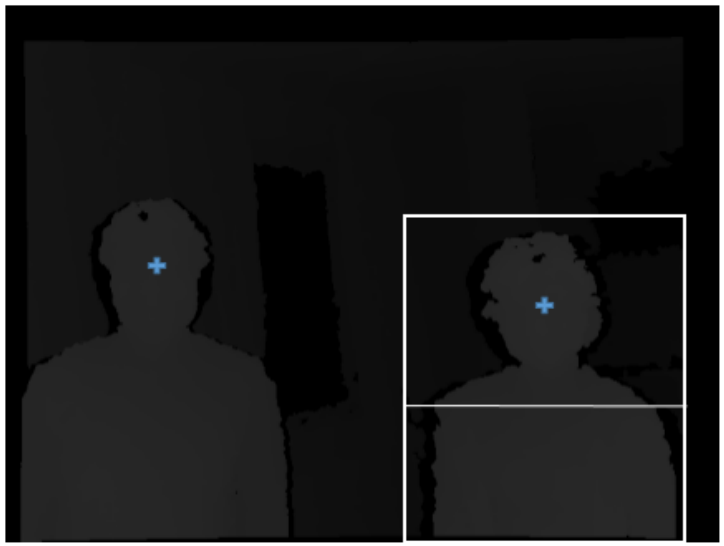




Torso Orientation Algorithm

Step 4:

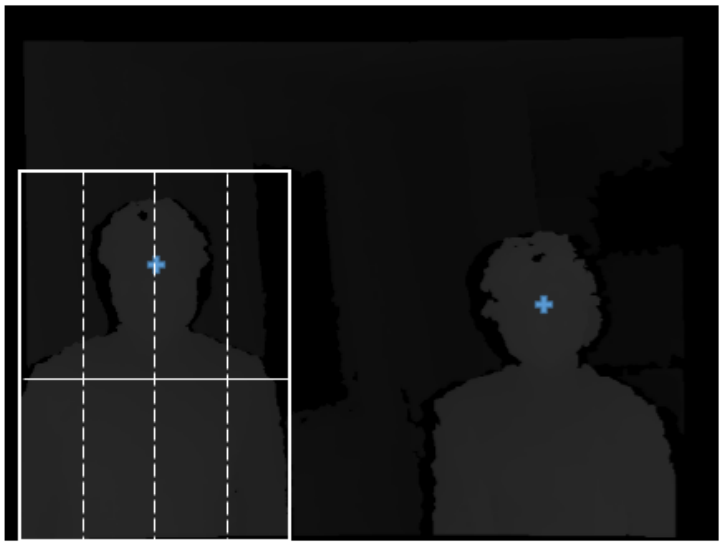
Draw rectangle that encloses the detected person.

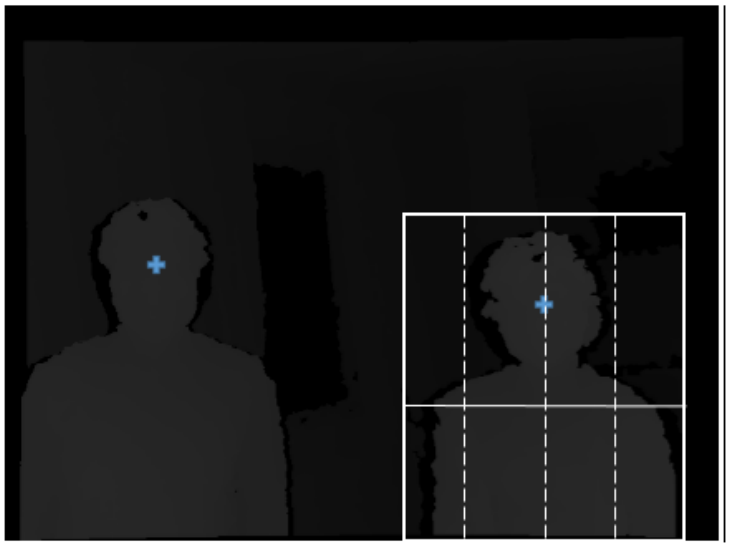


Torso Orientation Algorithm

Step 5:

Select the points corresponding to one-fourth and three-fourths of the person.

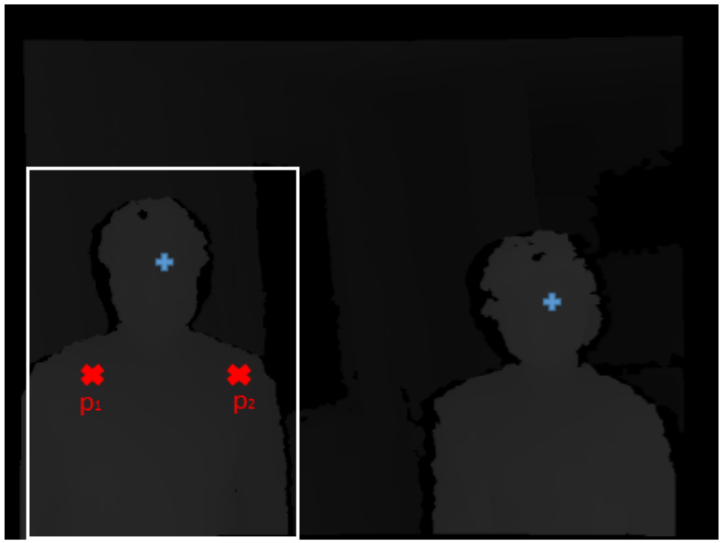


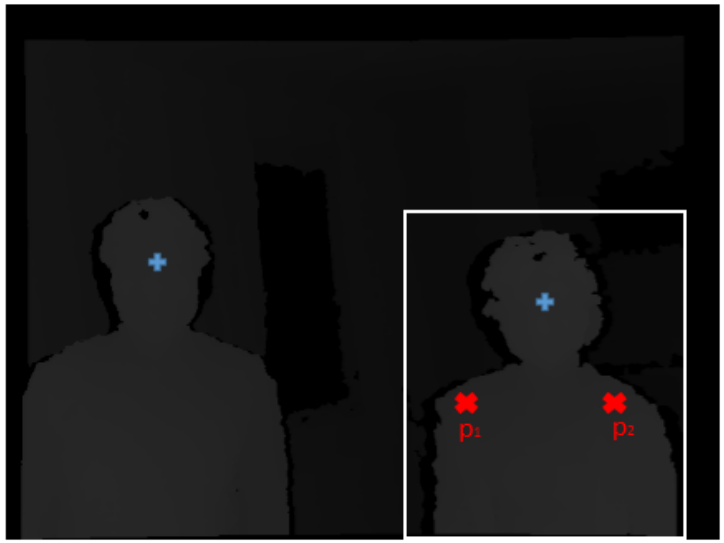


Torso Orientation Algorithm

Step 6:

Points are identified in the final.



