Object detection and tracking with background modeling

Hoang Tran
Linköping University
hoatr725@student.liu.se

Matheus Bernat
Linköping University
matvi959@student.liu.se

Viktor Ivarsson
Linköping University
vikiv480@student.liu.se

Yunhee Kim
Linköping University
yunki172@student.liu.se

Alejandro García
Linköping University
alega478@student.liu.se

Abstract

Object detection and tracking for moving objects such as humans can be implemented by applying background modeling to segment out the foreground that is then tracked throughout a video sequence. Median filtering is concluded as limited compared to Gaussian mixture models as a background model. For the tracker, simple overlap scores matching proves to work well except for the case of occluded objects. For tracking occluded objects, keypoint matching can be used to estimate displacements.

1. Introduction

This report contains a description of the techniques and implementation details of a tracking system. The goal of the constructed system was to track objects – mainly humans – through image sequences, while addressing problems such as shadows, occlusion and spurious motion.

The assumptions made are that both the cameras and the background are static, and that the humans are moving on a flat ground plane.

1.1. Pipeline

The system is constructed in a modular manner, where each module has a well defined responsibility. The modules are: data reading, background modeling, foreground segmentation, tracking and evaluation. The data reading module is responsible for reading the image sequences. Background modeling creates a binary image representing foreground and background, and also filters shadows. Foreground segmentation removes noise and detects newly appeared objects. The task of the tracking module is to reassign id's of reappearing objects and this way track them through the image sequence. Lastly, the evaluation module evaluates the tracking in the current image sequence by comparing the achieved id's to the known ground truth.

2. Background Modeling

This module has two main responsibilities: to create a binary image representing foreground and background (aka *background model*), and to filter shadows. Creating a background model was initially done with Median Filtering, and later with the more complex method of Gaussian Mixture Models (GMM). This part of the pipeline largely followed J.Woods thesis [1].

2.1. Median Filtering

Median filtering is an algorithm that approximates the median for each pixel value over time. For each time t and for each pixel value x_t, the median m_t is updated by equation 1, where α is the learning rate.

$$m_{t+1} = \begin{cases} m_t + \alpha & \text{if } x_t > m_t \\ m_t - \alpha & \text{if } x_t < m_t \\ m_t & \text{if } x_t = m_t. \end{cases} \quad (1)$$

The background is then segmented by thresholding the deviation from the median, in other words: $|x_t - m_t| < T$ where T is the threshold value. While implementing this for gray-scale images, there were a couple of problems since some objects in gray-scale are quite similar to the background, hence they will not get differentiated from the background. In Figure 1 and Figure 2, we present the comparison between gray-scale median filtering and three channel RGB median filtering.

One of the choices we had to make was how to initialize the median. In our case we decided on initializing the median as the values in the first frame. This created some problems during the first few frames as the first frame usually included some foreground as well as background. This led to the median fitting to the foreground object and hence, once the object moved, the background would be recognized as foreground, as we see in Figure 3.

Figure 1: Example of median filtering using gray-scale images.

Figure 2: Example of median filtering using RGB channels in images.

Figure 3: Example of problems in initialization for median filtering.

To combat this, α was set quite high to let the median adapt away from the first frame faster. This led to problems when one object stayed in the same pixel area for a long time as the median would converge to their values. In a real world scenario this would not be a problem as a low α would model the median quite well over a long time. Due to the limited potential of median filtering we moved on to try modeling the background with GMM.

2.2. Gaussian Mixture Modeling

Mixture model is a probabilistic model used to identify clusters from the original data. In this project, Gaussian function is used as a mixture component. Each pixel has its own one-dimensional GMM and it is online-updated.

For x_t denoted a pixel value at time t, its GMM is updated. The way it is updated depends on whether there is any mixture component matched with x_t. To do so, the distance between x_t and the K number of Gaussian mixture components needs to be estimated and this can be done by using Mahalanobis distance:

$$d_k^2 = \frac{(x_t - \mu_k)^2}{\sigma_k^2}, \forall k \in \{1, 2, ..., K\}. \tag{2}$$

With the estimated distance d_k, evaluating if x_t is matched with the K-th Gaussian or not is as follows given $\lambda = 2.5$:

$$match_k = \begin{cases} 0 & \text{if } d_k \geq \lambda \\ 1 & \text{if } d_k < \lambda \end{cases}, \ \forall k \in \{1, 2, ..., K\}. \tag{3}$$

If there is no mixture component which is matched with x_t, a new mixture component with $\mu = x_t$ and $\sigma^2 = \sigma_{init}^2$ is going to be added to the mixture model with a small mixture density w_{init}.

