

BACHELOR THESIS
ARTIFICIAL INTELLIGENCE

Radboud University



**Phosphene Movement HUDs:
Representing Movement in Phosphene Vision**

Author:
Loes Erven
s4538757

First supervisor:
dr. U. Güçlü
Artificial Intelligence,
Donders Institute,
Radboud University

Second supervisor:
drs. J. de Ruyter van Steveninck
Artificial Intelligence,
Donders Institute,
Radboud University



January 29, 2021

Abstract

Visual prosthetic devices provide a promising aid that aim to restore some form of vision in blind people. In this prosthetic vision a lot of information is lost. One of the problems that could arise from this is that it would be hard for a human to determine whether an object is moving. This can be crucial information depending on context. This project takes a look at ways of detecting this movement and in turn how to represent that information on phosphene level in the form of a Heads-Up Display (HUD).

Contents

1	Introduction	2
2	Preliminaries	3
2.1	Phosphenes	3
2.2	Heads-up Display	4
3	Related Work	5
4	Research	7
4.1	Methods	7
4.1.1	Absolute Movement	7
4.1.2	Displaying Relative Movement	8
4.2	Results & Discussion	10
4.2.1	Results	10
4.2.2	Discussion	13
5	Conclusions	15
A		18

Chapter 1

Introduction

This thesis is part of a bigger project which aims to develop a visual prosthesis to (partially) restore vision to the blind by stimulating the visual cortex. Every stimulating electrode causes a localised flash to appear in the visual field. We call these flashes “phosphenes”. As of now this can be done in relatively low resolution, meaning not everything a camera captures can be shown this way. Besides that it is not yet possible to intentionally manipulate the perceived colour, size and exact position of these phosphenes. All of this means that a choice has to be made about which aspects of a visual scene are important to show and which aspects are less so and can be left out. For instance in order to be able to identify objects around you, showing only the outlines (edges) of objects can be very helpful, while showing the colour of an object is generally less important. Besides a direct mapping of features in an image to a phosphene version of that image, high level information (information which can be deduced from those features but is not directly available without calculations) that is otherwise lost in translation could also be represented in a simplified manner.

In short my main questions are ‘how do I detect this movement?’ and ‘how can I display high level information like this in general?’ Or, put together:

‘How can movement be displayed in phosphene vision?’

An assumption for this thesis is that the only data available for input of the prosthesis are 2D images

Chapter 2

Preliminaries

2.1 Phosphenes

This thesis is part of a sub-project of Nestor Consortium by Neural Coding Lab. This project aims to restore (partial) vision to the blind with a visual prosthesis by stimulating the visual cortex and with that evoking phosphenes. Phosphenes in general refer to the concept of seeing light without light entering the eyes. These phosphenes can arise from internal phenomena like closed eye hallucinations, but also from external stimulation[1]. The latter is the case for this project. The idea is that the visual cortex is electrically stimulated causing the stimulated person to see dots of light without the use of their eyes. If these electrical pulses are sufficiently directed then the perceived phosphenes can be controlled to a certain extent. So far their colour and intensity can not be controlled. Their exact location is also hard to control and at the time of writing there is some variance to it, but an approximate location can indeed be controlled.[2]

Fig. 2.1 is an example of what this simulated phosphene vision looks like.



Figure 2.1: An example of simulated phosphene vision. From left to right: The raw image, the edges in the image, the phosphene image.

This is not the reality of what the image would look like when projected on the visual cortex. In reality the image would become warped like the inverse of a fish-eye lens because of differing receptive fields of our neurons, but for the sake of simplicity this is the representation of phosphene vision for this level in the project.

2.2 Heads-up Display

A Heads-up Display(HUD) is a display that presents information in such a manner that you barely have to look away from what you are usually looking at to be able to see what is on the display. In air crafts and cars that is your windshield, so the HUD can be projected on or in front of that on a transparent display, and in video games this is your screen, where information stays on the same place as you walk around, like a mini-map or a health bar for instance. In this project the concept of a HUD differs slightly in that you cannot look away from the “screen”, as it is literally all you can see. So here the concept of a HUD refers to a display that stays stationary as you move, as if it were tied to you physically.

Chapter 3

Related Work

Optical Flow

A well-known way of detecting movement is optical flow.

Optical flow looks at each element in an image sequence and determines in what direction and how fast it is moving relative to the observer, or as Horn and Schunck phrase it: “Optical flow is the distribution of apparent velocities of movement of brightness patterns in an image.” [3]

There are two main types of optical flow detection. Sparse and dense optical flow.