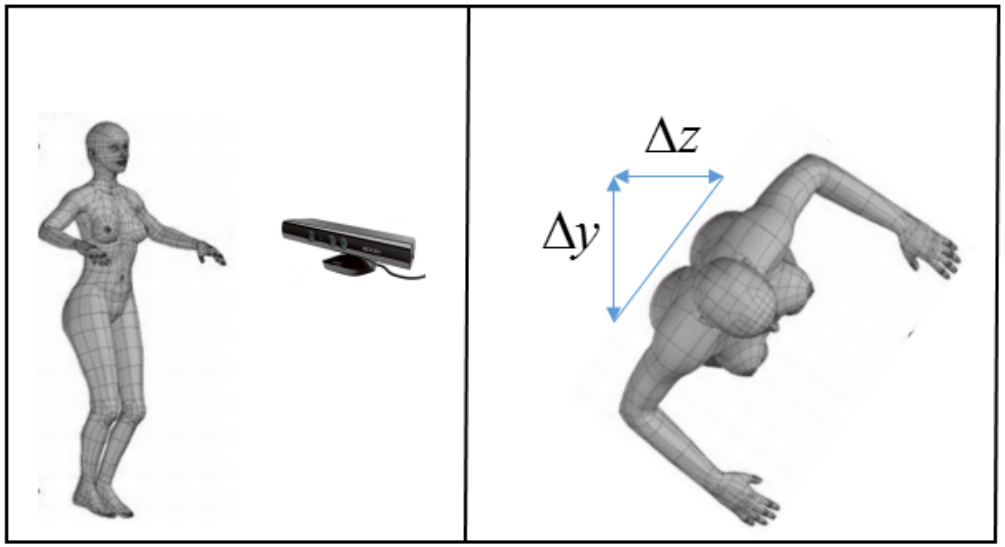


Torso Orientation Algorithm

Step 7

Compute the angle of torso orientation by using the following formula.

$$\theta = \arctan\left(\frac{z_2 - z_1}{y_2 - y_1}\right) = \arctan\left(\frac{\Delta z}{\Delta y}\right)$$



Torso Orientation Algorithm

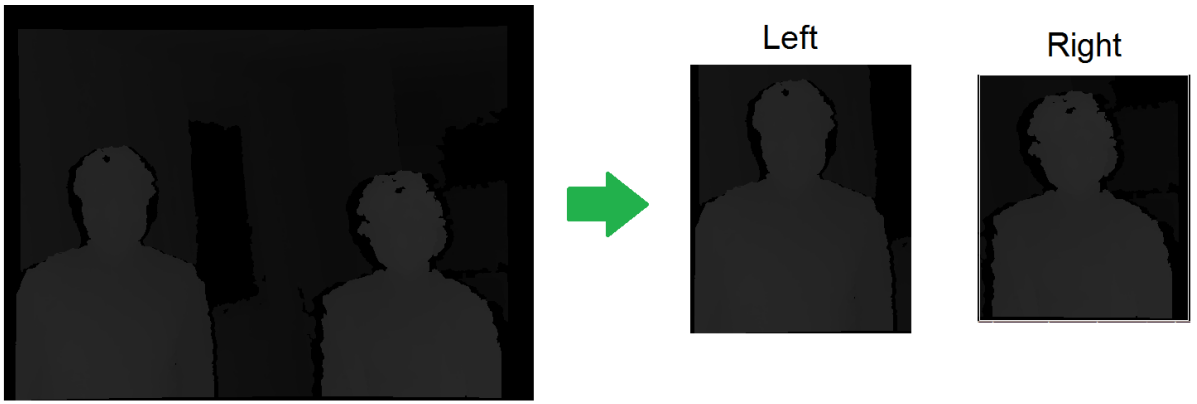
Results: One person

$\theta = -7.63$ *degrees*

Torso Orientation Algorithm

Results: Two persons

Left : $\theta = -21.04$ *degrees*
Right : $\theta = 17.85$ *degrees*



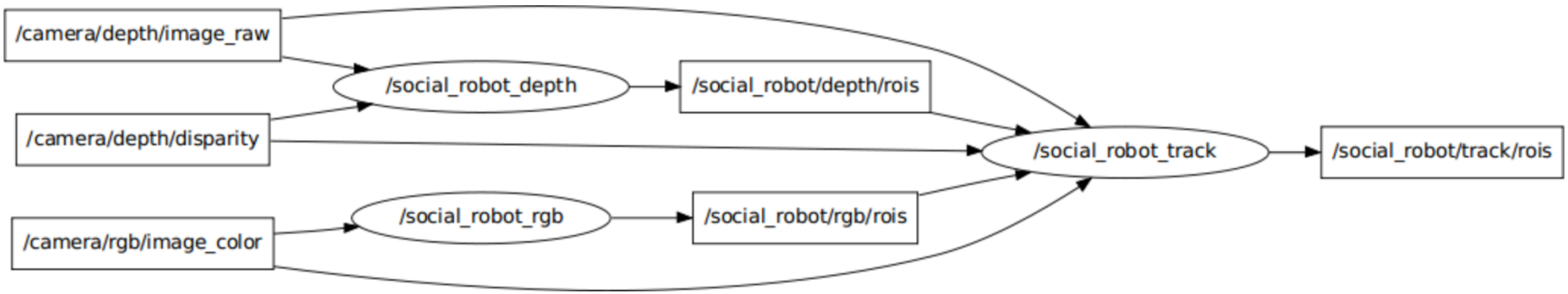
Software Architecture

ROS: Robot Operating System

ROS "provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. ROS is licensed under an open source, BSD license."

