



The effect of noise on computational T-Junction detection in images

Der Einfluss von Rauschen auf die Computer basierte
Erkennung von T-Kreuzungen in Bildern

Matheo Benedikt Möbis

Universitätsbachelorarbeit
zur Erlangung des akademischen Grades

Bachelor of Science
(*B. Sc.*)

im Studiengang
Informatik

eingereicht am 12.12.2025 am
Fachgebiet Computational Psychology der
Fakultät IV
der Technischen Universität Berlin

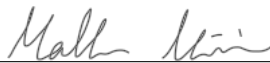
Betreuer	Prof. Dr. Marianne Maertens
Zweitgutachter	Prof. Dr. Guillermo Gallego

Declaration of Authorship

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe. Alle Stellen die den benutzten Quellen und Hilfsmitteln wörtlich oder inhaltlich entnommen sind, habe ich als solche kenntlich gemacht. Sofern KI-Tools verwendet wurden, habe ich Produktnamen, Hersteller, die jeweils verwendete Softwareversion und die Art der Nutzung (z.B. sprachliche Überprüfung und Verbesserung der Texte, systematische Recherche) benannt. Ich verantworte die Auswahl, die Übernahme und sämtliche Ergebnisse des von mir verwendeten KI-generierten Outputs vollumfänglich selbst. Die Satzung zur Sicherung guter wissenschaftlicher Praxis an der TU Berlin vom 15. Februar 2023.

https://www.static.tu.berlin/fileadmin/www/10002457/K3-AMBI/Amtsblatt_2023/Amtliches_Mitteilungsblatt_Nr._16_vom_30.05.2023.pdf habe ich zur Kenntnis genommen. Ich erkläre weiterhin, dass ich die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt habe.

Berlin, December 12, 2025



Matheo Benedikt Möbis

Abstract

The perception of depth is a crucial ability of the human visual system. This ability is not limited to the threedimensional world humans live in, but also extends to images displayed in 2D. One of the clues humans use to perceive depth in images are T-Junctions, which are caused by a luminance contrast at object borders. This perceived contrast at the borders can be reduced by masking images with noise [Bet+15], raising the question if a computer algorithm, that is able to detect T-Junctions in images, is affected by noise in a similar way. In this thesis, different stimuli are masked with noise at different center frequencies and the amount of detected T-Junctions by the algorithm is measured. The results show, that noise at frequencies between 0.04cpd and 0.13cpd disturbs the correct detection of T-Junctions whereas frequencies in the range $[0.17, 0.2]$ cpd improve the amount of correctly detected T-Junctions.

Zusammenfassung

Die Wahrnehmung von Tiefe ist eine wichtige Fähigkeit des menschlichen visuellen systems. Diese Fähigkeit ist nicht beschränkt auf die dreidimensionale Welt in der Menschen leben, sondern erstreckt sich auch auf Bilder, welche in 2D dargestellt werden. Einer der Anhaltspunkte den Menschen nutzen um Tiefe in Bildern wahrzunehmen sind sogenannte T-Kreuzungen, welche durch einen Luminanz Kontrast an den Rändern von Objekten hervorgerufen werden. Der wahrgenommene Kontrast an den Rändern kann durch die Maskierung von Bildern mit Rauschen reduziert werden [Bet+15], was die Frage aufwirft, ob ein Computer-Algorithmus, welcher in der Lage ist T-Kreuzungen in Bildern zu erkennen, in einem ähnlichen Maße von Rauschen beeinflusst wird. In dieser Bachelorarbeit werden verschiedene Stimuli mit Rauschen, bei verschiedenen Mittelfrequenzen, maskiert und die Anzahl der vom Algorithmus erkannten T-Kreuzungen gemessen werden. Die Ergebnisse zeigen, dass Rauschen bei Frequenzen zwischen 0.04cpd und 0.13cpd die korrekte Erkennung von T-Kreuzungen stört, während Frequenzen in dem Intervall $[0.17, 0.2]$ cpd die Anzahl korrekt erkannter T-Kreuzungen steigern.

Contents

Declaration of Authorship	iii
Abstract	v
Zusammenfassung	vii
Contents	ix
I Introduction	1
1 Motivation	3
2 Basic Terminology	5
2.1 Noise	5
2.2 White's illusion	6
2.3 Contours	7
2.4 High-Curvature Points	7
3 Procedure of the Algorithm	9
3.1 In- and Output	9
3.2 Contour-Detection	10
3.3 T-Junction and corner detection	15
3.4 Aim	15
II Methods	19
4 Stimuli	21
4.1 Creating stimuli with noise	21
5 Testing the algorithm	25
5.1 Executing the algorithm and handling the data	25

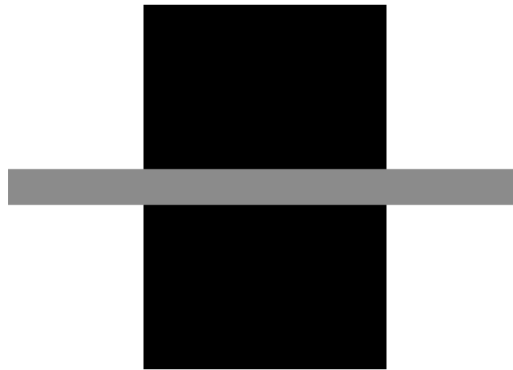
5.2	Variation of input parameters	26
5.2.1	Variation of τ	26
5.2.2	Variation of σ	26
5.3	Testing with noise-masked images	30
III	Results	31
6	Examining the data	33
6.1	First bar stimulus	33
6.2	Second bar stimulus	34
6.3	White's illusion	35
IV	Discussion	41
7	Interpretation	43
7.1	Influence of the base value	43
7.2	Comparison to Betz et al.	43
7.3	Difficulties and future work	45
8	Conclusion	47
	Usage of AI tools	49
	Bibliography	51

Part I

Introduction



(a) Rabbits



(b) Rectangles

Figure 1.1: On the right (a) an image of two 3D-printed stanford-rabbits and on the left (b) an image of two rectangles

When one looks at the photograph of two 3D-printed rabbits in Figure 1.1, two objects are perceivable in front of each other. To perceive depth in 3D, the ability to view the world with two eyes can be utilized, which creates depth when the two views are combined. But since the image of the rabbits is displayed in 2D, this mechanism can't be the reason why depth is perceived in this image. Distant objects also appear smaller than close objects due to perspective view, but if one looks at a simple image like the grey and black rectangles in Figure 1.1, a grey rectangle on top of a black one is perceived, although no perspective information is given. The results of past experiments showed that there are additional clues, called T-Junctions, the human visual system can utilize in order to perceive depth in images. In this thesis I am going to explore if a computer algorithm is able to identify T-Junctions in noisy images, which is motivated by the question, whether computer-algorithms fail to perceive depth in the same situations humans do.

The terms T-Junctions and noise, as well as other important concepts, are going to be explained in the next chapter.

2

Basic Terminology

2.1 Noise

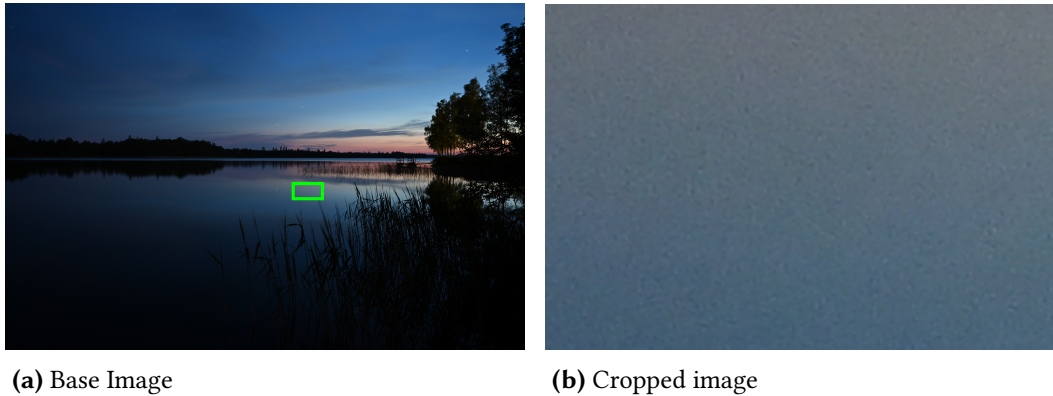


Figure 2.1: On the left (a) an image I took with a camera and on the right (b) a crop of that image to the marked green area. As one can see, the cropped image shows grain.

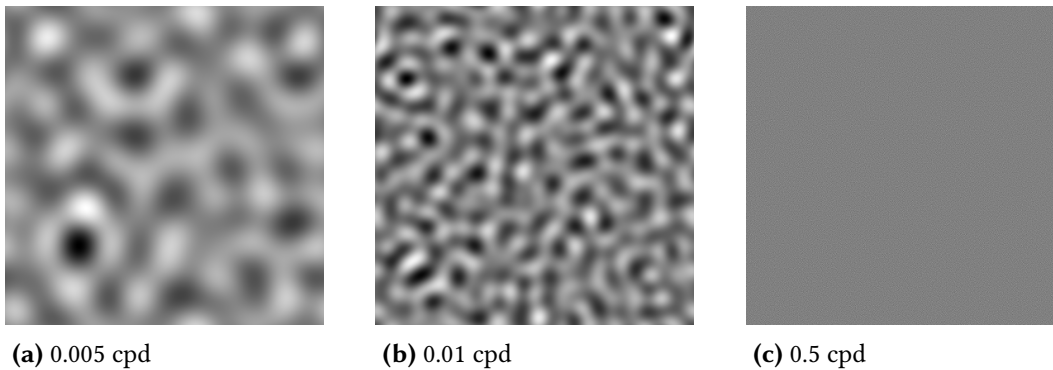


Figure 2.2: Narrowband-noise at different center frequencies.

Noise in the context of images describes a random disturbance of the pixel values that appears naturally when digital images are taken. This is depicted in Figure 2.1 where the cropped image shows some sort of grain. This grain is an example of

noise. In order to analyze effects of noise, it can be created artificially as one can see in Figure 2.2. One can also notice that the pattern in the three different images looks similar but it is repeating more often as one looks from the left to the right. How much a noise-pattern repeats itself is determined by its center-frequency with the unit „cpd“ which is short for „cycles per degree“ referring to the amount of repetitions in one degree of the observer’s viewing angle. Noise can be added to an existing image, which is called „noise masking“.

2.2 White’s illusion

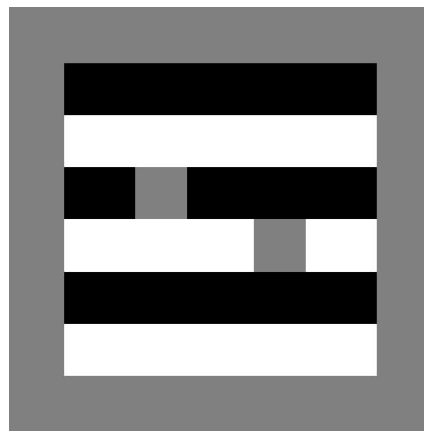
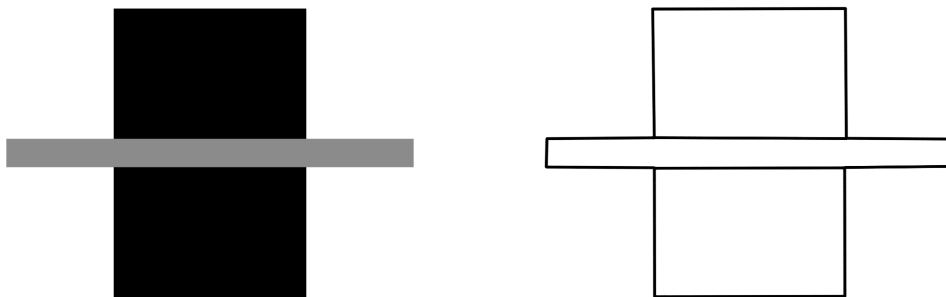


Figure 2.3: An image of white’s illusion[Bet+15]

When looking at Figure 2.3 one can notice that the two grey squares seem to have two different grey values. In fact it appears as if the square at the bottom right is darker than the square in the top right. This is an illusion: The two squares have exactly the same grey value when the digital pixel values are looked at. If this image is currently printed out, one could take a pair of scissors, cut the two squares out and compare them to each other, leading to the same result. Because of that one has to differentiate between the value of a grey value at a certain point, called luminance, and how it is perceived by a human being, called lightness.[Bol24] A simplified explanation for why the two square are perceived at a different lightness is, that the human visual system doesn’t focus on luminance but rather changes of luminance. Because of that a change from grey to black can be perceived as lighter than the change from grey to white.