Sparse Optical Flow looks at a smaller amount of pixels in a sequence of images. It looks at certain interesting features like edges or corners and determines the perceived movement of only those features. This is faster to compute than dense optical flow, but might not provide enough information depending on the application.

Dense Optical Flow looks at every pixel in a sequence of images and determines the perceived movement of all of them. Dense and sparse optical flow methods can also be combined, calculating sparse flow first and interpolating dense flow from that.[4]

Fig. 3.1 is an example of the way optical flow is represented. Such colour visualisations are usually used for representing dense optical flow. Arrows or lines are usually used in the case of sparse optical flow.

In the colour visualisation the colour at each location represents the vector at that location. The hue represents the direction, and the value the length of the vector.



Figure 3.1: An example of a dense flow representation using colours to indicate the vector. On the right is a legend of the colour coding.[5]

As you can see nearly everything at the left and right of the images appears to be moving to the sides. This is because the observer is moving forward. The movement captured by optical flow is the relative movement.

Horn and Schunck[3] determine dense optical flow with simple derivatives and a less simple error minimisation problem in their paper. This approach has been improved upon a lot but the research on it is nowhere near finished.[6]

Since the advent of artificial neural networks these have also widely been used to determine optical flow, with varying approaches.[7][8][9]

Chapter 4

Research

4.1 Methods

4.1.1 Absolute Movement

Optical flow is generally sensitive to motion parallax, since the detected movement is the movement relative to the observer. Something that is far away and racing past might appear to be moving at the same speed as something that is close by and stationary when you walk past it, especially in a 2-dimensional view. I have investigated a potential method for calculating the absolute movement from the known relative movement.

To calculate the optical flow and depth I used a model called GeoNet[9]. Later I also used the ground truths to rule out the possibility that the results were negatively influenced by the accuracy of the calculations. For this part I used the KITTI optical flow dataset. [10]

The investigated method was to normalise the optical flow by the depth. I started by using the standard normalisation function:

$$x_{normalised} = \frac{x - x_{minimum}}{x_{maximum} - x_{minimum}}.$$

This did not seem to significantly improve the accuracy of the optical flow. I then tried several other things like simply dividing the flow by the depth linearly, quadratically and exponentially, as well as myriad other such fairly simple calculations. These all either did not significantly improve the results or sometimes even made them worse.

At this point I considered whether the determined depth might be somehow warped, since it is calculated from a 2-dimensional plane (the image) instead of a single point (the location of the observer). If this were the case then an object at the edge of the image which has a calculated depth equal to that of an object in the middle of the image should in reality be

farther away from the observer. After digging into it I came to the conclusion that this is unlikely to be the case since the GeoNet network is trained on (and the depth ground truth consists of) data obtained by projecting points scanned by a laser located very close to the camera onto a 2d image, which yields an accurate representation of reality. This should by all means rule out any warping. For good measure I ran some trigonometric equations over the images that should fix the warping if any were there, and make the results less accurate if there were none. I could determine no warping.

Eventually I decided that I would limit the project to using relative movement, since simulated phosphene vision already limits the ability to tell *any* kind of movement apart from simply jittering phosphenes, and because of that I deemed it worth researching whether displaying the relative movement instead of absolute movement is already worth doing.

4.1.2 Displaying Relative Movement

Dataset

An example of a resulting flow image was shown in Section 3, Fig. 3.1. As you can see there, it is not necessarily trivial to see what data is relevant. Trying to display all of the available flow information in phosphene vision would simply be too much. If it were possible to control the colour of the phosphenes this might be feasible but as of now I saw no way to do that, so I had to decrease the amount of data to show. I decided to use bounding boxes around objects that were capable of movement for this, since these are the most relevant objects when displaying movement. My further reasoning for using bounding boxes was that I feel it to be information that can be used for other applications in the visual prosthesis as well (that or something similar, like instance segmentation, which could also be used for this), so it did not feel like a waste of computing power. Since the KITTI data set which includes ground truth for bounding boxes includes image sequences of only 4 frames long and I wanted to be able to show longer sequences in simulated phosphene vision, I opted to use a different data set, SYNTHIA-AL[11], which is a synthetic data set in a similar setting to the KITTI data set, namely traffic. The data set contains amongst other things ground truth for bounding boxes and much longer image sequences. In this data set the objects capable of movement and thus represented with a bounding box are cars, pedestrians and cyclists.