If there are more than one mixture component matched with x_t, then one mixture component among them should be selected because only one mixture component is updated at a time.

To select the mixture component that is going to be updated, let m be used to indicate m-th mixture component that is matched. m is computed as follows:

$$m = \arg\max_c \frac{w_c}{\sqrt{\|\sigma_c^2\|}}, \forall c \in \{1, 2, ..., K\}, \{match_c == 1\}. \tag{4}$$

In order to estimate the parameters for the selected m'th Gaussian mixture component with respect to x_t, a classic approach could have been used, namely the expectation maximization (EM) algorithm. However, to implement the exact algorithm is cumbersome so we used a more suitable algorithm which is derived from the EM algorithm. The algorithm is called the Stauffer and Grimsons algorithm[2]. The Stauffer and Grimsons algorithm is based on the EM algorithm but with much simpler computations that were adapted specifically for the background mixture models that can be updated online. The update for m-th mixture component will be done as follows (where α denotes the learning rate):

$$\begin{aligned} w_m &= (1 - \alpha) w_m + \alpha; \\ \rho_m &= \frac{\alpha}{w_m}; \\ \mu_m &= (1 - \rho_m)\, \mu_m + \rho_m \cdot x_t; \\ \sigma_m^2 &= (1 - \rho_m)\, \sigma_m^2 + \rho_m\, (x_t - \mu_m) \circ (x_t - \mu_m)\,. \end{aligned} \tag{5}$$

The mixture densities should be always summed up to 1. Therefore after updating w_m, the mixture densities have to be re-normalized to be summed up to 1. However, there is a risk when two new mixture components are added in a row from x_t to x_{t+1} due to the re-normalization step. The

mixture densities will become $w_k < w_{k+1}$. In order to avoid this, the re-normalization is only done for all mixture densities except m-th mixture component.

The mixture components with their mixture density, mean and variance should always be kept sorted in a descending order corresponding to $(\frac{w_1}{\sqrt{\|\sigma_1^2\|}}, \frac{w_2}{\sqrt{\|\sigma_2^2\|}}, ... \frac{w_K}{\sqrt{\|\sigma_K^2\|}})$ where K is the number of mixture components that the GMM is composed of.

Once the sorting is complete the threshold mixture component is calculated as

$$B = \arg\min_b \left(\sum_{k=1}^{b} w_k > T \right). \quad (6)$$

If any mixture component from the first to the B-th matches with x_t, then x_t is determined to be background; otherwise, if there is no match, then x_t is classified as foreground.

2.2.1 Pre-processing

The background segmentation has some issues to be handled. For instance, some pixels can be classified as foreground if the camera is not perfectly stationary but has a minor movement, although indeed, they are background. Another example is a tree with leaves that are swaying in the wind. It is true that something is going on in regards to the pixels where the leaves are. But in this project those kind of situations are managed by using low-pass filtering. As the result shown in Figure 4 and 5, with this pre-processing of the data, we can exclude some of the spurious background motions.

2.2.2 Comparison to median filtering

By comparing the results using GMM to those with median filtering, we concluded that GMM does not depend as much on an optimal α as median filtering is. There is a dilemma using median filtering when it comes to deciding α as shown in Figure 6. The result using GMM can be viewed in Figure 7 and as can be seen, the problem of deciding alpha is absent. In other words, the detected foreground objects do not change into background as quickly as in median filtering and the traces of movement from objects or people from the first frame and onward for few frames adapt to the change in intensity over time quicker.

2.3. Shadow Suppression

Shadows are detected as part of the motion in the image, so they end up interfering in the background modelling and are undesirably set as part of the foreground.

Figure 4: Original image (left) and GMM of the background without pre-processing (right). Left image contains much noise.

Figure 5: Original image (left) and GMM of the background with pre-processing (right). The noise is decreased, especially on the edges of windows on the top left of the image.

