

Link to data:

<https://atreus.informatik.uni-tuebingen.de/seafile/d/8e2ab8c3fdd444e1a135/?p=%2FEyeLad%20LabelingTool&mode=list>

EyeLad:Remote Eye Tracking Image Labeling Tool

Supportive eye, eyelid and pupil labeling tool for remote eye tracking videos.

Wolfgang Fuhl¹, Thiago Santini¹, David Geisler¹, Thomas Kübler¹, and Enkelejda Kasneci¹

¹*Perception Engineering, University of Tbingen, Tbingen, Germany*

{wolfgang.fuhl, thiago.santini, david.geisler, thomas.kuebler, enkelejda.kasneci}@uni-tuebingen.de

Keywords: data labeling, image processing, feature tracking, object detection, eye tracking data, remote eye tracking

Abstract: Ground truth data is an important prerequisite for the development and evaluation of many algorithms in the area of computer vision, especially when these are based on convolutional neural networks or other machine learning approaches that unfold their power mostly by supervised learning. This learning relies on ground truth data, which is laborious, tedious, and error prone for humans to generate. In this paper, we contribute a labeling tool (EyeLad) specifically designed for remote eye-tracking data to enable researchers to leverage machine learning based approaches in this field, which is of great interest for the automotive, medical, and human-computer interaction applications. The tool is multi platform and supports a variety of state-of-the-art detection and tracking algorithms, including eye detection, pupil detection, and eyelid coarse positioning. Furthermore, the tool provides six types of point-wise tracking to automatically track the labeled points. The software is openly and freely available at: www.ti.uni-tuebingen.de/perception

1 INTRODUCTION

Leveraging the power of machine learning methods is particularly interesting in tasks where algorithmic approaches are not able to cover the plurality of scenarios in which the algorithm is expected to perform. One such scenario is eye tracking, in which a key step is the correct detection of eye features such as the pupil, eye corners, and eyelids. Whereas the algorithmic detection rate of eye features is acceptable in constrained scenarios, in unconstrained and realistic ones they can be as low as 33% (Fuhl et al., 2016d; Fuhl et al., 2016e). It is worth noticing that not only is eye tracking challenging, but it also offers remarkable opportunities for automotive, medical, and human-computer interaction applications – e.g., driver activity recognition (Braunagel et al., 2015), medical disorder identification (Holzman et al., 1974), and usability research (Jacob and Karn, 2003).

Similarly to other fields, unsupervised machine learning has been employed in real eye tracking applications – e.g., automatic identification of eye movements (Tafaj et al., 2012). However, most applications are based on supervised learning as these methods have been shown to reach state-of-the-art performance in some tasks such as appearance-based gaze estimation (Zhang et al., 2015; Wood et al., 2016), pupil detection (Fuhl et al., 2016c), and blink detec-

tion (Appel et al., 2016). While powerful, these supervised learning methods rely on statistical learning and require a significant amount of annotated data to learn. For example, Krizhevsky et al. (Krizhevsky et al., 2012) had to employ about 1.3 million annotated images to achieve a robust decision function. Therefore, the capability of quickly annotating data is a driving factor to leverage supervised machine learning methods. As such, to reach acceptable performance in the detection of eye features across all use cases, an outstanding amount of ground-truth is required.

To accomplish this task, in this work we introduce a new open-source tool to annotate remote eye tracking images such as the ones shown in Figure 1. These images are relevant, for instance, in desktop settings (Jacob and Karn, 2003), interactive displays (Zhu and Ji, 2004), and automotive scenarios (Braunagel et al., 2015). Our tool makes it possible to use prior knowledge of previous images and also includes the state-of-the-art in detection algorithms for automatic labeling. In the following sections, the tool’s Graphical User Interface, built-in algorithms, and provided features for tracking are explained and visualized (when applicable). As a result, our tool provides a solid framework for quick and semi-automated annotation of remote eye tracking data.

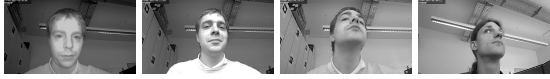


Figure 1: Example images from the data set provided by (Fuhl et al., 2016a). Such images are typically acquired from remote eye trackers – e.g., those built-in in car cockpits.