2.3 Contours

Looking at Figure 2.4 (a) one can see the image of the two rectangles again. Since the background is white and the first rectangle is black, there is a big difference in grey values between the two areas. This difference is called contrast and the line where the white area meets the black area is called an edge. The entirety of edges that differentiates one object from its background is called the object's contour. As one can see in Figure 2.4 (b) I drew these contours for the grey and the black rectangle by hand. Since the algorithm I am going to explain composes contours out of many different lines, I will often use the term „linesegment“ or „line“ when a part of a contour is meant.



(a) Original

(b) Contours

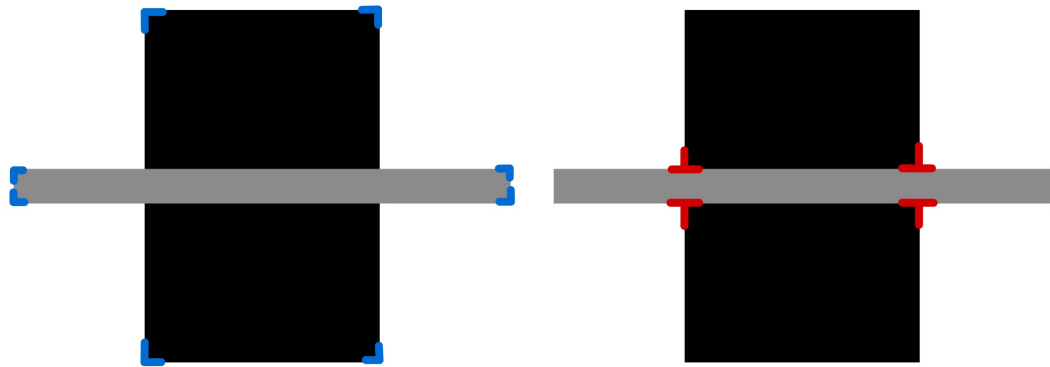
Figure 2.4: On the left (a) the original image and on the right (b) the hand-drawn contours of the image.

2.4 High-Curvature Points

A point where multiple line segments meet is called a „high-curvature point“ if the angle between the arriving line-segments is $\geq 15^\circ$ and $\leq 165^\circ$, describing a sharp turn in the curvature of the contour. For this thesis high curvature points are differentiated into two groups:

1. L-Junctions: indicated by 2 arriving line-segments

2. T-Junctions: indicated by 3 arriving line-segments



(a) L-Junctions

(b) T-Junctions

Figure 2.5: On the left (a) the hand-drawn L-Junctions in blue and on the right (b) the hand-drawn T-Junctions in red.

As one can observe in Figure 2.5, T- and L-Junctions are intuitively named since 3 arriving line-segments form the letter T and 2 arriving line-segments the letter L in most cases. Since T-Junctions are suggested to be a reason why humans perceive multiple layers in images, as explored by Zaidi et al.[ZSS97], this thesis will be heavily focused on the detection of this type of junction.

How T-Junctions are detected with an algorithm will be explained in the next chapter.

3

Procedure of the Algorithm

The algorithm that I am going to use, was proposed in the paper „Contours, Corners and T-Junctions Detection Algorithm“ by Buades et al.[BGN18]. Approaches described in other papers like Kang et al.[KCP05] and Dimiccoli et al.[DS09] also seemed promising but unfortunately no code was published, whereas Buades et al. provided the code as well as a demo-website¹. The following sections are a simplification of the explanation given in the paper by Buades et al. on pages 25-28.

3.1 In- and Output

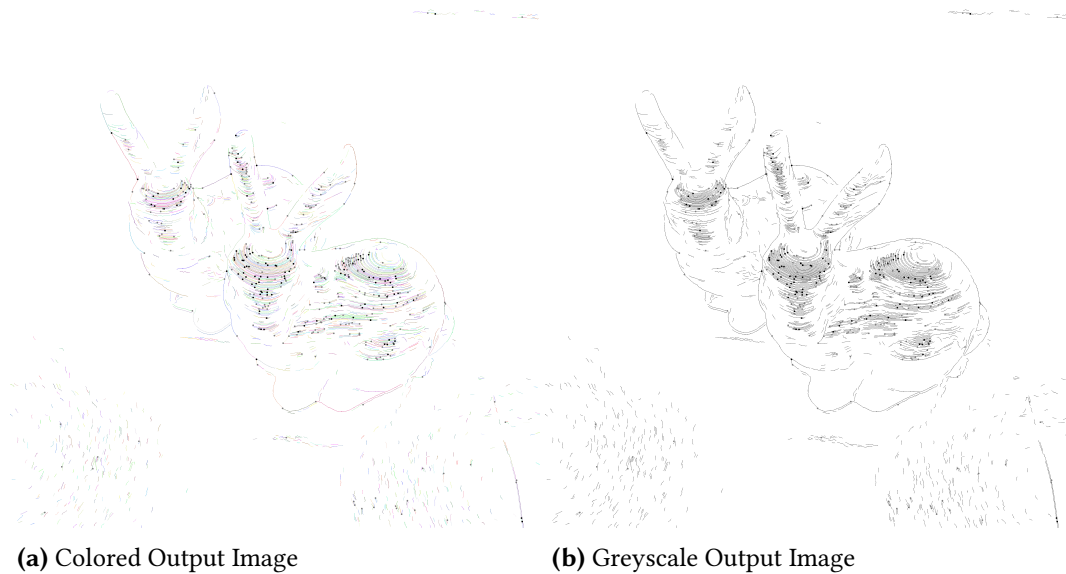


Figure 3.1: Output images of the algorithm with the rabbit-image from Figure 1 (a) as input

As input the algorithm gets an image which was converted to a grey-level colormap, like the image of the rabbits one can see in Figure 1 (a). In this image,

¹ <https://doi.org/10.5201/ipol.2018.218>

the algorithm detects edges and extracts corners and T-Junctions based on these edges. In Figure 3.1 one can see the output images: The line-segments in the first image are colored, whereas the second image has black line-segments. In addition to the images, two text files get created, which mark the exact coordinates of pixels that contribute to a contour as well as which of these pixels is classified as an T- or L-Junction. The first file contains the coordinates of the contours where each row represents the coordinates of one line-segment. The second file contains the coordinates of each junction that was detected where each row represents the pixel coordinates and the number of arriving line-segments at that coordinate. Here is one line that I copied from each of these files as an example:

- one row of the contour-file containing the coordinates for every pixel in one line segment:

```
1130 539 1130 540 1130 541 1131 537 1131 538 1130 542
1130 543 1131 535 1131 536 1130 544 1130 545 1131 545
1132 533 1131 534 1131 546 1131 547 1132 531 1132 532
1131 548 1131 549 1133 529 1133 530 1134 528
```

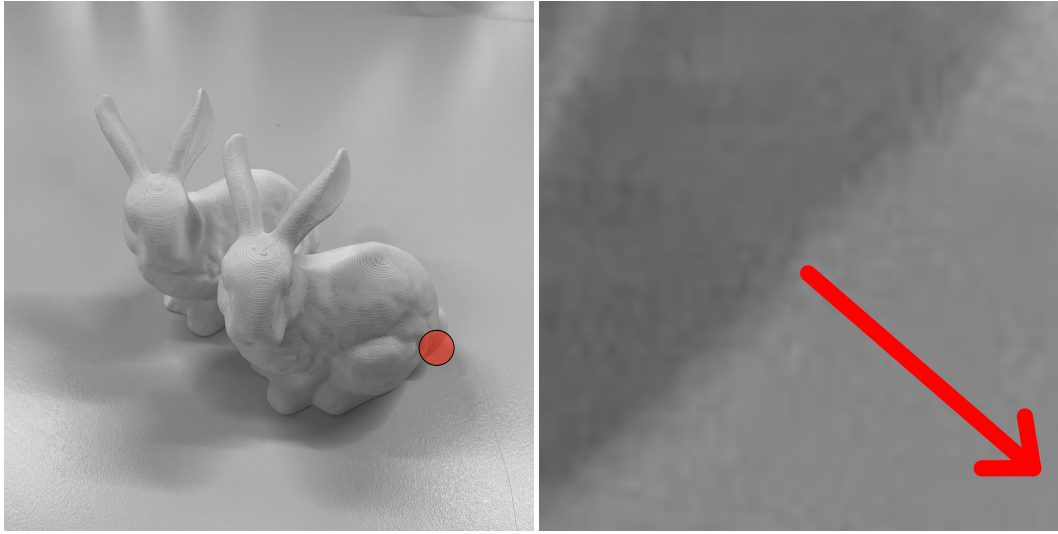
- one row of the junction-file containing the coordinate of a high curvature point, which is a L-Junction in this case:

```
2383 91 2
```

3.2 Contour-Detection

The first stage is, to extract edges in an image. To do so, the gradient for each individual pixel is determined, which is a vector that points in the direction of the steepest slope.

This is noticeable in Figure 3.2: When one looks at the bottom of the rabbit's tail and zoom in, a change from a dark grey to a lighter grey is observable. In the case of grey-scale images, this means that the vector points in the direction of the fastest change in grey-level. This vector is determined by using so called „steerable filters“. Without going into too much mathematical detail, steerable filters „look“ in a range of different directions and determine, by how much a function changes in these directions. This is called the directional derivative of a function. In the case of grey-scale images, the function is depicted by a change of grey values



(a) Stanford rabbits

(b) Gradient at a certain pixel coordinate

Figure 3.2: In (b) the image is zoomed into the area that was marked with the red dot in (a). The red arrow in (b) represents the direction of the gradient at the pixel coordinate, located at the base of the arrow.

along one of these directions. In Figure 3.3 the distribution of these directions is depicted. The number of sampled directions p determines the precision of the gradient and θ_0 the angle between each sampling direction. In the image $p = 8$ is used, but the proposed algorithm uses $p = 24$ resulting in $\theta_0 = 15^\circ$. For every sampled direction the directional derivative is evaluated, which is displayed by the length of the vectors in Figure 3.3. For explanation purposes the vectors in the image have different lengths, whereas in reality the vector with the highest derivative response is chosen. The returned vectors themselves have unit length. For simplification, I will often use the term „gradient value“ when the highest derivative response of steerable filters is meant. The directional derivative is also convolved with a 3D-Gaussian, with a standard deviation of σ , for noise reduction purposes. Since this thesis is concerned with the influence of noise on the output of this algorithm, one could think that the denoising operation could have an effect on the tests performed later on, but because this denoising operation is mostly used for shot-noise reduction, I assume that it will have no effect on narrowband noise. σ is set to a default value of 2 by the authors, but can be set as an input-parameter when the algorithm is executed.

The directions are sampled in the range $\theta = [0, 2\pi]$ (corresponding to $[0^\circ, 360^\circ]$), which can be written as a vector $\vec{\theta}$ in the form $\vec{\theta} = (\cos(\theta), \sin(\theta))^\top$. Since Buades