2.3.1 Use of the HSV color space

After the background model has created a binary image representing foreground and background it is possible to filter potential shadowed pixels by comparing the HSV field values. The HSV (Hue, Saturation, and Value) model is a generalization of the RGB model; with it we can carry out the hypothesis that a shadowed pixel will generally decrease in saturation and lightness value while the hue will remain relatively constant. The following shadow mask can be applied:

$$sp = \begin{cases} 1 & \text{if } \alpha \leq \frac{x_{f,L}}{x_{b,L}} \leq \beta \\ & \wedge x_{f,S} - x_{b,S} \leq \tau_S \\ & \wedge |x_{f,H} - x_{b,H}| \leq \tau_H \\ 0 & \text{otherwise ,} \end{cases} \quad (7)$$

where x_f is a current foreground pixel and x_b is a background pixel in the previous frame, and L denotes lightness in the HSV channel, S, saturation, H, hue. Pixels that do not meet the conditions will be set as background in the binary image (as zero). The rest are threshold hyperparameters, which after several tests have been set to α = 0.3, β = 0.9, τ_s = 0.3, τ_H = 0.5 to give reasonable results. Note that this method can split big objects into smaller ones, which eventually will make many bounding boxes to become true positives and false positives.

Figure 6: Original RGB image (left); After median filtering with $\alpha = 6$ (middle); After median filtering with $\alpha = 1$ (right). The red rectangles show that the pixels are changed into background. The green rectangles contain traces of people due to their movement made from the first frame and onward for a few frames.

Figure 7: Original RGB image (left) and GMM with $\alpha = 0.01$ (right).

Figure 8: Original RGB image (left) and GMM (right). Red rectangles show some parts of the foreground objects are lost. Blue rectangle shows the foreground object is detected without any lost part.

2.3.2 Horprasert method

The statistical approach for real-time background subtraction and shadow detection from Thanarat Horprasert, David Harwood and Larry S. Davis exploits the facilities of the RGB color space. Consists on measuring the distortion of a pixel's RGB color value in a current image that we want to subtract from the background, from the pixel's expected RGB color in the reference or background image. This is simply done by decomposing the distortion measurement into two components, which they call brightness distortion

Figure 9: Original MOT-09 video.

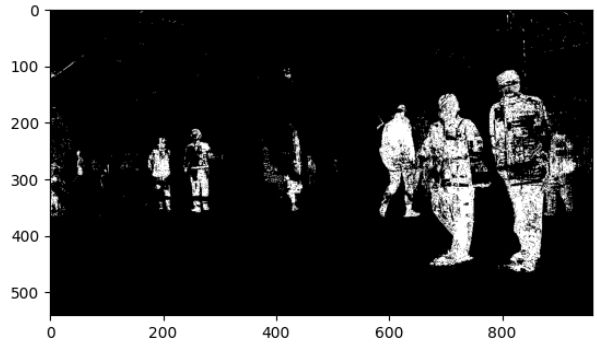

Figure 10: Example of MOT-09 without shadow supression.

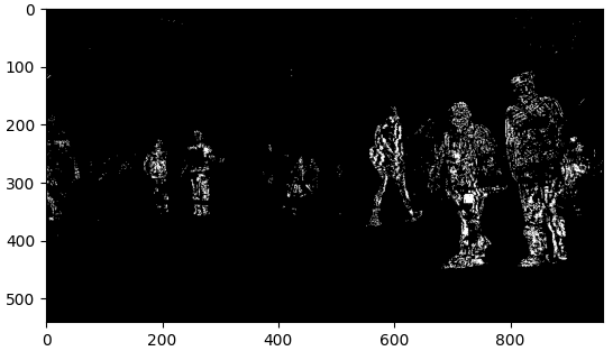

Figure 11: Example of MOT-09 with shadow supression.

and chromaticity distortion. Their model is based on the empirical observation that shadowed pixel values tend to follow straight lines connecting the values and the origin in the RGB color space. Basically brightness distorsion would be the deviation of a pixel value in the background image that brings the observed color close to the expected

chromaticity line, and the chromaticity distorsion is pixel values deviation perpendicular to the distance between the observed color and the expected chromaticity line.

3. Foreground Segmentation

This module performs noise removal followed by marking the connected regions as objects, and assigning those objects a new id. The removal of noise is done through morphological operations of the image, and the detection of regions as likely objects is done through connected component analysis.

3.1. Morphological image processing

In order to reduce noise from the background modeling we use morphological image processing. There are two different basic morphological operations dilation and erosion. When dilation is performed objects are expanded, while erosion will shrink objects. There is compound operations such as opening and closing which is a combination of the two basic operations. Opening is defined as erosion followed by dilation while closing is defined as dilation followed by erosion. The opening operation can be used to reduce the amount of noise such as small objects and spurs on larger objects in a binary image. The closing operation can be used to fill holes in objects.

