

Comparing different algorithms that generate phosphene images for visual cortical prosthesis

Author: J. Hartjes
Supervisor: Dr Y. Güçlütürk

Radboud University
Nijmegen, the Netherlands

Bachelor's Thesis

June 2019



Radboud Universiteit

Abstract

Visual cortical prosthesis (VCPs) are currently in development and there has already been some speculation about which algorithm is best to use in these implants. Semantic segmentation seems like a obvious choice because the algorithm also gives vision to self-driving cars. However, semantic segmentation is a slow and complex algorithm that predicts around 200 classes, which are not necessary when predicting phosphenes. Therefore, semantic segmentation is compared to two simpler algorithms, edge detection and a proposed neural network that predicts phosphene images from input images. The results indicated that although being the slowest of the methods, semantic segmentation makes the best phosphene images. The edge detection and neural network algorithms need some alterations to be able to make phosphene images that are usable in VCPs.

Introduction

Globally, over 36 million(2015) people suffer from blindness [1]. The consequences for these patients can have negative physical (accidents [2]), mental (social withdrawal and depression [3]), and economical (medical costs) implications. Visual prosthesis provide a way to restore some perception to visually impaired patients. Different approaches of prosthesis are: retinal-, cortical- and optic nerve implants. While all of these approaches have proven to be beneficial for patients [4][5][6], the focus of this paper will be on cortical prosthesis.

Visual cortical prostheses (VCP) are the most invasive approach of the prostheses. However, they are applicable to most of the patients because they require neither an intact eye or optic nerve to provide vision for patients. This is because VCPs interact directly with the visual cortex located in the occipital lobe. They stimulate small groups of neurons in this part of the brain, which results in the patient seeing a small dot of light [7][8]called a phosphene. By combining multiple phosphenes (phosphene mapping), a grid can be made [9]. The VCP can then present images onto this grid that the person will be able to see [10]. A VCP is connected to a camera, which is attached to the patients head, and uses an algorithm to transform images that the camera captures to a representation that the patient can see as a phosphene grid. This way, patients can receive information about their surroundings and perform basic cognitive tasks, for example object- and face recognition as well as tasks related to hand-eye coordination and virtual mobility [11].

Algorithms for VCPs are still in development but there are already promising ideas for these algorithms. These ideas include methods that use trans-formative reality [12] or iconic representations of the visual field [13] to provide prosthetic vision. Other approaches use semantic segmentation/labeling [14], which will be the main focus of this paper.

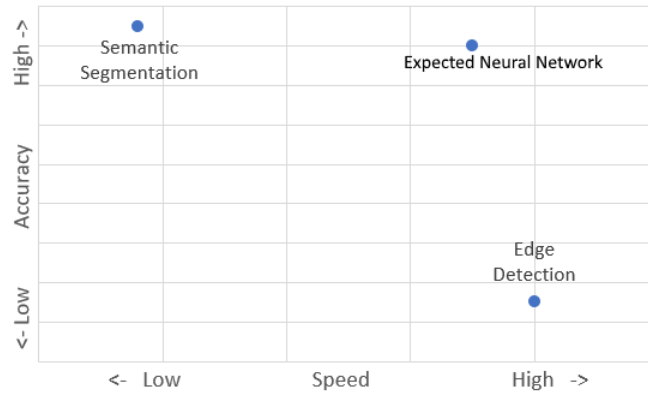
The semantic segmentation approach seems like a logical choice to use in the implants. This is partly because this algorithm also provides good vision for self-driving cars. However, due to the excessive processing requirements for these algorithms, they do not yet seem feasible. Computers with good GPUs can transform around 8 images per second with this method [15]. There are semantic segmentation algorithms that transform images faster, but these come at the price of lesser accuracy [16].

This paper compares the semantic segmentation method to two other methods in terms of accuracy and speed. The first method is a proposed neural network, which is a network that is trained to generate phosphene maps directly from images. Because these images can be trained with a less complex model than the state-of-the-art semantic segmentation models, it could provide a solution for the low frame rate in VCPs.

