Eindhoven University of Technology

Multiple face detection and tracking under occlusion

Author:
Raymond Koopmanschap
0894844

Coach:
Dr. C.A. (Cesar) Lopez Martinez

Supervisor:
Dr. Ir. M.J.G. (René) van de Molengraft

July 7, 2017

Report number:
CST 2017 078

Contents

Abstract

Human detection and tracking is a challenging task because of partial or full occlusions. To tackle this problem a face detection algorithm together with a Multiple Hypothesis Tracker (MHT) is used. The MHT algorithm keeps track of multiple hypothesis and chooses the one with the highest overall probability. The possibility of occlusions is thus handled by keeping track of different scenarios and chose the best solution when new useful measurement data is received. In this way, consistent tracking of a person can be obtained.

1 Introduction

Robust people detection and tracking is important for many applications. It is used in video-surveillance[11] and for several robotic applications such as mobile service robots [10]. Common algorithms like part-based pedestrian detection and background modeling fail to detect a person when a significant part of the scene is in motion or when most of the body parts are occluded [4]. Despite a lot of progress tracking also remains a challenging task due to noise, occlusions, and changes in the position and scale of the object. To solve this problem human detection and tracking are combined into a single scheme which includes a face detection algorithm, a transformation to a single point and a Multiple Hypothesis Tracker. The MHT will update the state when new evidence is provided by the measurements and choose the most probable hypothesis accordingly. In this way, occlusions can be handled naturally and the amount of false positives will be reduced, which will increase the performance of the overall detection and tracking structure.

The remainder of this report is organized as follows: section 2 will describe the environment of the implemented algorithm and background information of the project. Section 3 will outline the current state with respect to tracking and detection algorithms, it will show a comparison between the performance of different multiple target tracking algorithms and the relevant details of the current implemented work on AMIGO. Section 4 will describe the contribution of this report to the current models. Section 5 goes more in detail how the specific detection and tracking algorithms are combined and section 6 will show results about the performance of the detection and tracking.

2 Background

2.1 Introduction RoboCup

To enhance the development of mobile service robots in general and in domestic applications in specific, the Robocup@home league was introduced [2]. The mission of Robocup@home is to promote the development of service and assistive robot technology with high relevance for future personal domestic applications. It is the largest international annual competition for autonomous service robots and is part of the RoboCup initiative. A set of tests has to be executed in a realistic home environment setting. In this competition human-care robotics have to demonstrate their abilities mainly in the following domains: Human-Robot-Interaction and Cooperation, Navigation and Mapping in dynamic environments, Computer Vision, Object Recognition under natural light conditions and object manipulation.

2.2 Introduction AMIGO

One of the participants of those challenges is AMIGO the human-care robot of Tech United from the University of Eindhoven. AMIGO has finished first place in the European RoboCup@home league 2016 and second in the RoboCup world championship 2016 and is still developing. AMIGO has three main sensors: a Kinect camera who can collect RGBD data (1), a torso laser scan sensor (2) and a base laser scan sensor (3).

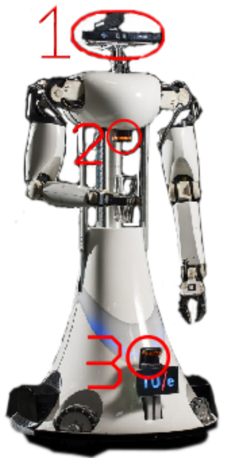


Figure 1: AMIGO

The implementation described in this report will, in the end, be used for AMIGO. The Kinect sensor is used to test the final implementation and to demonstrate the performance of the algorithm.

3 Related work

3.1 Face detection

The detection and tracking of persons consists of several stages. First, a detection algorithm is used to detect faces. The face detection will be done with openface [7]. It detects faces, transforms the face to feed it to the neural network which will classify the detections. Openface is a recent open source face detection implementation with state-of-the-art performance [6].

