Creating Slides from Video Lecture

Maksym Shylo[1], Anton Smirnov[2,3], and Valerii Krygin[2,4,1]

[1] National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine
maksym.shylo.work@gmail.com
https://ipt.kpi.ua/
[2] IRTC for IT and Systems of the NAS of MES of Ukraine, Kyiv, Ukraine
http://www.irtc.org.ua/
[3] Jazzros, Kyiv, Ukraine
anton.smn@protonmail.com
https://www.jazzros.com/
[4] Apostera, Kyiv, Ukraine
https://www.apostera.com/
valeriy.krygin@gmail.com

Abstract. Video recordings of lectures are no longer a rarity in the conditions of distance learning. Videos may be in an inconvenient format for students or contain different artifacts due to compression, camera quality, and other factors. It is useful to have a presentation of the study material, which contains only the text from the board because such a view of the material is most similar to the compendium. Moreover, some of the text may not be visible due to occlusions from people. To address these issues, we employ a neural network that removes people from video via content-aware-inpainting. To reduce duplication of slides due to camera shaking, we perform video-stabilization as a preprocessing step. Finally, we create slides by comparing changes in the frames, color correcting, and binarizing them. As a result, we get slides with the extracted text or drawings from the board, which will help to simplify the creation of e-learning materials for both new and existing lecture recordings. Teachers will also be able to quickly provide lecture material to students even if they teach several complex subjects.

Keywords: Video lecture · Slides from video · Removing a person from video · Video stabilization · Binarization

1 Introduction

Students and teachers have a great need to simplify the learning process and bring the lecture material as close as possible to the usual form of the synopsis. Since we live in a time when almost everyone has a smartphone at hand, a lecturer will need a phone or a webcam on a computer to record a lecture. The average student will not have to worry about the syllabus, as he receives the already processed and digitized text from the lecture in a comfortable slide

format. This paper describes an algorithm for creating slides based on a video lecture (fig. 1). A neural network is used to obtain the video without a lecturer. To avoid duplicated slides and better cleaning quality, we stabilize the video with a homography. Our slide creation algorithm uses Laplace operator, Otsu binarization, and mask extension. At the end of the program we have slides of the board (black- or whiteboard) in the form of a slideshow.

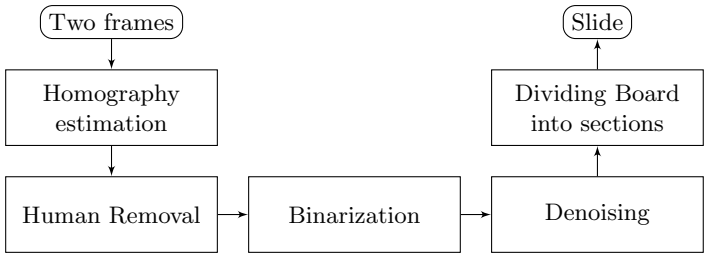


Fig. 1: Pipeline

2 Related Works

2.1 Whiteboard Scanning and Image Enhancement

This work describes the system of whiteboard edges detection, determination of the position of the board and reproduction of digitized material from photographs containing parts of the boards from different angles [12]. The method's aim is to scan the whiteboard and give its content with a rectangular shape. The algorithm works only with photos without obstacles while ours works with videos. Our program can process video lectures with different board color, while it's not known whether this method can handle chalkboards.

2.2 Whiteboard Disclosure Using Background Subtraction and Object Tracking

The work [4] processes the video recording, detects the teacher and removes him from the video. The authors emphasize that their program can also detect boards that are divided into parts. This implementation works good with different video resolutions and board conditions. In comparison with our work, this one does not include the creation of a slideshow and whiteboard digitizing.

2.3 Real-Time Whiteboard Capture and Processing Using a Video Camera for Teleconferencing

The work [6] presents an algorithm that analyzes a video, detects pen strokes and foreground objects. The big plus of this work is real-time video whiteboard

digitizing. They use contour stability analysis to detect a lecturer from the background, while we experiment with a neural network. The quality of result images represented in this work is quite low, so the whiteboard content is not readable. Also, we do not know how this algorithm behaves if the camera is shaking.

2.4 Whiteboard Video Summarization via Spatio-Temporal Conflict Minimization

The work [2] is most similar to ours. The authors present a method that creates image summaries(analog of ours slideshow) of handwritten whiteboard content from lecture videos recorded with still cameras. Their algorithm generates a Spatio-temporal index for table content. It helped for creating a temporal segmentation to detect key-frames. Also, they use a random forest algorithm, Otsu binarization for getting lecture summarization. The main question to this work is the output quality under the shaking camera conditions.

