Complexity Analysis of Multi Scale and Multi Channel Cascade Classifier Detector

Ramkumar Natarajan[*]

1 Summary of Discussions

This report is concentrated on the complexity analysis of the multi-scale and multi-channel decision tree based detector. The decision tree learned during the training stage is evaluated on the test image sets using a sliding window approach for multiple sizes of the window(scale invariance). These analysis and experiments are a part of the feature extraction in the grand plan of developing landmark based data association for incremental Smoothing and Mapping (using Bayes tree) (iSAM2)[5] at real-time by providing efficient access to marginal covariance matrices. To recollect a better context, I have listed the required background and literature survey in my previous report which could portray where I am coming from.

Following this section, I would shed some light on few relevant works that have extended the original work of Viola Jones in different contexts such as improving false alarm rate, modifications in Haar features, weighting schemes in Adaboost etc. There is no particular work that specifically deals with a detailed theoretical complexity analysis of the final detector although there has been empirical analysis both in the original paper[3] and [4]. These theoretical results would be followed by some of the experiments and new directions that summarize the planned future work. Finally, I have mentioned some of my random thoughts to explore, that occurred during the course of the analysis.

2 Papers Read

The Viola Jones rapid object detection[3] is a gold standard in the real-time object detection literature using HAAR features. The integral image for feature computation, Adaboost for feature selection and an attentional cascade for efficient computational resource allocation are the three key components behind achieving very high processing speed although the performance is same as the previous complex single stage classifiers. Following that, Lienhart *et. al,*[4] did an empirical analysis of the original work and also introduced 45^o rotated HAAR features and verified that the Gentle Adaboost performs better than Real and Discrete Adaboost in terms of detection accuracy. A complete algorithmic description that explains the implementation logic behind Open Computer Vision

[*]rnatarajan@wpi.edu

library[14] is provided by [13]. Almost all the previously existing analysis deal only with the complexity of training the Cascade Classifier and just mention that the detection rate is very fast. To the best of my knowledge, this is the first analysis of the computational complexity of the Cascade Classifier object detection as a function of various tunable parameters.

3 Theoretical Results

The trained classifier performs a sliding window search over the target image for a fixed window size during every iteration. The set of tunable parameters for the Cascade Classifier Detector are:

- The minimum window width and height of search, $W_{min} = \{w_{min}, h_{min}\}$
- The maximum window width and height of search, $W_{max} = \{w_{max}, h_{max}\}$
- The scale factor by which the window size grows from minimum to maximum, γ_0
- The minimum number of detections in the neighborhood to be selected as a true positive, β

The classifier is a cascaded set of decision trees and the number of such decision trees and the depth of each tree also influences the total time. However, since this value is constant for a given classifier it cannot be considered as a part of tunable parameters. Nevertheless, I have mentioned an upper bound on the number of evaluations of the entire decision tree over a given image.

REMARK: The classifier cannot detect target objects smaller than the trained window size.

Search window size and scale factor

The maximum and minimum window sizes and the scale factor are interrelated and hence considered together for analysis. The size of the window along both the width and height is increased from the minimum to maximum value at the scale factor ratio. This is done to stick to the constant aspect ratio at which the classifier is trained. Let w_t and h_t be the training width and height. The set of all scale factor values is given by

$$\Gamma = \{\gamma \mid w_t.\gamma \in [w_{min}, w_{max}] \cap h_t.\gamma \in [h_{min}, h_{max}], \gamma = \gamma_0, \gamma_0^2, ..., \gamma_0^n\} \ \forall \ n \in \mathbb{N} \tag{1}$$

where γ_0 is the initial value of scale factor which is also equal to the one input by the user. Generally, it is advised that γ_0 be in the positive neighborhood of 1, $\gamma_0 \to 1_+$ to get a finer search window size. The set of scale factors form a geometric progression with γ_0 being the common ratio between subsequent values. Now the set of image resolutions for which the integral image is to be calculated is given by

$$I = \{(x, y) | x = \frac{w_t}{r}, y = \frac{h_t}{r}\} \ \forall \ r \in \Gamma \tag{2}$$

The integral image is calculated for $\mid I \mid$ number of times. The number of elements in the set I is actually the number of terms in the G.P represented by Γ which can be calculated as follows.

