Person Detection using RGB-D Sensor and Thermo Camera

M. Pecka, V. Kubelka, K. Zimmermann and M. Reinstein

April 8, 2015

1 Introduction

Doc. Ing. F. Zlo, CSc. wants[1] to reduce the number of students by means of Robotron [1]. However, the publicly available face detector he has been using so far produces too many false positive detections (e.g. printed faces on posters). Given a proper calibration of provided sensors, you can augment bare RGB images with temperature and depth information. Avoid being exterminated by Doc. Ing. Zlo CSc., prove yourself useful and find a way to use the temperature and depth information to improve the detector so that it detects only faces of *real persons*. An example of how your algorithm should work is given in Figure 1. Please note that Doc. Ing. Zlo, who might appear in provided data, is not a *real person*.

2 Solution outline

We have used an RGB-D sensor and thermo camera to record target data into a BAG-file, see Section 4 for details. Your task is to improve the given face detector, which detects faces only in RGB images; see Section 6 for details. Since the thermo-camera and the depth sensor are also available, it is desirable to enrich the standard color image by temperature and distance of observed faces. You will find these measurements useful for improving the face detector. However, images provided by the sensors are not the same size nor aligned since the sensors are mounted next to each other so they observe the scene from different viewpoints. Hence, you will need to find the transformation that binds pixels of the images together. See Section 5 for details. Once the transformation among measurements is known, you are expected to build your own detector on

[1]Except the use-case of Doc. Ing. Zlo, a precise detector of real persons can be further utilized in several ways. It may also serve as a part of a victim detection system, which is essential for almost all Search&Rescue robots. Or it may be a part of a security video surveillance system. Last, but not least, it may be useful for interactive robots to detect people around them and start a conversation, offer help or some other task. Connected with (known) face recognition, the possibilities are even greater.

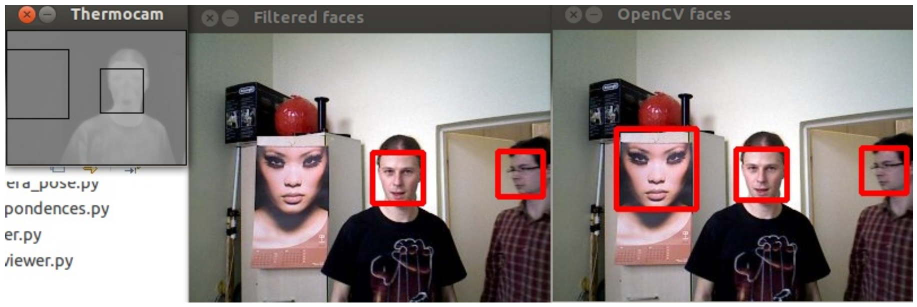

Figure 1: An example of detected faces filtering using the thermo camera data. Notice the face outside the thermo camera field of view – the thermo camera filter would have to leave it untouched even if it weren't a real person (you can't base your decision on missing data!).

top of the provided one, which reduces the number of false positive detections, see Section 3 for assignment details.

3 Assignment

The steps mentioned below should guide you through the completion of the semestral work.

1. Extract the data from BAG files
   - Download the code template from CourseWare [4] and also download the datasets belonging to this task [2, 3]. Extract the datasets e.g. to `iro/data`.
   - Fill in the places in the code marked with the `TODO:` comment to get the code working. You'll need to write your own subscribers to the relevant topics.
   - When you have the base of the code working, you should be able to call `rosrun` to launch your subscriber and while playing a dataset BAG file, the program should store the images and other data in MATLAB *.mat* files.
2. Calibrate the camera setup
3. Use Zhu-Ramanan face detector implemented in MATLAB
   - Initialize face detector:
     ```
     MY_THRESHOLD = x.yy;
     load_model;
     ```
   - Run the face detector on image `im`:
     ```
     [bounding_boxes, scores] = face_detect(im, model)
     ```

- See `README.md` for further help.

4. Implement your own classifier

   - Use temperature and depth measured within detected bounding boxes to implement your own classifier.
   - It is highly recommended to draw the ROC curves to see the influence of your classifier, otherwise you can easily do changes which look reasonable but harm the overall detection rate.
   - Ground truth will be provided after test datasets are recorded during the task assignment lab, link will appear on the course web site [4].