2 RELATED WORK

Annotation or labeling tools exist for different kinds of tasks. A more distant annotation tool for Indian languages is DONLabel (Deivapalan et al., 2008) which was developed due to the lack of available labeled data for this language. It can handle log audio recordings ≈ 60 seconds, can display text in different Indian languages, is usable through the web browser and there is no special knowledge needed to use it. The provided features are parametrized automatic segmentation and labeling, allows segment boundaries to be edited, removed or added, can play segments separately and zooming into segments.

An annotation tool for objects in images is SmartLabel (Wu and Yang, 2006). It is a semi automatic labeling tool where the user has to specify only a small region inside the object. SmartLabel (Wu and Yang, 2006) then tries to segment the complete object and in addition similar objects in the image. For the object completion SmartLabel uses Gaussian Random Fields (Gudder, 1978) with iterated harmonic energy minimization.

Another tool for object annotation is LabelMe (Russell et al., 2008), which was developed at the MIT. It is a web based tool allowing the user to define object outlines in images. In the background the annotated data is stored in a database. The objects are additionally described or named in words, which are enhanced by using WordNet (Miller, 1995).

A labeling tool for 2D sketches was developed by Worin et al. (Wolin et al., 2007). The authors focused on an efficient fragmentation interface combining automatic and manual techniques, pen based selection and labeling and visual feedback to the user.

Image Region Labeling Software (Goswami et al., 2016) is a MATLAB toolbox and allows the user to set polygonic regions with labels. Each region is set up based on vertices which can be changed, deleted or moved.

The Video Performance Evaluation Resource Ground Truth authoring tool (ViPER-GT) (Doermann and Mihalcik, 2000), written in Java, is developed for labeling regions in any kind of video. It allows to create descriptors for regions and enhance them with attributes. In addition it offers a time line and also zooming and panning.

Eye Picker (Bolme, 2016) is a Python GUI tool for setting and modifying landmarks on images. It is capable of handling complete folders with images and stores the labels as comma separated CSV file.

Recently developed annotation tools are Mindtagger (Shin et al., 2015) for labeling knowledge base, pUltra (Mavridou et al., 2016) for fluorescent annotation of Enterobacteria, and AVclass (Sebastián et al., 2016), which is used for malware labeling.

Some of the above mentioned annotation tools are not related to image processing but serve the same purpose, supporting annotators in their work which allows the validation of algorithms and the usage of machine learning techniques. Based on our knowledge there exist no tool specifically made for remote eye tracking data annotation, which is the motivation behind this work.

3 LABELING TOOL

In this section, the developed labeling tool for remote eye tracking data is described in detail. First the GUI will be shown, and its elements described. Afterwards all built-in algorithms for object detection and tracking are described. The last subsection describes planned improvements for the future.

3.1 GUI

Figure 2 shows the GUI of the proposed labeling tool. The global settings are selected in the top left red box (Figure 2A). These settings allow to choose tracking features, e.g., the intensity of the surrounding region, the gradient of this image region, or the binary result of the Canny edge detector (Canny, 1986). For the Canny edge detector, we used the implementation provided by Fuhl et al. (Fuhl et al., 2015) due to the automatic low-high threshold selection and the Gaussian derivative for gradient calculation. In that section of the GUI, one can also select the current video under annotation, choose between tracking all points from the previous frame or switching to the next frame and tracking everything. The selection box on the bottom right of this section shows the total video frame count, current frame, and allows navigating through the video.

The main visualization area (Figure 2E) shows the current frame from the video. This region is used for eye annotation. In the green box on the bottom left (Figure 2B), eye annotation controls buttons are shown. These controls allow to automatically detect the eyes, track the eyes from the previous frame or remove/add eye annotations. It is worth noticing that,