Let i denote the index of the terms in a G.P formed by initial value 1 and common ratio γ_0. Then the least and greatest value of i that scales w_t to the range $[w_{min}, w_{max}]$ are

$$i_l^w = \left\lceil log_{\gamma_0}\left(\frac{w_{min}}{w_t}\right)\right\rceil \tag{3}$$

$$i_g^w = \left\lfloor log_{\gamma_0}\left(\frac{w_{max}}{w_t}\right)\right\rfloor \tag{4}$$

where the subscripts l and g represent the least and greatest index values respectively and $\lfloor . \rfloor$ and $\lceil . \rceil$ are the least and greatest integer functions.

Similarly, the least and greatest value of i that scales h_t to the range $[h_{min}, h_{max}]$ are

$$i_l^h = \left\lceil log_{\gamma_0}\left(\frac{h_{min}}{h_t}\right)\right\rceil \tag{5}$$

$$i_g^h = \left\lfloor log_{\gamma_0}\left(\frac{h_{max}}{h_t}\right)\right\rfloor \tag{6}$$

To stick with the aspect ratio, only the terms that are common between the set of width and set of height ratios are considered. The number of common scale factors K which is same as $\mid \Gamma \mid$ and $\mid I \mid$ is in turn given by

$$K = \mid \Gamma \mid = \mid I \mid = min\{i_g^w - i_l^w, i_g^h - i_l^h\} \tag{7}$$

$$K = min\left\{\left\lceil log_{\gamma_0}\left(\frac{w_{max}}{w_{min}}\right)\right\rceil, \left\lceil log_{\gamma_0}\left(\frac{h_{max}}{h_{min}}\right)\right\rceil\right\} \tag{8}$$

Using the change of base rule,

$$K = min\left\{\left\lceil \frac{ln\left(\frac{w_{max}}{w_{min}}\right)}{ln\ \gamma_0}\right\rceil, \left\lceil \frac{ln\left(\frac{h_{max}}{h_{min}}\right)}{ln\ \gamma_0}\right\rceil\right\} \tag{9}$$

All example sub-windows used for training were variance normalized to minimize the effect of different lighting conditions. Normalization is therefore necessary during detection as well. The variance of an image sub-window can be computed quickly using a pair of integral images. Recall that $\sigma^2 = m^2 - \frac{1}{N}\sum x^2$, where σ is the standard deviation, m is the mean, and x is the pixel value within the sub-window. The mean of a sub-window can be computed using the integral image. The sum of squared pixels is computed using an integral image of the image squared (i.e. two integral images are used in the scanning process). During detection the effect of image normalization can be achieved by post-multiplying the feature values rather than pre-multiplying the pixels. Thus the integral image is calculated $2K$ number of times. The integral image is a summed area table data structure which is calculated efficiently using a recursive algorithm. Though the computation of summed area table depends on the resolution of the image, the task of evaluating the intensities over any rectangular area requires only four array references. This allows for a constant calculation time that is independent of the size of the rectangular area. Also for multichannel images with M channels, the number of integral image computations is $2MK$ times. This factor M is the *only* significant difference between multichannel cascade classifier detector and grayscale detector.

Minimum number of neighbors

The element set containing multiple detections of differently sized objects in target images are split into equivalency classes(clustering). We use a first order logic with a predicate that relates the objects in the set based on the test of similarity of rectangles. The running time of the clustering algorithm is actually *independent* of the threshold β that corresponds to the minimum number of neighbors jointly satisfying the predicate. The logic implements an $\mathcal{O}(N^2)$ algorithm for clustering a set of N elements into one or more equivalency classes [1] as described using disjoint-set data structure representation [2]. For every element in the disjoint-set data structure of size N^2 the predicate is evaluated which returns either true or false.
The predicate evaluates similarity of rectangles by taking any two elements from the input set along with the internal parameter ϵ, $0 < \epsilon < 1$.

