Eye pupil localization with an ensemble of randomized trees

Nenad Markus, Miroslav Frljak, Igor S. Pandzic, Jörgen Ahlberg and Robert Forchheimer

Original Publication:

Nenad Markus, Miroslav Frljak, Igor S. Pandzic, Jörgen Ahlberg and Robert Forchheimer, Eye pupil localization with an ensemble of randomized trees, 2014, Pattern Recognition, (47), 2, 578-587.
http://dx.doi.org/10.1016/j.patcog.2013.08.008

Eye pupil localization with an ensemble of randomized trees

Nenad Markuš[a,*], Miroslav Frljak[a], Igor S. Pandžić[a], Jörgen Ahlberg[b], Robert Forchheimer[b]

[a] *University of Zagreb, Faculty of Electrical Engineering and Computing Unska 3, 10000 Zagreb, Croatia*

[b] *Linköping University, Dept. of Electrical Engineering, Information Coding Group SE-581 83 Linköping, Sweden.*

Abstract

We describe a method for eye pupil localization based on an ensemble of randomized regression trees and use several publicly available datasets for its quantitative and qualitative evaluation. The method compares well with reported state-of-the-art and runs in real-time on hardware with limited processing power, such as mobile devices.

Keywords: eye pupil localization, boosting, randomized trees

1. Introduction

The human eyes play an essential role in everyday interaction, communication and other routine activities. Thus, tracking the eyes and eye pupils opens a wide range of applications. Eye pupil localization can be used in biometric systems, gaze/focus estimation, human-machine interfaces, character animation, etc. Using the eyes as a means of human-computer interaction is helping disabled people to improve their daily lives, and may become a hands-free alternative in other applications or an entertaining element in innovative games. Eye tracking is increasingly finding use in safety applications to detect situations such as sleepiness or lack of attention while driving or using hazardous machinery.

High quality gaze estimation systems most commonly use head-mounted or multiple near-infrared cameras. Such systems can achieve high accuracy but they

*Corresponding author, tel.: +38516129738, fax: +38516129832

Email addresses: nenad.markus@fer.hr (Nenad Markuš), miroslav.frljak@fer.hr (Miroslav Frljak), igor.pandzic@fer.hr (Igor S. Pandžić), jorgen.ahlberg@isy.liu.se (Jörgen Ahlberg), robert@isy.liu.se (Robert Forchheimer)

are expensive, intrusive or both. Furthermore, they often require calibration. Therefore, they are not well suited for user friendly applications working on off-the-shelf hardware where eye information is useful. There is a need for more versatile and simple systems and our research is directed this way. We are interested in the case when the images are supplied from a monocular video stream on consumer hardware, especially on mobile devices. These images may have low resolution, are often noisy and have bad contrast. Furthermore, we are interested in supporting a wide range of PC and mobile devices with limited processing power. To satisfy such requirements, a robust yet high-performance system is needed.

In this paper, we describe a framework for eye pupil localization. Small computational cost of the developed framework makes it ideal for achieving high performance using off-the-shelf hardware. Only one uncalibrated camera is needed and this makes the framework completely non-intrusive. The implemented system runs in real-time on mobile devices.

1.1. Related work

Our work is related to a large body of research on eye and gaze tracking. An extensive overview has been done by Hansen et al. [18].

Of particular interest is the research done on eye center localization. Methods that report state-of-the-art results have been described in [30, 32]. The basic idea of both approaches is to find the face within the image and then locate eye centers using geometric information (symmetry or curvature of the iris). Timm et al. [30] propose an approach based on analysis of image gradients. The idea is to define an objective function which obtains its maximum in the location where most gradient vectors intersect. Since the iris has a circular appearance in most frontal images, this also corresponds to the position of the pupil. Valenti et al. [32] use the curvature of isophotes (curves connecting points of equal intensity) to design a voting scheme for pupil localization. Additionally, they extend their approach by extracting a SIFT [24] vector for each candidate location and match it with examples in a database to obtain the final decision. In some cases this significantly increases the accuracy. Both methods are designed to deal with frontal faces and are accurate at solving this problem. Although they reportedly work in real-time, the authors have not demonstrated their performance on mobile devices.

The methods are fundamentally different from our approach since we use a machine learning algorithm to discover the structure in the problem rather than assuming one *a priori*. We formulate the problem of pupil localization as regression based on low-level image features. The crucial component is based on randomized trees. These were introduced to the computer vision and image processing community by Amit and Geman [1], and used since to solve classification [15, 22, 23, 26, 28, 33] and regression [7, 8, 9, 10, 11, 12, 17, 29, 36] problems. Randomized trees were a natural choice in our case due to the fact

that they handle large training sets with multiple inputs and outputs, and their high generalization power and fast computation.

A similar approach has been taken in [35] to estimate the gaze of the user. The authors decided to select a subset of Haar-like features and train a neural network to estimate the point of gaze on the screen.

1.2. Our contributions

We describe a method for pupil location estimation based on an ensemble of randomized regression trees. The method compares well with the current state-of-the-art in terms of accuracy and runs in real-time on hardware with limited computational resources, such as mobile devices. Additionally, we perform experiments that could prove valuable to other researchers and engineers in the development of image based regression algorithms for similar problems.

2. Method

We assume that the approximate locations of the eyes are known and that we can extract rectangular regions that contain them. This does not pose a problem today since very efficient and accurate face detectors exist. In the next step we use a regression tree ensemble to estimate the coordinates of eye pupils within these regions.

For the purpose of our experiments, we implemented a framework for regression tree construction and evaluation. Furthermore, we use a face detector available in OpenCV to find approximate locations of the eyes in videos and still images.