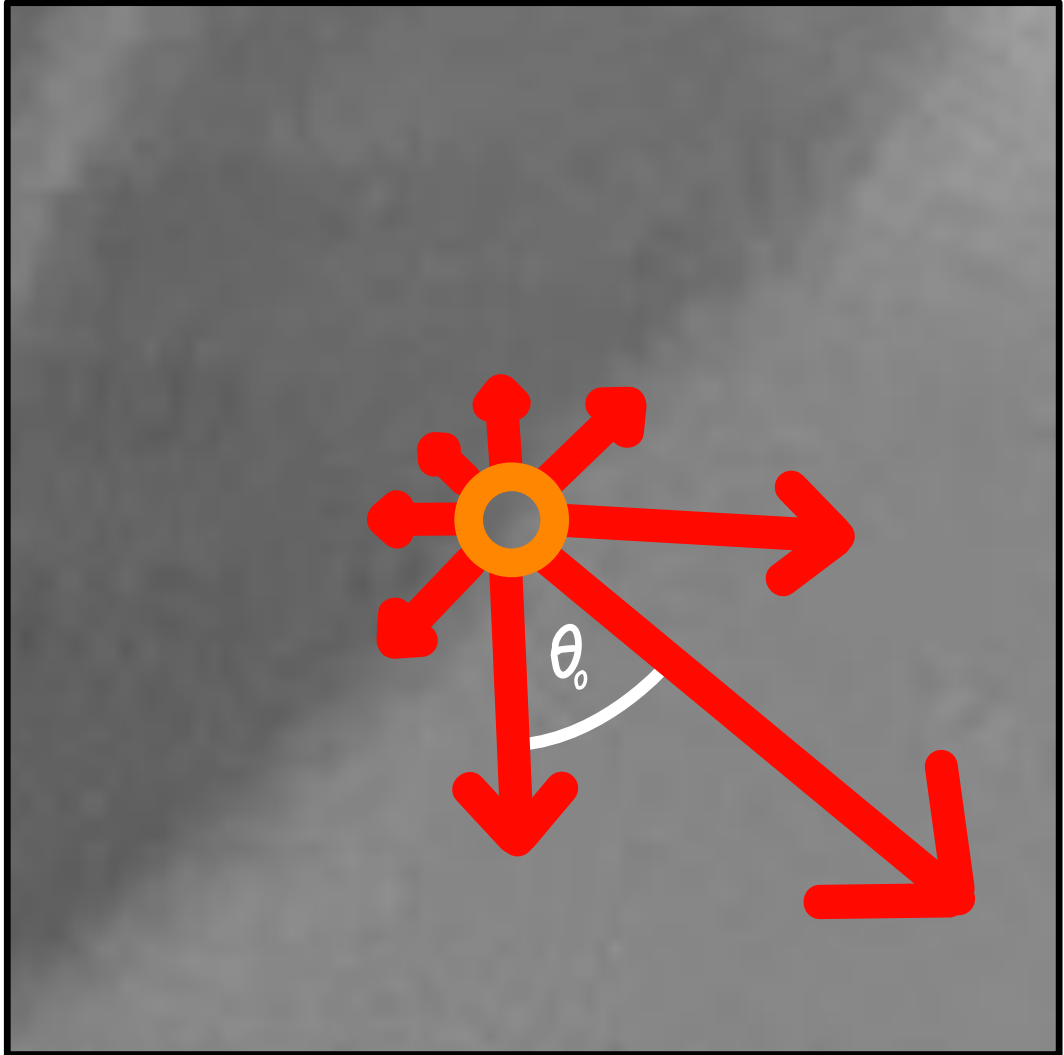


Figure 3.3: Sampling directions of steerable filters. The angle θ_0 is dependent on the number of sampling directions. In this image $\theta_0 = 45^\circ$ because 8 different directions are sampled and distributed equally over 360° . Short arrows represent a low directional derivative response in that direction and longer arrows a higher response.

et al. [BGN18] didn't differentiate between the direction vector $\vec{\theta}$ and the angle θ in their paper, the following usage of the vector-notation alters from their used notation, but is necessary in my opinion to properly understand the following process.

To sum this part up: The direction-vector $\vec{\theta}$, in whose direction the highest derivative response is returned, is the gradient. The algorithm now has the information, how each pixel is oriented. But this operation returns a value for every pixel in the image, as one can see in Figure 3.4 (b), so pixels that are not needed have to be discarded. To clear pixels that have a very low derivative response and therefore most likely don't belong to an edge, a threshold can be set, where responses that are below it are ignored. How the image looks like, after the threshold was applied is depicted in Figure 3.4 (c). This threshold was set by the authors to a fixed value of $\tau = 2.5$ but can be altered as an input parameter when the algorithm gets executed.

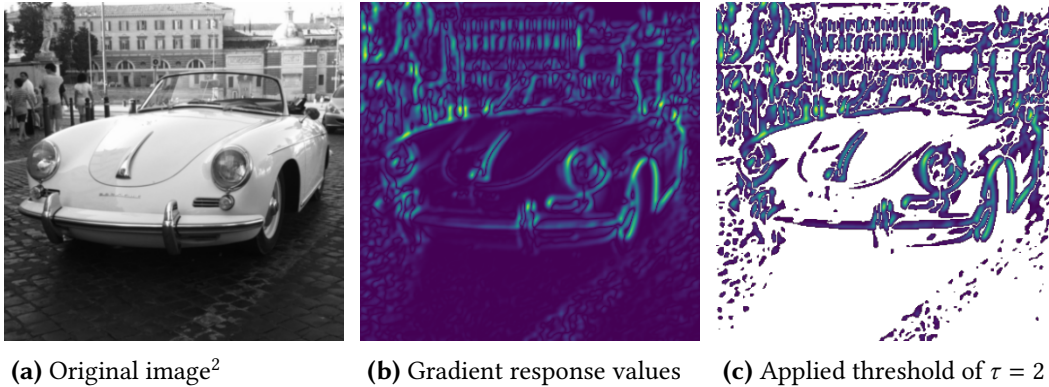


Figure 3.4: Determination of the gradient with the image in (a) as input. In (b) the returned gradient values are shown where yellow means a very high response and purple a very low one. In (c) the threshold was applied and only the values above the threshold are kept.

The pixels that are left, have a high possibility of being part of an edge, but as one can see in Figure 3.5 (b), the edge of the car's hood is represented by multiple pixels. This complicates the detection of high-curvature points later on, so an edge that is only one pixel wide is wanted. Because of that, a so called lateral inhibition is applied. This process runs over every retained pixel and looks if there is a neighboring pixel with a higher response, meaning that the current pixel doesn't represent the edge as well as the neighboring pixel. If that is the case, the current pixel is also discarded. The problem is that neighboring pixels can't be treated equally, since a neighboring pixel could have a higher response than the

² <https://ipolcore.ipol.im/demo/clientApp/demo.html?id=218>

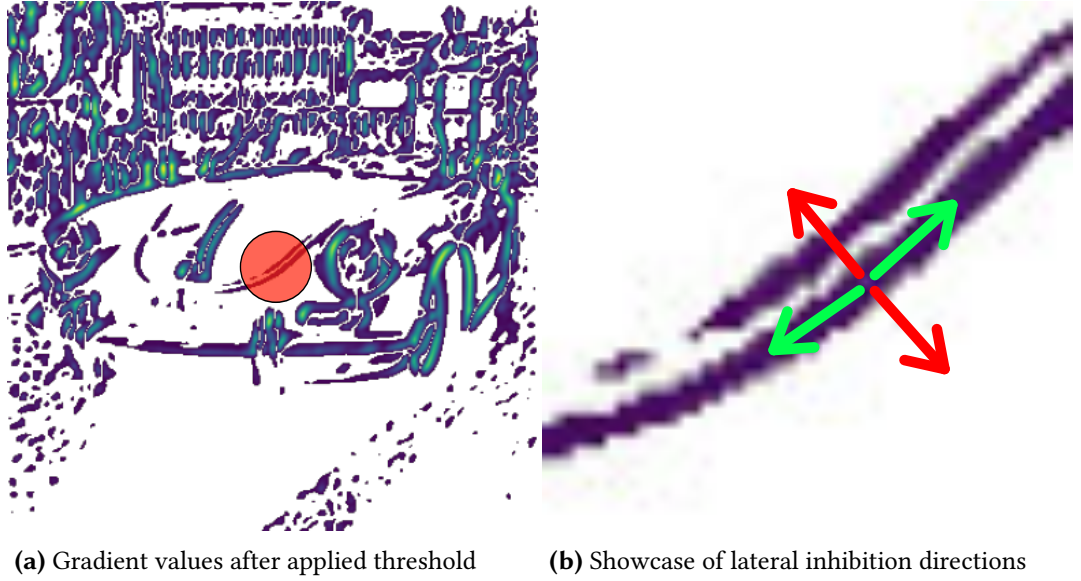


Figure 3.5: In (b) the image is zoomed into the area that was marked with the red dot in (a). The red arrows in (b) represent the direction in which pixels get inhibited, whereas the green arrows show the directions where pixels have to be kept and reinforced.

current one, but both pixels are important to keep, to resemble the course of an edge. In Figure 3.5 (b) this is represented by the red and green arrows: The pixels in the direction of the green arrows have to be retained, whereas the pixels in the direction of the red arrows have to be discarded. Because of that, a neighborhood has to be defined that excludes the pixels which lie in the direction of the green arrows:

$$\{(x', y', \theta'_0) \mid |\vec{d} \cdot \vec{\theta}| > |\vec{d} \cdot \vec{\theta}_\perp| \text{ and } |\theta - \theta'| < \theta_0\},$$

where x and y resemble the coordinates of the current pixel and θ its orientation. Following from that, x' , y' and θ' are the same information for a pixel adjacent to the current pixel. $\vec{d}$ is the vector that points from the current pixel to the neighboring pixel, defined as $\vec{d} = (x - x', y - y')$. $\vec{\theta}_\perp$ describes a perpendicular orientation-vector to a given one, so if $\vec{\theta} = (\cos(\theta), \sin(\theta))^\top$ then $\vec{\theta}_\perp = (-\sin(\theta), \cos(\theta))^\top$. For intuition: If θ describes an angle then $\theta_\perp = \theta + 90^\circ$.

When this process is done, every retained pixel is set to a value of one and the rest of the image-pixels to zero, which results in an image with the probable edges set

to black and the rest set to white. Because an inhibition was applied to the image, the retained pixels have to be reinforced. To do so an association is applied, which is the exact opposite of the process explained above.

3.3 T-Junction and corner detection

After the association process is done, what is left is a local density around each pixel in the same sample range as before ($\theta = [0, 2\pi]$). This means that now the information is given, how much a pixel contributes to other pixels in a given direction. From that it can be derived that if a pixel has a high response to more than one direction, this pixel is a high-curvature point. To do so, a threshold of 30% was set by Buades et al., so that a pixel has to contribute to at least 30% of the sampled directions. When p symbolizes the number of samples in the range $[0, 2\pi]$, then the threshold is $\rho = 0.3p$. Based on this information, the algorithm iterates over every high curvature point found and checks in a radius of $r = 2\sigma$ how many line-segments arrive in this area. By this amount T- and L-Junctions can be classified, as one can observe in Figure 3.6.

3.4 Aim

Like for white's illusion (Figure 2.3), it is assumed that humans detect T-Junctions based on the change of luminance at object borders. [ZSS97] In addition, Betz et al. [Bet+15] conducted experiments to test if humans perceive the effect of white's illusion when the image was masked with narrowband noise at different center frequencies. They found, that at low and high center frequencies humans were able to perceive the effect, while mid-range center frequencies seemed to disturb the effect as displayed in Figure 3.8. Based on this, it can be assumed that humans fail to perceive T-Junctions under the same conditions as in the conducted experiments for white's illusion. Since humans perceive lightness differently although the luminance is not changed, this raises the question how these results compare to a computer algorithm, which only focuses on raw pixel values.

Following from that, the aim of this thesis is going to be, how noise-masking of different images at different center frequencies affects the amount of detected T-Junctions by the proposed algorithm.

In the next part I am going to explain how the algorithm was used and which influence different input parameters have on the output, if no noise-masking was applied. Also the creation of the noise masking is going to be illustrated.