3 Video Stabilization

Shaking camera is not a surprising factor for video lectures. The camera may shake accidentally when it's attached to a desk where students write the lecture. Intentional camera movements should be considered as well: in order to track a big board with a low-resolution camera you need to point it to areas of interest. These situations are very natural and they may obstruct the lecture perception. We propose to estimate the image distortion with a homography in coarse-to-fine manner.

3.1 Homography Estimation

The board is assumed to be flat. Transformation of a projection of the board to a matrix of a moving pinhole camera (camera that produces no radial distortions) can be described with homography [5]. Camera movements during the lecture may vary from insignificant subpixel shifts to flicks. Our aim is to produce the video where the camera behaves like a static one.

Given two frames P and N (Previous and Next), we can get four pairs of corresponding points from them

$$M = \left\{ \left\langle \boldsymbol{x}_i^P, \boldsymbol{x}_i^N \right\rangle : i = \overline{1,4} \right\}, \quad \boldsymbol{x}_i^P \in \mathbb{R}^2 \times \{1\}. \tag{1}$$

Corresponding points are those ones which are projections of the same point in space. If these points are points of a flat board, they are connected by the relation

$$H \cdot \boldsymbol{x}_i^N \approx c \cdot \boldsymbol{x}_i^P, \quad \forall i = \overline{1,4}. \tag{2}$$

Note that the points are from the homogeneous space, so to calculate coordinates of their projections onto the image we need to divide the first and second

coordinates by the third. We assume that none of the points are at the infinity and the third coordinate never turns in zero.

The equation (2) can be easily solved with respect to unknown homography H. This matrix can be used to stabilize the video — we apply it to entire N and it fits the P and ready for further processing.

3.2 Coarse-to-Fine Image Registration

Finding the set of points M for (1) is a difficult task. Image coordinates are discrete, so it's impossible to achieve absolutely accurate correspondence. We need a subpixel accuracy for further procedures to work well. For this purpose we propose the coarse-to-fine image registration pipeline.

Initial Matching As the first homography estimation (which can be enough) we can use feature descriptors like SIFT, SURF or ORB [1,7,10] to find feature points and RANSAC [3] to estimate the homography. Let's now denote the M' as a set of all correspondences found by a feature matcher. The problem we would like to solve is

$$\sum_{\langle \boldsymbol{x}^P, \boldsymbol{x}^N \rangle \in M} \left[\!\!\left[\left\| \frac{H \cdot \boldsymbol{x}_i^N}{\left(H \cdot \boldsymbol{x}_i^N\right)_z} - \boldsymbol{x}_i^P \right\| < \varepsilon \right]\!\!\right] \to \max_{\substack{H \in \mathbb{R}^{3\times 3}, \\ \det H \neq 0}}, \quad \varepsilon > 0. \tag{3}$$

It's intractable in general, so the common approach is to choose four random pairs from M', solve the (2) and check the sum (3). We repeat the procedure as many times as we want and choose the homography that suits this penalty better than others.

Pyramidal Homography Estimation We want to achieve subpixel accuracy of images matching. If it doesn't happen with the previous matching, we go to the next step. This step is another coarse-to-fine method in this coarse-to-fine approach. To distinguish them we call this one "pyramidal homography estimation". The idea is to find exactly four pairs of corresponding points for (1). Let us introduce a function T that takes an image N and a set of corresponding points M, calculates homography H, applies it to N and returns a transformed image. Given a loss function $L : \mathbb{R}^{w\times h} \times \mathbb{R}^{w\times h} \to \mathbb{R}$ (for example, L_2 norm in $\mathbb{R}^{w\times h}$-dimensional Euclidean space), we can formulate the problem of finding M as

$$L\left(P, T\left(N, M\right)\right) \to \min_M .$$

The problem is very hard in general too, so we should use brute-force-like approach here. We have already improved the P and N match with the previous coarse step, so now we can use brute-force search with a relatively small number of steps. For example, we get four corner points on the P image and find their correspondences in square of size 2 with step 0.5 on the N image. This needs $8^{\frac{2}{0.5}} = 2^{12}$ calls of L and T functions. The number looks small but the L may

be a sum of squares of pixelwise differences. Thus, for $1'000 \times 1'000$ images we need to transform nearly four billions pixels to find the best M for one pair of frames. This takes a very long time so we use the pyramidal approach: instead of iteration over all possible combinations we start with only three offsets for each corrdinate: -2, 0, 2. Then, after finding the best ones of them, we start again with new three offsets $b_i - 1$, b_i, $b_i + 1$ (where b_i is different for each coordinate). And so on as long as we need.