2.1. Regression trees

Regression trees [5] are tools for function approximation. The basic idea is to split recursively the original problem into two simpler ones, each solvable by a model of reduced complexity. The splits are performed at internal nodes of the tree, based on problem-specific binary tests. Terminal nodes contain simple models that approximate the desired output. In practice, we treat the tree construction process as a supervised learning problem. In other words, we have to choose the binary tests in internal nodes and output models at terminal nodes based on a finite set of input-output pairs in a training set.

Let us briefly describe the configuration of a single tree in our implementation. The binary tests at internal nodes of the tree are based on pixel intensity differences (this is motivated by good results obtained, for example, by [6, 28]). We define a binary test on image I as

$$\text{bintest}(I; \mathbf{l}_1, \mathbf{l}_2, t) = \begin{cases} 0, & I(\mathbf{l}_1) - I(\mathbf{l}_2) \leq t \\ 1, & \text{otherwise} \end{cases}$$

where $I(\mathbf{l}_i)$ is the pixel intensity at location $\mathbf{l}_i$ and t is the threshold. Locations $\mathbf{l}_1$ and $\mathbf{l}_2$ are in normalized coordinates. This results in their resolution independence and means that the binary tests can easily be resized based on the data obtained from depth cameras or trackers/detectors that provide information about the scale (resizing is needed when the user moves around and changes his/her distance from the camera as this affects the scale of eyes within the video frame). Each terminal node of the tree contains a constant vector that models the output.

The construction of the tree is supervised. The training set consists of images annotated with values in $\mathbb{R}^2$. In our case, these values represent the location of eye pupil in normalized coordinates. The parameters of each binary test in internal nodes of the tree are optimized in a way to maximize clustering quality obtained when the incoming training data is split by the test. This is performed by minimizing the following quantity:

$$Q = \sum_{\mathbf{x} \in C_0} \|\mathbf{x} - \bar{\mathbf{x}}_0\|_2^2 + \sum_{\mathbf{x} \in C_1} \|\mathbf{x} - \bar{\mathbf{x}}_1\|_2^2, \tag{1}$$

where C_0 and C_1 are clusters that contain pupil coordinates $\mathbf{x} \in \mathbb{R}^2$ of all eye patches for which the outputs of binary test were 0 and 1, respectively. The vector $\bar{\mathbf{x}}_0$ is the mean of C_0 and vector $\bar{\mathbf{x}}_1$ is the mean of C_1.

As the set of all pixel intensity differences is prohibitively large, we generate only a small subset during optimization and compute a threshold for each of them. Although there are many strategies to generate this subset [6], in our experiments we select each pixel difference by sampling two locations from a uniform distribution on a square $[-1, +1] \times [-1, +1]$.

For each pixel intensity difference feature, an optimal threshold can be computed in the following way. Sort the training images by the value of their feature response and then perform a linear search over the array for the best threshold. The test that achieves the smallest error according to equation 1 is selected. Thus, the computational complexity of binary test optimization is $O(F \cdot S \log S)$, where S is the number of training samples at the node and F is the number of generated features. The training data is recursively clustered in this fashion until some termination condition is met. In our experiments, we limit the depth of our trees to reduce training times and memory requirements. The output at each leaf node is obtained as the mean of pupil coordinates arriving at that node.

2.2. Tree ensembles

It is well known that a single tree will most likely overfit the training data. On the other hand, an ensemble of trees can achieve impressive results. The most common ways of combining multiple trees are random forests [4] and gradient boosting [14]. The former grows deep trees independently and then averages the

result of prediction of each individual tree in the ensemble. The latter grows trees sequentially. Each new one added to the ensemble is learned to reduce the remaining training error further. Its output is shrunk by a scalar factor (called shrinkage, $\nu \in [0, 1]$) that plays a similar role as does learning rate when training neural networks.

Both methods have as parameters the maximum depth of each tree, d, and the number of trees in an ensemble, T. Memory required to store the regression ensemble scales as $O(2^d \cdot T)$ and time for its evaluation as $O(d \cdot T)$. Additionally, gradient boosting has shrinkage ν as a parameter. This parameter does not affect the speed or memory requirements during runtime.

One of the reasons behind the success of tree ensembles is the randomness present in the learning process. Randomness is injected in the induction of each single tree by bootstrap sampling the training set and randomly choosing features as split candidates at each internal node. This procedure reduces the correlation between different trees in the ensemble and leads to better generalization [19]. The other benefit of this approach is the reduction of training time needed to learn each individual tree.

Both methods show good results in practice, as evidenced in numerous experiments conducted by other researchers and engineers. Thus, we have decided to use numerical measures in order to determine which method to use.

2.3. Random perturbations

We have observed that the output of regression trees is noisy and can be unreliable in some frames, especially if the video stream is supplied from a low quality camera. This can be attributed to variance of the regressor as well as to the simplicity of binary test at internal nodes of the trees: Pixel footprint size changes significantly with variations in scale of the eyes and we can expect problems with aliasing. These problems can be reduced during runtime with random perturbations. The idea is to sample multiple rectangular patches at different positions and scales around the eye and estimate the pupil location in each of them. The resulting pupil coordinate is the median over the detections for each spatial dimension. Similar solutions were also proposed by Dollár et al. [10] and Cao et al. [7].

The runtime complexity of the method scales as $O(p \cdot T \cdot d)$, where p is the number of sampled rectangles (random perturbations). Notice that this complexity does not depend on the resolution of the patch.