GeoNet was trained with a KITTI data set, but luckily it generalises to the SYNTHIA-AL data set, so there was no need to retrain it. The only thing I needed to change was to cut the input images to the correct size.

Now with these bounding boxes available I had a way to limit the amount of information to show in a HUD.

The way I then calculated the movement for each object was to take the mean vector inside the bounding box. This works fine if there are no occlusions, but it gives significant problems if there are. If the object is occluded by only another object that has a bounding box then this is relatively trivial to solve, but if it is not it becomes more difficult.

HUDs

There were several aspects to consider when deciding how to represent the HUD. To name a few significant considerations:

- The low resolution of the phosphenes means that it is impractical to show complex images or shapes in the HUD.
- It has to be clear where the image ends and the HUD begins.
- The HUD should not impede the view of the image.
- It should be possible to use a HUD for multiple things.
- It should be clear and intuitive what the information in the HUD means.
- The HUD should only display relevant information.

Taking all these considerations into account there were a few options that came to mind:

1. A set corner containing a simple shape which is there if something is moving and disappears if there is not.
2. A set corner containing an arrow showing the direction of a moving object which disappears if nothing is moving.
3. Nr. 1 and 2 except the location of the HUD changes according to which quadrant the moving object is in, the HUD being in one of four set locations, the corners.
4. Nr. 1 and 2 except the location of the HUD changes according to the direction in which the object is moving, the HUD being in one of four set locations, the corners.
5. A border around the image containing simple shapes in the direction of the movement of each object.

All of these approaches have drawbacks however. The most notable are: for 1 and 2, what do you display when there are multiple objects? For 3 and 4, it might become chaotic if something is switching between quadrants a lot and it might be confusing to use multiple corners and *only*. For 5, it

simply takes up a lot of space that we do not have in the first place. I decided to explore two of these further with a circle as simple shape. First, to satisfy Occam’s razor, the first option with the added feature that the circle represents the fastest object in the image, and that the size of the circle depends on the speed of that object so that it is bigger when the speed is higher and smaller when the speed is lower. Second, the fifth option, again with the added feature that represents speed, since it seemed to me most likely option to be worth the trade-off between size and confusion, and the information it yields.

4.2 Results & Discussion

4.2.1 Results

Fig. 4.1 shows a visualisation of the HUDs mentioned, in an image with no occluded relevant objects.

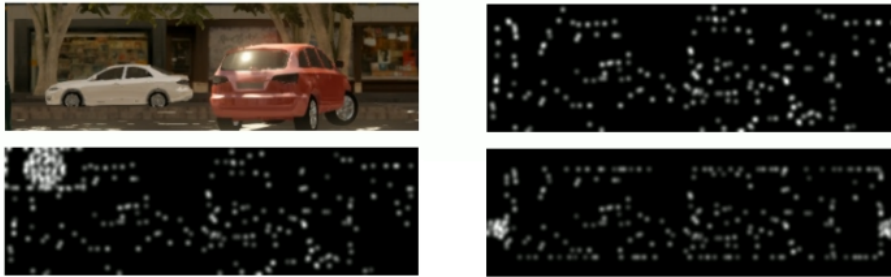


Figure 4.1: Visualisation of the HUDs. From left to right, from top to bottom the image shows: the raw image, the phosphene image without HUD, the phosphene image with a corner HUD, the phosphene image with a border HUD.

It is hard to show the reality of simulated phosphene vision in single images, so it might be hard to see what the HUDs add in single images, so the appendix contains a URL to a video of these same HUDs.

The corner HUD is not hard to implement since the circle always stays in the same place. The border HUD is a little more complex but not too hard to implement as well. It was made by taking each calculated vector then calculating the intersections of the line it is positioned on and the image borders, checking in which general direction the vector is pointing and putting a circle at the corresponding intersection.

The problem of occlusion is most obvious in the border HUD as can be seen in Fig. 4.2, since the circles are obviously located in the wrong

direction, but if the fastest object is occluded, the corner HUD also shows the wrong speed.

This is also a big part of why the indicators in the border HUD jump around so much. I tried adding a simple decay function and while this did make the jumping a lot calmer, it was a lot more confusing too.



Figure 4.2: Top: an image with occluded objects. There is a barely visible white car at the far left, occluded by a tree. Bottom: the border HUD without the phosphene mask for clarity.