For the basic morphological operations a structuring element is needed, it's possible to use many different shapes and sizes. We went with a 3x3 squared structuring element. To perform erosion with a 5x5 structuring element all we had to do was perform erosion twice with a 3x3 kernel, three times for 7x7 and so on. The result of this part of the pipeline can be seen in Figure 12 [3].

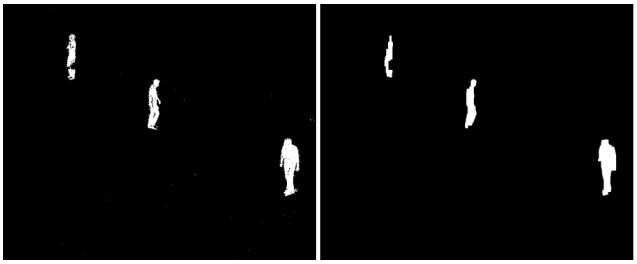

Figure 12: Original binary image (left) and image after the morphological image processing (right).

3.2. Connected component labeling

The image where we want to label objects is a binary image, therefore we chose to use connected component labeling (CCL) to identify objects. We looked at other options such as the watershed algorithm [4] but ultimately decided to go with CCL. There are many implementations of the CCL, we used the OpenCV default algorithm for 8-connectivity called Block-Based CCL with Decision Trees. Basically, the algorithm passes over the image and gives each 8-connected object a unique label [5].

With the image labeled we can extract the bounding boxes and apply them to the original image, this is shown in Figure 13.

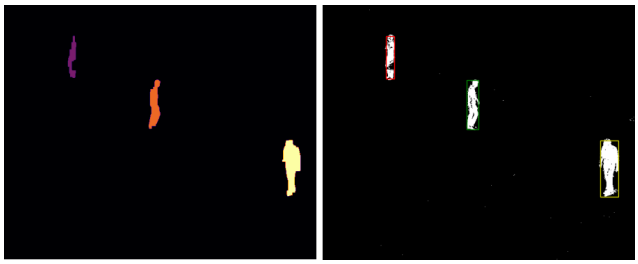

Figure 13: Labeled image (left) and the original binary image with bounding boxed on top (right).

4. Tracking

The task of the tracking module is to track objects through the given image sequence. By tracking, what is meant is that an object will be given the same id throughout the sequence. Among the techniques used in tracking are Kalman filter for prediction of currently unseen objects [6] (Section 4.1), object matching through pixel overlap (Section 4.3), and object matching through matching of key points (Section 4.2).

In short the tracker we develop works like this:

1. Receive identified object bounding boxes for frame f_{i+1}.
2. Predict the movement of bounding boxes for frame f_i
3. Calculate the overlap of the bounding boxes for frame f_{i+1} and predicted bounding boxes of framef_i.
4. Deal with seemingly merged bounding boxes by removing them from f_{i+1} (see Section 4.3).
5. Assign the id's from f_i to bounding boxes in f_{i+1} for matched objects and set unassigned bounding boxes as missing.
6. Use key point matching to estimate the movement of missing bounding boxes and add them to frame f_{i+1}.

4.1. Kalman Filtering

To predict the position of an object, a constant velocity model is used for the Kalman filter. In two dimensions, this is given by the state vector $s = (p_x, p_y, v_x, v_y)$, where

p_x, p_y are the x and y coordinates, while v_x, v_y are the respective velocities. Given in state-space form, this gives equation 8, where T is the sample time and v_t and e_t are assumed to be zero-mean Gaussian noises with covariances Q and R respectively:

$$s_{t+1} = \begin{pmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} s_t + \begin{pmatrix} T^2/2 & 0 \\ 0 & T^2/2 \\ T & 0 \\ 0 & T \end{pmatrix} v_t;$$
$$\mathrm{y} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \mathrm{s}_t + e_t. \tag{8}$$

Since the positional values of an object are the pixel coordinates of the top left corner of the bounding box, some problems occur. If the bounding box suddenly changes its size, it is quite possible that the position of the top left corner is radically changed compared to the predicted state. However, we did not encounter significant difficulties with this. Tracking the center of the bounding box gave similar problems, so the original tracking was kept.