The second method that I will be testing is edge detection. This method is extremely fast compared to the other methods [17]. It is expected that this approach leads to a less accurate results due to

the noisy nature of edge detection. However, if the results are accurate, then, after more research, this approach could also provide a promising solution to obtain more FPS in VCPs. Picking the right algorithm for VCPs comes with a trade-off between accuracy and speed. My hypothesis is that semantic segmentation will have a high accuracy and low speed, while edge detection will have a low accuracy and high speed. I also expect that the proposed network will have an accuracy close to that of semantic segmentation and a speed close to the speed of edge detection. In figure 1, the visualization of the trade-off and the expected results are depicted.

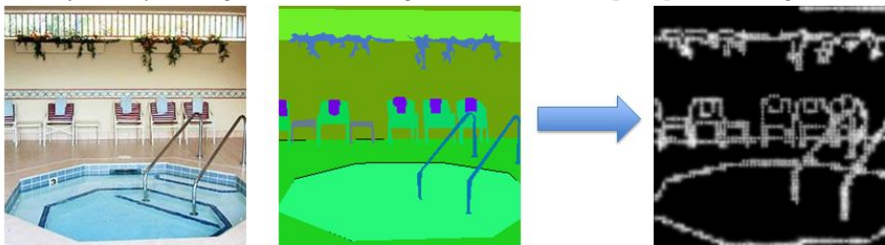
Figure 1: *Visualization of my hypothesis. The axis represent accuracy and speed.*



Methods

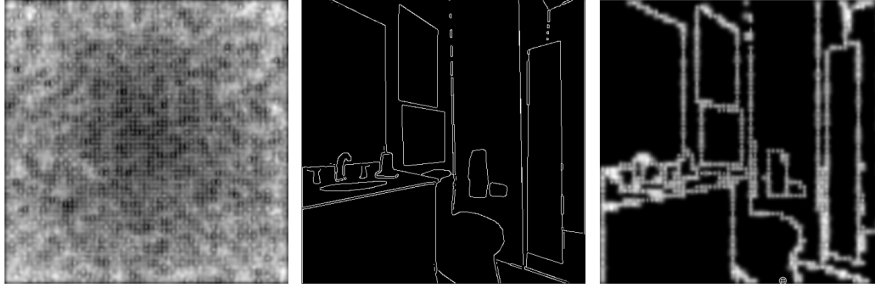
To be able to compare the different methods (semantic segmentation, edge detection and the neural network), every algorithm is used to transform the same images to phosphene maps, so that the comparison is fair. The ADE20K data-set is used for this [18]. The data-set is normally used to train semantic segmentation algorithms and consists of 20.000 training images and 2.000 validation images. The ground truth images of the ADE20K data-set are used to create our own ground truth images. These are then the best possible phosphene map for that corresponding picture (figure 2). This is done by using edge detection on the semantically segmented ground truth, and then transform that to a phosphene map using a phosphene filter. The phosphene filter transforms an

Figure 2: *Generating the ground truth from the ADE20K dataset. The left two pictures show the image and the ground truth of the ADE20K dataset. The right picture is our own ground truth, which is made by transforming the ADE20K ground truth to a phosphene image.*



edge-detected image to a phosphene grid by multiplying the two pictures (figure 3). Phosphene maps will have a size of 63x63 phosphenes, resulting in a grid of 3969 phosphenes, which is close to 4000, the number of electrodes that is possible to use in VCPs.