   Note, that correctly drawn ROC curve is a compulsory part of your report. Both `TP` and `FP` are relative with respect to *all* ground truth faces in all BAG files – for example: if you have 10 incorrect detections and 50 correct detections out of 100 ground truth faces, then `TP` $= 0.5$ and `FP` $= 0.1$). This also means `TP` can not have values greater than 1 (the best we can do is detect all faces), but `FP` can surely go beyond 1 (if the detector shows too much false faces). A correctly detected face is a bounding box B covering ground truth bounding box B_0 by more than 50%, i.e.

$$\frac{B \cap B_0}{B \cup B_0} \geq 0.5 \qquad (1)$$

5. Create a PDF file containing **only** explicit answers to the following questions:

   (a) What is the reprojection error of your calibration matrix?
   Output: Real number from interval $[0; \infty]$.

   (b) You are given a calibration matrix `P` of your camera. Let us suppose that you increase the camera resolution two-times (i.e. `rows`$= 2\times$ `rows`, `cols`$= 2\times$ `cols`), while its world position is preserved. Calibration matrix of the new camera is $\mathtt{H} \cdot \mathtt{P}$. What is the matrix $\mathtt{H} \in \mathcal{R}^{3\times 3}$?
   Output: Real matrix $\mathtt{H} \in \mathcal{R}^{3\times 3}$.

   (c) Draw the ROC curve of (i) the provided RGB face detector (with parameters used for your final detector) and (ii) your final RGB-D-T face detector, with FP ranging from 0 to at least 1.
   Output: One graph with two ROC curves with FP on horizontal axis and TP on the vertical axis.

   (d) What is `TP` for `FP` $= 0.05$? (Highest achieved TP will be rewarded by a *bottled beer signed by all IRO teachers and personal congratulation of doc. Ing. Zlo, CSc. in front of the whole classroom*).
   Output: Real number from interval $[0; 1]$.

   (e) It is obvious that previously defined ROC curves can be identical or they can touch each other. Is it also possible that one ROC curve

strictly intersects the other (i.e. there is one FP-interval in which the first ROC has strictly lower FN and another FP-interval in which the second ROC has strictly lower FN)?
Output: Binary answer $\{\text{yes}, \text{no}\}$**.**

(f) Let us take your detector with fixed θ corresponding to `FP` $= 0.05$ on your testing set consisting of L images with fixed resolution (and low density of faces). Let us create a new testing set by adding L background images (images without faces) in the same resolution. How does the `FP`, `FN` change when detector is evaluated on the new testing set?
Output: Two real numbers N,M $\in [0; \infty]$ **corresponding to the claim that the resulting false positive is approximately N** $\times$ **FP and resulting false negative is approximately M** $\times$ **FN.**

6. Submit your work

   - Your work has to be uploaded to the Upload system. The report is submitted to task `answers`. All your codes should be zipped and submitted to task `code`. Do not include any binary files in the submitted package (BAG files, MAT files...).

4 Sensors and data description

The first sensor is the *ASUS Xtion PRO LIVE* providing standard color images (640x480 pixels) as well as depth images of the same size (a combination of these two images is denoted as RGB-D). These two images are already calibrated for you – there is both the RGB and depth information at all pixel coordinates. The depth information is expressed in *meters*; zero depth indicates no depth information at that particular pixel. *Depth* stands for the Z coordinate of the corresponding 3D point – a plane parallel with the image sensor will have a constant value indicated by the depth sensor.

The second sensor is a thermo-camera, namely *thermoIMAGER TIM 160* from MICRO-EPSILON. This camera captures infrared images (160x120 pixels), where the value of each pixel is the temperature of the corresponding surface observed by the camera (approximate temperature in our case, emissivity is not taken into account). In the provided dataset, the temperature is encoded into an integer (done by the thermo-camera), to obtain a real-world value, you need to follow this formula:

$$T_{pixel}[^\circ\mathrm{C}] = \frac{rawPixelValue - 1000}{10} \qquad (2)$$

where $rawPixelValue$ is the integer you receive in the thermo-image pixel. Make sure you convert the values into float or double types before division.