Brute-force search needs $8^{\frac{\ell}{s}}$ steps, where ℓ is a square size and s is the step size. Pyramidal approach needs $8^{3 \cdot \frac{\ell}{s}}$, which is 8^6 for our example with $\ell = 2$ and $s = 0.5$.

4 Removing a Person from Video

In this section we describe how a person is removed from video frames. Removing a person from a frame is essential because it resolves the problem of occlusion, providing frames with only content of the board thus making later stages of the pipeline simpler.

For this task we employ a neural network for human detection, which is one of possible choices to tackle this problem.

4.1 Neural Network Architecture

For neural network we chose a pretrained network from YOLO (You-Only-Look-Once) [8,9] family of neural network architectures. Specifically YOLOv5s, which is an incremental improvement both in accuracy and in performance over classical YOLOv3 architecture.

The reason for this, is because we wanted to process videos in near real-time fashion and this network is able to run at 30 FPS even on mobile devices.

This neural network frames the problem of object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. Thus, it predicts both bounding boxes and class probabilities directly from images in one evaluation.

The architecture of this neural network is separated into two parts, the backbone and the head. The backbone consists of convolutional layers followed by bottleneck layers used for feature extraction, while the head consists of convolution and upsampling layers for the prediction of bounding boxes and its class probabilities.

Output of the neural network is additionally processed by a Non-Maximum-Supression module for bounding box filtering.

4.2 Person Removal

To remove a person we need to first detect it and then inpaint or replace it without modifying the content on the board behind it. Each frame $\mathcal{F}$ we detect all possible humans and mark those regions as regions to inpaint $\mathcal{R}_i \subseteq \mathcal{F}$.

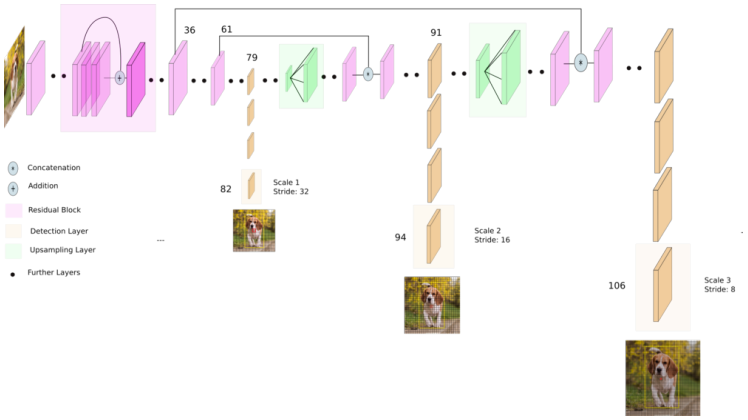


Fig. 2: Simplified YOLOv5 architecture diagram. Credits: Ayoosh Kathuria

Additionally, we keep a buffer $\mathcal{B}$ of unoccluded regions from previous frames. When a region is marked as a region to inpaint $\mathcal{R}_i \subseteq \mathcal{F}$, we retrieve corresponding unoccluded regions from previous frames in that buffer and replace it with them

$$\mathcal{R}_i = \mathcal{B}_{\mathcal{B} \cap \mathcal{R}_i},$$

where $\mathcal{B} \cap \mathcal{R}_i$ is the corresponding region in the buffer $\mathcal{B}$ for the $\mathcal{R}_i$.

We update the buffer $\mathcal{B}$ from the new frames using unoccluded regions, this way we retain the most recent information. Update rule for the buffer

$$\mathcal{B}_{\overline{\mathcal{R}_i}} = \overline{\mathcal{R}_i},$$

where $\overline{\mathcal{R}_i} = \mathcal{F} \setminus \mathcal{R}_i$ is the unoccluded region in current frame $\mathcal{F}$.

This simple technique allows us to keep the original content of the board without any modifications, compared to other inpainting methods. The only drawback of this method is when video stabilization fails to completely eliminate motion of the camera, the whole frame shifts w.r.t previous frames and the buffer can no longer be used for precise inpainting. This results in small artifacts near the edges of the regions.