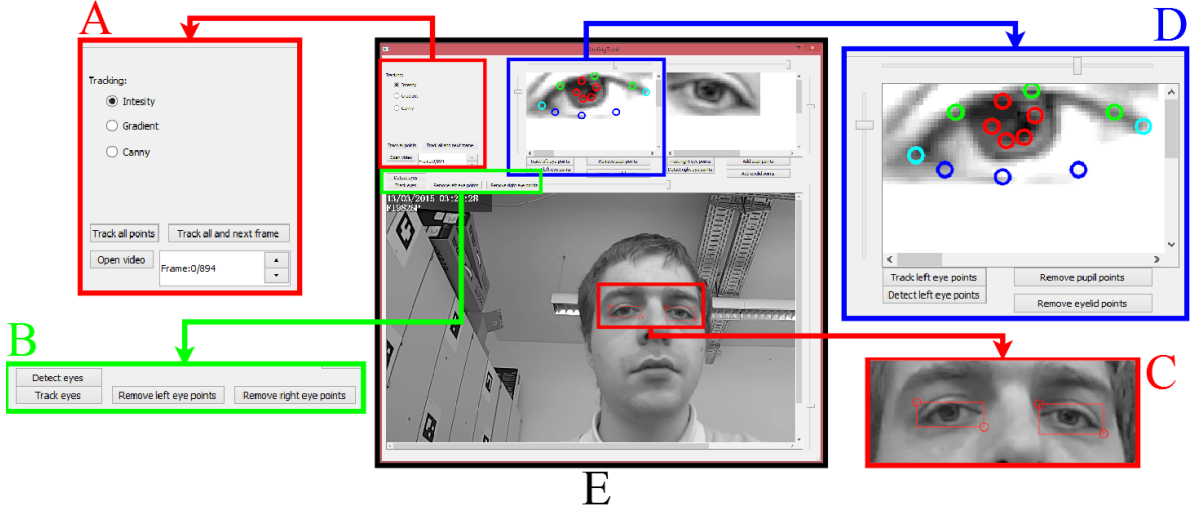


Figure 2: The graphical user interface of our labeling tool (E). The red box on the left (A) shows the general adjustment settings and the frame counter with position. The right red box (C) shows the labeled eyes, where the circles are movable by mouse. The green box (B) shows the main window buttons and the blue box (D) the buttons needed for eye feature labeling. The slider above the main window is for normalization and the slider on the right side for zooming in the same.

in its current state, the tool assumes only one person is visible in the frame; future iterations of the tool are planned to include cases where multiple subjects are present.

The slider above the current frame visualization allows the user to adjust the normalization of that image, increasing the contrast. Similarly, sliders can be found above the left and right eye boxes to adjust the contrast in the eye box images (see the top of Figure 2D). The first normalization step is given by

$$I(x,y) = \begin{cases} I(x,y) & \text{if } I(x,y) < H_p(I) \\ H_p(I) & \text{otherwise} \end{cases}, \quad (1)$$

where $I(x,y)$ is the intensity value of the image at pixel x,y and $H_p(I)$ is the relative threshold based on the intensity histogram of the image. p is the value specified by the slider in the range $[0,1]$, and specifies the percentage of intensity values used to select the threshold. Afterwards, the remaining intensity values are stretched over the range $[0,255]$, i.e.,

$$I(x,y) = \frac{255 * (I(x,y) - \min(I))}{\max(I) - \min(I)}. \quad (2)$$

Examples of this normalization are shown in Figure 3. As can be seen in the first row, the normalization allows the user to more easily and accurately identify the pupil border points and the eyelid points. Furthermore the second row is an example where the illumination of the eye was insufficient due to the head rotation. Additionally, the reflection (small white dot and glass frame) forces the imaging sensor to compress the eye intensity values, leading to a large gap in



Figure 3: Normalized images using equation 1 and equation 2. Each row shows the original input image (first column) and normalized images (second and third columns).

the intensity histogram. As a result, the image is perceived as mostly dark by the annotator. The second image in this row shows the same image normalized ($p = 0.75$), allowing the annotator to clearly see the eye features.

The slider on the right side of the current frame visualization allows the user to adjust the zoom within the range $[25\%, 500\%]$. This feature is also present in the eye boxes (see right side of Figure 2D).

In Figure 2, the red box on the bottom right highlights the eye boxes (Figure 2C). These boxes are defined by two points marked by red circles, which can be dragged and dropped with the mouse; during this process, the region of interest is automatically updated.