3.2 Tracking algorithms

For people tracking many algorithms exist. Often a Kalman filter is used to make a prediction of the detections in time, next data association will take place and lastly the update of the position is calculated by using the prediction of the Kalman filter and associated measurement. For multiple target tracking, many different algorithms can be used, each with their own variations. The global nearest neighbor (GNN) is the most basic one, however, a disadvantage is its incapability of handling occlusions because if it is unable to update his state with new available measurements, the track is lost. More advanced algorithms for tracking try to solve this problem. Popular are the joint probabilistic data association filter (JPDAF) [12] and the Multiple Hypothesis Tracker [14]. Another popular algorithm that exists in many variations is the particle filter which is also used for human tracking [4]. Little research is done to compare advanced data association algorithms. A study worth mentioning is [13] which compares various advanced tracking methods. In this work, the multiple hypothesis tracker shows overall good performance. However comparing different algorithms is difficult since true positive, false positive and false negative rates depend on how track creating and maintaining is done which differ strongly per implementation [13]. Nonetheless, the MHT has an advantage over the JPDAF or GNN in handling occlusions naturally by delaying ambiguous scenarios until new useful evidence arise [8]. This makes it a promising candidate for testing the desired implementation.

3.3 The Multiple Hypothesis Tracker

The MHT maintains multiple hypotheses and chooses the one with the most probable state. Each measurement has a probability that can originate from an object not present in the world model yet, represent a false positive or originate from an object that is already in the world model. Prior probabilities for each of them have to be initialized and those values will affect the result of the algorithm. For example, a higher prior probability for new objects leads to faster associating a new measurement with a new object instead of a false positive. After the prior values are initialized, gate computation is performed 2, [5].

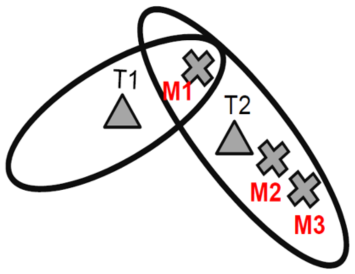


Figure 2: Gate computation, tracks are displayed with triangles and measurements with crosses

Each hypothesis contains a set of tracks with respective covariance. The decision if a measurement is inside the track gate is dependent on the distance between the measurement and track, the corresponding covariance of the track and the measurement uncertainty matrix. This notion of calculating distances between measurements and tracks is called the mahanolobis distance and is often used in tracking algorithms. The gate computation reduces the amount of possible combinations by removing unlikely associations, where the distances between the tracks and measurements are too far away from each other.

Next hypothesis are formed. For each measurement, at least two scenarios are possible, a false alarm or a new measurement and depending on the gate computation a measurement might be associated with an existing track. After the hypotheses are calculated, the states with a very low probability for a certain amount of steps are deleted to increase computational efficiency. Next, the most probable state is chosen, the tracks are updated using a Kalman filter and after new measurements arrive the process will repeat itself 3.