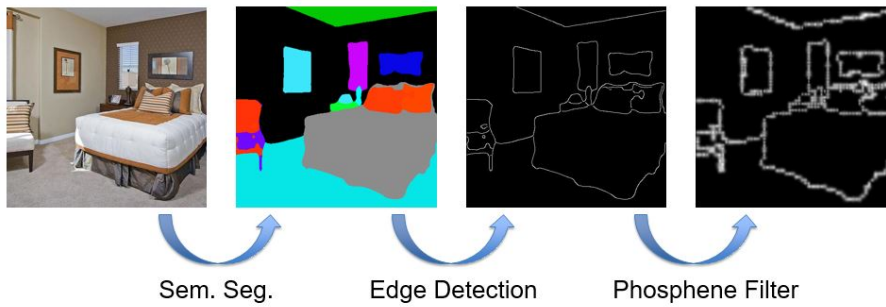
Figure 3: *Example of the phosphene filter. The left picture shows the phosphene grid (including noise) with every phosphene active. The middle picture shows an edge-detected image. The right picture shows the phosphene representation of the picture after multiplying both images.*



Semantic Segmentation Algorithm

To generate phosphene maps with this method, an existing neural network is used to generate the semantic segmented images. This is the PSPresnet50 [19], a network implemented in chainer. Because this network is pre-trained, we only have to call the predict function to get segmented images. After the segmented images are generated, edge detection is used to transform them into phosphene maps, as is shown in figure 4. When the phosphene maps are generated, they are comparable to the ground truth.

Figure 4: *Processing stages of the semantic segmentation algorithm. The left most image is the input, and the right most image is the output.*



Proposed Algorithm

A neural network is trained to generate phosphene images directly from input images. To do this, I make use of an existing network that is used to transform sketches into photo-realistic versions of them [20]. Because this network is used to transform pictures into other representations, we can train it to transform pictures into phosphene representations. This network can be trained with our input images and ground truth to predict phosphene images. The network uses convolutional layers to learn features in the input images and learns to predict phosphene images based on these features by adapting weights accordingly. When the network is trained, it takes the input image and outputs binary phosphene images. This is done for every validation image to generate our data.

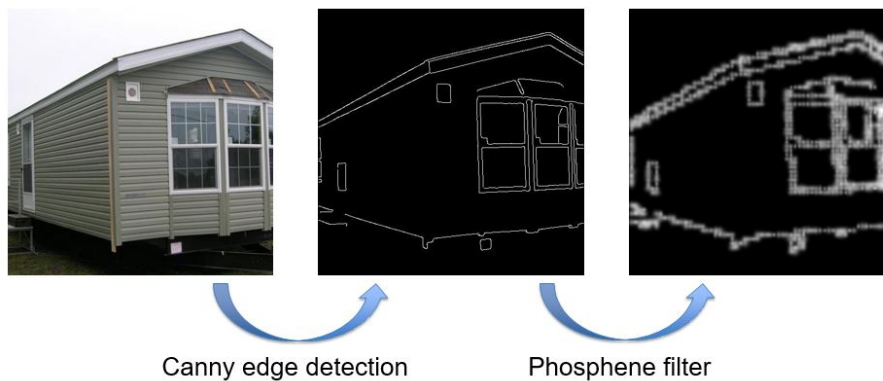
The input of the network are the images, which have a size of 473x473 and 3 channels (RGB). These images are zero-padded to 504x504 to fit the input dimensions of the network. The output and the ground truth has only one channel and a size of 63x63: a binary matrix which represents whether a phosphene is on or off. The architecture of the network is as follows: 4 convolutional layers, 5 residual blocks, 2 deconvolutional layers and finally 1 convolutional layer.

After training the network for the first time, the network learned to output only zeros, which represents black pixels. This gave an accuracy of 87% because there was a class imbalance (13% phosphenes and 87% non-phosphenes/black). Which means that the ground truth pictures are mostly black, which makes it hard to train the network. Therefore I changed the loss function to calculate pixel-wise loss and then apply weights to that to counter the class imbalance. However, this slowed the network down by quite a lot.

Edge Detection Algorithm

For the edge detection method, we can generate phosphene maps by using edge detection (canny) on the validation images of ADE20K. For this transformation the same threshold is used for every image. Edge detected images are then being transformed to phosphene maps by using the phosphene filter (figure 5). However, these images are not related to, or trained from, the ground truth phosphene maps. Therefore we can not compare them to the ground truth images as we do with the other approaches, but only to the results of the other methods.