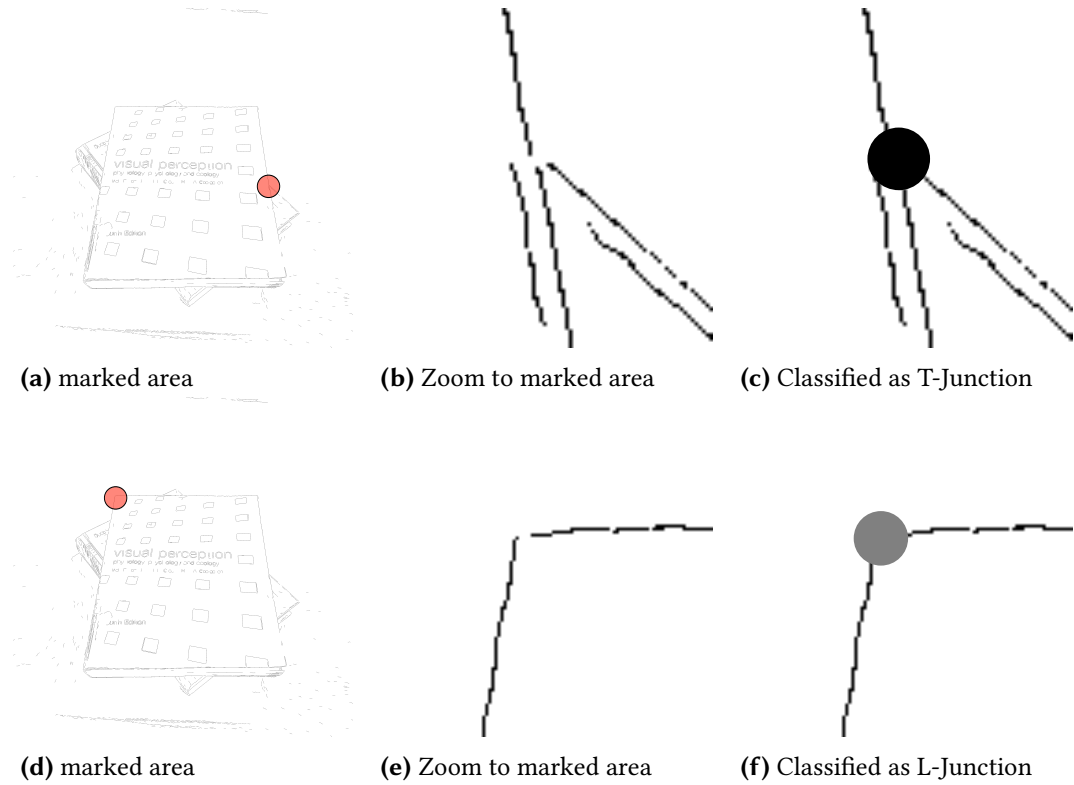
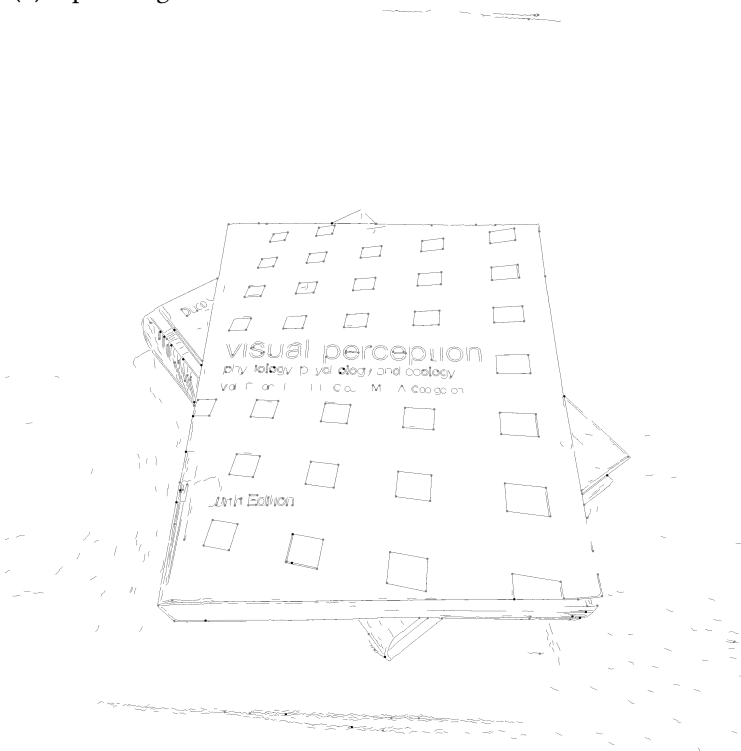


Figure 3.6: Using the image from Figure 3.7(a) as input, the output of the algorithm is shown in (a) and (d) without a classification of T- and L-Junctions. (b) and (e) represent a zoom to the red areas that were marked in (a) and (d). As one can see, there is a point in (b) where three line segments meet each other as well as two line segments in (e). These points are marked by the algorithm with black dots for T-Junctions and grey dots for L-Junctions. How the actual output looks like is depicted in Figure 3.7



(a) Input Image



(b) Output of the algorithm

Figure 3.7: In- and output of the algorithm, using the output-file without colored contours. As one can see, small black and grey dots have been added to the areas shown in Figure 3.6 (a) and (d).

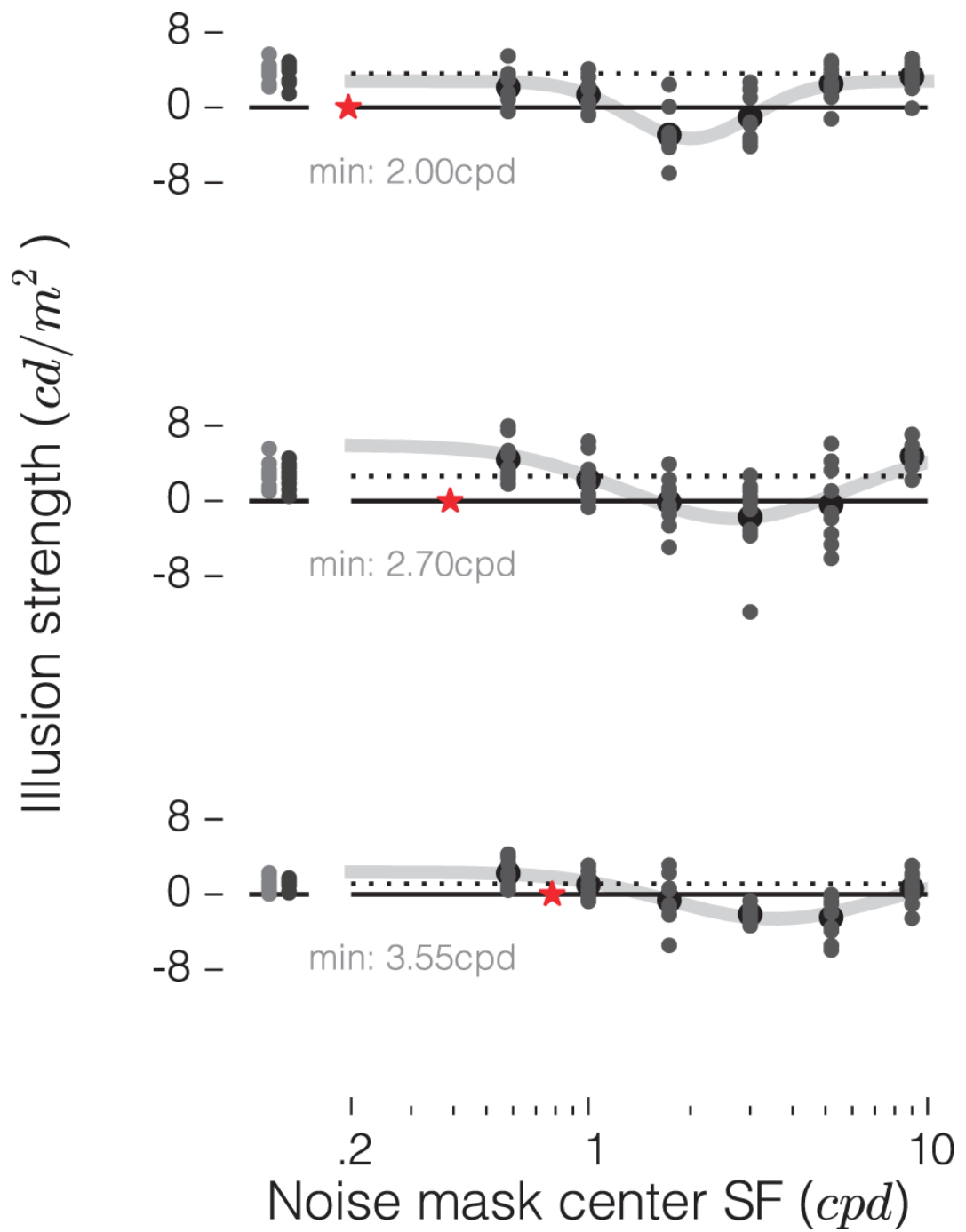


Figure 3.8: Results from Betz et al. On the x-axis the center frequency is displayed and the y-axis shows the strength of white's illusion.[Bet+15, Page 8]

Part II

Methods

Looking at the output of the algorithm in Figure 3.1 one can see that it is hard to determine if T-Junctions were identified correctly or not. In order to measure if the amount of detected T-Junctions as well as their position is correct, it is necessary to have an image where these parameters are easy to count for a human being. This count can then be used to evaluate if the output deviates from it. Counting T-Junctions manually in images like Figure 3.1 or Figure 3.7 is a task that may lead to miscounts. Because of that I created two own stimuli and used an image of white's illusion, as one can see in Figure 4.1. White's illusion was used in the work by Betz et al. [Bet+15], which is why I chose this stimulus in addition. Using these stimuli, it was pretty easy to count the T-Junctions, which is displayed in Figure 4.2. This count can now be used to compare the results to later on.

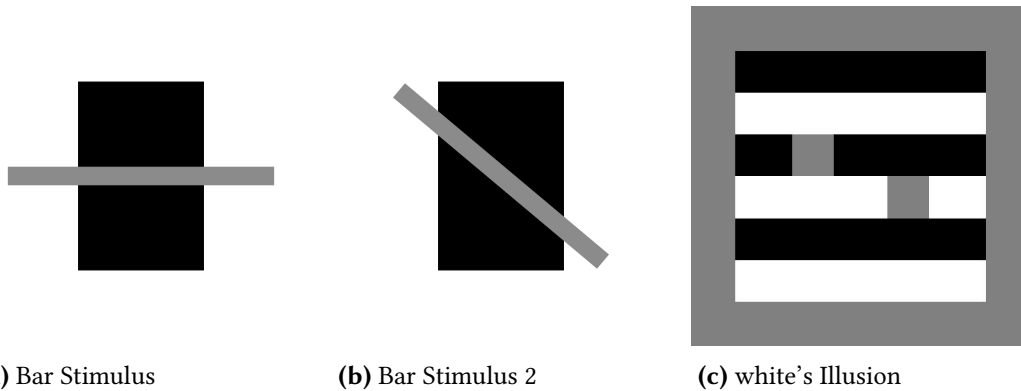


Figure 4.1: The two self-created stimuli (a,b) and an example of white's illusion (c) from [Bet+15, Betz et al.]

4.1 Creating stimuli with noise

Now noise was added to the chosen stimuli. To do so, narrowband noise-masks were sampled in the range of $[0.005, 0.5]$ cpd with a stepsize of 0.005 cpd using the Python stimupy library [SMV23]. When conducting experiments with human participants, the center frequency of the noise mask is dependent on the viewing distance of the

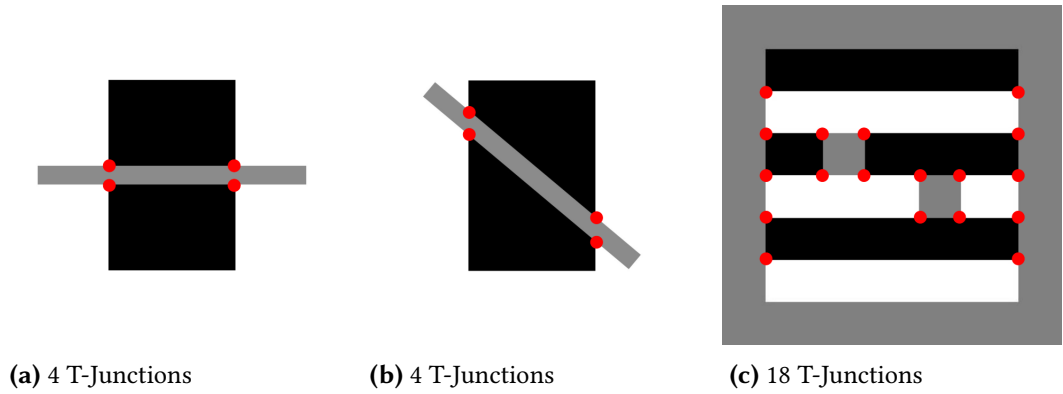


Figure 4.2: Every T-Junction in the 3 stimuli was marked with red dots and counted.

observer. The higher the distance, the higher the frequency is perceived. Because of that, one needs to know the amount of pixels that lie within one degree of the observers viewing angle, to determine at which center frequency the observer will perceive the noise mask. Since the experiment is not conducted with human participants, no viewing distance is given so the masks were sampled at 1 pixel per degree. This means that one pixel represents one degree of the viewing angle of the observer. When comparing the stimuli to stimuli used in experiments with human participants, the center frequency needs to be adjusted. For instance: When one looks at a monitor with a width of $w = 40$ and a horizontal resolution of $r = 1024$ from a distance of about $d = 50$ a factor can be calculated given by the formula³:

$$x = \left(\tan^{-1} \left(\frac{\frac{w}{r}}{d} \right) \cdot \frac{180}{\pi} \right)^{-1}$$

$$\Rightarrow \left(\tan^{-1} \left(\frac{\frac{40cm}{1024px}}{50cm} \right) \cdot \frac{180}{\pi} \right)^{-1} = 22,34ppd$$

Now the center frequency range given at 1ppd can be adjusted to the described monitor by just multiplying with the calculated factor:

$$cpd = cpd_{px} \cdot ppd_{new}$$

Following from that the range of $[0.005, 0.5]$ cpd would translate to about $[0.11, 11.7]$ cpd. The bandwidth was constant with a value of 1.0. The noise mask was applied by adding the grey-values of the two images. In order to avoid that the addition would

³ <https://osdoc.cogsci.nl/3.2/visualangle/>

lead to values above 255, which is white for the grey value range $[0, 255]$, the values of each pixel were scaled down:

$$\text{masked image} = 0.5 \cdot \text{image} + 0.5 \cdot \text{noise-mask}$$

In Figure 4.3 the noise mask (a) as well as the image, that the mask is going to be applied to (b), are shown. After they were scaled and added like explained above, this results in the image (c). Since the masks were created in the given range, 100 masks were created. Every mask applied to 3 images results in a total of 300 created stimuli. 15 of these stimuli are shown in Figure 4.4. When one looks at the second row from the top in Figure 4.4, the noise mask for the two stimuli on the left appears to be much more subtle than for white's illusion on the right. This is also true for the rest of the stimuli but not as noticeable as in the second row. The reason for that difference is, that the noise mask was created in a similar way for all stimuli, but the two stimuli on the left have a resolution of 1080×1080 pixels and the image of white's illusion on the right has a resolution of 400×400 pixels. Because the images are scaled to the same size in this Figure, the image of white's illusion got „stretched“ which makes the noise pattern appear at a different center frequency for the viewer. For the algorithm however this should make no difference, since it analyzes the images on a pixel level.