A Kalman filter could also be used to track the changes in the bounding box dimensions, but since they could be radically changing between the frames, the group decided that it was unnecessary. Tinkering with the parameters of the Kalman filter such as the process and measurement noise covariances did not yield big improvements over different datasets so we ended up deciding on having the noise covariances both initialized to 1. The covariance matrix P_0 was initialized to 1 for the positional parameters and 0.1 for the velocities.

4.2. ORB algorithm for key point detection

In order to deal with objects being occluded and thus not being identified as unique by foreground segmentation, tracking of keypoints was implemented. The ORB-algorithm from OpenCV is the one we used. It is similar to the SIFT and the SURF algorithm, with the advantage of being free to use and having improved performance [7].

In summary, the implementation was the following: for a frame t, identify the keypoints in the frame that lie within the foreground of the binary image. In the same way, detect the interesting keypoints in frame $t + 1$. Then, through another OpenCV function (Flann-matcher), match the keypoints from frame t to the keypoints in frame $t + 1$. See the matching of the keypoints in Figure 14. Finally, the matches were subsequently saved to a scores table for doing object matching, as described in Section 4.3.

4.3. Matching

New objects are compared to old objects using certain measures. For the case where the matching score is high enough, the objects are "matched", i.e. the new object is assigned the id of the old object. The measures used to compare objects are pixel overlap between the bounding boxes of the objects, and matchings of the keypoints.

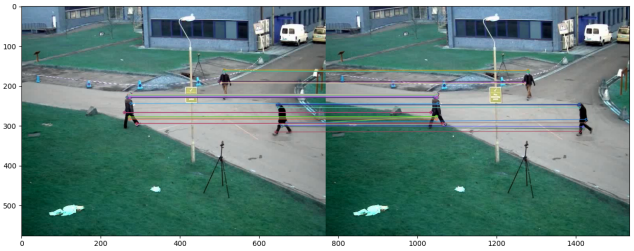

Figure 14: Keypoint matchings between two subsequent frames.

The first step of the procedure is to construct a so called *scores table*, where the existing objects from the most recent frames are in the rows, and the objects of the current frame in the columns. For instance, the element ij of the table contains a combined value of the objects bounding box pixel overlap, and the number of keypoint matches between object i and new object j. Both the pixel overlap and the number of keypoint matches are normalized before being added up together. In the sum of pixel overlap and keypoint matches, one can weigh them differently depending on how reliable a measure is in comparison to the other. The second step of the procedure is, for each new element j, to choose the element i such that $i = \mathrm{argmax}_i\, table(i, j)$. Finally, each new object is assigned the matched object id.

An approach taken in order to avoid smaller objects from being merged to bigger sized objects was to pop the greater objects from the list of existing objects. A *bigger sized object* was defined as an object that had large enough score with more than one new object, for example if two boxes had a 40% overlap with a new box. This generally meant that the smaller objects had been connected as one larger object by foreground segmentation so this object was removed. Then occlusion management dealt with the tracking of the box id instead as seen in section 4.4.

4.4. Occlusion management

One method to improve tracking when objects are occluded was to track the objects with missing matches with the Kalman filter until it either got a match again and therefore a measurement update or reached a maximum amount of frames missing, where we would then remove the object completely. This had the problem of not being very accurate most of the time. For model cases, for example, when a person walks behind a sign with constant speed, it worked flawlessly, but in most other cases, it tended to not be very accurate. Since the tracking was done on the top left corner

of the bounding box, it was not very accurate in predicting the movement. It also tended to track small objects that were mislabeled quite harshly and there would sometimes be boxes flying around, so we scrapped this idea.

The other method we used instead was to use the keypoint matching to track occluded objects. Once a box did not have a match, it was assigned as missing. Then for all the missing boxes we used key point matching for the scene inside the box. These keypoints then gave a bunch of vectors that could be seen as the movement from one frame to the other for this object. Taking the median of these vectors to prevent outliers and mismatches, the missing object would then be moved by this vector. This proved to give a better result in dealing with occlusions. However, there are still some problems with it. Sometimes an object would be split into a large box and a smaller one for the first frame, and in the next, it would be fully connected. This then resulted in the small box going missing, and due to the keypoint matching, it would stick to this object, even though the object already had a full bounding box. This method also cannot deal with full occlusion as there are no key points to gather in that case.