2.4. A chain of multi-scale ensembles

We have observed that accuracy and robustness of the method critically depend on the scale of the rectangle within which we perform detection. If the rectangle is too small, we risk that it will not contain the eye at all due to the uncertainty

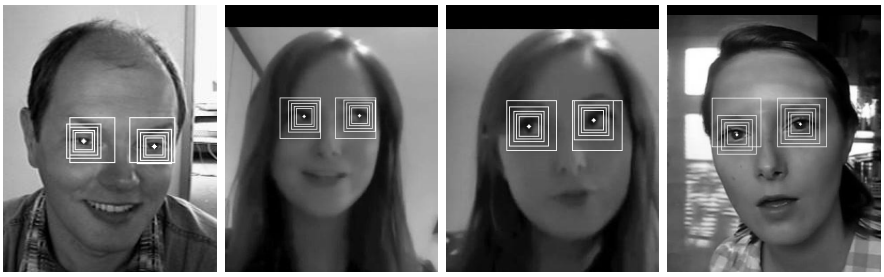


Figure 1: Some typical rectangle sizes for multi-scale pupil location estimation.

introduced by face tracker/detector. If the rectangle is big, the detection is more robust but accuracy suffers. Thus, we propose a solution similar to the one provided by Ong et al. [25]. The approach also resembles the standard pyramidal optical flow algorithm [3].

The idea is to learn multiple tree ensembles, each for estimation at different scale. The method proceeds in a recursive manner, starting with an ensemble learned for largest scale. The obtained intermediate result is used to position the rectangle for the next ensemble in the chain. The process continues until the last one is reached. Its output is accepted as the final result. Images illustrating the typical size of rectangles for multi-scale estimation can be seen in figure 1.

We provide numerical results in the experimental section to back our arguments.

3. Training the method

3.1. Annotated data

For our experiments we use the UULM [34] and Gi4E [27] publicly available datasets. Additionally, we constructed our own database for the purposes of the paper. It consists of various students, people from our lab and images acquired on the internet.

All databases consist of images of a number of different subjects with varying head rotation and gaze direction. The images are of varying resolution (800×600 and 640×480 pixels) and each is annotated with locations of eye pupils. All datasets together contain around 5000 face images.

3.2. Training set preparation

Under real world conditions, we must expect that face trackers and detectors, on which our method relies for rough eye region localization, will introduce errors in image patch extraction. Thus, in order to make our system more robust, we intentionally introduce position and scale perturbations in the training data.

For each annotated face in available databases we extract a number of eye patches and corresponding pupil coordinates. All these patches are generated from the *basic rectangle* using random perturbations. The basic rectangle has a predetermined aspect ratio and is obtained from annotated eye corner coordinates which provide both eye region location and a scale factor. Each position perturbation is sampled from a uniform distribution on a rectangle determined by -25% to $+25\%$ of the width and height of the basic rectangle. Similarly, each scale perturbation is sampled from a uniform distribution on an interval determined by -25% to $+25\%$ of the basic rectangle scale.

3.3. Generalization ability of learned tree ensembles

In order to establish the generalization ability of tree ensembles for different parameter values we have performed numerous experiments and it is not possible to show all results due to limited space. Thus, we have decided to present only the graphs that demonstrate the reduction of mean squared error (in image coordinates) achieved by different ensembles for various tree depths, d.

For each annotated image of the eye, we generate 100 training samples according to the method described in the previous section. The UULM database and our annotated images are used for training and Gi4E is used for validation. Thus, the training set consists of around 700 000 images and the validation set of around 200 000 images. We grow each tree on a 50% bootstrap sample of the prepared training data. The number of binary tests considered at each internal node is $4 + 4d'$, where d' is the depth of the node. For boosting, the shrinkage parameter is fixed to 0.4 as we have noticed that this value gives good results.

The whole evaluation procedure was repeated 5 times and the results were averaged. In figure 2 we plot the mean squared errors against the number of binary test evaluations, $d \cdot T$, for different ensemble parameters. We can observe that boosting outperforms random forests in our setup. Note that the errors are not normalized, i.e., they are in image coordinates. Thus, as the validation images vary in resolution, these errors do not have an intuitive relation to the accuracy of pupil localization. In later sections we will use a more appropriate measure to compare our method with other approaches.

All tree ensembles mentioned in the rest of the paper were trained using data from all mentioned databases. To generate the training set, we used 400 random perturbations per annotated image of the eye.

3.4. Comments on training time

We implemented our framework for regression tree construction in standard C. The training runs in one thread although it could be trivially parallelized (in our case it was just not necessary). For comparison, training time for an ensemble of one hundred trees of depth equal to ten takes about six hours on a training set with six million examples, which is acceptable.

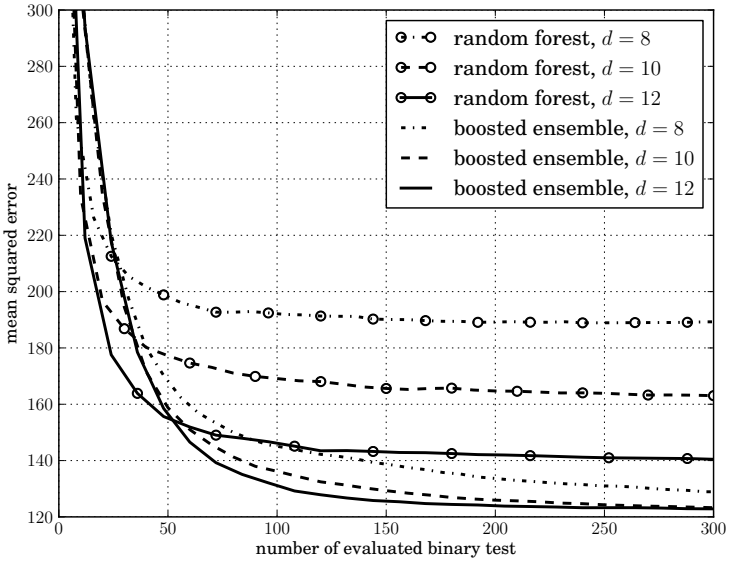


Figure 2: Mean squared errors on validation data plotted against the number of evaluated binary tests, $d \cdot T$, for different maximum depth d of trees in boosted ensembles and random forests. Number of binary test evaluations is proportional to the amount of computation needed at runtime.