These two sensors are attached to a common holder (see Figure 2 for a rough idea, the actual setup differs), yet the exact configuration is not known (by the configuration, we mean the rotation and translation of camera optical centers). The only known parameter is the RGB-D camera *calibration matrix* K. This parameter is sufficient – combined with the depth information – to project each pixel of the RGB-D camera to corresponding 3D space coordinates. That is, for each color-depth image pair, you are able to get 640×480 points $X_{rgbd} = [x, y, z]^T$ that create a colored *point-cloud*. The images are recorded

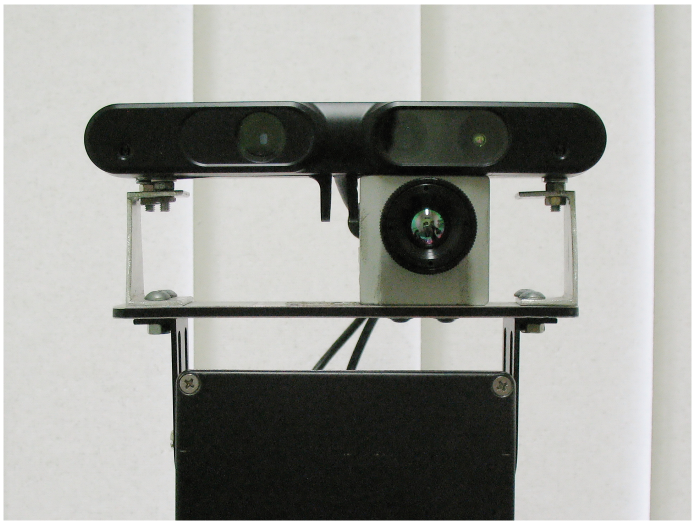

Figure 2: The two sensors involved: *ASUS Xtion PRO LIVE* (top), *thermoIMAGER TIM 160* (bottom).

using ROS, therefore the output provided to you are *BAG* files containing all the images as messages published at appropriate topics. The provided BAG files are:

- `iro_caminfo_2015.bag`: contains *camera_info* topics, you can find the K matrix for the RGB-D sensor inside
- `iro_calib_data_2015.bag`: image topics with the calibration sequence (the hot metal plate)
- `iro_train_data_2015.bag`: image topics with real and artificial faces; use these for detector design
- `iro_test_data_2015.bag`: same as the previous one, but ground truth will be provided for this dataset, use it to evaluate your result

The first two bag files are available on the course web site, the other two will be provided after the class where this assignment is presented – students of IRO will be involved in the recording of these datasets.

The bag files contain several topics, the *_caminfo_* bagfile contains three topics describing parameters of the cameras:

- `/camera/depth_registered/camera_info`: internal parameters of the depth sensor
- `/camera/rgb/camera_info`: internal parameters of the RGB camera
- `/thermo_cam/camera_info`: internal parameters of the thermo-camera

Please note that we do not guarantee correctness of the parameters with one exception, the K matrix of the RGB camera.

The other bagfiles consist of three image topics:

- `/sync/depth_registered/image_raw`: depth images; value of each pixel stands for depth in meters (z coordinate)
- `/sync/rgb/image_raw`: RGB color images
- `/sync/thermo_cam/image_raw`: thermo-camera images; pixel values encode measured temperature, refer to the sensor description in this section

Feel free to call `rostopic list -v` to get a detailed list of all topics available and `rostopic info TOPIC_NAME` to get more information about the chosen topic. Of course the relevant topics will only be published when the BAG file is played – you may find the `-l` option of `rosbag play` helpful when examining the topics.

TECHNICAL NOTE: The sensors generate images with rate higher than 80Hz. To keep the bag files reasonably large, we diminished the rate to approx. 3Hz by saving only a subset of all available images. Moreover, we chose the subset to be as synchronous as possible. Thus, you can consider the triplets of images to be synchronous keeping in mind that small delays may occur.

Doc. Zlo will choose one representative dataset from those created during the labs and make some poor assistant manually create ground truth. You will be asked to use this ground truth to evaluate your ROC curves and possibly other results. The ground truth will be saved in Matlab MAT file and will have the same name as the corresponding BAG file. Each row of the MAT file will contain a record of the type described by Table 1.

Timestamp.secs	Timestamp.nsecs	x_1	y_1	x_2	y_2

Table 1: The format of a row of the provided ground truth. First two columns together form the timestamp of the corresponding picture, and the x_1, y_1 values are coordintes of a face's top left corner, whereas x_2, y_2 represent the bottom right corner. For one timestamp, there can be more than one record (more faces in one picture) and there doesn't have to be any record for a given timestamp (no faces in the picture).