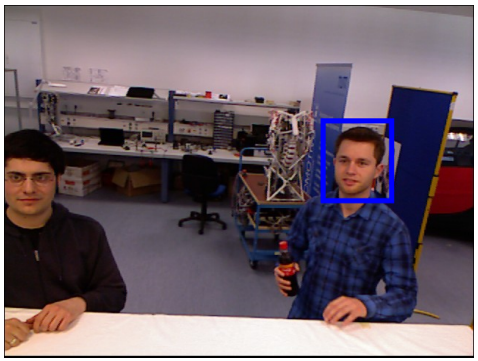
KINECT for XBOX 360

Software Architecture



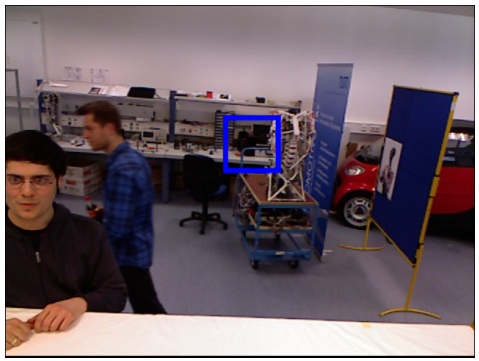
Limitations

- A face is detected and sent to the tracking ...



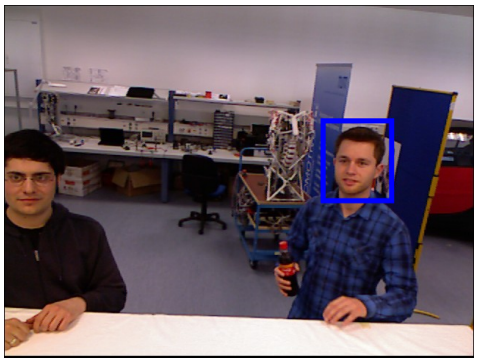
Detector

Tracking



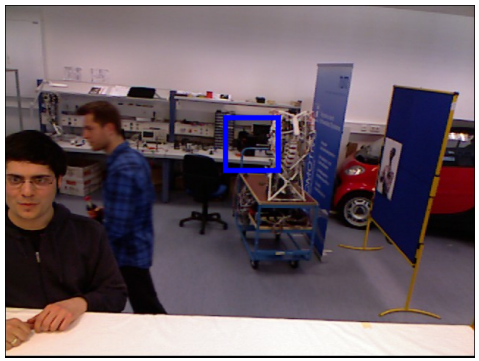
Limitations

- A face is detected and sent to the tracking ...



Detector

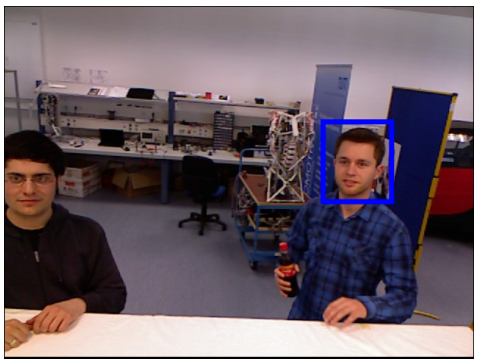
Tracking



- But a couple of frames have already passed based on the processing speed.

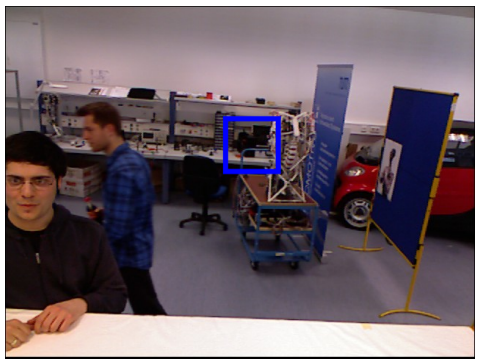
Limitations

- A face is detected and sent to the tracking ...



Detector

Tracking



How to overcome this limitation

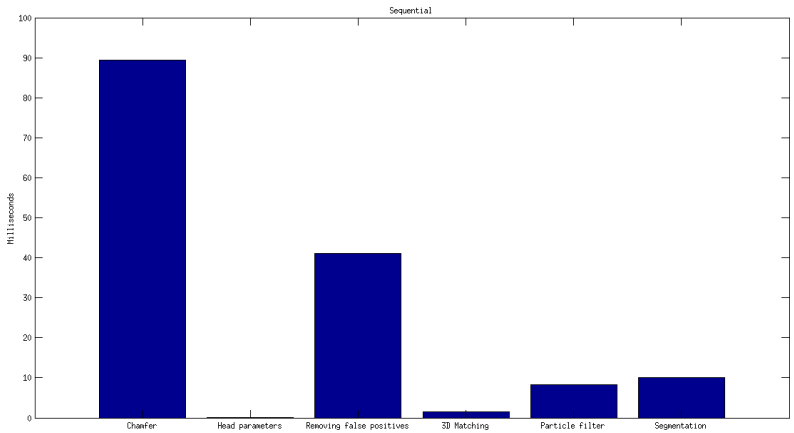
Trying to do dynamic communication between detector and tracking.