4. Experiments

We intend to make a system capable of real-time performance on hardware of limited processing power, such as mobile devices. Thus, we need to take care of both memory and computation requirements. The complexity of the estimation structure can be regulated by varying the number of trees and the maximum depth of each tree. We fix the run-time complexity by imposing a constraint on the number of binary tests that can be evaluated during estimation to 1000. Furthermore, to limit the memory requirements, we fix the maximum depth of each tree to $d = 10$. This limits the number of trees to $T = 100$. For learning boosted ensembles the shrinkage was set to $\nu = 0.4$. In our preliminary experiments we have observed that these parameters give good tradeoff between accuracy and performance. It is left to determine the appropriate number of stages for multi-scale estimation process and the method for training each ensemble. We address these questions through experimental analysis.

We are interested in evaluating the usefulness of the method in relevant applications such as pupil localization in still images and pupil tracking in video streams. In our implementation, eye pupil localization in still images is summarized by the following steps:

1. Obtain a face bounding box using a face detector.
2. Estimate eye regions using simple anthropometric relations.

3. Estimate pupil location for each eye region using a chain of multi-scale tree ensembles.

Pupil tracking uses the previously described method for initialization and proceeds by repeating the following steps for each frame:

1. Estimate positions of the eyes based on pupil locations in previous frame.
2. Estimate scale of each eye region based on distance between the eyes.
3. Estimate pupil location for each eye region using a chain of multi-scale tree ensembles.

In the rest of this section we discuss quantitative and qualitative results obtained by the method. Additionally, we measure the required processing times on different hardware.

4.1. Quantitative results

The normalized error is adopted as the accuracy measure for the estimated eye locations. It was proposed by Jesorsky et al. [21] and is defined as follows:

$$e = \frac{\max\{D_L, D_R\}}{D}, \tag{2}$$

where D_L and D_R are the Euclidean distances between the found left and right eye centers and the ones in the ground truth, and D is the Euclidean distance between the eyes in the ground truth. Roughly, an error of $e \leq 0.25$ corresponds to the distance between the eye center and the eye corners, $e \leq 0.1$ corresponds to the range of the iris, and $e \leq 0.05$ corresponds to the range of the pupil.

We plot regression error characteristic (REC) curves [2] to evaluate the accuracy of our method for different parameter choices on different datasets. The first experiment examines the accuracy on a set of still images. The other evaluates real-time pupil localization in a video stream.

4.1.1. Experimentation with still images

The BioID database (`http://www.bioid.com`) is used for testing. It consists of 1521 frontal face grayscale images with significant variation in illumination, scale and pose. In several samples of the database the subjects are wearing glasses. In some instances the eyes are closed, turned away from the camera, or completely hidden by strong highlights on the glasses. Due to these conditions, the database is considered difficult and realistic. Some snapshots can be seen in figure 3. Our approach yields inaccurate estimations if the eyes are closed or strong reflections on the glasses occur as we did not include such examples in the training set.

One important thing to note before presenting numerical results is that we omitted the images where the detector did not find the face, following [32].

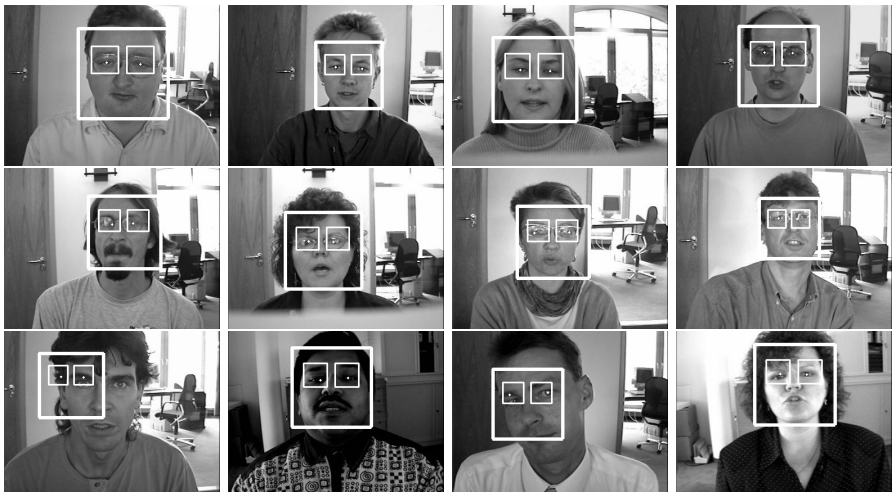


Figure 3: First row illustrates some successful detections on the BioID database. The second and third one illustrate some typical failures.

This did not significantly alter the results since approximately 1% images were discarded this way.

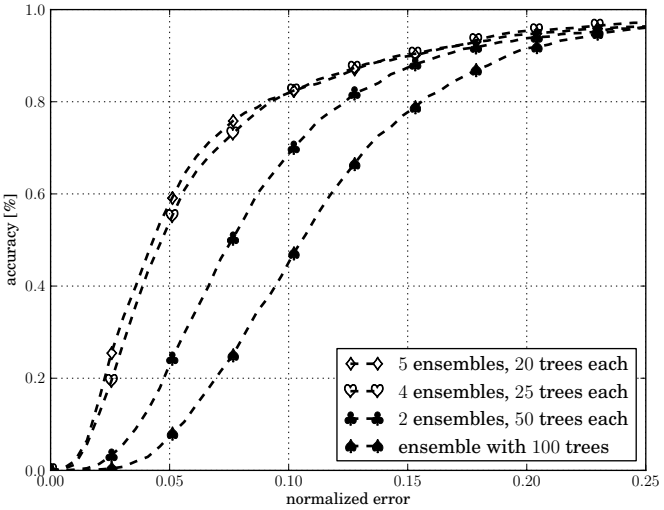
Let us first examine the benefits of multi-scale estimation (we do not perform random perturbations in this experiment, i.e., we set $p = 1$). Quantitative results can be seen in figure 4. We can observe that this procedure significantly improves the accuracy for $e \leq 0.15$. As we are mostly interested in accurate pupil localization, we can conclude that the multi-scale approach provides significant benefits.