5 Calibration

You are given images captured by a Microsoft-Kinect-like sensor (see Figure 3, 1st and 2nd image) and by a thermo-camera (see Figure 3, 3rd image). Your task is to find transformation matrix `P` between the RGB image pixel coordinates and corresponding pixel coordinates in the thermo-camera image (for those pixels for which the calibration matrix exists). This procedure is known as calibration and an example result is depicted in Figure 4.

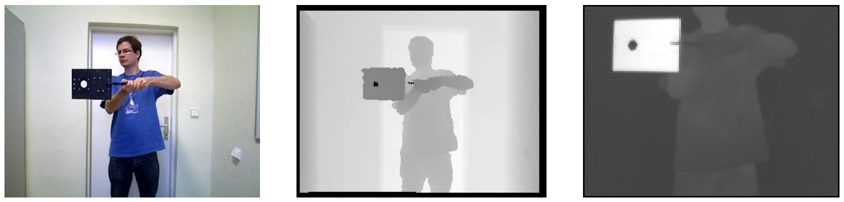

Figure 3: Three types of images provided in the *rosbag* file: a standard color image (left), a depth image (middle) and a thermo-cam image (right). The assistant is holding a hot metal sheet, that can be easily identified in all three image types.

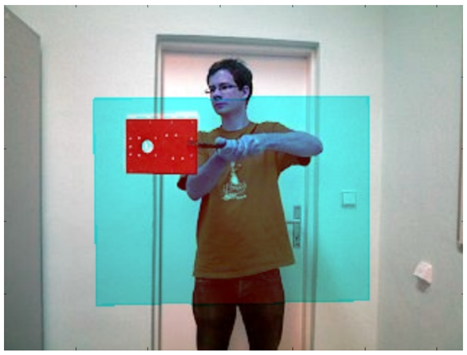

Figure 4: An example of the calibration result. Pixels corresponding to the thermo-camera field-of-view are colored according to their temperature (blue is cold, red is hot).

During the second IRO lecture and labs, the pin-hole camera model for projection of 3D points (in the world coordinate frame) into 2D camera plane was presented. The convention adopted by ROS differs in coordinate frame orientation, as described in Figure 5. Therefore, always visualize the data you work with!

Let us make world coordinate frame coincident with the coordinate frame of the RGB camera. Since RGB images are enriched by depth, we can easily obtain 3D homogeneous coordinates $X_{rgb} \in \mathcal{R}^4$ in such world coordinate frame. The

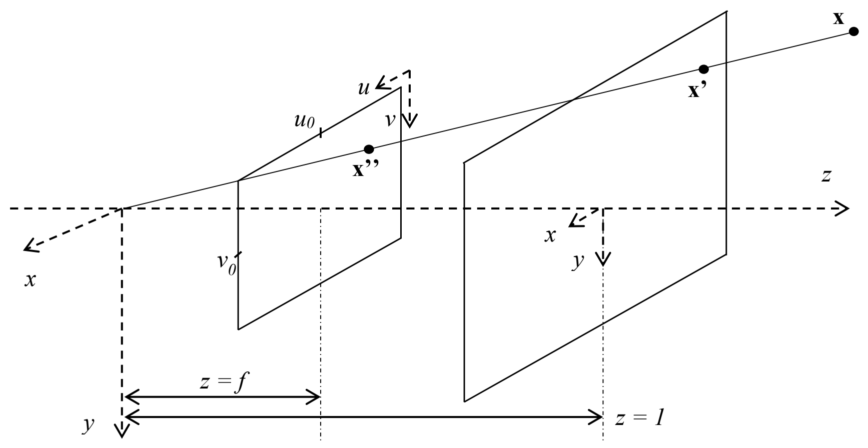

Figure 5: Projection from a 3D scene to an image plane by the ROS conventions.

projection of X_{rgb} into the thermo camera homogeneous coordinates $\mathbf{X}_{thermo} \in \mathcal{R}^3$ follows the standard pin-hole camera model

$$\mathbf{X}_{thermo} = \mathtt{P}\mathbf{X}_{rgb} \tag{3}$$

with the unknown projection matrix P, which projects 3D points expressed in the `rgb`-frame to the image plane of the thermo-camera. Your task is to formulate the search of vectorized matrix $\mathbf{p}$ as the solution of the *overdetermined* set of homogeneous linear equations $\mathtt{A}\mathbf{p} = \mathbf{0}$, and find its *non-trivial* solution $\mathbf{p}^* \in \mathbb{R}^n$ *in the least squares sense*; where non-trivial means $\mathbf{p}^* \neq 0$, overdetermined means that there are more independent equations than unknowns (i.e. dim rng($\mathtt{A}$) $\geq n$) and the least square sense means solution of the following optimization problem