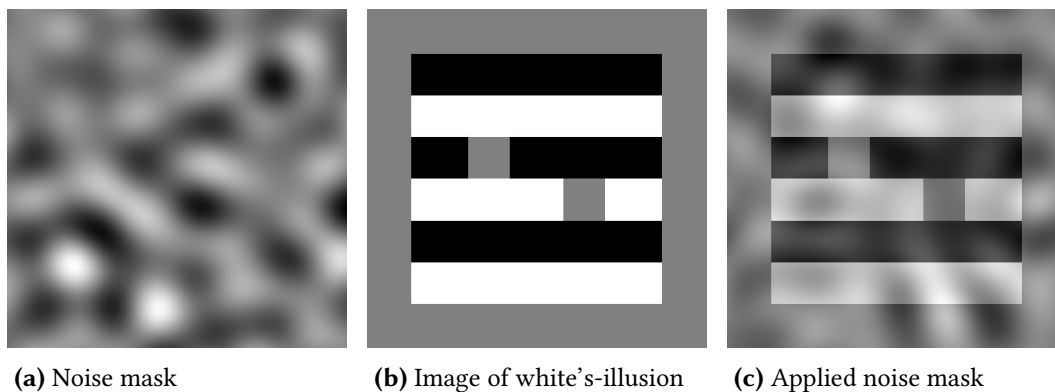


Figure 4.3: Process of noise-masking: From left to right: noise-mask, image, noise-masked image

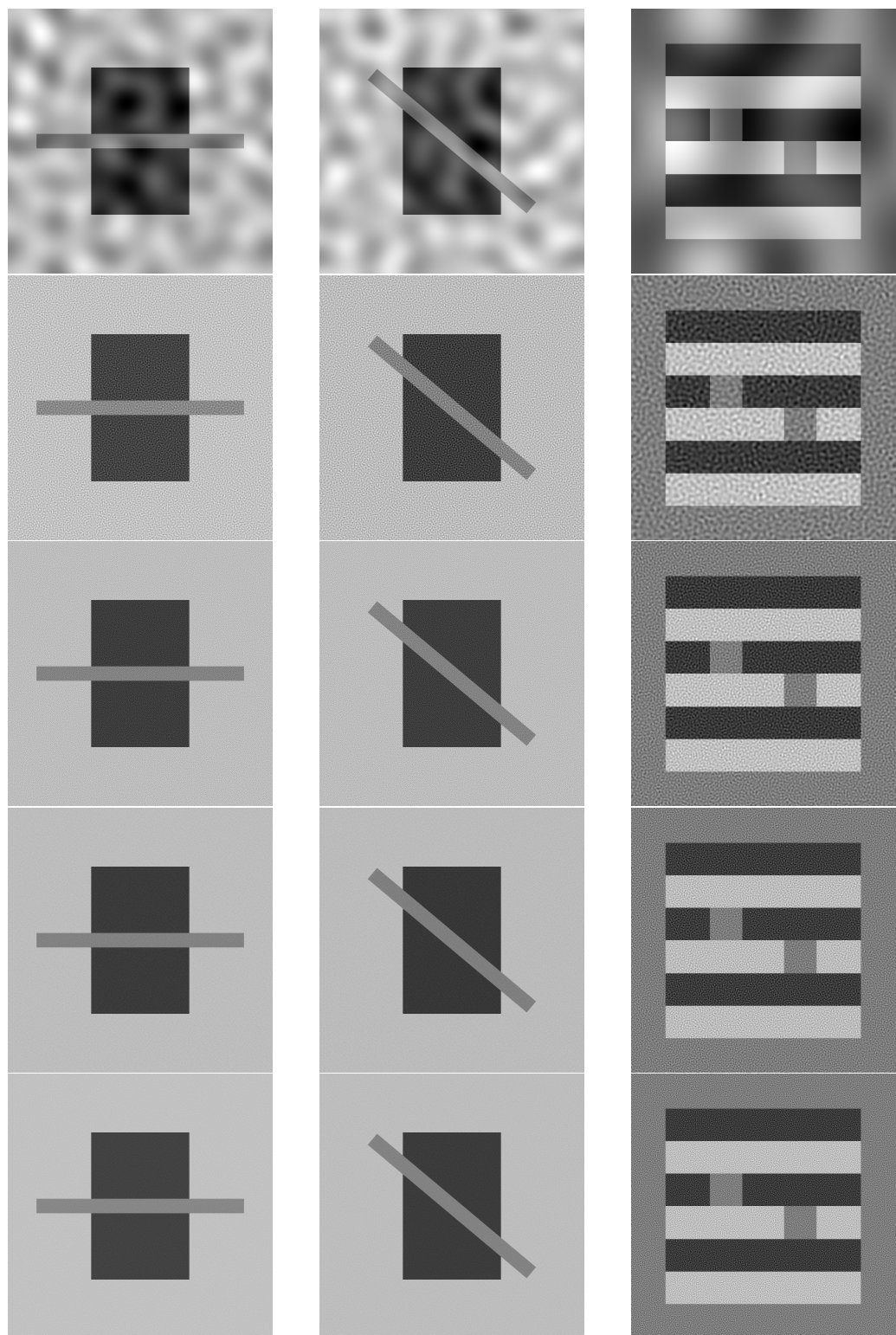


Figure 4.4: All stimuli from 0.005cpd to 0.405cpd with steps of 0.1cpd

5

Testing the algorithm

5.1 Executing the algorithm and handling the data

Since the algorithm had to be tested with many different input-images and parameters, I created a Jupyter Notebook environment which was capable of executing the algorithm multiple times automatically. This was possible by using the python subprocess library which lets the user execute bash commands via python. The algorithm was implemented in c++ so an executable was generated during the compilation, whose execution is shown below:

```
./{executable} -t {tau$} -s {sigma} {input-image} probs/fileprob.s  
{sigma} contoursGray.png contoursColor.png contours.txt junctions.txt
```

With subprocess, this translated to the version below:

```
import subprocess  
subprocess.run([f"./{code_location}/{executable}",  
               "-t", f"{tau}",  
               "-s", f"{sigma}",  
               f"{input_image}",  
               f"{probs}",  
               f"{output_image_grey}",  
               f"{output_image_color}",  
               f"{output_pos_contours}",  
               f"{output_pos_corners}"])
```

The input parameters were tau (τ), sigma (σ) and the input-image, as already explained earlier. The rest of the command determined, where the output was saved.

Since the L- and T-Junctions that were classified by the algorithm were stored in a txt-file, I wrote a Python script that iterated over these files and created a pandas dataframe. In this dataframe the amount of detected junctions was stored in relation to the image name and the noise mask that was applied to it. The complete jupyter notebook as well as all plots and images are available at my Github repository⁴

⁴ <https://github.com/RasElasedAustralis/BA-thesis-code>

5.2 Variation of input parameters

Before I started to use these noise-masked stimuli as an input for the algorithm, I first wanted to evaluate how the output looks like, if no noise was added. In addition to that, I tested how the output of the algorithm changes, if the images without noise are used as input and the two parameters τ and σ , which are fixed by default, are changed. As a short recap:

- τ is the threshold, that only keeps pixels whose gradient values are above it.
- σ is the the standard deviation of the gaussian that is used for noise reduction purposes.

5.2.1 Variation of τ

I started off by using the Stimuli from Figure 4.1 as input, testing thresholds between 1 and 3. The variation of the parameter showed no effect in that range. However, when the threshold was varied between 2 and 18, using the natural images from Figure 1.1 and Figure 3.1 as input, the output varied as one can see in Figure 5.1. Following from that, it can be defined that the variation of this parameter has no significant effect on the stimuli I used, if it stays in the range $[1, 3]$. This is the reason I used the default value of $\tau = 2.5$ for all following tests.

5.2.2 Variation of σ

For σ I tested with values in the range $[1, 4]$, since that is the same range the authors use on their demo-website⁵. To do so, the algorithm was executed multiple times in this range for σ in steps of 0.25 with the stimuli from Figure 4.1 used as input. As one can see in Figure 5.2, the amount of detected L-Junctions and T-Junctions changes for the image of white's illusion with varying standard deviations. To analyze how this parameter influences the amount of detected T-Junctions, a pandas dataframe was created by iterating over every junction-file using python. Based on this dataframe, the plots in Figure 5.3, Figure 5.4 and Figure 5.5 were created.

When looking at these plots, one can observe that the amount of detected T-Junctions not only varies heavily for each stimulus, but also between each stimulus: While the amount of detected T-Junctions in Bar-Stimulus-1 (Figure 5.3) stays at a maximum of 4 for the standard deviations between 1.0 and 2.0, this amount varies heavily for the image of white's illusion (Figure 5.5).

5 <https://ipolcore.ipol.im/demo/clientApp/demo.html?id=218>

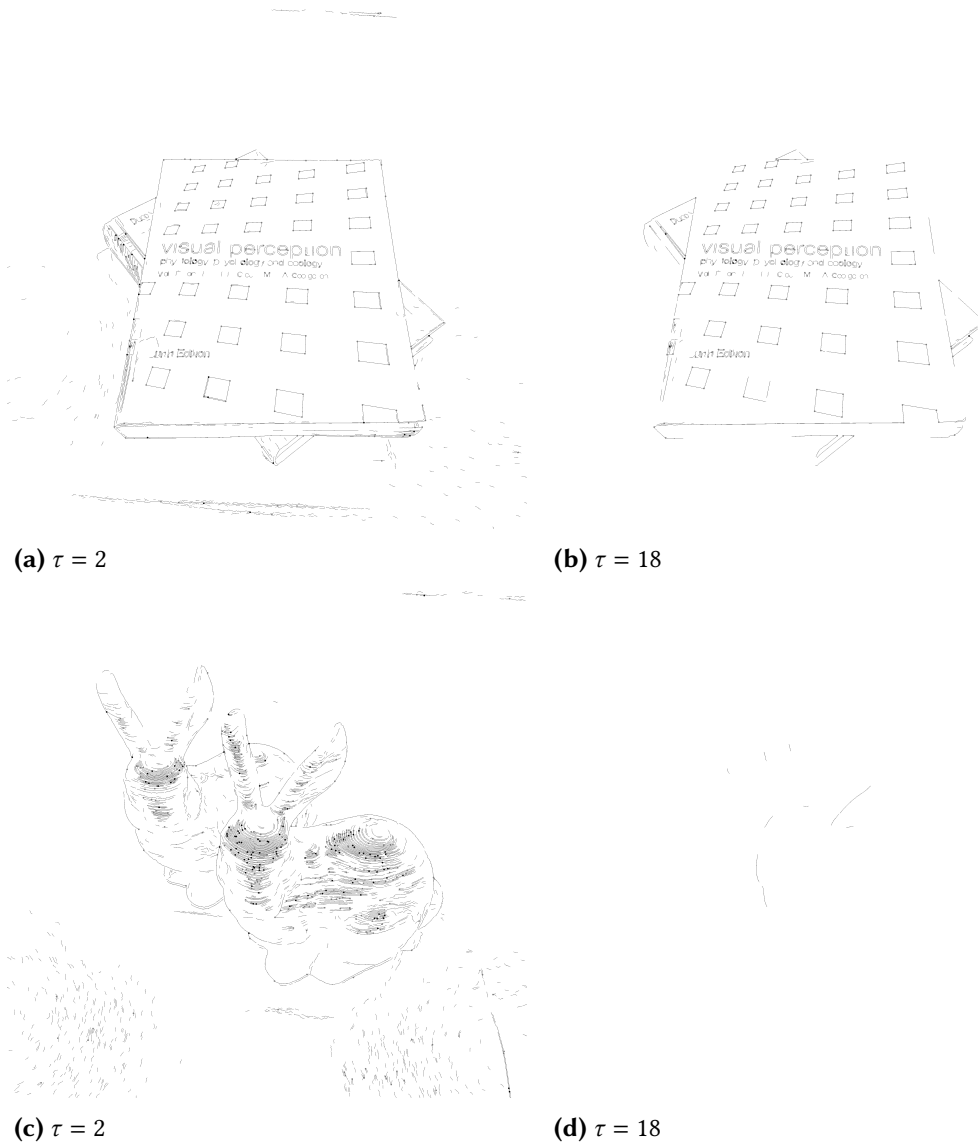


Figure 5.1: Depicted are two different natural images used with varying thresholds τ in the range $[2, 18]$. In (a) and (b) the image from Figure 2.6 was used and in (c) and (d) the image from Figure 1.1 was used.