Two rectangles are similar if their sides are proportional.

This is checked by finding if all the sides of one rectangle is within $\Delta = \epsilon * (min(w_1, w_2), min(h_1, h_2))/2$ of the other rectangle.
Two rectangles which are within a threshold Δ on all the four sides, for a Δ formulated using minimum of width and height of the participating rectangles, are also within a threshold Δ for a Δ formulated using the maximum of their width and height.

Hence a Δ formed by the minimum of the their sides is a more restricted metric. The reason for formulating Δ is to account of uncertainty in similarity. The representative rectangle for all the equivalency class in the disjoint-set data structure is obtained by averaging all the constituent rectangles which is in turn returned as the detected rectangle.

Depth of the decision tree and target image resolution

Though it looks like the detector is evaluated for various window sizes from minimum to maximum size, algorithmically the target image is shrunk from its original resolution to various sizes depending upon scale factors such that the objects of interest of sizes between minimum and maximum window size falls inside the constant window size. In other words, we are decreasing the size of the target image itself rather than increasing the size of the search window and rescaling the features appropriately. We take this rather long route of rescaling the image because the detection could be done using the same window size by which the classifier is trained. Although the values of the feature learned at every level of the classifier is normalized by the area of the feature, no clear explanation about guarantees on generality of learned classifier value for arbitrary rescaling of the feature considering linear interpolation of image and truncation/roundoff errors in feature rescaling is given in the original paper[3]. Obviously, by fractional rescaling the new correct positions become fractional. A plain vanilla solution is to round all relative look-up positions to the nearest integer position. However, performance may degrade significantly, since the ratio between the two areas of a feature may have changed significantly compared to the area ratio at training due to rounding. One solution is to correct the weights of the different rectangle sums so that the original area ratio between them for a given haar-like feature is the same as it was at the original size[4]. Nevertheless, by doing this reverse operation we end up linearly decimating the target image $\mid \Gamma \mid$ times and calculating integral image for each of them. From a memory perspective, the integral image for all the scales of the target image are computed and stored in a continuous serial buffer of size

$$M \times \sum_{\gamma \in \Gamma} \left(\frac{W}{\gamma} + 1 \right) \left(\frac{H}{\gamma} + 1 \right)$$

where $W \times H$ is the resolution, M is the number of channels of the target image and Γ is the set of scale factors.

For a decision tree of S stages and depth d_s per stage where $s \in S$, an upper bound on the number of evaluations of all the stages of the decision tree is given by

$$\leq \sum_{\gamma \in \Gamma} \left\{ \left(\frac{W}{\gamma} - w_t + 1 \right) \left(\frac{H}{\gamma} - h_t + 1 \right) \times \sum_{s \in S} d_s \right\} \tag{10}$$

where $W \times H$ is the resolution of the target image, $w_t \times h_t$ is the training resolution and Γ is the set of all scale factors.

Note: An implementation detail is that, for integral image resolutions whose corresponding scale factor is less than 2, the cascaded decision tree is only evaluated over every other pixel along width and height. Also, the entire continuous buffer of integral image across various resolutions is striped into memory chunks of size

$$\delta_i = 32 \times \frac{H}{\gamma_i W} \quad \forall \gamma_i \in \Gamma \tag{11}$$

where 32 is code specific. In total, there are $\sum_i \delta_i$ chunks which are processed in parallel using TBB multi-threading library.