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} \|\mathtt{A}\mathbf{p}\|, \text{ subject to } \|\mathbf{p}\| = 1. \tag{4}$$

Unfortunately, the optimization is strongly dependent on the origin and scale of the Euclidean coordinate system in which the correspondence pairs $\mathbf{x} = [x\ y\ z]^\top \in \mathbb{R}^3$ and $\mathbf{x'} = [u\ v]^\top \in \mathbb{R}^2$ (corresponding to homogeneous coordinates X_{rgb} and X_{thermo}) are expressed. To suppress such undesirable property, we *normalize* coordinate system by transforming points $\mathbf{x}$ to a new set of points $\mathbf{y}$ such that the centroid of the points $\mathbf{y}$ is the coordinate origin and the average distance from the origin is $\sqrt{3}$ (then, the "average" point's distance is 1). For example, points $\mathbf{y}_i$ are computed from points $\mathbf{x}_i$ as follows:

$$\mathbf{y}_i = \sqrt{3}\Big(\frac{\mathbf{x}_i - \overline{\mathbf{x}}}{\sigma}\Big), \tag{5}$$

where $\overline{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$ is the centroid and $\sigma = \frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}_i - \overline{\mathbf{x}}\|$ is the average distance. Similarly, points $\mathbf{y}'_i$ are computed from points $\mathbf{x}'_i$ as $\mathbf{y}'_i = \sqrt{2}\left(\frac{\mathbf{x}'_i - \overline{\mathbf{x}'}}{\sigma'}\right)$. Therefore, it is highly desirable to build matrix `A` from normalized points $\mathbf{y}_i$ and $\mathbf{y}'_i$, perform optimization and then incorporate compensation for the effect of normalization into resulting matrix P as follows:

$$P_{denormalized} = N_{therm}^{-1} P N_{rgbd} \tag{6}$$

where the inversion of N_{rgbd} compensates the effects of the 3D *RGBD* points normalization:

$$N_{rgbd} = \begin{bmatrix} \frac{\sqrt{3}}{\sigma} & 0 & 0 & -\overline{x}_1\frac{\sqrt{3}}{\sigma} \\ 0 & \frac{\sqrt{3}}{\sigma} & 0 & -\overline{x}_2\frac{\sqrt{3}}{\sigma} \\ 0 & 0 & \frac{\sqrt{3}}{\sigma} & -\overline{x}_3\frac{\sqrt{3}}{\sigma} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

where σ and $\overline{\mathbf{x}}$ are the translation and scaling factors from 5. The second matrix N_{therm} compensates the effects of the 2D thermo coordinates normalization:

$$N_{therm} = \begin{bmatrix} \frac{\sqrt{2}}{\sigma'} & 0 & -\overline{x'}_1\frac{\sqrt{2}}{\sigma'} \\ 0 & \frac{\sqrt{2}}{\sigma'} & -\overline{x'}_2\frac{\sqrt{2}}{\sigma'} \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

where σ' and $\overline{\mathbf{x}'}$ are the translation and scaling factors for 2D thermo image coordinates normalization. The resulting matrix $P_{denormalized}$ can be used to project 3D *RGBD* points into the 2D thermo image coordinates without any need of normalization.

5.1 Calibration dataset

To perform the calibration it is necessary to identify correspondences between the camera images. You are provided with a *BAG* file containing a calibration dataset, see Figure 3. There are three images that depict an assistant holding a hot metal sheet. The metal sheet is being moved continuously in front of the sensors so the field of view of the thermo-camera is roughly covered. The movement is repeated for several distances from the camera to ensure there are enough correspondences – the corners of the metal sheet, for example – to successfully perform the calibration.

Sample Python code that processes the topics stored in the BAG files has been prepared for you – download link is located in the semestral project section of the course web page [3]. Its purpose is to extract images stored in the bag files and export them to MATLAB arrays in *.mat* files. It is almost complete, you are supposed to finish it – in the source code, there is a part marked as `TODO:` with instructions.