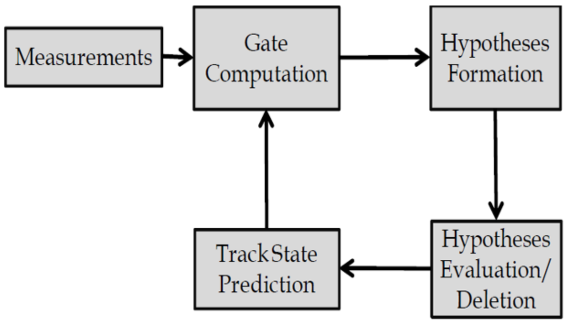


Figure 3: Multiple hypothesis tracker overview

3.4 Probabilistic Multiple Hypothesis Anchoring

Although the basic MHT keeps track of multiple hypotheses, occlusions still remains a challenging task because when an existing track is occluded it does not have a corresponding associated measurement and therefore can not be assigned and updated with that respective measurement. Thus a model which describes the state after occlusion is preferred. Hence an extension of the MHT, the probabilistic multiple hypothesis anchoring (PMHA) [9] will be used to test the capabilities of the proposed scheme. This extension includes a probabilistic anchoring algorithm which enables to create a world model which can handle class and color attributes in a probabilistic manner while also using multiple hypotheses to keep track of objects. If an existing track can be updated with a new measurement a constant velocity Kalman filter is used to predict the position. However, if no updates are received for a fixed amount of time the Kalman filter is replaced by a fixed Gaussian distribution with a mean of the last known position and a fixed covariance which can be freely chosen. As a result, the precision of the occluded state can be chosen by varying the covariance. Furthermore to indicate the uncertainty in the motion model the Kalman filter is given a high process noise because a constant velocity Kalman filter has only acceptable accuracy for short time periods. Thus the model will rely more on new measurements which provide valuable information to update the position.

The tracking results of the overall PMHA implementation relies on properly chosen several parameters including the before mentioned prior probabilities for a new object, an existing object and a false positive. Furthermore, a covariance matrix to model the uncertainty of a given measurement, the process noise of the Kalman filter and the fixed state Gaussian distribution has to be initialized.

Summarizing, the advantages of the PMHA are the natural occlusion handling, being able to handle multiple and an unknown number of object detections and delaying ambiguous scenario until useful evidence arrives. A general disadvantage of the MHT is the rapidly growing hypothesis trees thus complicated measures are needed to keep computational efficiency high. However if properly handled the MHT is a promising algorithm.

4 Contribution

The main contribution to this project is the combination of existing methods into one framework. A person's face will be detected using openface which produces as output a region of interest (ROI). This ROI is transformed to a 3D point in space which in turn is, together with a covariance, fed in the PMHA algorithm. This report will bridge the gap between the currently existing methods and add an appropriate covariance matrix to the measurements done by openface. Consequently, a face can be tracked in the presence of occlusions and the output will be a position with respect to the Kinect camera and a 3 by 3 covariance matrix of the corresponding position.

5 Implementation

First detections are done with the Kinect camera to find faces. For that purpose, a face detection algorithm is used. Openface is used to obtain region of interests from RGBD images of the Kinect [1]. The regions are defined as an x,y offset and corresponding width respectively height of the ROI expressed in the amount of pixels it contains.4

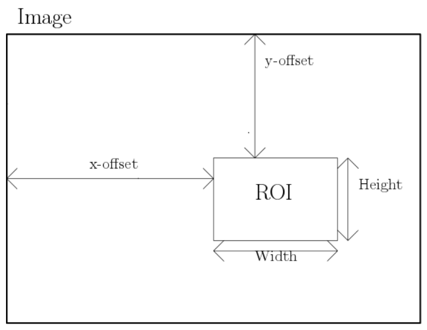


Figure 4: Region of interest dimensions

The region of interest obtained by openface needs to be transformed to an x,y,z position in order to be useful for wire. This can be computed given a ROI and the depth information of the camera [3]. This position will be calculated with as reference frame the Kinect. 5

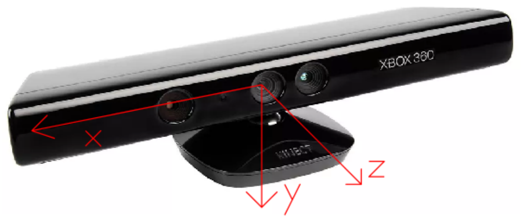


Figure 5: x,y,z convention

After the 3D position is obtained, tracking is done. To feed the position to the PMHA a 3 by 3 covariance matrix is needed that describes the variances in the x,y,z directions. It is assumed that the off-diagonal entries, the covariances, are zero. Hence the variances of x,y and z are independent of each other. The aforementioned assumption is reasonable because a change in x-direction does not imply a change in y-direction or any other possible combination of directions.