Survey

With these HUDs I performed a short and informal pilot survey, testing for two sequences for both HUDs and without a HUD on how pleasant the sequence is to look at, how clear it is what is happening and how distracting the HUD was. The results are not very conclusive since the survey was small scale, but even so there are some interesting results.

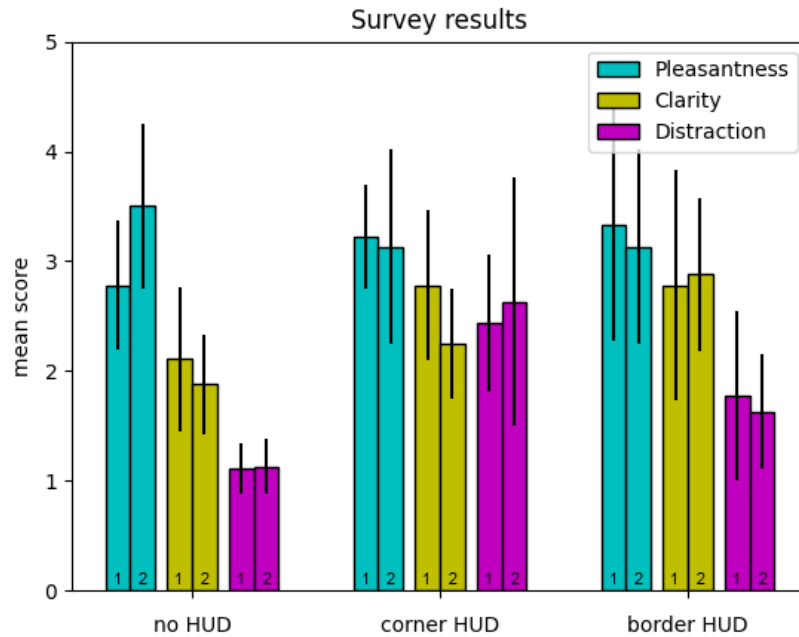


Figure 4.3: A bar graph of the results of the survey. The results of sequence 1 and 2 are shown side to side in the same colour. The error bar shows the standard error.

From Figure 4.3 we can see that the HUDs do not necessarily seem more pleasant to look at than the sequences without HUD, but they do seem to add some clarity over the display without a HUD. They also seem to be more distracting. There does not seem to be a big difference in clarity and pleasantness between the two HUDs, though the corner HUD seems to be quite a bit more distracting than the border HUD. In order to know whether these differences are significant, I performed paired t-tests on the results. For this, the results of both sequences were grouped together to compare Pleasantness, Clarity and Distraction between different the different displays. There were a few significant results:

Clarity of corner HUD > Clarity of no HUD, $p = 0.029$.

Clarity of border HUD > Clarity of no HUD, $p = 0.00095$.

Distraction of corner HUD > Distraction of no HUD, $p = 0.000048$

Distraction of border HUD > Distraction of no HUD, $p = 0.0098$

Distraction of corner HUD > Distraction of border HUD, $p = 0.016$

For all other results $p > 0.05$.

A non-significant but notable result is the Clarity of the border HUD over the Clarity of the corner HUD, with $p = 0.055$.

Furthermore I performed normality tests on all results. The distraction

for no HUD is the only factor for which it can confidently be said that it is not distributed normally. This goes for both sequences.

Besides the reported results, two people have indicated a clear preference of the border HUD over the corner HUD. One of them remarked that the preference might have to do with not being used to the jittery visual. This would mean that it might be worth testing whether the HUDs still add something after someone has used phosphene vision without HUDs for a while.

Nobody has indicated a preference for a different display.

Of course a more in depth study would need to be done to get a clearer idea of the effectiveness of the HUDs, but these results do seem to point to a positive effect of the HUDs, especially the border HUD.

4.2.2 Discussion

Limitations & Future Recommendations

Something these implementations still lack is the third dimension, or z-axis. It is not possible to get this information from the optical flow, but it is possible when you have bounding boxes or depth information. For the corner HUD nothing needs to be changed about the implementation of the HUD, since all it needs is the length of the vector, but for the border HUD this does need to change. One way of representing this extra dimension is by moving the circle to the outside of the border when the object is moving away, and to the inside when the object is moving closer. Another way that is only possible when the resolution is high enough, would be representing it in a way not unlike it is represented in physics, with a cross shown when it is moving away, and a circle shown when it is moving towards the observer. Another feature that can be added when taking the z-axis into account would be an alarm when an object is moving towards the observer with a high speed. A way of representing this could be making the entire HUD flash white. This might still make it hard to see where the object is, but it might also be clear enough at that point. This is a possible feature that could add a lot but would need to be thoroughly tested before implementing.