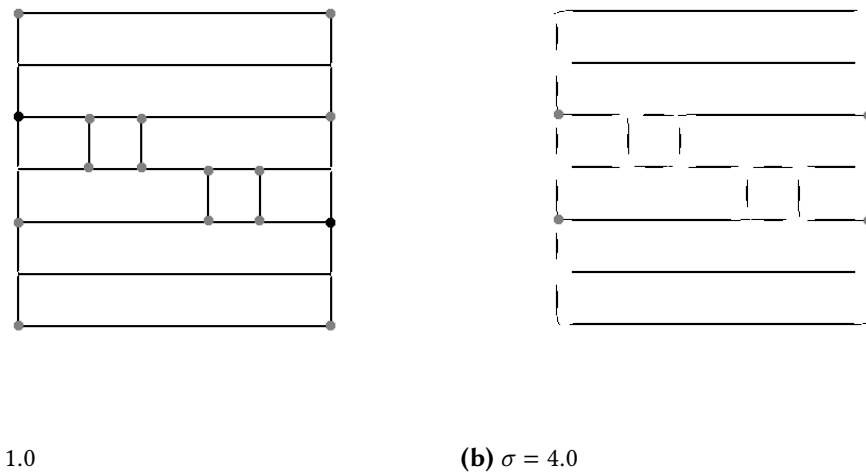


Figure 5.2: Output of the algorithm after white's illusion was used as the input at different standard deviations.

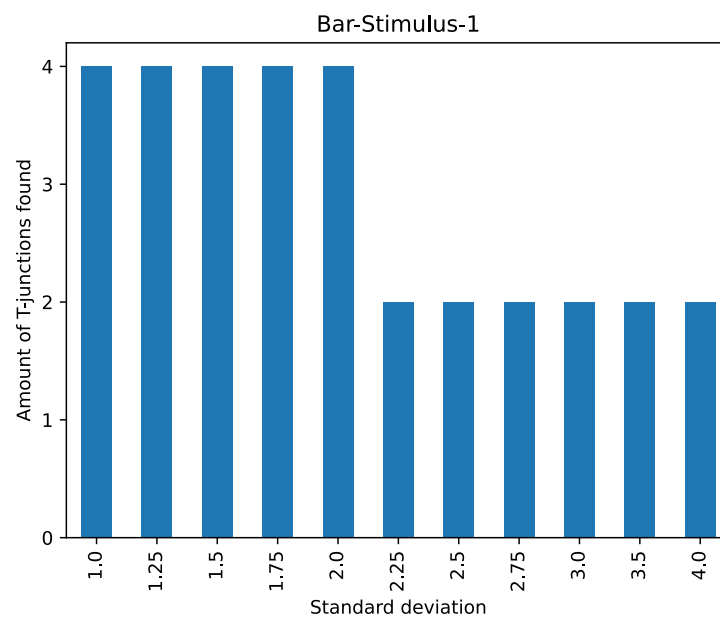
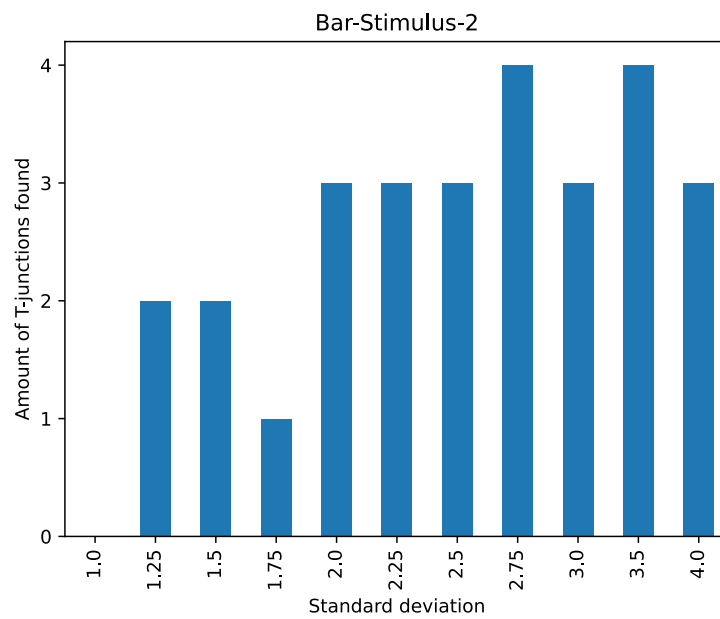
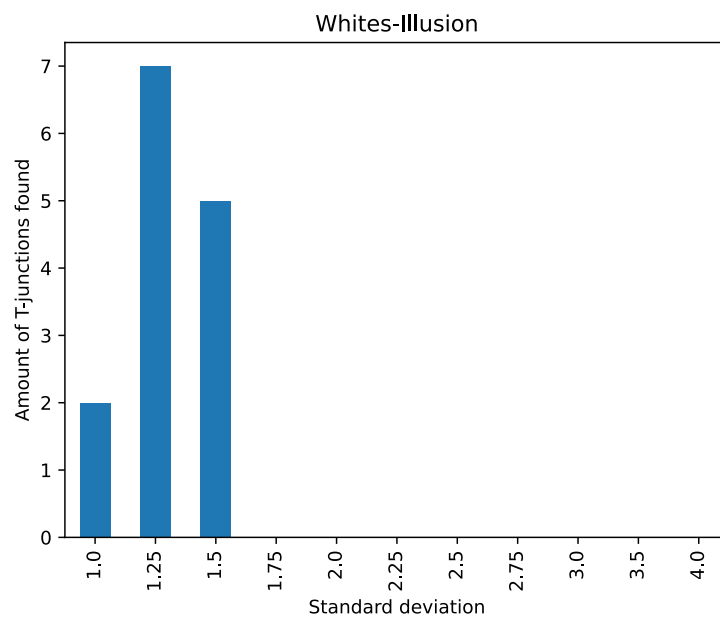


Figure 5.3: Bar stimulus 1

**Figure 5.4:** Bar stimulus 2**Figure 5.5:** white's illusion

The amount of found T-Junctions for this stimulus varies between 2 and 7 in the standard deviation range [1.0, 1.5]. In contrast to that, the highest amount of detected T-Junctions for Bar-Stimulus-2 is in the range of [2.75, 3.5]. The problem that this creates is, that every image the algorithm receives as input seems to have a different optimum for the standard deviation that has to be used for it. Since the objective of this thesis is the variation of the noise-masking, a variation of an additional parameter would not be feasible. Because of that I also used the default value of $\sigma = 2$.

5.3 Testing with noise-masked images

To test the algorithm by Buades et al., I used the noise-masked stimuli displayed in Figure 4.4. The parameters τ and σ were set to fixed values of $\tau = 2.5$ and $\sigma = 2.0$, as described before. To simplify the notation, the amount of detected T-Junctions for one stimulus will be represented by the variable T . The base values for T without noise can be read from the plots in Figure 5.3, Figure 5.4 and Figure 5.5:

- Bar stimulus 1: $T = 4$
- Bar stimulus 2: $T = 3$
- white's illusion: $T = 0$

Also the true values for T are given, by the counts given in Figure 4.2:

- Bar stimulus 1: $T = 4$
- Bar stimulus 2: $T = 4$
- white's illusion: $T = 18$

Part III

Results

6

Examining the data

6.1 First bar stimulus

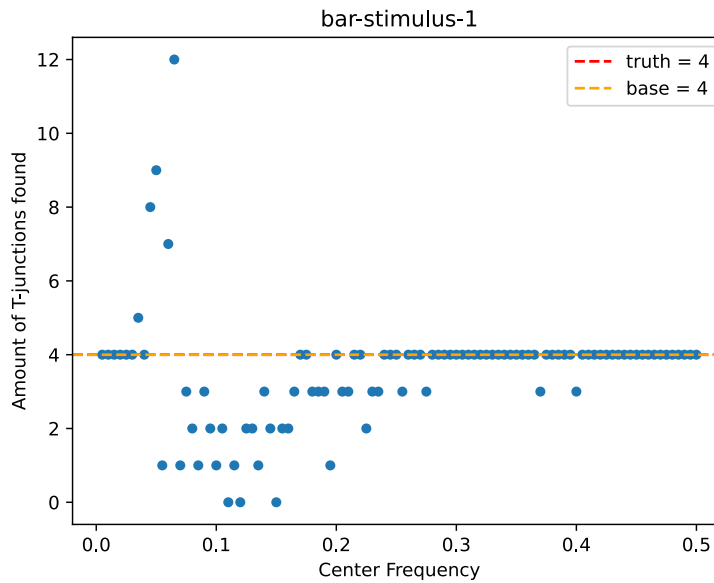


Figure 6.1: The amount of detected T-Junctions for the first bar stimulus. The truth and the base value match, so only one of the lines that mark these values is visible.

Looking at the plot in Figure 6.1 it shows the center-frequency of the used noise-masks on the x-axis and the amount of detected T-Junctions for the image, that was masked with this noise-mask, on the y-axis. The true amount of T-Junctions is marked with a red dashed line and the amount that was detected without noise in the stimulus is indicated by an orange dashed line. In the case of this plot, the two values are the same, resulting in the depiction of only the orange line. At low center frequencies between 0.005cpd and 0.02cpd the amount of detected T-Junctions stays at the base and true value of four. The in- and output for the stimulus at 0.005cpd is shown in Figure 6.4 (a) and (b). What is noticeable that the contours of the stimulus seem to get detected correctly, as well as the four T-Junctions. Also some „contour-patches“ are formed in the lower left corner of the output. At slightly

higher center frequencies between 0.03cpd and 0.07cpd a big increase of detected T-Junctions can be noticed. The peak of detected T-Junctions is reached at 0.065cpd with twelve detected T-Junctions. The in- and output at this center frequency is shown in Figure 6.4 (c) and (d). At this center frequency the output shows that some T-Junctions get detected correctly but most of them are detected in positions where no T-Junction can be found in the image without any noise. Increasing the center frequency of the noise mask further leads to a strong decrease in detected T-Junctions. The lowest points are reached between 0.1cpd and 0.13cpd. Looking at the in- and output in Figure 6.4 (e) and (f) at a center frequency of 0.12cpd, no T-Junctions were detected. Also the contours of the bars seem to be disrupted. At even higher center frequencies of 0.15cpd and above, the amount of T-Junctions stays the same at the true value of four. Again, the in- and output are shown in Figure 6.4 (g) and (h) at a center frequency of 0.17cpd. The contours seem to get detected correctly again as well as the T-Junctions.

6.2 Second bar stimulus

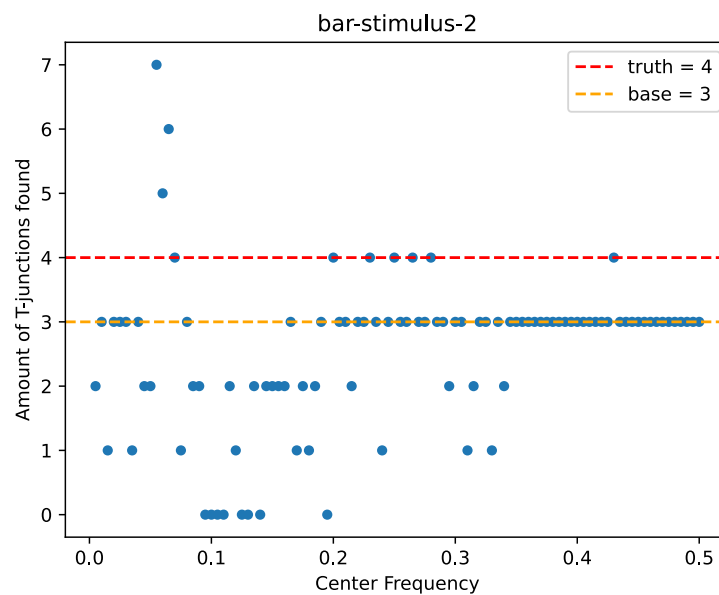


Figure 6.2: The results of the noise-masked versions of the second bar-stimulus.