To obtain the variances in the three directions and test the assumption of independence, measurements are done. After the required results are obtained an appropriate covariance matrix is realized. In this way, the data can be sent to the PMHA algorithm which will give as output the mean coordinates x,y,z with respect to the Kinect frame and a covariance matrix to indicate the precision of the resulting position to perform tracking and handle occlusions by using the before mentioned multiple hypothesis structure. An overview can be seen below with as objects the detected faces including a 3D position, covariance matrix and identification number 6.

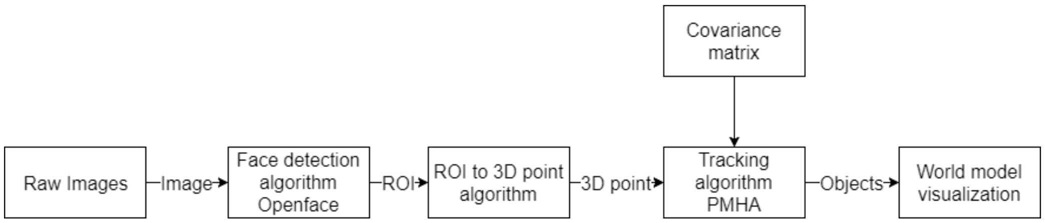

Covariance matrix
Raw Images
Image
Face detection algorithm Openface
ROI
ROI to 3D point algorithm
3D point
Tracking algorithm PMHA
Objects
World model visualization

Figure 6: Detection and tracking scheme

6 Measurements and results

6.1 covariance matrix of stationary person

Multiple measurements are executed to test the performance of the implementation. The first measurement is testing if the variances and covariances are distance dependent. Thus a point further away from the Kinect has a higher covariance matrix. To test this a human is standing in front of the Kinect camera at a distance in the z-direction of 1, 1.5, 2 and 2.5 meter at each position for 2 minutes.The x,y,z position of the face is recorded and accordingly, the variances and covariances will be calculated given this data with n the number of measurements taken 1, 2, 3.

$$\mu = \sum_{i=1}^{n} \frac{x_i}{n} \tag{1}$$

$$Var(x) = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2 \tag{2}$$

$$Cov(x,y) = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{1}{2}(x_i - \mu_x)(y_i - \mu_y) \tag{3}$$

Table 1: Variances of measurements when standing still

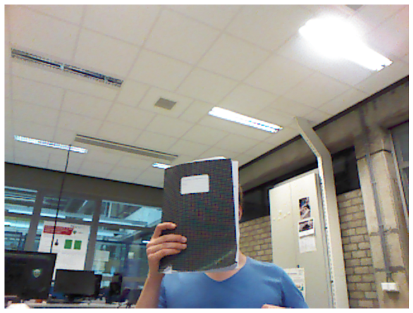
	x	y	z
1 m	$0.9837 * 10^{-4}$	$0.6473 * 10^{-4}$	$0.1237 * 10^{-4}$
1.5 m	$0.0305 * 10^{-4}$	$0.8117 * 10^{-4}$	$0.3719 * 10^{-4}$
2 m	$0.0966 * 10^{-4}$	$0.0437 * 10^{-4}$	$0.6315 * 10^{-4}$
2.5 m	$0.0005 * 10^{-4}$	$0.0001 * 10^{-4}$	$0.3569 * 10^{-4}$

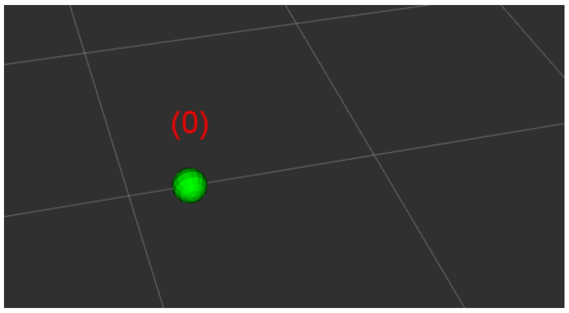
The obtained variances are fluctuating around an order of magnitude 10^{-4} which is rather low. This can be explained by the fact that a substantial movement is needed to obtain a different ROI. Thus if a person is sitting perfectly still the data is nearly the same and does not differ. The covariances are 2 orders of magnitude lower than the variances 10^{-6} and as assumed indeed negligible. As can be seen from the measurements the variances are not distance dependent.