Figure 5: *Processing stages of the edge detection algorithm. Input images are being edge detected by using the canny edge detection method. Edge detected images are then transformed to phosphene images by using the phosphene filter.*



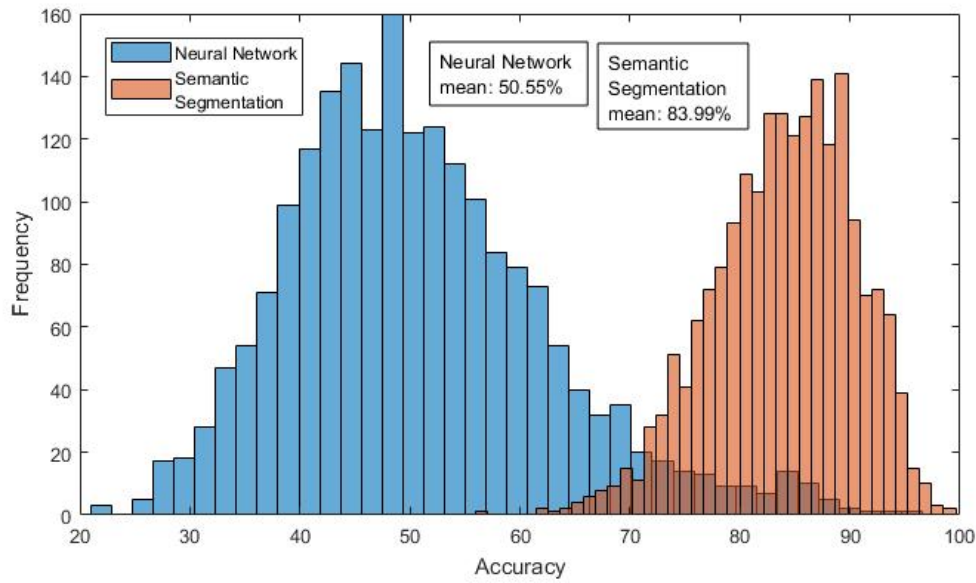
Results

Semantic Segmentation

The speed of this method is around 8 FPS [15].

The average accuracy of this method is 83.99%. This is calculated by comparing the generated phosphene images to the ground truth and counting how many of the pixels/phosphenes match. The distribution of the accuracy's for semantic segmentation (and for the neural network) can be seen in figure 6.

Figure 6: Accuracy's of semantic segmentation and the neural network. The figure shows the distribution of accuracy's for all 2000 validation images, as well as the mean accuracy.



The ground truth has on average 532.2 phosphenes per image while the algorithm predicts on average 488.3 phosphenes per image. Meaning, the algorithm predicts on average 43.89 too few phosphenes. This is calculated by subtracting the number of phosphenes in predicted images by the number in the ground truth. The distribution of this calculation can be seen in figure 7, as well as the distribution for the neural network.

In figure 8, a heatmap is depicted that shows the accuracy for every pixel of the phosphene images.

Figure 7: Number of predicted phosphenes compared to the ground truth for both semantic segmentation and the neural network. This figure shows the distribution of whether the phosphene image had too many or too few phosphenes compared to the ground truth for all validation images.