The blue box on the top right (Figure 2D) shows a detailed view for the controls and visualization of the right eye feature annotation area. Here the pupil, eyelid and eye corner points can be tracked using information from the previous frame based on the features set in the global settings (Figure 2A). For pupil

detection we integrated ElSe (Fuhl et al., 2016d) with the modifications from (Fuhl et al., 2016a), enabling it to be used for remote eye tracking images. For automatic eyelid detection we modified the method proposed in (Fuhl et al., 2016b) to return equally distant points from the Bézier spline. The eye corner detection uses the method (Fuhl et al., 2016b) to identify eyelids and selects the two points with maximum distance as eye corners. The red circles in the blue eye box (Figure 2D) are the pupil outline points, the green circles are the upper eyelid points, the dark blue circles are the lower eyelid points, and the cyan points are the eye corners. These can be toggled using the Add/Remove buttons.

The saving and loading of labeling points is done in the background as to not interrupt the user. For each change, everything is saved and, if a video is loaded, an equally named txt file is searched and opened (or created if it does not exist). If the file has not the correct data format a backup is created. The data format used by our tool is CSV, where we use semicolon as data separators.

3.2 Eye Detection

For eye detection, we integrated the Haar Cascade (Viola and Jones, 2004) face and eye detection from OpenCV (Bradski, 2000). In the first step, we used the face detection selecting only the top result. In the resulting face region, we use the eye detection from OpenCV (Bradski, 2000) and select only the two top results. The resulting two squares are then converted into our two point format and visualized as shown in Figure 2C.

3.3 Pupil Detection

The integrated pupil detection algorithm is ElSe (Fuhl et al., 2016d), which in its original form starts with edge detection and filtering. Due to the low resolution from the eye region in remote images, this step is likely to fail. In (Fuhl et al., 2016a), a modification using only the second step of ElSe was proposed (Fuhl et al., 2016d) and evaluated on remote images, outperforming existing methods. Therefore, we integrated this implementation, which is a surface difference circle weighted by a mean circle on the inverted intensity. This, however, only yields a rough pupil center estimation. We used this algorithm with different circle sizes and selected the result with maximal response. Afterwards, equally distant points on the resulting circle are selected as pupil contour points.



Figure 4: The three kinds of features used for tracking. The first image is the input image and represent raw intensity values. The second image is the gradient magnitude, and the last image is the result from the Canny edge detection.

3.4 Eyelid and Eye Corner Detection

For eyelid point selection, we integrated the method described in (Fuhl et al., 2016b). The algorithm starts by calculating a likelihood map based on horizontal edge value, mean, standard deviation and skewness. On this likelihood map, a Canny edge detector is applied. Afterwards pairs of edges are evaluated based on enclosed intensity, obliquely to one another and the shift of their mean position. The highest rated pair is chosen as eyelids. This method was developed for fast eyelid aperture estimation but also delivers an eye lid outline based on two combined Bézier curves. In the aperture estimation the two most distant points from those two curves are searched, which we use as eye corners. On the upper and lower curve, equally distant points are set as eye lid points.

3.5 Point Tracking

For tracking, we used the surrounding region of each labeled point and searched in the new image the position that minimizes the function

$$T(x, y, I, N) = \sum_{i=-\frac{n}{2}}^{\frac{n}{2}} \sum_{j=-\frac{n}{2}}^{\frac{n}{2}} |I(i, j) - N(x + i, y + j)|, \quad (3)$$

where I is the image patch of the labeled image, and N is the new image in which the position has to be searched. x, y is the current candidate location and n the patch window size. We integrated different tracking features, which means that I and N in Equation 3 can be intensity, gradient, or binary images. In figure 4, examples of all three features are shown on the complete image. In our implementation the Canny is calculated on the small inspected patches and not the complete image due to the possible non existence of edge pixels. The first image shows the input or raw intensity values. In the second image in figure 4 the gradient magnitude of the first Gaussian derivative is shown. The last image shows the Canny edge detector (Canny, 1986) response.

3.6 Future Improvements

Due to the specialization of our tool to remote settings, we intent to include further point sets for labeling. This includes nose, eyes, eye brows, face outline, and iris, which are important for head pose estimation and face recognition, making the labeling tool more generic and applicable to other face related tasks. In addition, due to the increasing interest in crowd tracking (Saxena et al., 2008; Eshel and Moses, 2008), we plan to include capabilities annotation capabilities when more than a single subject is present in the image. This could be useful to develop algorithms estimating the gaze of those subjects supporting the research in this area.