However, when a person is moving the variances differ from the obtained results. In general, the variance increases with a higher speed. Thus the values for the covariance matrix need to be chosen accordingly. In addition to this, the maximum allowable acceleration of the Kalman filter prediction model can be chosen in the PMHA which allows for specifying the noise level of the prediction model. A higher allowable acceleration will handle fast movements better but can interpret two measurements close together as a movement of one object to the other especially when the former will be occluded after a certain moment.

6.2 Occlusion handling

The second experiment shows how occlusions are handled in the final implementation. The measurement setup is as follows. One face is detected and modeled as a 3D point in the world model. This face is then occluded and moved to the right. Then a new face is added to the world model. Openface alone would fail in this experiment because if a occlusion takes place the measurement is lost. On the other hand the PMHA can handle occlusions and as a result, a consistent representation of the faces is obtained. First, a face is detected and thereafter the face will be occluded 7.

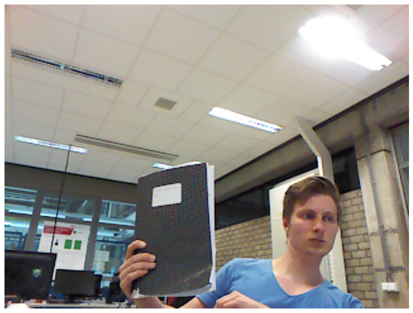

(a) Occlusion

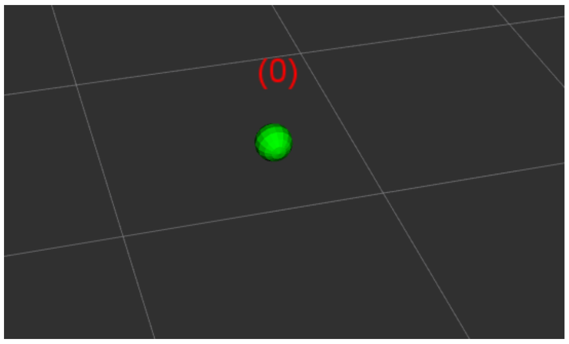


(b) Occlusion in world model

Figure 7: Occlusion after a first detection

Next, the face is shown after the occlusion. The world model is correctly updated.

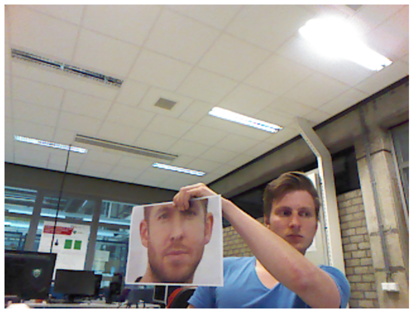

(a) Showing face



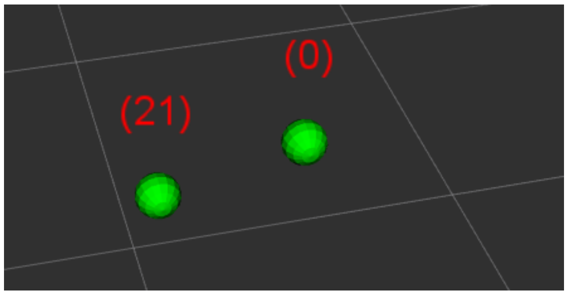
(b) Showing face in world model

Figure 8: Showing face after occlusion

Finally when a new face is shown the world model is correctly updated recognizing the new face as a new object. As can be seen from the numbers above the green dots. In this way, the algorithm keeps track of different objects and will remember which object in the world model belongs to which face.