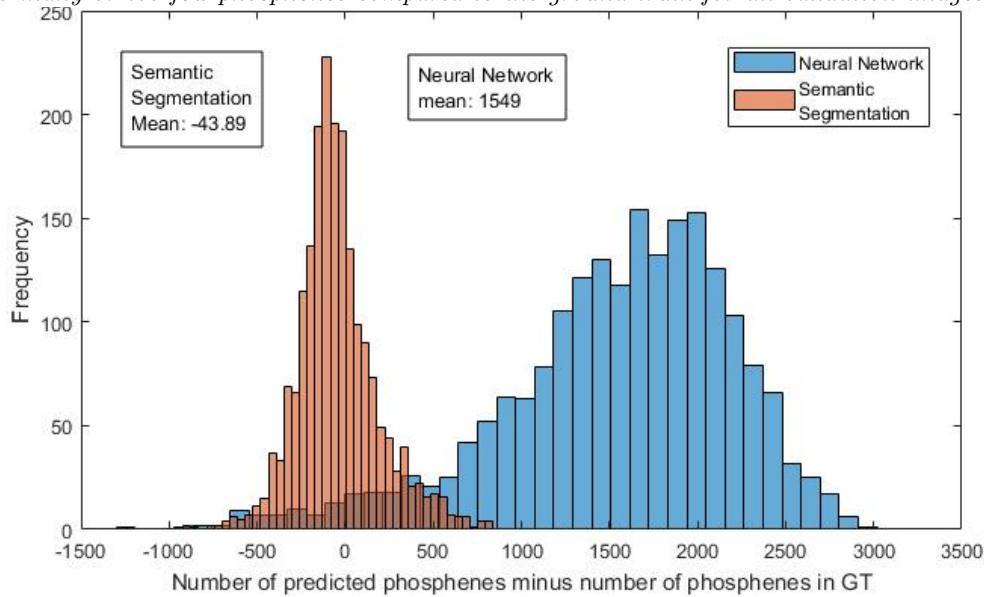
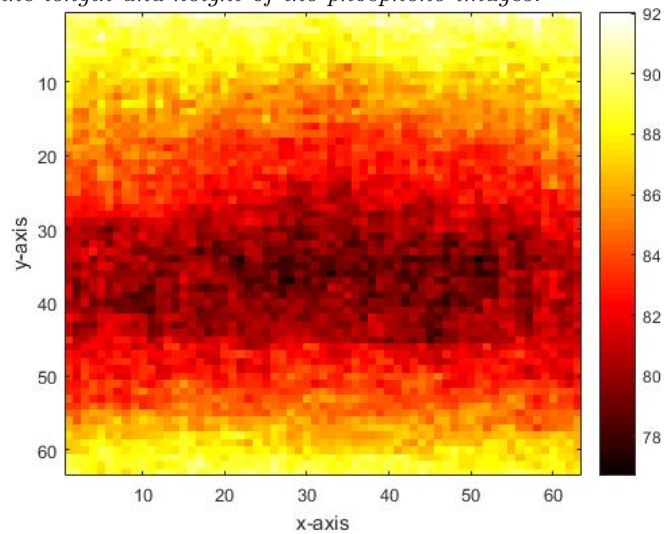


Figure 8: This figure shows the accuracy of semantic segmentation for every pixel of the image. The axis correspond to the length and height of the phosphene images.

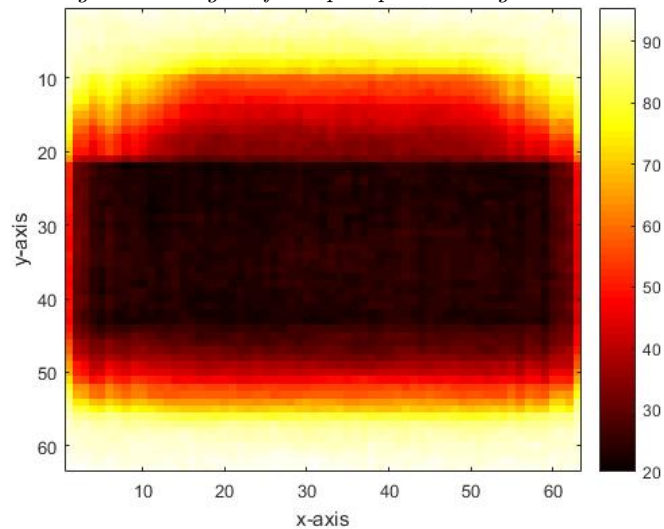


Proposed Method

The speed of this method is around 10 FPS after changing the loss function but it was a lot faster before that (over 40 FPS). The predicted images made with this algorithm look bizarre, the top and bottom of the image consist of zeros and there will be a 'blob' of phosphenes in the middle, which doesn't look like anything (see appendix for examples). However, this strategy gives the network an accuracy of 50.55%. Figure 6 shows the distribution of accuracy's for the network.