4 Experiments run

The so far described Haar Cascade Classifier[3] is used to detect the product labels in an online fashion. The biggest advantage of the Cascade Classifiers are their rapid speed of detection despite very long training time. The complexity analysis summarised in this report gives idea on the sensitivity of each classifier parameter on the detection time. All the experiments are conducted using *Gaeta+*, a BossaNova Robotics proprietary robot with the images captured from the camera stack mounted on it. The camera stack contains 11 cameras arranged vertically to cover the entire span of a typically setup store aisle shelf as shown in Fig.[fig]. The camera stack is also supported by a vertical LED array to capture bright and sharp images at very high shutter speeds while the robot is moving. The images are projected on a plane fitted in the pointcloud(pointcloud data is collected by a vertically mounted LIDAR) constructed at the end of the aisle. This generates a panorama of the entire aisle. The product labels are detected on this panorama. It is imperative to detect the labels very quickly so as the solutions such as out-of-stock of store products could be reported readily. To give a context about the average size of the target object, the robot tracks the aisle sections from a fixed distance that falls in the depth of field of the camera. However, there are always cases in which the depth of various sections and product label holding pegs in the aisle are different and hence the target object size may vary significantly. A conservative search size may detect all object instances but it is time critical to have a tight estimate that is just enough to detect all such instances.

From the theoretical results, it is clear how different values of the parameters are mathematically related. For this experiment we trained a product label with a training window size 63×35 using Gentle Adaboost[6][7]. In order to prove that the detection time is logarithmically related to the size of the search space defined by maximum and minimum window size, I ran the algorithm with multiple search search sizes decided by minimum and maximum window size. For the simplicity of understanding and denoting, the search size $\mathcal{S}$ can be represented as the length of line joining the right bottom corners of the minimum and maximum sized search rectangles.

$$\mathcal{S} = \sqrt{w_{max}^2 + h_{max}^2} - \sqrt{w_{min}^2 + h_{min}^2} \tag{12}$$

The resolution of each source image captured by the camera that form the panorama is 1920×2560. The average size of the target object to be detected when the tracking distance of the robot from the shelf is $75cm$ is 410×235. But as mentioned before, because of the various aisle section depths and camera perspective the search rectangle size has to vary across the average size. The search size characterized by the ratio of maximum and minimum window size is started from $W_{min} = (480, 274)$, $W_{max} = (518, 296)$ and broadened till $W_{min} = (63, 35)$, $W_{max} = (980, 560)$. In every iteration, the gap between the minimum and maximum window size is reduced by a step size $s = 24$ pixels, *i.e* the minimum height is increased by 12 pixels and maximum height is decreased by 12 pixels with their corresponding widths being aspect ratio adjusted. The final time of detection for a given search gap is found by averaging the detection time over 291 source images having slightly varying target object size. Fig. 1 shows the various search size represented by the ratio of maximum and minimum window sizes plotted against the detection time(blue dots connected by a blue line). It can be seen from the shape of the curve that it approximates the logarithmic complexity. The green curve is the true log curve plotted with the ratio of maximum and minimum window size, *i.e.* $log_\gamma(\frac{W^i_{max}}{W^i_{min}})$. This experiment empirically verifies that the detection time scales logarithmically with the search size.

The second experiment is to empirically verify that the detection time varies *inverse logarithmically* with the increase in the scale factor, which is the step size from the minimum to maximum window size of detection. The experiment