(a) Showing second face



(b) Showing second face world model

Figure 9: Showing face after occlusion

7 Conclusion & future work

In this project, the open source implementation of openface is combined with the PMHA to handle occlusions of faces in a natural way. First, by transforming the ROI of a face given by openface to a 3D point in x,y,z coordinates relative to the Kinect frame. Then by adding an appropriate covariance matrix and using correct values of other parameters defined in the PMHA tracking will be achieved. Tests are performed to measure what the value for the covariance matrix is for a stationary face detection and if the variances are depth dependent. Furthermore a test is performed to show the difference of a stand alone face detection algorithm and a multiple hypothesis tracking approach that is able to handle occlusions.

The model can be extended by adding a different class to detect people instead of only faces. For example, a torso laser scanner which detects torsos and a part-based algorithm can be used to construct which torsos and faces belong to the same person. In general, this can be extended to other parts of the human body as well for example legs and shoulders, making people detection more robust. In addition to this the added covariance matrix can be made speed dependent, thus increases with higher rates of changes in the x,y,z position in time, instead of constant which will improve the accuracy of the model.

References

[1] Openface ROS. https://github.com/tue-robotics/image_recognition/tree/master/openface_ros.

[2] RoboCup@Home. http://www.robocupathome.org.

[3] Roi to 3D point. https://github.com/tue-robotics/rgbd/blob/master/src/get_3d_point_from_image_roi_node.cpp.

[4] Irshad Ali and Matthew N. Dailey. Multiple human tracking in high-density crowds. *Image and Vision Computing*, 30(12):966–977, dec 2012.

[5] Angelos Amditis, George Thomaidis, Pantelis Maroudis, Panagiotis Lytrivis, and Giannis Karaseitanidis. Multiple Hypothesis Tracking Implementation.

[6] Brandon Amos, Bartosz Ludwiczuk, and Mahadev Satyanarayanan. Models and Accuracies - OpenFace. http://cmusatyalab.github.io/openface/models-and-accuracies/.

[7] Brandon Amos, Bartosz Ludwiczuk, and Mahadev Satyanarayanan. OpenFace: A general-purpose face recognition library with mobile applications. 2016.

[8] S.S. Blackman. Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, 19(1):5–18, jan 2004.

[9] J Jos Elfring, S Sjoerd Es, van den, MJG René Molengraft, van de, and M Maarten Steinbuch. Semantic world modeling using probabilistic multiple hypothesis anchoring. *Robotics and Autonomous Systems*, 61(2), 2013.

[10] S. Feyrer and A. Zell. Detection, tracking, and pursuit of humans with an autonomous mobile robot. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289)*, volume 2, pages 864–869. IEEE, 1999.

[11] Álvaro García-Martín and José María Martínez. People detection in surveillance: classification and evaluation. *IET Computer Vision*, 9(5):779–788, oct 2015.

[12] B. Habtemariam, R. Tharmarasa, T. Thayaparan, M. Mallick, and T. Kirubarajan. A Multiple-Detection Joint Probabilistic Data Association Filter. *IEEE Journal of Selected Topics in Signal Processing*, 7(3):461–471, jun 2013.

[13] G.W. Pulford. Taxonomy of multiple target tracking methods. *IEE Proceedings - Radar, Sonar and Navigation*, 152(5):291, 2005.

[14] D. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, dec 1979.