Another important moment is lighting condition. As it changes over time, for example due to camera refocus or simply due to changes in light sources, content in the buffer will differ slightly in color when used to inpaint newer frames.

In the result we get a relatively clean image of the board, that is used for the next stages in the pipeline. As a further improvement we can additionally track all moving objects, not necessarily humans, using techniques like optical flow and remove them, since we can safely assume that the board is static.

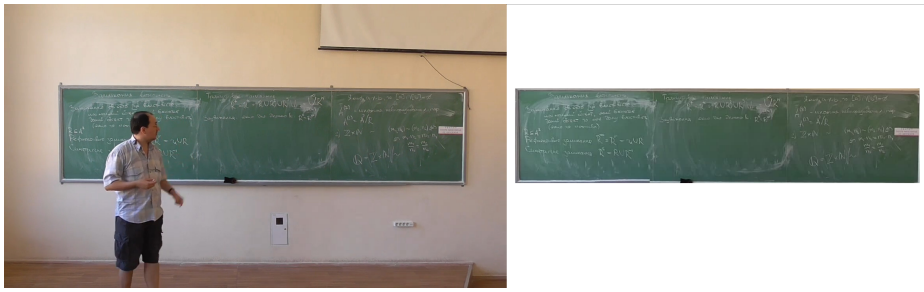
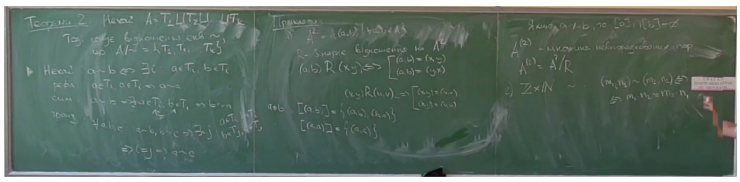


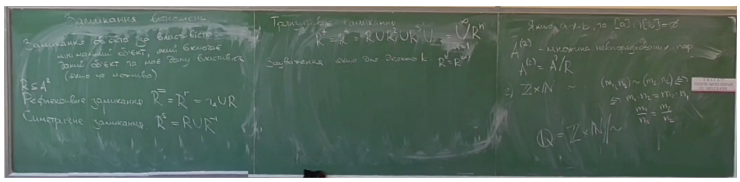
Fig. 3: Comparison of the raw frame from camera and the result of removing a human. Source: https://youtu.be/a7TUp4p-pIk

5 Creating Slides from Video

After removing the lecturer and stabilization, the video is processed to create the slides. We iteratively take two frames: Previous and Next (fig. 4), hereinafter referred to as P and N, respectively. The frames are taken from the video with a certain distance (number of frames) between them. We set the distance because there is no need to check every frame because there is almost no change of content on the board between adjacent frames.



(a) Previous frame



(b) Next frame

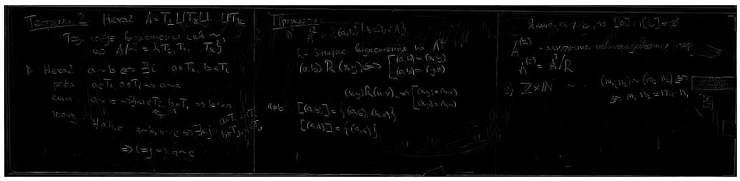
Fig. 4: Example of Previous and Next frames

5.1 Slides Comparison

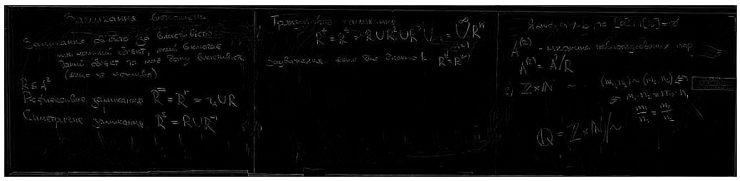
Images are converted from RGB to grayscale format. We used the Laplace operator [11] of finding edges and denoising for the frames. We use this operator

because it gives better results than other differential operators, based on our observations of the input material. This operator reduces the noise level and emphasizes the contents with suitable quality (fig. 5).

$$L(x,y) = \nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \tag{4}$$



(a) Previous frame



(b) Next frame

Fig. 5: Previous and Next frames after laplace operator processing

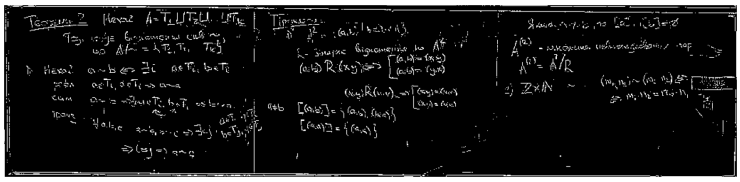
We use the Otsu binarization method [11] to obtain a mask of the contents of the board (fig. 6). It is fast and gives good binarization quality. As we can see on the image mask, the binarization divides the contents of the board into two classes on the example frames.