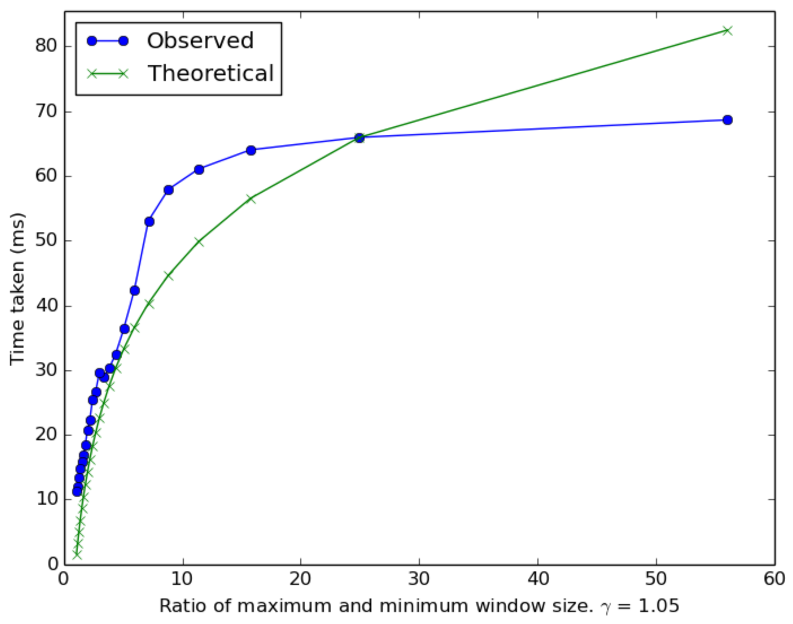

Figure 1: Comparison of ratio of maximum and minimum search window size with time taken taken for detection. It is clear from the shape of curve(blue) connecting the time taken for every search size that it varies logarithmically. The ideal curve(green) is just shown as a reference of a log curve and is not metrically accurate because we are trying to explain the theoretical complexity.

was conducted with the constant minimum window size $W_{min} = (63, 35)$ and maximum window size $W_{max} = (980, 560)$. The scale factor, γ is increased in steps of 0.02 starting from $\gamma = 1.01$. It can be seen from the plot in Fig. 2 that the detection time decreases inverse logarithmically with the increase in scale factor as explained theoretically by Eqn. 9. The green curve in Fig. 2 is the ideal curve that is plotted with the different scale factors and constant detection window size using the formula derived in Eqn. 9, *i.e.* $log_{\gamma_i}(\frac{W_{max}}{W_{min}})$.

5 New Directions

One interesting extension which I am planning on taking up following this work is to modify the way the Adaboost ranks the weak classifiers and calculates its error rate during the training process of the classifier. In the current implementation, at every stage of the classifier we build a decision tree with a set of weak classifiers(HAAR features) whose linear combination can classify at least a desired percentage of positive training data as true positives. Each weak classifier can be visualized as a hyperplane in the sample space trying to classify correctly as many samples as possible. So this is a idiosyncratic feature which when evaluated at some particular position of all positive training sample across all channels and root mean squared(the metric used to evaluate the score of a weak classifier in conventional Viola Jones algorithm[3] is the difference between

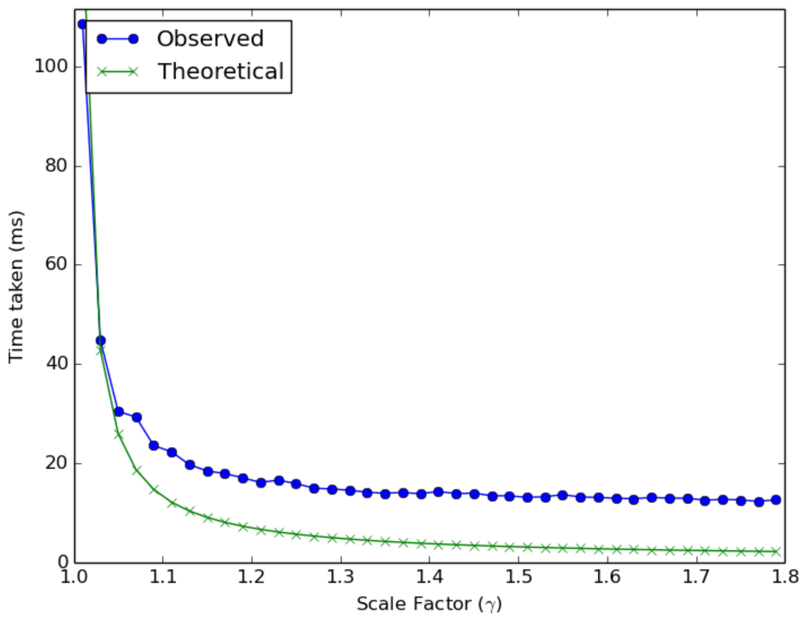

Figure 2: Comparison of scale factor with time taken taken for detection. It is clear from the shape of curve(blue) connecting the time taken for every scale factor that it varies *inverse* logarithmically. The ideal curve(green) is just shown as a reference of a inverse log curve and is not metrically accurate because we are trying to explain the theoretical complexity.

the sum of intensities in the dark and light regions of the feature) produces similar results.