The results can be improved even further by using random perturbations, as explained in section 2.3. Quantitative results can be seen in figure 5. We observe a significant increase in accuracy when p increases. Of course, the number of computations increases linearly with p. Thus, in order to justify the use of this improvement we demonstrate in section 4.3 that real-time frame rates can be achieved on mobile devices with moderate values of p.

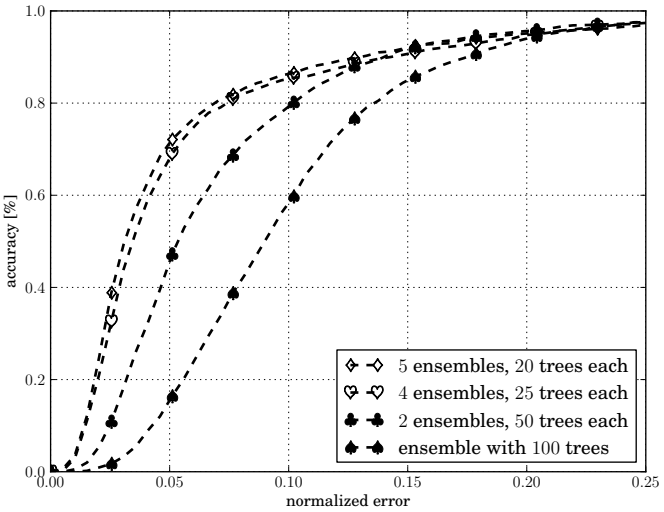
We compare our results in table 1 to the two state-of-the-art methods for eye center localization. We can observe that our method compares well to the competition in terms of accuracy. Of course, there is always the problem of dataset bias [31]: We cannot conclude which method is superior based just on accuracy results on one dataset.

We noticed a few sources of discrepancies with annotated data (examples can be seen in figures 3 and 6):

- ***Presence of eyeglasses***. In some images the eyes are completely hidden by strong highlights on the glasses. Our method fails under these conditions. Figure 6a illustrates this case.

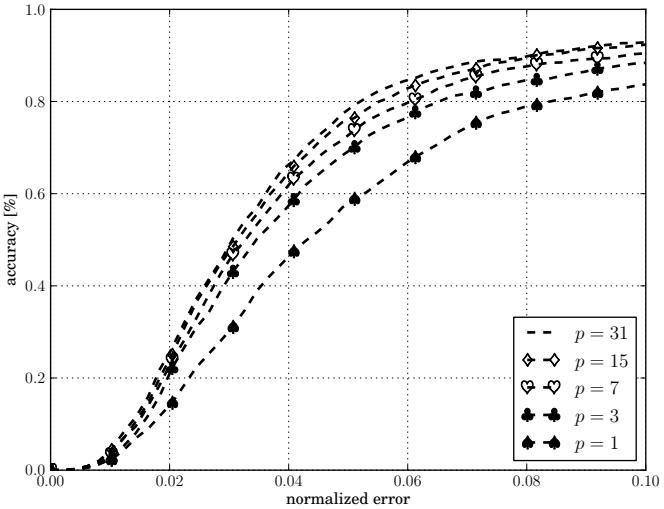


(a) Estimation using random forests.

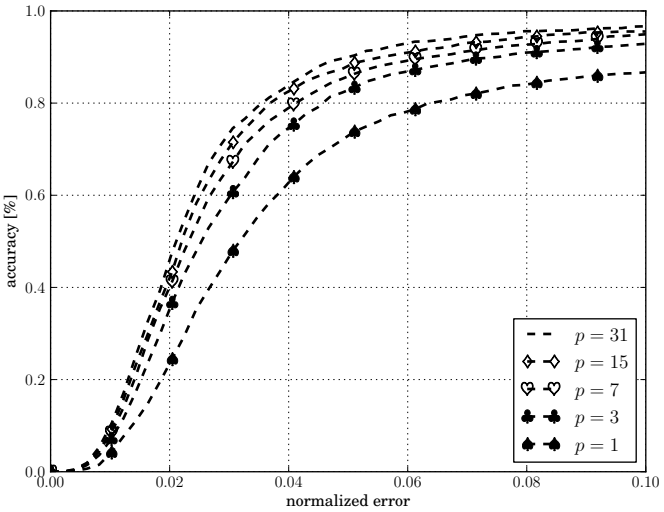


(b) Estimation using boosted ensembles.

Figure 4: Increase in accuracy when using a multi-scale estimation approach. We observe that boosting gives better results.



(a) Estimation using random forests at 5 different scales, 20 trees per scale.

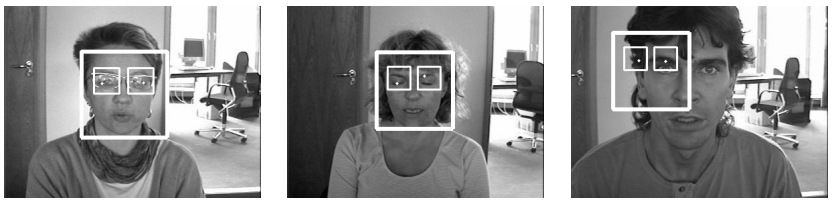


(b) Estimation using boosted ensembles at 5 different scales, 20 trees per scale.

Figure 5: Increase in accuracy when using different number of random perturbations, p. We can observe that boosting gives significantly better reslults.

Method	$e \leq 0.05$	$e \leq 0.1$	$e \leq 0.25$
Our results, $p = 7$	85.7	95.3	99.7
Our results, $p = 31$	**89.9**	**97.1**	**99.7**
Timm et al. [30]	82.5	93.4	98.0
Valenti et al. [32], basic method	80.6	85.8	96.6
Valenti et al. [32], extended method	86.1	91.7	97.9

Table 1: Comparison of normalized error scores on the BioID database. The values in the table are percentages.