First approach

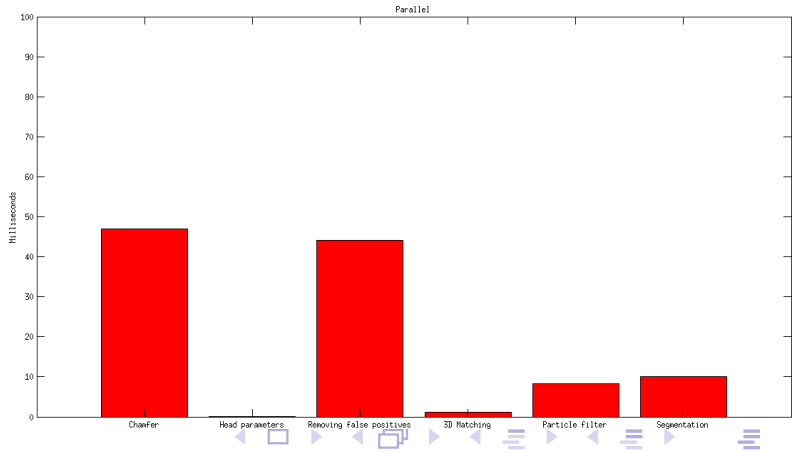
Functions to parallelise

1. Chamfer function $\longrightarrow$ Template Matching.
2. False Positives Removal has many loops and conditions.

Sequential



Parallel



Parallel Programming

Implemented

- In False Positives Removal function many loops can be parallelise.
- Particle filters uses one thread for each track.
- Detection and tracking are running in different threads.

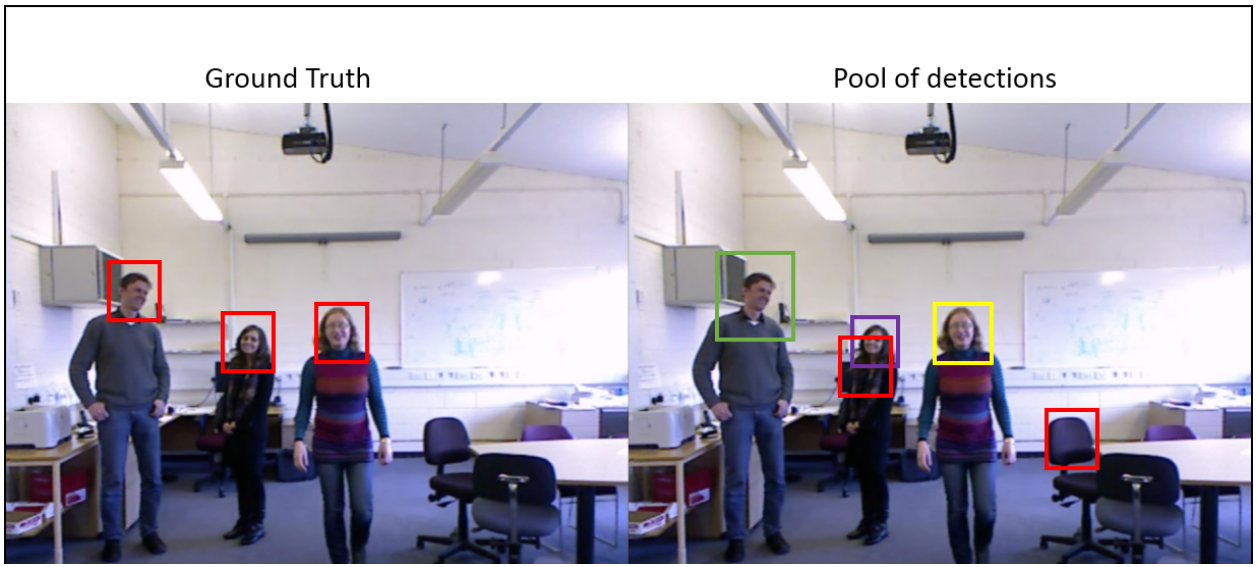
TODO

- For Template Matching function two different approaches have been discussed.
  1. Implement our own function and speed up the process.
  2. Parallelise from the existing function.

Results

Evaluation

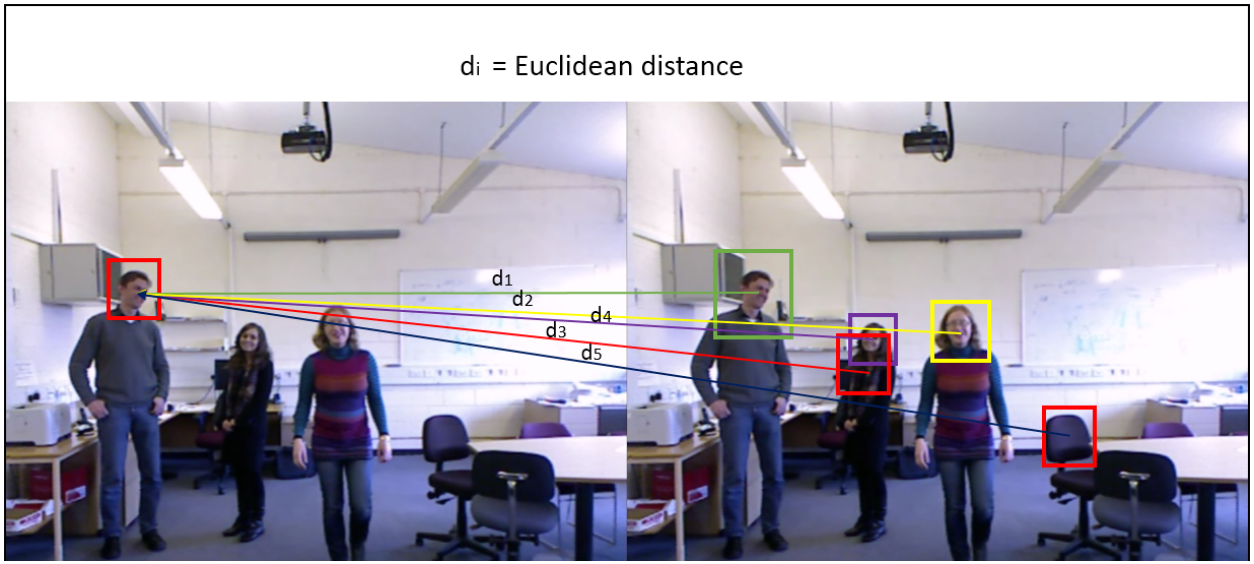
Given the ground truth and a pool of detections per frame.