An important observation is that not all channels contribute equally at various positions of the object. For example, while trying to detect objects in a camouflage the depth cloud, if available, is a more representative channel than the image intensity. The edges in the depth pointcloud could be best represented by a HAAR feature(an edge detector in a simplest level) with a minimal feature error rate, $min \sum_i w_{t,i}(y_i - f_t(x_i))^2$, where $w_{t,i}$ is the weight of the ith training sample of weak classifier t, y_i is the groundtruth of ith training sample and $f_t(x_i)$ is the classification output of ith training sample with weak classifier t. A similar argument could be attempted for features representing the difference in LIDAR intensities when detecting an object composed of different physical materials. However, taking a root mean square of the feature values evaluated over all the individual channels comes with the inherent assumption that every channel contains equal amount of information. By the above example and argument this does not seem to be a representative metric. The adaboost scoring mechanism for each weak classifier could be modified to compute the error rate of the feature individually and together for all the available channels. The potential downside of this is that it could increase the training time by a factor, perhaps by the number of channels. However, by doing this each branch of the decision tree could then contain a weak classifier that is specific to a particular channel of the image. This generalization could therefore learn important variations across and within multiple channels at different regions of the object thereby adding a new dimension of information to already existing setup.

With the completion of the proposed hypothesis above, the first part of the thesis, "Feature Extraction and Data Association by marginalizing covariances in iSAM2" comes to completion. Following that, we shift towards fusing this with the data association part of Incremental Smoothing and Mapping using the Bayes Tree (iSAM2) by *Kaess et al.*[5][8]. Most of these state-of-the-art SLAM solutions assume known correspondences between the features and landmark measurements. All these methods bring upon subtle numerical optimizations on the representation of measurement Jacobians as information matrix and the extraction of marginal covariance from square root information matrix[9]. Irrespective of any approach the covariance matrix has to be computed, either fully or marginally, to implement any data association algorithm other than Nearest Neighbor like Maximum Likelihood(ML) Data Association[10] or Joint Compatibility Branch and Bound(JCBB)[11]. Kaess explicitly addresses the estimation of exact and conservative marginal covariances for the data association problem in iSAM[12] but just mentions that *simple recursive algorithms that are formally equivalent to dynamic programming could be used on Bayes tree structure of iSAM2* to compute the marginal covariance. However, data association in conjunction with iSAM2 has not been explicitly studied and the problem mentioned above is still open. This would be the second phase of my thesis.

6 Plan of Action

For the upcoming two weeks, I am planning on finalizing the design for implementing the hypothesis proposed in the previous section. An interesting note is that, although the training time is going to increase by a factor as we are evaluating several channel wise feature values in place of one, the detection is going to be constant time. This is because we are still learning the same number of weak classifiers per stage and decision tree. Also each of those weak classifiers are going to contain information about the channel on which they should be evaluated and the stump threshold. As we are computing the integral image for all those channels nevertheless, the evaluation and hence the total detection time is constant. I am also planning on introducing some intelligent caching mechanism that could reuse the already computed feature values with respect to their identity in subsequent stages of the classifier over the same image which could help reducing the training time.