The results for the second bar stimulus, shown in Figure 6.2, have similarities with the results from the first bar stimulus, like the low amount of detected T-Junctions

at 0.005cpd as well as the first spike at around 0.06cpd and the low amount at 0.13cpd. The in- and outputs for the stimuli at these center frequencies are shown in Figure 6.5 (a)-(d) respectively. The difference however is the discrepancy between the true amount of T-Junctions that should be detected and the amount that was detected without any noise masking. Another difference at center frequencies between 0.2cpd and 0.3cpd is, that an increase of detected T-Junctions above the base amount of three is shown, reaching the true amount of four. In Figure 6.5 (g) and (h), showing the in- and output at 0.2cpd, one can see that these classifications are correct.

6.3 White's illusion

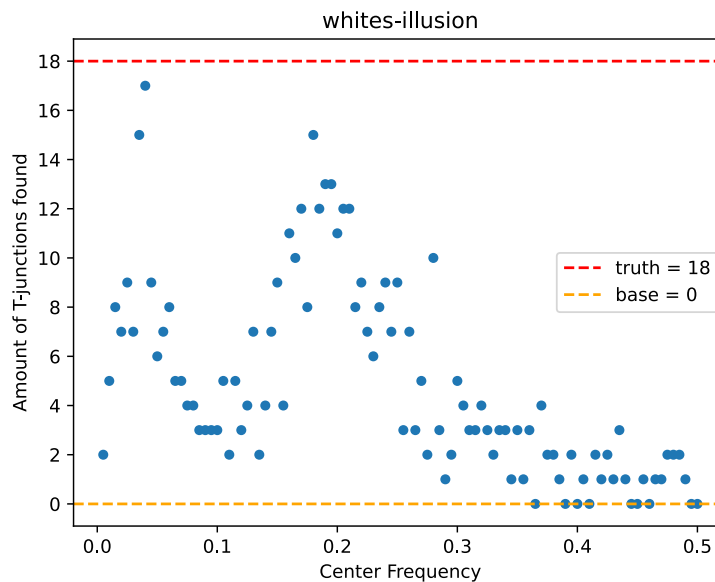
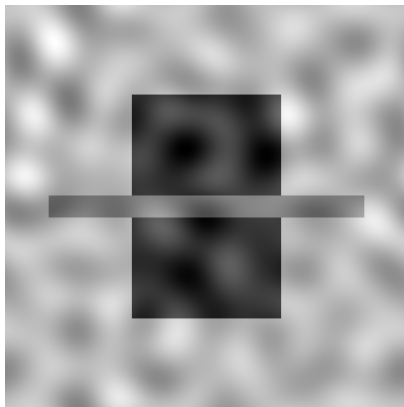


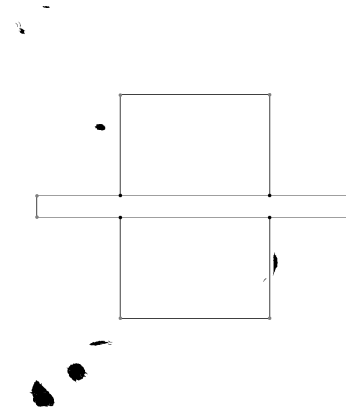
Figure 6.3: The results of the noise-masked versions of white's illusion.

In Figure 6.3 one can see that the results for white's illusion follow a similar pattern in comparison to the results from the second bar stimulus in Figure 6.2. Like for the first and the second bar stimulus, a low amount of T-Junctions gets detected between 0.005cpd and 0.15cpd with a high amount of incorrectly detected T-Junctions at 0.04cpd. The in- and outputs in that range are shown in Figure 6.6 (a)-(f). The big difference however, is the large discrepancy of the true and base value as well as the second spike of detected T-Junctions at around 0.2cpd. At

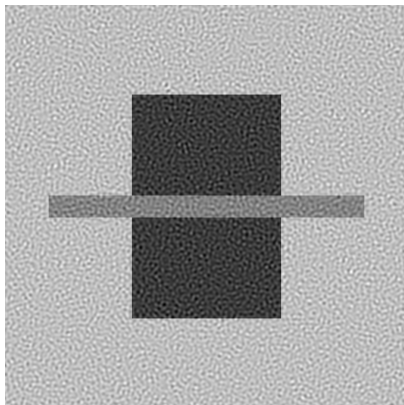
0.18cpd the largest amount of correctly identified T-Junctions, with a value of 15, can be seen. For this value, the in- and output is shown in Figure 6.6 (g) and (h).



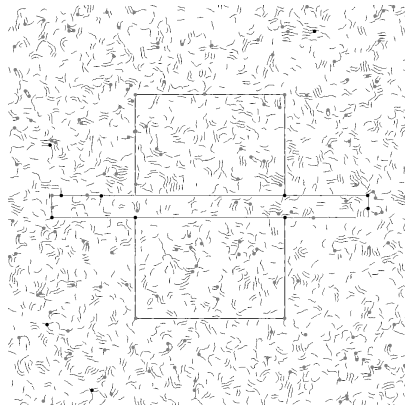
(a) Input at 0.005cpd



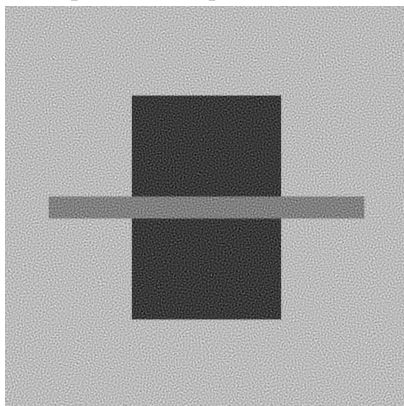
(b) Output at 0.005cpd



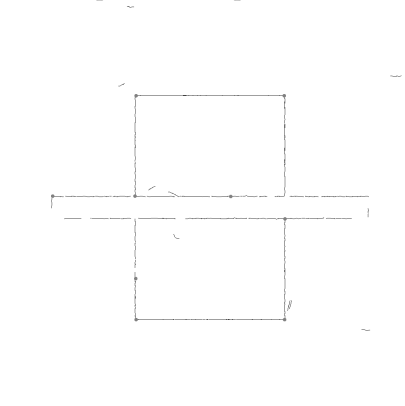
(c) Input at 0.065cpd



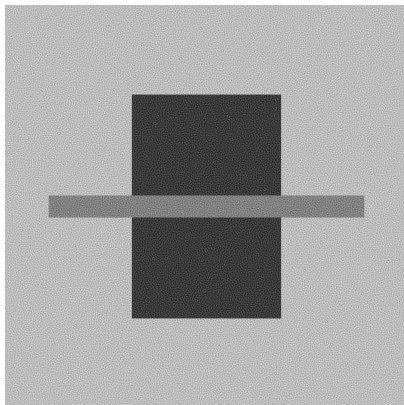
(d) Output at 0.065cpd



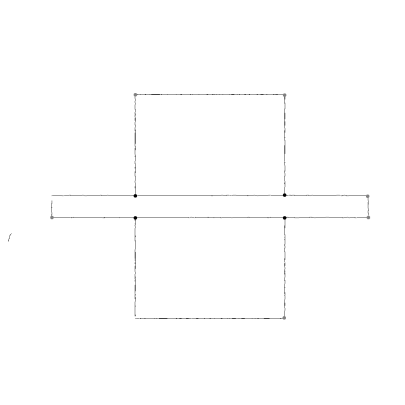
(e) Input at 0.12cpd



(f) Output at 0.12cpd

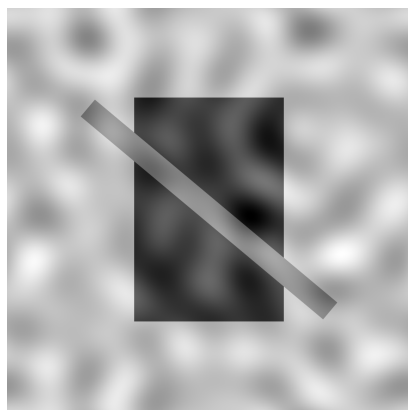


(g) Input at 0.17cpd

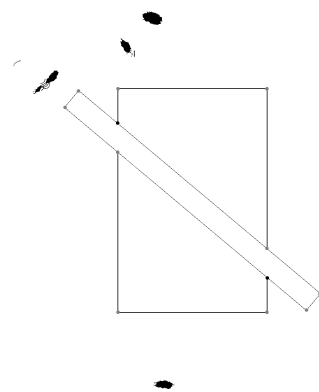


(h) Output at 0.17cpd

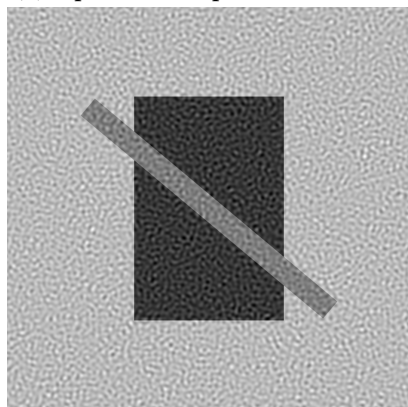
Figure 6.4: In- and output for the masked first bar stimulus at four different center frequencies



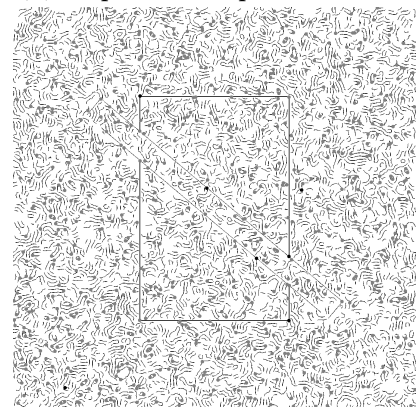
(a) Input at 0.005cpd



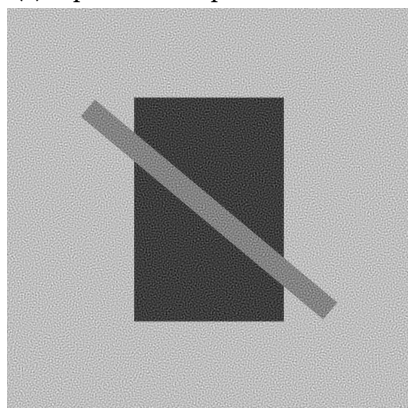
(b) Output at 0.005cpd



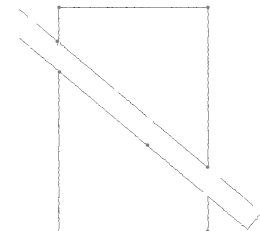
(c) Input at 0.055cpd



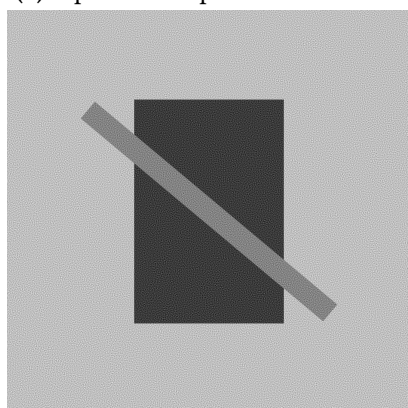
(d) Output at 0.055cpd



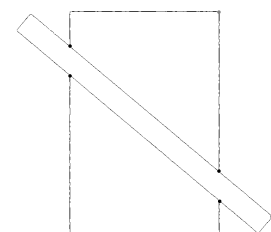
(e) Input at 0.13cpd



(f) Output at 0.13cpd

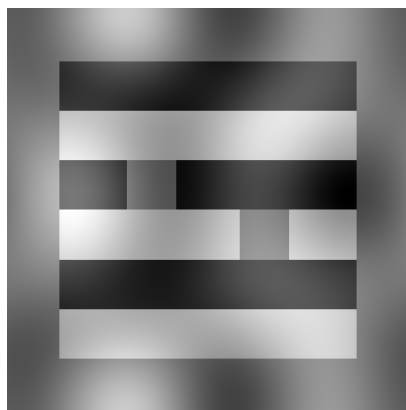


(g) Input at 0.2cpd

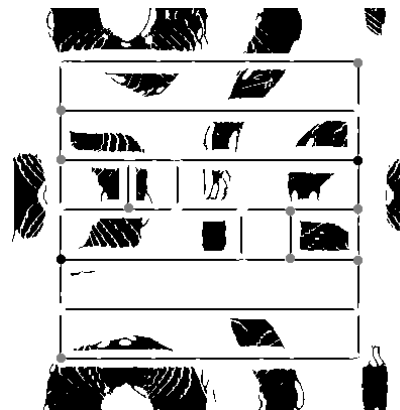


(h) Output at 0.2cpd

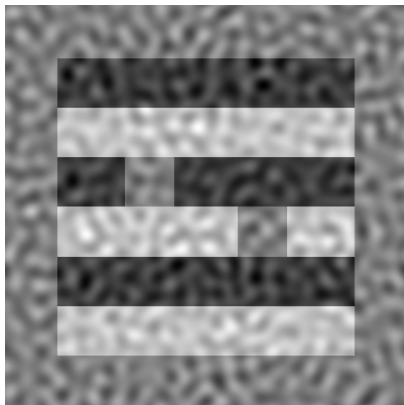
Figure 6.5: In- and output for the masked second bar stimulus at four different center frequencies.