(a) Failure due to strong highlights on the glasses.

(b) Failure due to eye closure.

(c) Failure of face detection procedure.

Figure 6: Snapshots from the BioID database illustrating some typical failures of our method.

- ***Closed eyes***. Eye center localization can not work on closed eyes when the iris is not visible. Thus, the results obtained from our system are unpredictable during eye closure. An example can be seen in figure 6b.
- ***Eye region extraction failure***. Some bad results are due to the limitations of the used face detection procedure. An example can be seen in figure 6c.

4.1.2. Experiment on video

We use the Talking Face video database (`http://www-prima.inrialpes.fr/FGnet/data/01-TalkingFace/talking_face.html`) to evaluate our system quantitatively in real-time applications. The video contains 5000 frames taken from a video of a person engaged in conversation. Each frame was annotated semi-automatically using an active appearance model [16] trained specifically for the person in the video. The annotations are sufficiently accurate to represent the facial movements during the sequence. They contain the locations of eye centers, among other characteristic points on the face.

We used an estimation chain consisting of 5 boosted ensembles with 20 trees of depth equal to 10. The number of random perturbations was set to $p = 15$. The video showing tracking results is available as supplementary material.

The normalized error (equation 2) averaged over the video sequence was equal to 0.0236 and its median 0.0201. Accuracy curve can be seen in figure 7 and error distribution over the frames in figure 8. These show that most of the

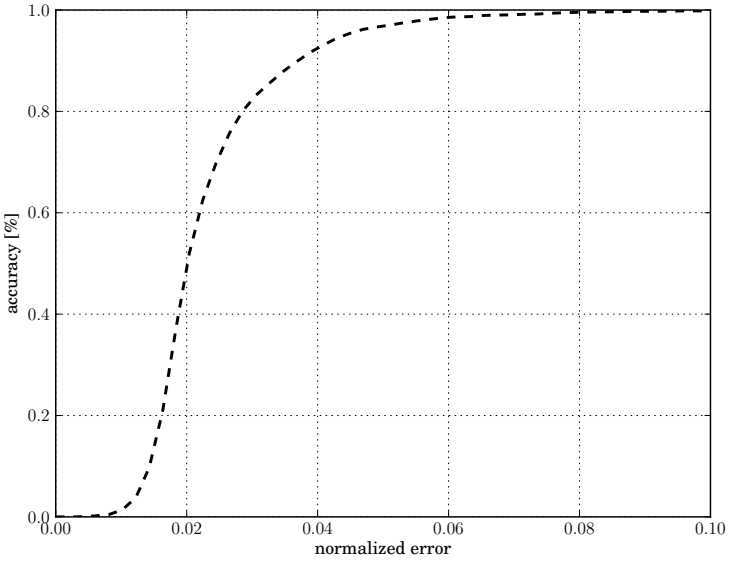


Figure 7: Accuracy curve for the Talking Face video pupil tracking.

time our results were in the range of pupil radius from annotated data. Some snapshots from the video can be seen in figure 9.

Although the average error is sufficiently low, we analyze the error spikes present in the graph. We noticed a few sources of these discrepancies with the annotated data:

- ***Closed eyes***. Eye center localization can not work on closed eyes when the iris is not visible. Thus, the results obtained from our system are unpredictable during eye closure. The proper solution is to include eye closure detection, which we plan in the future. An example can be seen in figure 10a.
- ***Error in annotated data***. Because the annotated data was obtained semi-automatically, it contains occasional errors. For example, this happens around the 4000th frame (a big spike in the error graph). Figure 10b illustrates the case. We can observe that our system is more accurate than the annotated data in this frame.
- ***Failure of our method***. In some frames our method outputs incorrect eye pupil locations even when all appropriate conditions are met. An example of such a case can be seen in figure 10c.

4.2. Qualitative results

Besides the quantitative analysis on datasets that provide ground truth data, we performed less formal qualitative experiments on still images and video in-

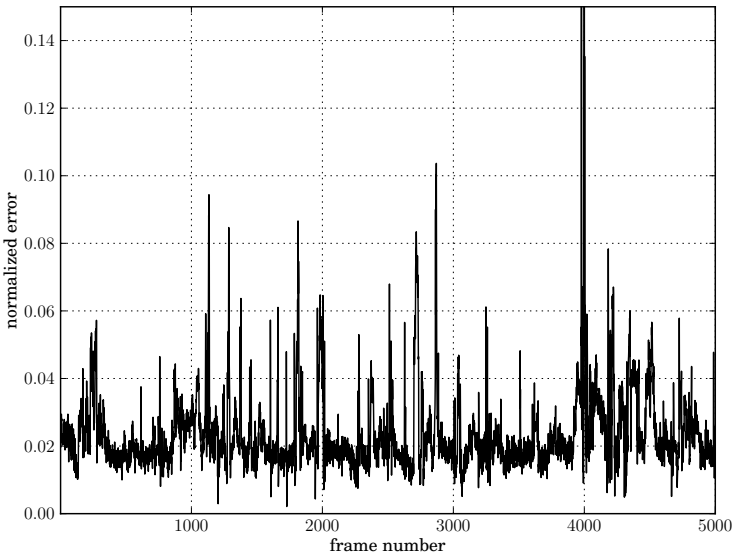


Figure 8: The distribution of normalized errors over the frames of the Talking Face video.

Figure 9: Snapshots from the Talking Face video in which our system outputs the correct locations of eye pupils.

(a) Failure due to eye closure. (b) Error in annotated data. (c) Our method outputs wrong results.

Figure 10: Snapshots from the Talking Face video illustrating some typical discrepancies with annotated data. Annotations are represented with a cross and the output of our system with a circular target symbol.