The network predicts on average 1549 too many phosphenes when compared to the ground truth. Figure 7 shows the distribution of the number of phosphenes when compared to the ground truth. In the heatmap shown in figure 9, the accuracy of every pixel in the phosphene images is displayed.

Figure 9: *This figure shows the accuracy of the neural network for every pixel of the image. The axis correspond to the length and height of the phosphene images.*



Edge Detection

Canny edge detection has a speed of around 20 FPS [17].

The phosphene images from this method are not linked to the ground truth and I can therefore only subjectively judge how good the results are. However, after looking at the phosphene images, it is clear that this method is very noisy and that the images generated with this specific threshold are not really usable in implants that will be used in different environments. For clarity, some example pictures can be found in the appendix.

Conclusion

Overall, the outcome images with semantic segmentation look really good and most of the items/features in the predicted images can be recognized pretty easily (see appendix). In the heatmap (figure 8) we can see that the top and bottom of the picture have a better accuracy than the pixels in the middle of the picture. This is probably because the top and bottom of the picture are mostly sky and ground and the more interesting objects that are harder to predict are in the middle of the pictures.

The neural network outputs blobs of phosphenes in the middle of pictures. This might be a result of the change in loss function. Because the class imbalance was countered, the network predicts way too many phosphenes. When looking at the heatmap (figure 9), it is obvious that the blobs of phosphenes in the middle of the pictures account for the overall low accuracy of the network. We can see that the network learned that there are rarely phosphenes on the top and bottom of the pictures and gets a high accuracy there because it mostly puts zeros there.

From figure 6 and 7 it is obvious that semantic segmentation outperforms the neural network. Semantic segmentation has better accuracy's than the neural network. The neural network also predicts way too many phosphenes while the semantic segmentation network only predicts a little too few.

In the results is found that edge detection is too noisy too use in VCPs but it might give decent results in static environments with clear edges. To generate the edge detected images, the same threshold was used, and this can be seen in the results. Some images show only half of the features in the picture and some show large groups of phosphenes that don't have any distinct shape. A possible explanation is that the images differ in brightness and thus have different contrasts between pixels. There were some phosphene images that looked pretty good, these were images that had clear edges, for example pictures of indoors settings or buildings.

Finally, we can conclude that with the current settings, the neural network is very bad and not usable in VCPs. The architecture of the network is unable to learn this problem with these settings. We can also say it is worse than edge detection because edge detection at least tries to represent what is in the picture, which will always be better than a obscure group of phosphenes in the middle. Semantic segmentation will remain the best option, although it is the slowest of the methods.

Discussion & Further Research

For future projects an adjusted neural network with a different architecture could be trained to predict phosphene images.

Another option would be to create a more complex algorithm for edge detection that takes different thresholds for every image into account and might for example make use of the 3 color channels to learn to mark edges. Another option to generate better images with this method is a more advanced algorithm that takes into account the context of the image.

Different semantic segmentation algorithms can also be taken into account.

Some other possibilities to create phosphene methods, that have not been discussed in this paper, are; depth detection and instance segmentation. These algorithms might also be good methods to generate phosphene images and could thus be researched.