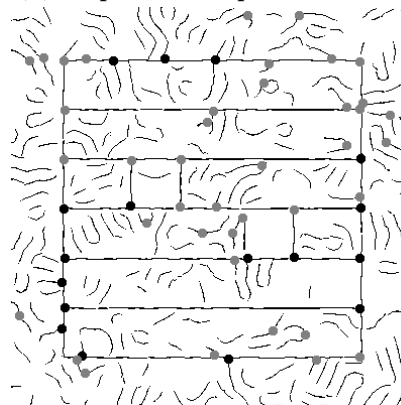
(a) Input at 0.005cpd



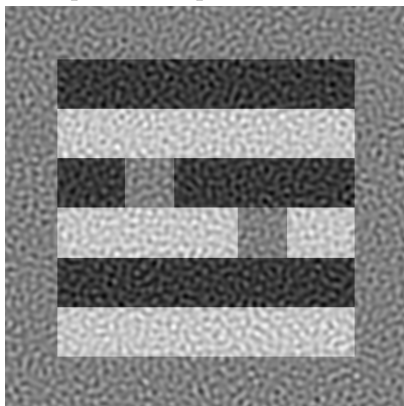
(b) Output at 0.005cpd



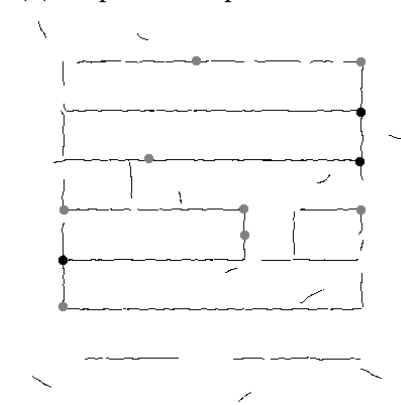
(c) Input at 0.04cpd



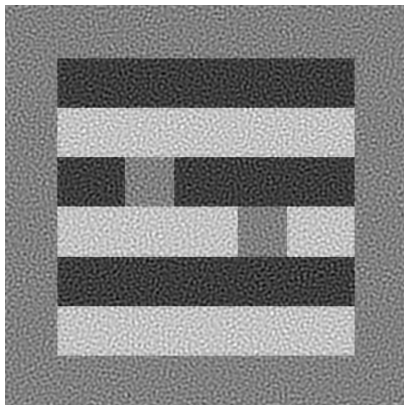
(d) Output at 0.04cpd



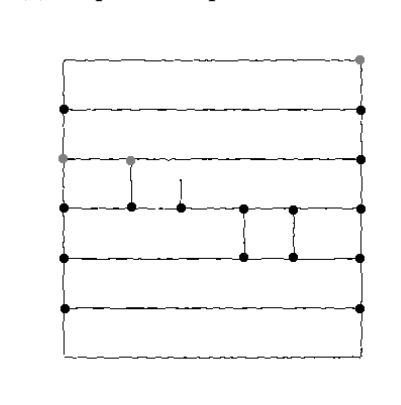
(e) Input at 0.1cpd



(f) Output at 0.1cpd



(g) Input at 0.18cpd



(h) Output at 0.18cpd

Figure 6.6: In- and output for the masked white's illusion at four different center frequencies

Part IV

Discussion

7.1 Influence of the base value

As mentioned in the previous chapter, the base amount of detected T-Junctions differed from the true value for the second bar stimulus and white's illusion. Because the base amount for white's illusion was zero, this leads to the problem that no differentiation can be made whether the low amount of detected T-Junctions was caused by the noise masking or if T-Junctions just get detected in the same way they would without noise masking. This is different for the second bar stimulus in Figure 6.2, since the base value is not zero. In Figure 6.2 it looks like center frequencies higher than 0.3cpd have no influence on the amount of detected T-Junctions. This is confirmed by the plot of the first bar stimulus in Figure 6.1.

In summary, one can say that the effect of narrowband noise disturbs the detection of T-Junctions at low center frequencies. At higher center frequencies the effect is dependent on the difference between the true amount of T-Junctions in the image and the detected amount without noise. If both values are the same, no significant disturbance or improvement can be measured above 0.2cpd. If the values differ, noise maskings at a center frequency between 0.2cpd and 0.3cpd can improve the amount of detected T-Junctions. The intensity of the improvement seems to be dependent on the difference between the truth and the base value.

7.2 Comparison to Betz et al.

My initial assumption when starting the experiment was, that low center frequencies would not affect the output of the algorithm. As one can see in Figure 7.1 (a) the grey value of the noise changes slowly and was therefore expected to lead to a low disturbance of the gradient value. Following from that, a high amount of detected T-Junctions was expected for low center frequencies. Mid center frequencies, shown in Figure 7.1 (b), were expected to have a high influence on the detected T-Junction amount since the noise pattern shown in Figure 7.1 (b) disturbs the contrast at the edges, leading to a low amount or misclassified T-Junctions. At higher center frequencies, a low disturbance was expected, since the noise-pattern would be so fine that it would barely influence contrast at the edges. This hypothesis is based on

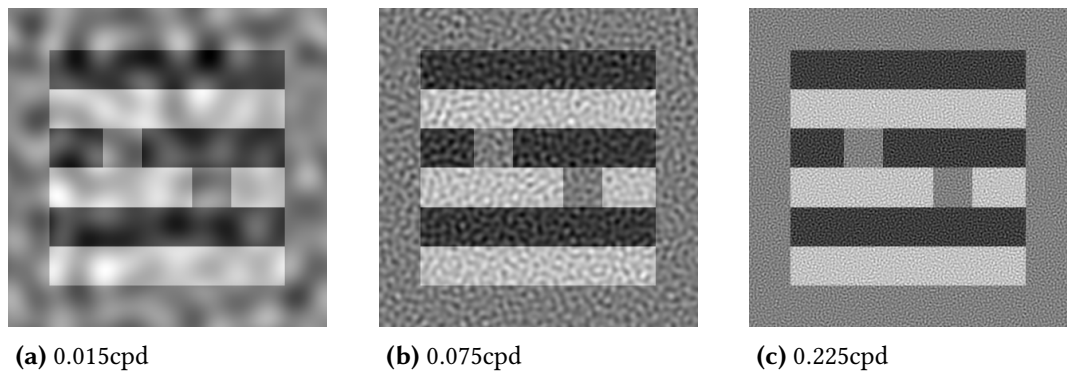


Figure 7.1: white's illusion, masked with narrowband noise at a low center frequency (a), mid center frequency (b) and high center frequency (c).

the intuition that follows from the results by Betz et al. [Bet+15], who came to these results with human participants, as shown in Figure 3.8. Before comparing the results to each other, the center frequency ranges need to be adjusted as described in Chapter 4.1. For the experiment by Betz et al., a Monitor with a width of 40cm and a horizontal resolution of 1024px was used. The participants were looking at the monitor from a distance of 70cm. Using the described formula, the resulting pixels per degree for their experiment is 31, 28ppd. Applying this factor to the range I tested in, results in the range [0.15, 15.6]cpd. The range the authors tested in, is [0.58, 9]cpd. For the plot, showing the results for white's illusion, in Figure 6.3 this corresponds to the range [0.018, 0.28]cpd. As one can see, the pattern of the plot in this range doesn't have any visual similarities to the plot shown in Figure 3.8. But this does not necessarily mean that no similarities exist when looking at specific regions of the plots:

The results by Betz et al. show an inhibitory effect of noise in the range of 2cpd to 4cpd. This corresponds to the range [0.064, 0.13]cpd in my results, where noise also had an inhibitory effect on the correctly detected amount of T-Junctions by the algorithm: Either it led to a lot of false positives, as shown in Figure 6.6 (c) and (d), or the contours were fragmented, leading to even less detected T-Junctions, as shown in Figure 6.6 (e) and (f). The inhibitory effect stops in the results by Betz et al. between 6cpd and 9cpd, which corresponds to the range [0.19, 0.29] in my results. Noise in this range led to an improved amount of detected T-Junctions by the algorithm. However, this comparison is limited because of the low base value for the second bar stimulus and whites illusion.

7.3 Difficulties and future work

During the research phase for this thesis I found two papers by Kang et al.[KCP05] and by Dimiccoli et al.[DS09]. Since the code for both papers was not published, I wasn't able to test another algorithm and compare the results to the ones I got from the algorithm by Buades et al.[BGN18]. Dr. Dimiccoli kindly responded to my request but unfortunately the code was no longer available to her. In the future, one could implement the algorithms proposed in the mentioned papers and compare the results to the ones from this thesis. In addition the algorithm by Buades et al. could be tuned to work better in no-noise scenarios, which could lead to a better comparability. This thesis focused on narrowband noise so one could follow up with testing different noise types like pink noise.

In this thesis I masked different stimuli with narrowband noise at different center frequencies and measured how many T-Junctions were detected for each stimulus at each center frequency by a computer algorithm. In order to have a truth value, the amount of T-Junctions in each stimulus was counted manually. To have base values to compare this amount of T-Junctions to, the algorithm was also executed with the unmasked stimuli. The results show, that narrowband noise at low center frequencies between 0.04cpd and 0.13cpd disturbs the correct classification of T-Junctions by the algorithm. At higher center frequencies between 0.17 and 0.2cpd, the amount of correctly detected T-Junctions was increased, if the base amount was lower than the truth. The results were also compared to the findings by Betz et al.[Bet+15]. This comparison showed, that noise masking leads to an inhibitory effect in the same center frequency range for both experiments. However the comparability of the two experiments is limited: Due to the low base value for two of the stimuli, the reason for a low amount of detected T-Junctions is unclear. Even though the results of this thesis show a similar pattern when compared to each other, no clear statement can be made with respect to other experiments.

Usage of AI tools

For this thesis, ChatGPT⁶ was used to program subtasks in python. Prompts like „How do I create a 3D plot of grey values in python?“ were used to speed up the work in that regard. Prompts like „How do I display code in latex?“ were used to speed up the research process for certain latex intricacies.

⁶ chatgpt.com

Bibliography

- [Bet+15] Torsten Betz, Robert Shapley, Felix A. Wichmann, and Marianne Maertens. **Noise masking of White’s illusion exposes the weakness of current spatial filtering models of lightness perception.** *Journal of Vision* 15:14 (Oct. 2015), 1–1. ISSN: 1534-7362. DOI: 10.1167/15.14.1. eprint: https://arvojournals.org/arvo/content_public/journal/jov/934563/i1534-7362-15-14-1.pdf. URL: <https://doi.org/10.1167/15.14.1> (see pages v, vii, 6, 15, 18, 21, 44, 47).
- [BGN18] Antoni Buades, Rafael Grompone von Gioi, and Julia Navarro. **Contours, Corners and T-Junctions Detection Algorithm.** *Image Processing On Line* (2018), 24–36 (see pages 9, 13, 45).
- [Bol24] Hannah Boldt. **At which point does noise masking cause a layer scission?** (2024). URL: https://www.psyco.tu-berlin.de/files/theses/thesis_boldt.pdf (see page 6).
- [DS09] Mariella Dimiccoli and Philippe Salembier. **Exploiting T-junctions for depth segregation in single images** (2009), 1229–1232. DOI: 10.1109/ICASSP.2009.4959812 (see pages 9, 45).
- [KCP05] Sung Kwan Kang, Young Chul Choung, and Jong An Park. **Image Corner Detection Using Hough Transform** (2005). Ed. by Jorge S. Marques, Nicolás Pérez de la Blanca, and Pedro Pina, 279–286 (see pages 9, 45).
- [SMV23] Lynn Schmittwilken, Marianne Maertens, and Joris Vincent. **stimupy: A Python package for creating stimuli in vision science.** *Journal of Open Source Software* 8:86 (2023), 5321. DOI: 10.21105/joss.05321. URL: <https://doi.org/10.21105/joss.05321> (see page 21).
- [ZSS97] Q. Zaidi, B. Spehar, and M. Shy. **Induced Effects of Backgrounds and Foregrounds in Three-Dimensional Configurations: The Role of T-Junctions.** (1997). URL: <https://doi.org/10.1068/p260395> (see pages 8, 15).