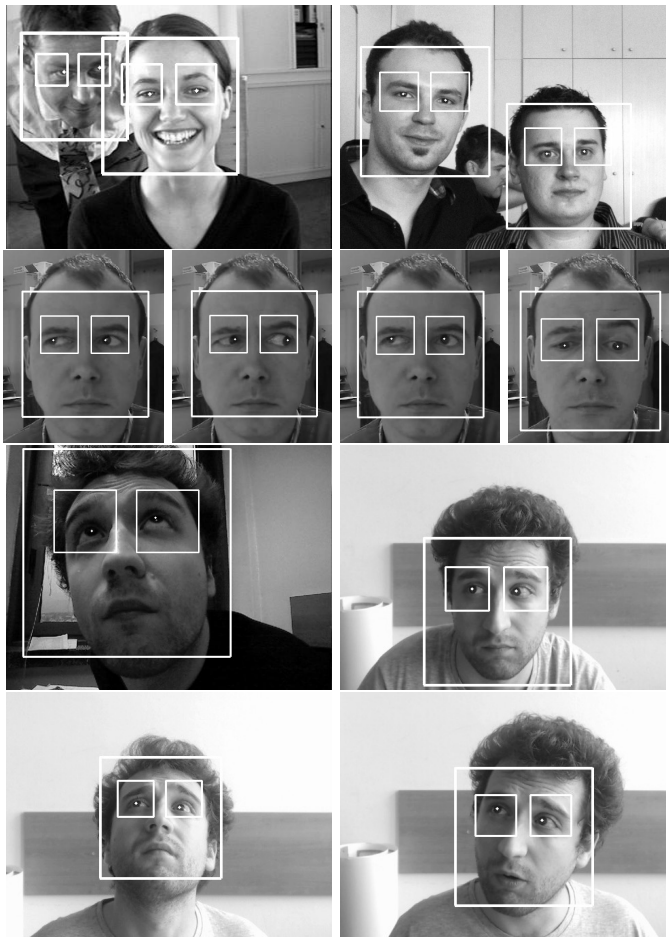


Figure 11: Pupil localization in still images.

put, both on PCs with webcams and on mobile devices. A video showcasing these results is available at `https://www.youtube.com/watch?v=7J30yNHlXlQ` (which is also available as supplementary material). Furthermore, for readers who wish to test the method themselves, the windows-based demo application is available (`http://hotlab.fer.hr/_download/repository/puploc.zip`).

Figure 11 illustrates some examples of eye pupil localization in still images. The system correctly estimates the pupil coordinates even when eye rotations are present. Figure 12 illustrates some results for eye regions extracted during tracking in a video stream. We can observe that the method provides accurate results for large variations in position, scale and rotation. This demonstrates that our approach can successfully deal with real-world conditions.

We have observed occasional failures that can be attributed to the reasons dis-

Figure 12: Example eye patches with estimated pupil locations.

Device	CPU	Time [ms]			
		$p = 3$	$p = 7$	$p = 15$	$p = 31$
PC1	3.4GHz Core i7-2600	0.02	0.05	0.12	0.24
PC2	2.53GHz Core 2 Duo P8700	0.06	0.13	0.29	0.60
iPhone 5	1.3GHz Apple A6	0.12	0.27	0.58	1.20
iPad 2	1GHz ARM Cortex-A9	0.22	0.53	1.13	2.32
iPhone 4S	800MHz ARM Cortex-A9	0.28	0.66	1.42	2.93

Table 2: Average time required to process one image region for different number of random perturbations, p. The ensemble consists of 100 trees, each of depth 10.

cussed in sections 4.1.1 and 4.1.2.

4.3. Analysis of performance and required memory

We measure the average time to process one eye-region rectangle and locate the pupil using our method with standard parameters ($T = 100$ and $d = 10$). The system is implemented in C and runs in a single thread. Most computations are performed in integer arithmetic to further speedup the processing. The obtained results are reported in table 2.

Considering that typical face tracking performance may be around 30 to 50 frames per second on most devices, this gives a frame processing time of 20 ms, and considerably more in case of full face detection. This means that detection of both eye pupils presents a negligible performance impact within a face detection or tracking system. Also note that the system can track eyes on its own at these computational requirements (section 4.1.2).

A structure of $T = 100$ trees of depth $d = 10$ requires approximately 1.3 megabytes of storage. We believe that this is acceptable both at runtime and distribution with an application.

5. Conclusion and future directions

We have described an eye pupil localization framework based on an ensemble of randomized regression trees. The implementation is capable of achieving

real-time performance on mobile devices. The developed software provides a low-cost alternative to complex and costly commercial eye tracking systems.

The described method is very general and can be applied to solve similar problems, such as tracking of learned 2D templates. An interesting application would be to track multiple points scattered on a rigid object and use RANSAC algorithm [13] to recover the motion, similar to the work done by Zimmermann et al. [37].

Future work includes the possibility of using randomized trees for eye closure estimation. It would be interesting to see if we could obtain good results by synthesizing a training set using methods from computer graphics. Another interesting direction of research is to use our method for estimating the point of gaze of the user. This would require the integration of a 3D model to our framework (similar to [20]).

6. Acknowledgements

This research is partially supported by Visage Technologies AB, Linköping, Sweden, by the Ministry of Science, Education and Sports of the Republic of Croatia, grant number 036-0362027-2028 "Embodied Conversational Agents for Services in Networked and Mobile Environments" and by the European Union through ACROSS project, 285939 FP7-REGPOT-2011-1.

References

[1] Amit, Y., Geman, D., 1997. Shape quantization and recognition with randomized trees. Neural Computation 9, 1545–1588.

[2] Bi, J., Bennett, K. P., 2003. Regression error characteristic curves. In: Proceedings of the 20th International Conference on Machine Learning. pp. 43–50.