Results

Evaluation

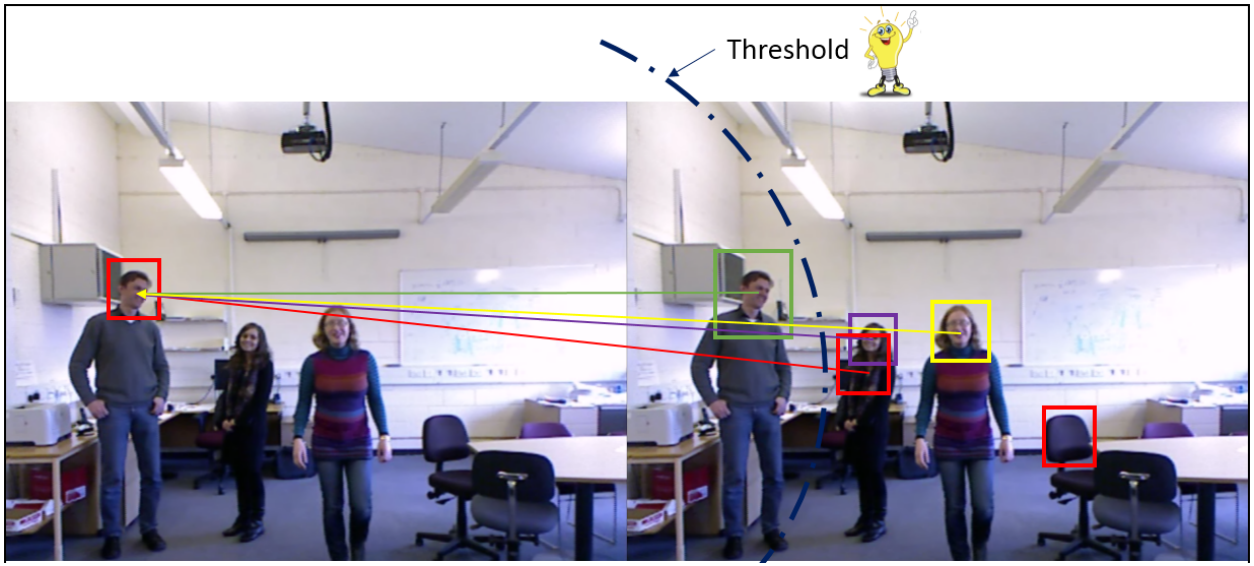
For each target in ground truth, compute euclidean distance and take the minimum one.



Results

Evaluation

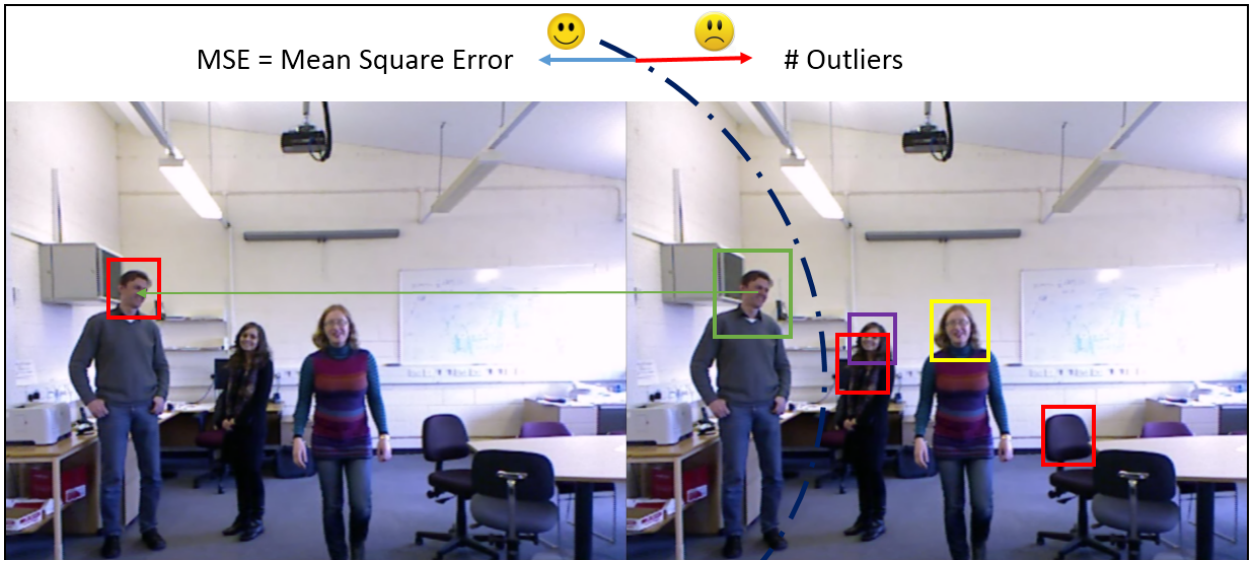
Compare the selected target with a threshold to completely separate outliers.



Results

Evaluation

True positive detections and outliers are used to estimate the algorithm performance.



Results

Root Mean Square Error (RMSE)

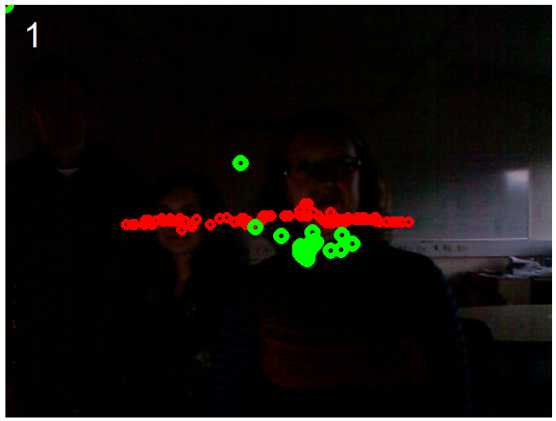
$$d_i = \sqrt{(x_i^g - x_i^r)^2 + (y_i^g - y_i^r)^2}$$