We use the average of pixelwise XORs to measure the relative difference between the frames P and N (binary analog of the mean squared error). We compare the difference with a threshold provided by a user. This constant plays a significant role in the decision to treat frame N as a new slide or not. Let's introduce the following notation: h — height of the frame, w — width of the frame, $t \in [0;1]$ — the threshold given by the user, $P_{i,j}$, $N_{i,j}$ — values of pixel from the i-th row and j-th column in frames P and N respectively. To measure the relative changes between frames P and N we use

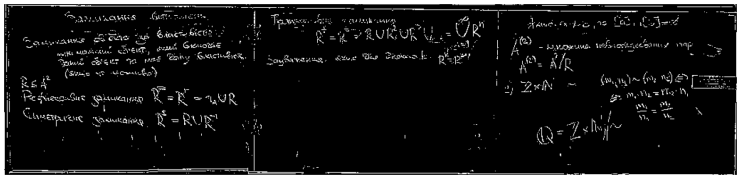
$$k(P,N) = \frac{\sum_{i=1}^{h}\sum_{j=1}^{w} P_{i,j} \oplus N_{i,j}}{h \cdot w}. \tag{5}$$

We say that the frame N is a new slide if its mean XOR from the frame P is more than t

$$s(h,w,t,P,N) = \begin{cases} yes, & t > k(P,N), \\ no, & otherwise. \end{cases} \tag{6}$$



(a) Previous frame



(b) Next frame

Fig. 6: Previous(a) and Next(b) frames after Otsu binarizating

Then we take a new frame as a N and the old N becomes the new P, and the procedure starts again.

5.2 Slide Processing

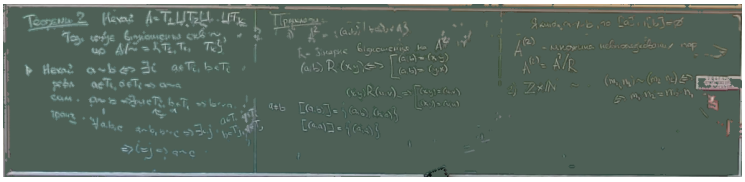
We use the binary dilation [11] for the noise reduction with a circle mask with radius 2px. Then, we use the original RGB frame (the origin of the binary contents mask) and remove all the background using the dilated mask. Then we fill the background by its average color to achieve more aesthetic results (fig. 7).

5.3 Dividing Board into Sections

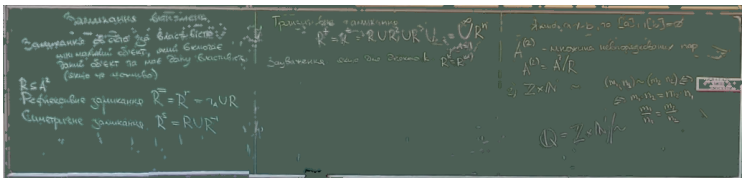
It's good to have slides of the whole board, but what if the board is divided into sections? It means that you will get slides where some parts of the board didn't change. It would be great to have slides that contain only those sections that changed. We created a possibility to get slides if a user indicates board dividing (fig. 8).

Let's define number of slides as d, number of divided sections as q, height of divided parts as g and width of divided part as b. We have H — a matrix of labels with size $|d| \times |q|$, where each element equal 1 if content of this section has changed and 0 otherwise. We have S — a matrix of divided slides with size $|d| \times |q| \times |g| \times |b|$. A constant t is a treshold of changes between $S_{i,j,\cdot,\cdot}$ and $S_{i+1,j,\cdot,\cdot}$ sections. We only take sections where $D_{i,j} = 1$ and get new slides

$$D_{i,j} = \begin{cases} 1 & t > k\left(S_{i,j,\cdot,\cdot}, S_{i+1,j,\cdot,\cdot}\right), \\ 0 & otherwise. \end{cases} \tag{7}$$