Background Subtraction Optical flow is not the only method of detecting movement that could be used for the application I researched. One such method is background subtraction. It takes the perceived background of an image and then looks at the difference between that and the next image in a sequence. Anything that is different should then be movement. This is relatively trivial when it involves completely stationary cameras, as it only needs to take into account changes in weather and illumination. However when the camera moves issues arise. The new image can not be compared

to the known background image in the same way it was before, because the entire image will have changed thus everything in the image will be registered as movement, even though it is only the camera that is moving.

There are several solutions for this. I will briefly name two of them here.

One solution is building a bigger, panoramic background [12], which is especially useful for surveillance cameras, which will only ever see a very limited part of the world and will always see a spot from the same angle.

Another solution is taking a background image which is constantly updated with new information. Approaches that do this look at features in the existing background image (like the corners of a building) and compare those with features in the new image in order to be able to warp the old background as if the observer were standing in the position it is in in the new image. [13][14] After this warping the regular background subtraction can be performed with little extra calculations.

Chapter 5

Conclusions

To conclude, although it remains to be seen how this would translate to actual users of the visual prosthesis, as it is hard to imagine how and how fast the brain would be able to adapt to such a type of vision, there does seem to be something to representing movement with a HUD-like approach. To me it also seems worth it to look more into representing different types of high-level information that would otherwise be harder than normal to pick up in phosphene vision. However a challenge lies in getting those types of representations to work together without becoming too chaotic to be of use. In possible further research in which the absolute instead of relative movement is needed I would recommend using the background subtraction for moving cameras as briefly discussed in section 4.2.2 instead of trying to expand on the optical flow, unless there is a different reason information on optical flow would be needed.

Bibliography

- [1] G. S. Brindley and W. S. Lewin. The sensations produced by electrical stimulation of the visual cortex. *The Journal of Physiology*, 196(2):479–493, 1968.
- [2] Eduardo Fernandez. Development of visual Neuroprostheses: trends and challenges. *Bioelectronic Medicine*, 4(1):12, 2018.
- [3] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1):185 – 203, 1981.
- [4] Jérôme Revaud, Philippe Weinzaepfel, Zaïd Harchaoui, and Cordelia Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. *CoRR*, abs/1501.02565, 2015.
- [5] Mark Gituma. *Generating Optical Flow Using NVIDIA flownet2-pytorch implementation*. 2019.
- [6] Deqing Sun, Stefan Roth, and Michael J. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *Int. J. Comput. Vision*, 106(2):115–137, 2014.
- [7] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.
- [8] Anurag Ranjan and Michael J. Black. Optical flow estimation using a spatial pyramid network. *CoRR*, abs/1611.00850, 2016.
- [9] Z. Yin and J. Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. *CoRR*, abs/1803.02276, 2018.
- [10] Moritz Menze, Christian Heipke, and Andreas Geiger. Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, 2018.

- [11] Javad Zolfaghari Bengar, Abel Gonzalez-Garcia, Gabriel Villalonga, Bogdan Raducanu, Hamed H Aghdam, Mikhail Mozerov, Antonio M Lopez, and Joost van de Weijer. Temporal coherence for active learning in videos. *arXiv preprint arXiv:1908.11757*, 2019.
- [12] Pietro Azzari and Alessandro Bevilacqua. *Joint Spatial and Tonal Mosaic Alignment for Motion Detection with PTZ Camera*. ICIAR'06. Springer-Verlag, Berlin, Heidelberg, 2006.
- [13] Soo Wan Kim, Kimin Yun, Kwang Moo Yi, Sun Jung Kim, and Jin Young Choi. Detection of moving objects with a moving camera using non-panoramic background model. *Machine Vision and Applications*, 24(5):1015–1028, 2013.
- [14] Amitha Viswanath, Reena Kumari Behera, Vinuchackravarthi Senthamilarasu, and Krishnan Kutty. Background modelling from a moving camera. *Procedia Computer Science*, 58:289 – 296, 2015.

Appendix A

URL to the phosphene video:
youtu.be/mjspeJ6aFI0