Planned algorithmic improvements are the integration of histogram of oriented gradients (HOG) (Dalal and Triggs, 2005) for eye and face detection and the convolutional neuronal networks from openFace (Amos et al., 2016), which are based on the Google FaceNet (Schroff et al., 2015). For tracking, we plan to add the ORB features (combination of FAST Keypoint Orientation (Viswanathan, 2009), Rotation-Aware Brief (Calonder et al., 2010)) (Rublee et al., 2011), Scale-invariant feature transform (SIFT) (Lindeberg, 2012), Maximally stable extremal regions (MSER) (Matas et al., 2004), and Speeded Up Robust Features (SURF) (Bay et al., 2008) due to their robustness, which could be useful for extremely challenging images.

Additionally, we want to investigate the possibility to apply learning based tracking approaches as in SmartLabel (Wu and Yang, 2006), where they used Gaussian Random Fields (Gudder, 1978). In the remote eye tracking case, it would also be possible to apply convolutional neuronal networks on image patches or training Random regression forests (Svetnik et al., 2003) on image features. Due to the possibility of using our labeling tool on color images further improvements include also the visualization of points and the justifiability of their size.

4 CONCLUSIONS

We propose a labeling tool only dependent on OpenCV and Qt, making it portable and cross-platform. We included different features for tracking and state-of-the-art algorithms for automatic detection. The user can easily adjust image contrast and zoom layer, improving usability and accuracy of the resulting labels. Each labeling point can be dragged and dropped with the mouse for an intuitive interaction. The tool was validated by labeling more than

35.000 images¹. We hope it will serve the community and are open for suggestions to further improve it.

REFERENCES

- Amos, B., Ludwiczuk, B., and Satyanarayanan, M. (2016). Openface: A general-purpose face recognition library with mobile applications. Technical report, CMU-CS-16-118, CMU School of Computer Science.
- Appel, T., Santini, T., and Kasneci, E. (2016). Brightness- and motion-based blink detection for head-mounted eye trackers. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp Adjunct 2016, Heidelberg, Germany, September 12-16, 2016*, pages 1726–1735.
- Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359.
- Bolme (2016). Eye picker. <https://github.com/bolme/pyvision/wiki/Eye-Picker>. Accessed: 16-11-27.
- Bradski, G. (2000). Opencv. *Dr. Dobb's Journal of Software Tools*.
- Braunagel, C., Kasneci, E., Stolzmann, W., and Rosenstiel, W. (2015). Driver-activity recognition in the context of conditionally autonomous driving. In *IEEE 18th International Conference on Intelligent Transportation Systems (ITSC)*.
- Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer.
- Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, 1(6):679–698.
- Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE.
- Deivapalan, P., Jha, M., Guttikonda, R., and Murthy, H. A. (2008). Donlabel: an automatic labeling tool for indian languages. *Energy*, 2:4.
- Doermann, D. and Mihalcik, D. (2000). Tools and techniques for video performance evaluation. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 4, pages 167–170.
- Eshel, R. and Moses, Y. (2008). Homography based multiple camera detection and tracking of people in a dense crowd. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.
- Fuhl, W., Geisler, D., Santini, T., Rosenstiel, W., and Kasneci, E. (2016a). Evaluation of state-of-the-art pupil detection algorithms on remote eye images. In *Proceedings of the 2016 ACM International Joint Confer-*

¹These are currently under a non disclosure agreement and could not be made available at the time of writing.