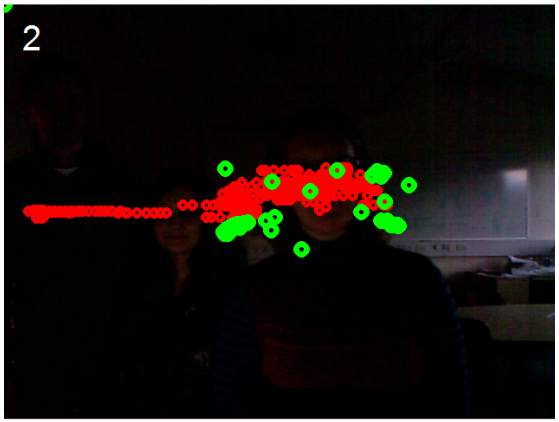
$$RMSE = \sqrt{\frac{1}{N_{frames}} \sum_{i=1}^{N_{frames}} (d_i - \bar{d})^2}$$

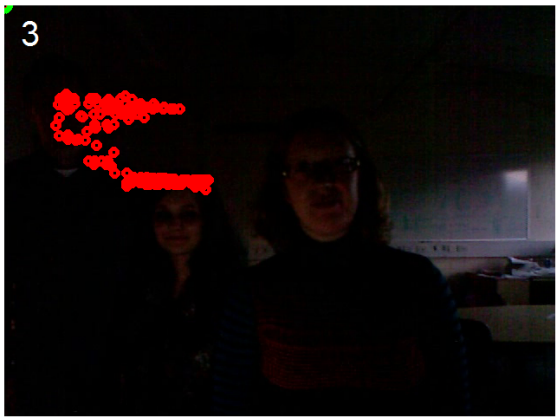
Outliers ratio (OR)