References

- [1] Rupert R. A. Bourne. “Magnitude, temporal trends, and projections of the global prevalence of blindness and distance and near vision impairment: a systematic review and meta-analysis”. In: (2017). DOI: DOI:[https://doi.org/10.1016/S2214-109X\(17\)30293-0](https://doi.org/10.1016/S2214-109X(17)30293-0).
- [2] Rebecca Ivers et al. “Visual impairment and risk of hip fracture.” In: *American journal of epidemiology* 152 7 (1999), pp. 633–9.
- [3] Gwyn C. Jones et al. “Effects of depressive symptoms on health behavior practices among older adults with vision loss.” In: *Rehabilitation psychology* 54 2 (2009), pp. 164–72.
- [4] Alice T Chuang, Curtis Edward Margo, and Paul B Greenberg. “Retinal implants: a systematic review.” In: *The British journal of ophthalmology* 98 7 (2014), pp. 852–6.
- [5] Jean Delbeke et al. “The microsystems based visual prosthesis for optic nerve stimulation.” In: *Artificial organs* 26 3 (2002), pp. 232–4.
- [6] Soroush Niketeghad and Nader Pouratian. “Brain Machine Interfaces for Vision Restoration: The Current State of Cortical Visual Prosthetics”. In: *Neurotherapeutics* 16 (Sept. 2018). DOI: 10.1007/s13311-018-0660-1.
- [7] Giles Skey Brindley and Walpole Sinclair Lewin. “The visual sensations produced by electrical stimulation of the medial occipital cortex.” In: *The Journal of physiology* 194 2 (1968), 54–5P.
- [8] Edward J. Tehovnik et al. “Phosphene induction and the generation of saccadic eye movements by striate cortex.” In: *Journal of neurophysiology* 93 1 (2005), pp. 1–19.
- [9] N.R. Srivastava et al. “Estimating Phosphene Maps for Psychophysical Experiments used in Testing a Cortical Visual Prosthesis Device”. In: *Proceedings of the 3rd International IEEE EMBS Conference on Neural Engineering* (June 2007), pp. 130–133. DOI: 10.1109/CNE.2007.369629.
- [10] Michael Beauchamp et al. “Dynamic Electrical Stimulation of Sites in Visual Cortex Produces Form Vision in Sighted and Blind Humans”. In: (Nov. 2018). DOI: 10.1101/462697.
- [11] Nishant R Srivastava, Philip R Troyk, and Gislin Dagnelie. “Detection, eye-hand coordination and virtual mobility performance in simulated vision for a cortical visual prosthesis device”. In: *Journal of neural engineering* 6 (July 2009), p. 035008. DOI: 10.1088/1741-2560/6/3/035008.
- [12] W. L. D. Lui et al. “Transformative Reality: Improving bionic vision with robotic sensing”. In: *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. Aug. 2012, pp. 304–307. DOI: 10.1109/EMBC.2012.6345929.
- [13] Jesus Bermudez-Cameo et al. “RGB-D Computer Vision Techniques for Simulated Prosthetic Vision”. In: *IbPRIA*. 2017.
- [14] Lachlan Horne et al. “Semantic labelling to aid navigation in prosthetic vision”. In: *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (2015), pp. 3379–3382.
- [15] W. H. Li. “Wearable Computer Vision Systems for a Cortical Visual Prosthesis”. In: *2013 IEEE International Conference on Computer Vision Workshops*. Dec. 2013, pp. 428–435. DOI: 10.1109/ICCVW.2013.63.
- [16] Alberto Garcia-Garcia et al. “A Review on Deep Learning Techniques Applied to Semantic Segmentation”. In: *CoRR* abs/1704.06857 (2017).
- [17] Jong-Chyi Su. “Comparison of Edge Detectors”. In: 2014.

- [18] Bolei Zhou et al. “Scene Parsing through ADE20K Dataset”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.
- [19] Hengshuang Zhao et al. “Pyramid Scene Parsing Network”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 6230–6239.
- [20] Yağmur Güçlütürk et al. “Convolutional Sketch Inversion”. In: *Computer Vision – ECCV 2016 Workshops*. Ed. by Gang Hua and Hervé Jégou. Cham: Springer International Publishing, 2016, pp. 810–824. ISBN: 978-3-319-46604-0.

Appendix: Examples of images by every algorithm