5. Results

When evaluating the designed tracker, we calculated several evaluation metrics. First, we created a score table that calculated the Jaccard index for every detection compared to the ground truths. Then we solved the assignment problem with *linear_sum_assignment* from *scipy.optimize*. We defined a true positive detection (TP) as a detection that has a Jaccard index of at least 0.2 with its associated ground truth bounding box. A false positive detection (FP) is defined as a detection with a Jaccard index of less than 0.2 or a detection that doesn't have an associated bounding box. Lastly a false negative (FN) is a ground truth bounding box that has a Jaccard index less than 0.2 or that is missing an associated detection. The results are shown in Table 1.

Sequence	TP	FN	FP
MOT17-02	1703	28300	371
MOT17-04	7342	100663	508
MOT17-09	2392	8019	421

Table 1: Basic evaluation metrics.

With these basic metrics we calculate the *precision* as $\frac{\sum TP}{\sum TP + \sum FP}$ and *recall* as $\frac{\sum TP}{\sum TP + \sum FN}$. We also calculated the amount of times the *indentity switches* for the detections aswell as the *average true positive overlap* as $\frac{\sum Jaccard\_index(TP)}{length(TP)}$. The results are shown in Table 3.

As an example we can see the comparison between the ground truth and our tracker in Figure 15. We can see

Sequence	Precision	Recall	id switches
MOT17-02	0.72	0.05	322
MOT17-04	0.77	0.06	701
MOT17-09	0.81	0.22	280

Table 2: Evaluation results.

Sequence	Precision	Recall	id switches
MOT17-02	0.82	0.06	86
MOT17-04	0.94	0.07	182
MOT17-09	0.85	0.23	155

Table 3: Evaluation results.

that our tracker detects large objects well, but it misses the mother and child at the entrance of the store to the right.

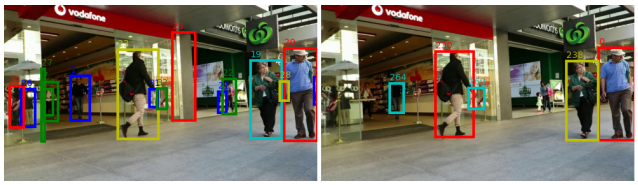

Figure 15: Sequence MOT17-09, ground truth (left) our detections (right).

6. Conclusions

Due to the nature of the data sets, the background modeling was better handled by GMM compared to median filtering. To further improve the background modeling, one could consider to pre-process the data. Without preprocessing the data used with GMM, a problem can arise if x_t (that is supposed to be classified as foreground) is matched with the background Gaussian. In this scenario, x_t will be classified as background. One possible way to handle this issue is to manipulate the foreground object intensities.

For tracking, overlap scoring was quite effective and the prediction step using a Kalman filter slightly improved it. However, for the occlusion management, a Kalman filter that assumes constant velocity is not robust enough to stand up for the challenge. In this case the keypoint matching gave better results, but would fail once an object was fully occluded. The solution to the merged boxes problem was also quite primitive and a more robust solution should be researched on. A research paper *Globally Optimal Solution to Multi-Object Tracking with Merged Measurements* [8] deals with this problem as well as improves on the matching algorithm by using the Hungarian algorithm [9].

References

[1] J.Wood. Statistical background models with shadow detection for video based tracking. 2007.

[2] Chris Stauffer and W Eric L Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings. 1999 IEEE computer society conference on computer vision and pattern recognition (Cat. No PR00149)*, volume 2, pages 246–252. IEEE, 1999.

[3] Auckland university, morphological image processing. https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm. Accessed: 2021-03-24.

[4] A. Bieniek and A. Moga. An efficient watershed algorithm based on connected components. *Pattern Recognition*, 33(6):907–916, 2000.

[5] OpenCV, structural analysis and shape descriptors. https://docs.opencv.org/4.5.1/d3/dc0/group__imgproc__shape.html#gae57b028a2b2ca327227c2399a9d53241. Accessed: 2021-03-24.

[6] M. Millnert F. Gustafsson, L. Ljung. *Signal Processing*. Studentlitteratur AB, 2010.

[7] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[8] Rui Caseiro João F. Henriques and Jorge Batista. Globally optimal solution to multi-object tracking with merged measurements. In *2011 International Conference on Computer Vision*, 2011.

[9] H.W. Kuhn. The hungarian method for the assignment problem. In *Naval Research Logistics Quarterly 2*, pages 83–97, 1955.