$$OR = \frac{N_{outliers}}{N_{frames}}$$

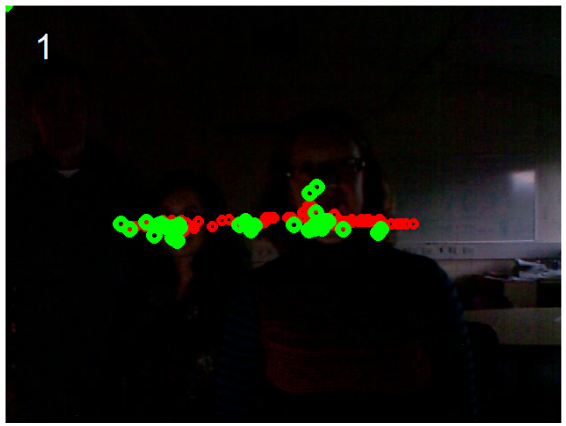
Results: Detections

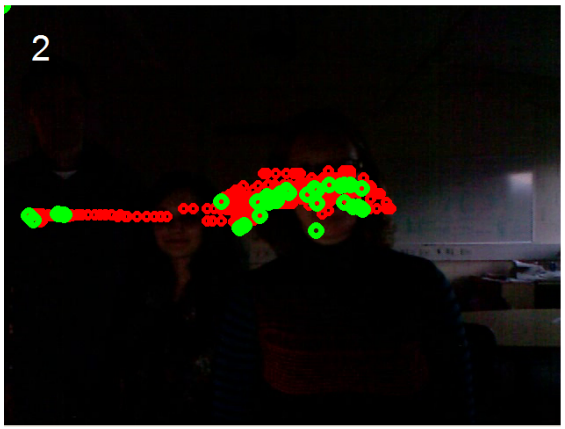


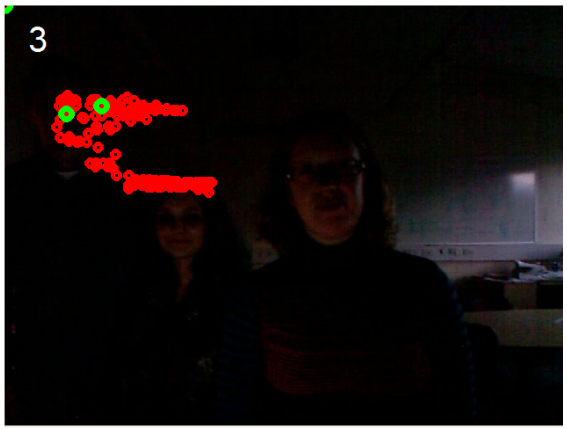




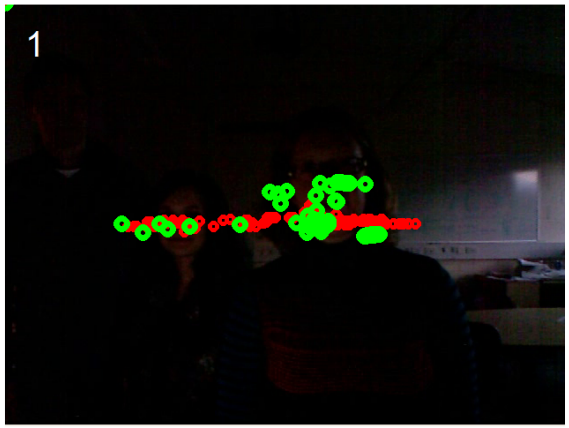
Results: Tracking in RGB

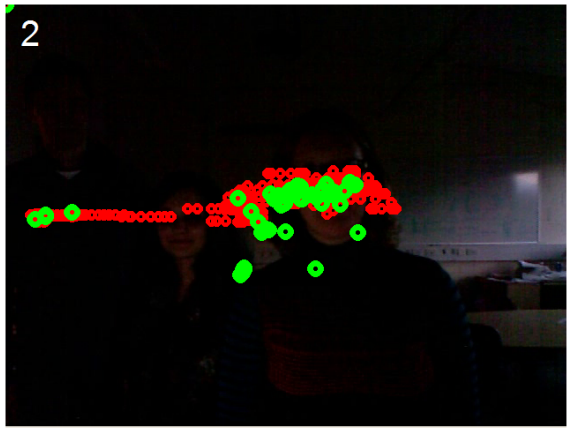




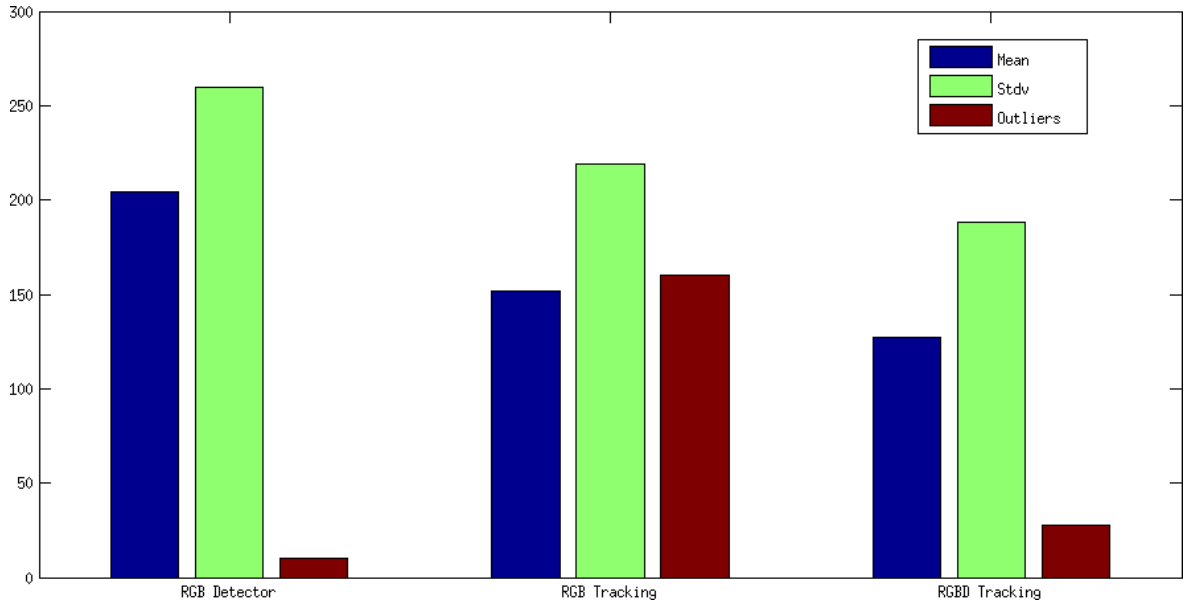


Results: Tracking in RGB + Depth





Results: Outliers



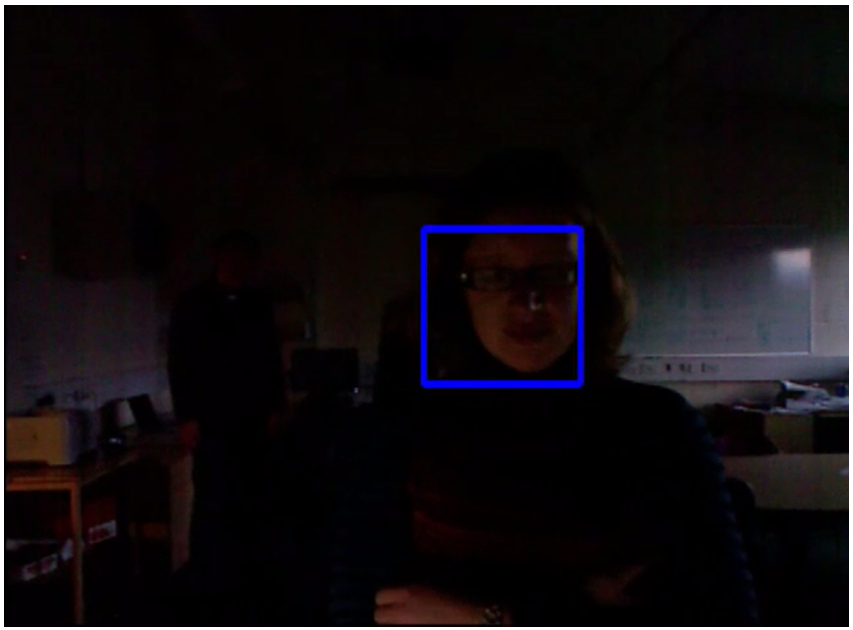
Results: Confusion Matrix

Lab scenario with bad light conditions.

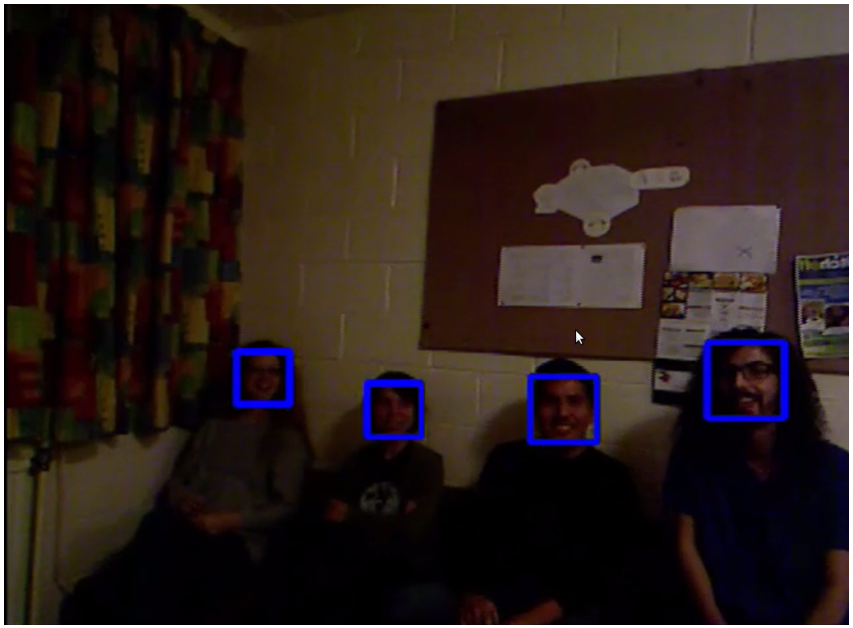
	TP	FP	Accuracy	Error
RGB + HS	157	107	0.302	0.698
RGBD + HS	524	255	0.359	0.641
RGBD + HSD	1070	106	0.469	0.530

- RGB + HS: detection in RGB and using hue & saturation in tracking.
- RGBD + HS: detection in RGB and depth, and hue & saturation in tracking.
- RGBD + HSD: detection in RGB and depth, using hue, saturation and depth in tracking.