[3] Bouguet, J. Y., 1999. Pyramidal implementation of the lucas kanade feature tracker description of the algorithm. Tech. rep., Intel Microprocessor Research Labs.

[4] Breiman, L., 2001. Random forests. Journal of Machine Learning Research 45, 5–32.

[5] Breiman, L., Friedman, J., Stone, C. J., Olshen, R. A., 1984. Classification and Regression Trees. Chapman and Hall.

[6] Calonder, M., Lepetit, V., Ozuysal, M., Trzinski, T., Strecha, C., Fua, P., 2011. Brief: Computing a local binary descriptor very fast. TPAMI 34, 1281–1298.

[7] Cao, X., Wei, Y., Wen, F., Sun, J., 2012. Face alignment by explicit shape regression. In: CVPR. pp. 2887–2894.

[8] Criminisi, A., Konukoglu, E., Shotton, J., 2010. Regression forests for efficient anatomy detection and localization in ct studies. In: Medical Computer Vision 2010: Recognition Techniques and Applications in Medical Imaging, MICCAI workshop. pp. 106–117.

[9] Dantone, M., Gall, J., Fanelli, G., Gool, L. V., 2012. Real-time facial feature detection using conditional regression forests. In: CVPR. pp. 2578–2585.

[10] Dollár, P., Welinder, P., Perona, P., 2010. Cascaded pose regression. In: CVPR. pp. 1078–1085.

[11] Fanelli, G., Dantone, M., Gall, J., Fossati, A., Gool, L. V., 2012. Random forests for real time 3d face analysis. IJCV 101, 437–458.

[12] Fanelli, G., Gall, J., Gool, L. V., 2011. Real time head pose estimation with random regression forests. In: CVPR. pp. 617–624.

[13] Fischler, M. A., Bolles, R. C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24, 381–395.

[14] Friedman, J. H., 2001. Greedy function approximation: A gradient boosting machine. Annals of Statistics 29, 1189–1232.

[15] Gall, J., Yao, A., Gool, N. R. L. V., Lempitsky, V., 2011. Hough forests for object detection, tracking, and action recognition. TPAMI 33, 2188–2202.

[16] Gao, X., Su, Y., Li, X., Tao, D., 2010. A review of active appearance models. IEEE Systems, Man, and Cybernetics, Part C: Applications and Reviews 40, 145–158.

[17] Glocker, B., Pauly, O., Konukoglu, E., Criminisi, A., 2012. Joint classification-regression forests for spatially structured multi-object segmentation. In: ECCV. pp. 870–881.

[18] Hansen, D. W., Ji, Q., 2010. In the eye of the beholder: A survey of models for eyes and gaze. TPAMI 32, 478–500.

[19] Hastie, T., Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Learning. Springer.

[20] Heyman, T., Spruyt, V., Ledda, A., 2012. 3d face tracking and gaze estimation using a monocular camera. Tech. rep.

[21] Jesorsky, O., Kirchbergand, K. J., Frischholz, R., 1992. Robust face detection using the hausdorff distance. In: Audio and Video Biom. Pers. Auth. pp. 90–95.

[22] Kalal, Z., Mikolajczyk, K., Matas, J., 2012. Tracking-learning-detection. TPAMI 34, 1409–1422.

[23] Lepetit, V., Fua, P., 2006. Keypoint recognition using randomized trees. TPAMI 28, 1465–1479.

[24] Lowe, D. G., 1999. Object recognition from local scale-invariant features. In: ICCV. Vol. 2. pp. 1150–1157.

[25] Ong, E.-J., Lan, Y., Theobald, B., Bowden, R., 2011. Robust facial feature tracking using selected multi-resolution linear predictors. TPAMI 33, 1844–1859.

[26] Özuysal, M., Fua, P., Lepetit, V., 2007. Fast keypoint recognition in ten lines of code. In: CVPR. pp. 1–8.

[27] Ponz, V., Villanueva, A., Cabeza, R., 2012. Dataset for the evaluation of eye detector for gaze estimation. In: Proceedings of the 2012 ACM Conference on Ubiquitous Computing. pp. 681–684.

[28] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A., 2011. Real-time human pose recognition in parts from single depth images. Tech. rep., Microsoft Research.

[29] Sun, M., Kohli, P., Shotton, J., 2012. Conditional regression forests for human pose estimation. In: CVPR. pp. 3394–3401.

[30] Timm, F., Barth, E., 2011. Accurate eye centre localisation by means of gradients. In: Int. Conference on Computer Vision Theory and Applications. Vol. 1. pp. 125–130.

[31] Torralba, A., Efros, A. A., 2011. Unbiased look at dataset bias. In: CVPR. pp. 1521–1528.

[32] Valenti, R., Gevers, T., 2012. Accurate eye center location through invariant isocentric patterns. TPAMI 34, 1785–1798.

[33] Villamizarn, M., J. Andrade-Cetto, A. Sanfeliu, F. M.-N., 2012. Bootstrapping boosted random ferns for discriminative and efficient object classification. Pattern Recognition 45, 3141–3153.

[34] Weidenbacher, U., Layher, G., Strauss, P.-M., Neumann, H., 2007. A comprehensive head pose and gaze database. In: 3rd IET International Conference on Intelligent Environments, Ulm (Germany). pp. 455–458.

[35] Zhang, Y., Bulling, A., Gellersen, H., 2012. Towards pervasive eye tracking using low-level image features. In: Symposium on Eye Tracking Research and Applications. pp. 261–264.

[36] Zhou, S. K., Georgescu, B., Zhou, X. S., Comaniciu, D., 2005. Image based regression using boosting method. In: ICCV. pp. 541–548.

[37] Zimmermann, K., Matas, J., Svoboda, T., 2009. Tracking by an optimal sequence of linear predictors. TPAMI 31, 677–692.