(a) Previous frame



(b) Next frame

Fig. 7: Previous(a) and Next(b) frames after applying dilated mask on RGB slides

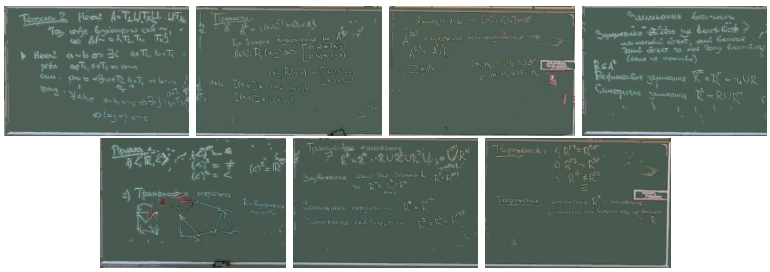


Fig. 8: Divided slides

6 Conclusions and Further Work

We have implemented a working pipeline, that transforms a video lecture to a slide show. On the first stage, we apply stabilization based on pyramidal homography estimation. After that, we remove all detected humans from the frame by inpainting them with the content they are occluding, utilizing a neural network for human detection. Finally, we create slides from the processed frames, using Laplace operator and Otsu binarization to filter out background and retain only board drawings. Additionally, our algorithm is able to create slides, when the board is divided into several parts as is often common in lecture halls.

Our further steps are as follows:

- use more efficient and accurate method of obstacles removing: contours analysis like in [2] or motion tracking instead of artificial neural network;
- use more suitable binarization method to fight shadows and low-quality video (strokes on a board can be 1px wide);

– use vectorization of the resulting writings to add scaling ability, which is helpful for aesthetics and clarity.

References

1. Bay, H., Ess, A., Tuytelaars, T., Gool, L.V.: Speeded-up robust features (surf). Computer Vision and Image Understanding **110**(3), 346 – 359 (2008). https://doi.org/10.1016/j.cviu.2007.09.014, http://www.sciencedirect.com/science/article/pii/S1077314207001555, similarity Matching in Computer Vision and Multimedia
2. Davila, K., Zanibbi, R.: Whiteboard video summarization via spatio-temporal conflict minimization. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). vol. 01, pp. 355–362 (2017). https://doi.org/10.1109/ICDAR.2017.66
3. Fischler, M.A., Bolles, R.C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381395 (Jun 1981). https://doi.org/10.1145/358669.358692, https://doi.org/10.1145/358669.358692
4. Gonzalez, A., Suh, B., soo Choi, E.: Whiteboard disclosure using background subtraction and object tracking (2012)
5. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, 2 edn. (2004). https://doi.org/10.1017/CBO9780511811685
6. He, L.w., Zhang, Z.: Real-time whiteboard capture and processing using a video camera for teleconferencing. Tech. Rep. MSR-TR-2004-91 (September 2004), https://www.microsoft.com/en-us/research/publication/real-time-whiteboard-capture-and-processing-using-a-video-camera-for-teleconferencing/
7. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision **60**(2), 91–110 (Nov 2004). https://doi.org/10.1023/B:VISI.0000029664.99615.94, http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94
8. Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: Unified, real-time object detection. CoRR **abs/1506.02640** (2015), http://arxiv.org/abs/1506.02640
9. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. CoRR **abs/1804.02767** (2018), http://arxiv.org/abs/1804.02767
10. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.R.: Orb: An efficient alternative to sift or surf. In: Metaxas, D.N., Quan, L., Sanfeliu, A., Gool, L.V. (eds.) ICCV. pp. 2564–2571. IEEE Computer Society (2011), http://dblp.uni-trier.de/db/conf/iccv/iccv2011.html#RubleeRKB11
11. van der Walt, S., Schönberger, J.L., Nunez-Iglesias, J., Boulogne, F., Warner, J.D., Yager, N., Gouillart, E., Yu, T., the scikit-image contributors: scikit-image: image processing in Python. PeerJ **2**, e453 (6 2014). https://doi.org/10.7717/peerj.453, https://doi.org/10.7717/peerj.453
12. Zhang, Z., He, L.W.: Whiteboard scanning and image enhancement. Digital Signal Processing **17**(2), 414 – 432 (2007). https://doi.org/10.1016/j.dsp.2006.05.006, http://www.sciencedirect.com/science/article/pii/S1051200406000595