Results: Demo



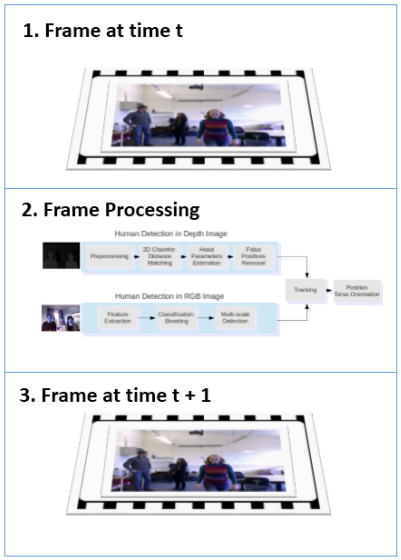
Results: Demo cont.



Conclusions

Real-time vs amount information

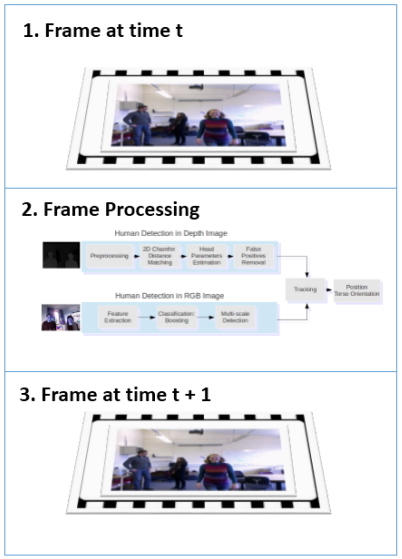
Kinect performs 30 frames per second, while 1 out of 7 frames can be processed.



Conclusions

Implications on Evaluation

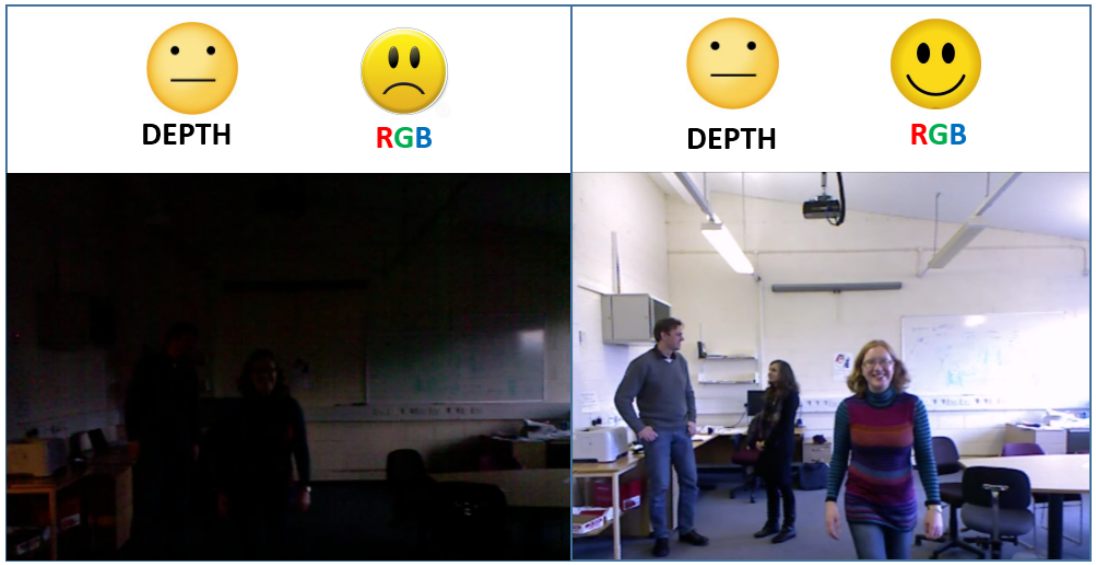
The amount of frames in ground-truth is different from the number of processed frames.



Conclusions

Lighting conditions

Kinect offers a wide spectrum of scenarios with different lighting conditions.



Future work

Classifier of features
In depth image

Classifier

The type of values used for depth information is 16 bits.
A classification algorithm on depth image that provides robustness to the template matching.

Future work

Torso Orientation

- It is sensitive to different poses.
- The use of first order moment in horizontal and vertical directions, can improve the estimation of the centroid by implementing a robust segmentation algorithm.
- Extension to roll and pitch angles.

Different poses

Segmentation

Future work

Depth definition

Depth sensor

Kinect does not provide information that allows to characterize features with high definition.

Future work

Evaluation

The incorporation of the depth information in the accuracy estimation can provide a better estimation of the algorithm performance.

$$d_i = \sqrt{(x_i^g - x_i^r)^2 + (y_i^g - y_i^r)^2 + (z_i^g - z_i^r)^2}$$

References I

- M. Luber, L. Spinello & Kai O. Arras, *People Tracking in RGB-D Data With On-line Boosted Target Models*, University of Freiburg, Germany.
- L. Spinello & Kai O. Arras, *People Detection in RGB-D Data*, University of Freiburg, Germany.
- Y. Freund & R. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting*, in Computational Learning Theory, 1995.
- N. C. Oza & S. Russell,, *Online bagging and boosting*, in Artificial Intelligence and Statistics, 2001, pp. 105, 112.

References II

- H. Grabner & H. Bischof, *On-line boosting and vision*, in Proc. of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, New York, USA, 2006.
- Keni Bernardin and Rainer Stiefelhagen, *Evaluating multiple object tracking performance: the CLEAR MOT metrics*. J. Image Video Process. Article 1 (January 2008), 10 pages.
- John J. Leonard, Bradley A. Moran, Ingemar J. Cox & Matthew L. Miller, *Underwater Sonar Data Fusion Using an Efficient Multiple Hypothesis*. IEEE Robotics and Automation, 1995.

THANK YOU