Appendix

3D point to world evidence python script

```
#!/usr/bin/env python

# this is the service to transform a 3D point to a WorldEvidence
# message
import rospy
from beginner_tutorials.srv import *
from wire_msgs.msg import WorldEvidence
from wire_msgs.msg import ObjectEvidence
from wire_msgs.msg import Property


def callback(data):

    worldev = WorldEvidence()
    worldev.header.stamp = rospy.Time.now()
    # set the worldevidence frame to the camera frame to see
    # the visualization from the perspective of the Kinect.
    worldev.header.frame_id = "camera_rgb_optical_frame"

    list_objev = []

    # make a list of all object evidences. One object evidence has
    # two properties a position with covariance and a class label
    # with a discrete probability.

    # This class label is not used here but is added for proper use
    # in the future.

    for point_seq in data.points:
        objev = ObjectEvidence()
        obj_props = Property()
        obj_props.attribute = "position"
        obj_props.pdf.type = 1
        obj_props.pdf.dimensions = 3
        obj_props.pdf.data = [1.0, point_seq.point.x,
        point_seq.point.y, point_seq.point.z,
        0.01, 0.0, 0.0, 0.01, 0.0, 0.01]
        obj_props2 = Property()
        obj_props2.attribute = "class_label"
        obj_props2.pdf.type = 5
        obj_props2.pdf.dimensions = 1
        obj_props2.pdf.domain_size = -1
        obj_props2.pdf.values = ["face"]
        obj_props2.pdf.probabilities = [1]

        objev.properties = [obj_props, obj_props2]
        list_objev.append(objev)

    worldev.object_evidence = list_objev

    return Geometry_msgs_to_world_evidenceResponse(worldev)

def geometry_to_world_evidence():
    rospy.init_node('geometry_test_server')
    rospy.Service('geometry_to_world_evidence',
        Geometry_msgs_to_world_evidence, callback)

    print "Starting"
    rospy.spin()

if __name__ == "__main__":
    geometry_to_world_evidence()
```

Main script which subscribes to a camera image and publishes world evidence messages to wire

```
#!/usr/bin/env python
import rospy
from sensor_msgs.msg import Image
from sensor_msgs.msg import RegionOfInterest
from geometry_msgs.msg import PointStamped
from image_recognition_msgs.srv import Recognize
from image_recognition_msgs.msg import Recognition
from wire_msgs.msg import WorldEvidence
from rgbd.srv import Project2DTo3D
from beginner_tutorials.srv import *

srv = None
srv2 = None
srv3 = None
pub = None

def callback(data):

    try:
        # transforms the image to an ROIs
        resp = srv(image=data)
        # initialize time that is needed to transform a 2D roi
        # to a 3D point because it also uses the depth info
        # of the kinect image at the time that the roi is send.
        time = rospy.get_rostime()
        a=[]
        # to make a list of all ROIs
        for test in resp.recognitions:
            a.append(test.roi)

        # 2D roi to 3D point service
        resp2 = srv2(time, a)
        point=resp2.points

        # from a 3D point to a message in WorldEvidence format -->
        # a format that wire can use for the PMHA.
        resp3 = srv3(point)
        pub.publish(resp3.ev)
    except IndexError, e:
        print e
        rospy.logerr("Service call failed: %s" % e)

    return 1

def listener():
    # define all services, subscribers and publishers
    global srv
    global srv2
    global srv3
    global pub

    rospy.init_node('listener', anonymous=True)
    rospy.wait_for_service('/recognize')
    rospy.wait_for_service('project_2d_to_3d')
    rospy.wait_for_service('geometry_to_world_evidence')

    rospy.Subscriber("/camera/rgb/image_rect_color", Image,
        callback, queue_size=1, buff_size=999999999)

    srv = rospy.ServiceProxy('/recognize', Recognize)
    srv2 = rospy.ServiceProxy('project_2d_to_3d', Project2DTo3D)
    srv3 = rospy.ServiceProxy('geometry_to_world_evidence',
        Geometry_msgs_to_world_evidence)

    pub = rospy.Publisher("/world_evidence", WorldEvidence,
        queue_size=10)

    rospy.spin()

if __name__ == '__main__':
    listener()
```