6 Face detection

6.1 The face detector

The Zhu-Ramanan face detector is a detector based on mixtures of trees with a shared pool of parts, see [6] for details. In its full strength, it provides much more information about faces in the image than is needed for our task (e.g. landmark locations and their relative poses). So we slightly altered its output to just return the smallest bounding box containing all the returned landmarks for every detected face. Therefore, you can treat the detector as a classifier solving a binary decision task, with labels $\mathbf{y} \in \{\mathtt{F}, \mathtt{B}\}$ (i.e. Face and Background), on the set of all rectangles in the image.

6.2 Evaluation of detector quality

A graph depicting the relation between true positives $\mathtt{TP}(\theta) = 1 - \mathtt{FN}(\theta)$ and false positives $\mathtt{FP}(\theta)$ as a function of classification threshold θ is called *ROC curve*, see Figure 6. We also introduced ROC curves during the 6th lab (Bayesian decision tasks). ROC curve determines the quality of the classifier – for example an ideal classifier would have some threshold θ for which we would have $\mathtt{TP}(\theta) = 1$ and $\mathtt{FP}(\theta) = 0$, i.e. detect 100% faces and make no mistake. However, it is not the case in real world problems and we have to choose from the trade-off between TP and FP.

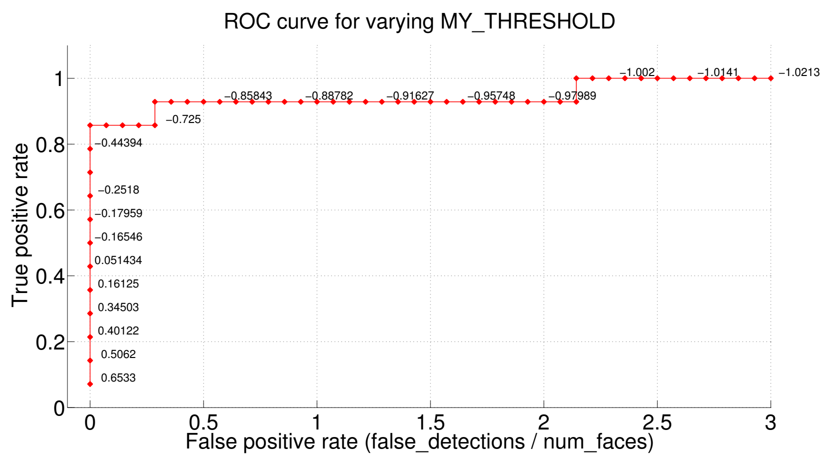

Figure 6: An example ROC curve. This one was evaluated on different data than you are provided with, so its shape may differ from the one you get.

6.3 Face detector usage

You find the face detector package on the semestral work page on CourseWare [5]. Just download it, unzip somewhere, and start Matlab in that folder. Follow the instructions in `README.md` to run (and eventually compile) the detector.

The detector has a single parameter you should treat as the θ parameter in the ROC curve – it is called `MY_THRESHOLD` and is located in the `load_model.m` script. This script also contains other parameters you can play with (e.g. the model to use), but evaluation of this task only expects you alter `MY_THRESHOLD`.

You can play with this parameter to obtain a ROC curve for the provided face detector. The easiest way is to run the detector with a very low threshold, record the resulting bounding boxes and scores, and then just compare these scores to various thresholds.

References

[1] Bugemos. Robotron. `http://bugemos.com/komiksy/Student/2-10.gif`.

[2] Martin Pecka and Vladimir Kubelka. Ros bag files with calibration and test images. `http://ptak.felk.cvut.cz/tradr/share/IRO/semestral_project/bag_files`.

[3] Martin Pecka and Vladimir Kubelka. Supplementary code for extracting images from bag files. `http://ptak.felk.cvut.cz/tradr/share/IRO/semestral_project/rec_imgs_to_mat_for_IRO.zip`.

[4] IRO Team. Task 2 cw page. `https://cw.fel.cvut.cz/wiki/courses/a3m33iro/semestralni_uloha`.

[5] Xiangxin Zhu and Deva Ramanan. Face detection, pose estimation, and landmark localization in the wild, source codes for iro course. `http://ptak.felk.cvut.cz/tradr/share/IRO/semestral_project/face_detector.zip`.

[6] Xiangxin Zhu and Deva Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2879–2886. IEEE, 2012.