- ence on Pervasive and Ubiquitous Computing, *UbiComp Adjunct 2016, Heidelberg, Germany, September 12-16, 2016*, pages 1716–1725.
- Fuhl, W., Kübler, T., Sippel, K., Rosenstiel, W., and Kasneci, E. (2015). Excuse: Robust pupil detection in real-world scenarios. In *International Conference on Computer Analysis of Images and Patterns*, pages 39–51. Springer.
- Fuhl, W., Santini, T., Geisler, D., Kübler, T. C., Rosenstiel, W., and Kasneci, E. (2016b). Eyes wide open? eyelid location and eye aperture estimation for pervasive eye tracking in real-world scenarios. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp Adjunct 2016, Heidelberg, Germany, September 12-16, 2016*, pages 1656–1665.
- Fuhl, W., Santini, T., Kasneci, G., and Kasneci, E. (2016c). Pupilnet: Convolutional neural networks for robust pupil detection. *arXiv preprint arXiv:1601.04902*.
- Fuhl, W., Santini, T., Kübler, T., and Kasneci, E. (2016d). Else: Ellipse selection for robust pupil detection in real-world environments. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications, ETRA '16*, pages 123–130, New York, NY, USA. ACM.
- Fuhl, W., Tonsen, M., Bulling, A., and Kasneci, E. (2016e). Pupil detection for head-mounted eye tracking in the wild: an evaluation of the state of the art. *Machine Vision and Applications*, pages 1–14.
- Goswami, D., Kalkan, S., and Krueger, N. (2016). Probabilistic classification of image structures. <http://de.mathworks.com/matlabcentral/fileexchange/21871-image-region-labeling-software>. Accessed: 16-11-27.
- Gudder, S. (1978). Gaussian random fields. *Foundations of Physics*, 8(3-4):295–302.
- Holzman, P. S., Proctor, L. R., Levy, D. L., Yasillo, N. J., Meltzer, H. Y., and Hurt, S. W. (1974). Eye-tracking dysfunctions in schizophrenic patients and their relatives. *Archives of general psychiatry*, 31(2):143–151.
- Jacob, R. and Karn, K. S. (2003). Eye tracking in human-computer interaction and usability research: Ready to deliver the promises. *Mind*, 2(3):4.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.
- Lindeberg, T. (2012). Scale invariant feature transform. *Scholarpedia*, 7(5):10491.
- Matas, J., Chum, O., Urban, M., and Pajdla, T. (2004). Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767.
- Mavridou, D. A., Gonzalez, D., Clements, A., and Foster, K. R. (2016). The pultra plasmid series: A robust and flexible tool for fluorescent labeling of enterobacteria. *Plasmid*, 87:65–71.
- Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.
- Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. IEEE.
- Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. (2008). Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3):157–173.
- Saxena, S., Brémond, F., Thonnat, M., and Ma, R. (2008). Crowd behavior recognition for video surveillance. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 970–981. Springer.
- Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823.
- Sebastián, M., Rivera, R., Kotzias, P., and Caballero, J. (2016). Avclass: A tool for massive malware labeling. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 230–253. Springer.
- Shin, J., Ré, C., and Cafarella, M. (2015). Mindtagger: a demonstration of data labeling in knowledge base construction. *Proceedings of the VLDB Endowment*, 8(12):1920–1923.
- Svetnik, V., Liaw, A., Tong, C., Culberson, J. C., Sheridan, R. P., and Feuston, B. P. (2003). Random forest: a classification and regression tool for compound classification and qsar modeling. *Journal of chemical information and computer sciences*, 43(6):1947–1958.
- Tafaj, E., Kasneci, G., Rosenstiel, W., and Bogdan, M. (2012). Bayesian online clustering of eye movement data. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 285–288. ACM.
- Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *International journal of computer vision*, 57(2):137–154.
- Viswanathan, D. G. (2009). Features from accelerated segment test (fast).
- Wolin, A., Smith, D., and Alvarado, C. (2007). A pen-based tool for efficient labeling of 2d sketches. In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling*, pages 67–74. ACM.
- Wood, E., Baltrušaitis, T., Morency, L.-P., Robinson, P., and Bulling, A. (2016). Learning an appearance-based gaze estimator from one million synthesised images. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications, ETRA '16*, pages 131–138, New York, NY, USA. ACM.
- Wu, W. and Yang, J. (2006). Smartlabel: An object labeling tool using iterated harmonic energy minimization. In *Proceedings of the 14th ACM international conference on Multimedia*, pages 891–900. ACM.
- Zhang, X., Sugano, Y., Fritz, M., and Bulling, A. (2015). Appearance-based gaze estimation in the wild. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4511–4520.
- Zhu, Z. and Ji, Q. (2004). Eye and gaze tracking for interactive graphic display. *Machine Vision and Applications*, 15(3):139–148.