Before delineating the experimental setup required for validating the claim it is important to understand the two potential ways in which this new approach could be beneficial as opposed to $L2$ norm as the metric for calculating the feature value. Firstly, the $L2$ norm of intensity variations irrespective of any color space of operation across all the image channels under-represents the significant channel that consistently has the variation in intensity. The magnitude of the vector could contain a bias component that captures the intensity variation in one that channel but it might not be consistent because of the following reason. Secondly, there is a possibility that the channels other than the significant channel with respect to a feature may always be a source of random noise making it unreliable consistently. Regarding the setup of experiments for validating the hypothesis proposed in previous section, a set of synthetic images has to be generated with the above two target cases in mind. The classifier should then be trained and tested over this synthetic data.

7 Random Thoughts

Before training the set of images over its pixel intensities, the rotation in the dataset could be removed by computing the principal component axes. The principal component axes of the positive images is actually the best representative color space for the given dataset. Rather than experimenting between various color spaces like Lab, HSV etc., it sounds theoretically correct to choose a mathematically grounded color space. This approach would capture dominating hidden channel as a result of PCA.

The resolution of the negative image plays an important role in the training process. Given a negative image of very high resolution compared to the training size, we sequentially sample the portions of the negative image horizontally starting from top left. At every stage of the classifier, we evaluate the sampled negatives using the already trained stages and collect as many sample until we stumble upon *"numNeg"*(number of negative images used for training) false positives. Although it could be considered that the variations are gradual on the negative image when moved horizontally through each sample and by the time we sample *"numNeg"* false positives we would have attained enough variance in negative space it does not explicitly give control to a trainer for capturing the variance in negative space. Instead we could have equal resolution samples as that of the training size from the negative space and hence assure enough variance while sampling itself.

References

[1]. Arthur, David, and Sergei Vassilvitskii. "k-means++: The advantages of careful seeding." Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, 2007.

[2]. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001), "Chapter 21: Data structures for Disjoint Sets", Introduction to Algorithms (Second ed.), MIT Press, pp. 498524

[3]. Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. Vol. 1. IEEE, 2001.

[4]. Lienhart, Rainer, Alexander Kuranov, and Vadim Pisarevsky. "Empirical analysis of detection cascades of boosted classifiers for rapid object detection." Joint Pattern Recognition Symposium. Springer Berlin Heidelberg, 2003.

[5]. Kaess, Michael, et al. "iSAM2: Incremental smoothing and mapping using

the Bayes tree." International Journal of Robotics Research 31.2 (2012): 216-235.

[6]. Schapire, Robert E., and Yoram Singer. "Improved boosting algorithms using confidence-rated predictions." Machine learning 37.3 (1999): 297-336.

[7]. Freund, Yoav, Robert Schapire, and N. Abe. "A short introduction to boosting." Journal-Japanese Society For Artificial Intelligence 14.771-780 (1999): 1612.

[8]. Kaess, Michael, et al. "The Bayes tree: An algorithmic foundation for probabilistic robot mapping." Algorithmic Foundations of Robotics IX. Springer Berlin Heidelberg, 2010. 157-173.

[9]. Kaess, Michael, and Frank Dellaert. "Covariance recovery from a square root information matrix for data association." Robotics and autonomous systems 57.12 (2009): 1198-1210.

[10]. Bar-Shalom, Yaakov, and Xiao-Rong Li. "Estimation and Tracking: Principles, Techniques, and Software [Reviews and Abstracts]." IEEE Antennas and Propagation Magazine 38.1 (1996): 62.

[11]. Neira, Jos, and Juan D. Tards. "Data association in stochastic mapping using the joint compatibility test." IEEE Transactions on robotics and automation 17.6 (2001): 890-897.

[12]. Kaess, Michael, Ananth Ranganathan, and Frank Dellaert. "iSAM: Incremental smoothing and mapping." IEEE Transactions on Robotics 24.6 (2008): 1365-1378.

[13]. Wang, Yi-Qing. "An Analysis of the Viola-Jones face detection algorithm." Image Processing On Line 4 (2014): 128-148.

[14]. Open Computer Vision Library. http:/sourceforge.net/projects/ opencvlibrary/