BACHELOR

3D reconstruction of streamer discharges using mathematical algorithms based on predictive back-end reconstruction, minimum spanning trees and sub-Riemannian geometry

de Veld, F.T.

Award date:
2019

Link to publication
3D Reconstruction of streamer discharges using mathematical algorithms
Based on predictive back-end reconstruction, minimum spanning trees and sub-Riemannian geometry

Author
Frank de Veld (1010914)

Supervisors
ir. Siebe Dijcks
dr. ir. Sander Nijdam
ir. Erik Bekkers
prof. dr. Luc Florack

June 26, 2019
Abstract

In this report a systematic approach is described on how to reconstruct streamer discharges in three-dimensional space. These streamer discharges are ionized channels of gas which occur often in high-voltage applications. Little is known about these streamers and currently all information about them is obtained from analysis of two-dimensional pictures or by means of simulations.

The aim of this project was to tackle this challenge of reconstructing a three-dimensional model of the streamer discharge. This challenge boils down to two main problems: reconstructing the streamer discharge in two-dimensional pictures and connecting several pictures of the same streamer discharge. In the setup, stereo-photographic and stroboscopic techniques have been used, with the result that pictures of two angles of the same streamer discharge were taken and that each picture was a combination of several photos with tiny shutter times. For the two-dimensional reconstruction two simple algorithms were used, one based on simple top-to-bottom sequencing and one predictive algorithm. Also, two advanced algorithms were used, based on more mathematical theory on sub-Riemannian distance and graph theory. After sequencing, the parts of the streamer discharge were connected via means of interpolation or by means of geodesics. The connection between the images was done by researching unique, defining properties of different segments of the streamer discharge and coupling these between the two images.

Four different algorithms are discussed with their main advantages and disadvantages and ultimately one main algorithm based on sub-Riemannian distances is advised to be used. Streamer discharges with up to eight branches and two overlapping streamer channels can be automatically reconstructed with great confidence. Further research should be focused on varying more parameters and doing extensive data-analysis on properties like velocity, branching angle and branching length, all of which are fundamentally three-dimensional properties.
# Contents

1 Introduction 1

2 Theory on Streamer Discharges 3
   2.1 Definition and Initialization of Electrical Streamers 3
   2.2 Initiation Cloud 4
   2.3 Behaviour of streamer discharges 5
      2.3.1 Positive and negative streamer discharges 6
      2.3.2 Streamer propagation 7
   2.4 Branching of streamer discharges 7
   2.5 Streamer discharges in nature 10

3 Mathematical Theory 11
   3.1 Metrics and norms 11
   3.2 Riemannian and sub-Riemannian space 13
      3.2.1 Nilpotent approximation for SE(2) 15
   3.3 Perceptual Grouping and Fast Marching Algorithms 17
      3.3.1 Perceptual Grouping Algorithm 17
      3.3.2 Fast Marching Algorithm 18
   3.4 Graph Theory 19
      3.4.1 Kruskal’s algorithm 19
   3.5 Interpolation 20
      3.5.1 Spline Interpolation 20
      3.5.2 Hermite Interpolation 21

4 Experimental Setup 22
   4.1 Gas-related parts of the setup 22
   4.2 Electricity-related parts of the setup 23
   4.3 Imaging-related parts of the setup 24
   4.4 Calibration 26

5 Creating a 3D Image 28
   5.1 Obtainable information 28
   5.2 Noise reduction 29
   5.3 Identification of Streamer Dots 30
   5.4 Analysis of Streamer Dots 31
   5.5 Connection between two two-dimensional images 31
   5.6 Experimental Remarks 33
   5.7 Error Analysis 34
   5.8 Two-dimensional reconstruction and basic image analysis 34
      5.8.1 Starting algorithm for creating three-dimensional images 35
      5.8.2 Predictive algorithm for streamer discharges with simple branching 37
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.8.3</td>
<td>Graph-based algorithm for streamer discharges with moderate branching</td>
<td>39</td>
</tr>
<tr>
<td>5.8.4</td>
<td>Curvature-based algorithm for streamer discharges with intricate branching</td>
<td>40</td>
</tr>
<tr>
<td>5.9</td>
<td>Connection between two-dimensional images</td>
<td>42</td>
</tr>
<tr>
<td>5.10</td>
<td>Transition to three-dimensional model</td>
<td>43</td>
</tr>
<tr>
<td>5.11</td>
<td>Comparison with existing methods</td>
<td>44</td>
</tr>
<tr>
<td>6</td>
<td>Results</td>
<td>46</td>
</tr>
<tr>
<td>6.1</td>
<td>Most basic algorithm results</td>
<td>46</td>
</tr>
<tr>
<td>6.2</td>
<td>Predictive Algorithm results</td>
<td>47</td>
</tr>
<tr>
<td>6.3</td>
<td>Minimum Spanning Tree results</td>
<td>50</td>
</tr>
<tr>
<td>6.4</td>
<td>Nilpotent Approximation results</td>
<td>52</td>
</tr>
<tr>
<td>6.5</td>
<td>Interpolation results</td>
<td>54</td>
</tr>
<tr>
<td>6.6</td>
<td>Miscellaneous Results</td>
<td>56</td>
</tr>
<tr>
<td>7</td>
<td>Conclusion</td>
<td>59</td>
</tr>
<tr>
<td>8</td>
<td>Discussion</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>Bibliography</td>
<td>64</td>
</tr>
<tr>
<td>A</td>
<td>Mathematica Code</td>
<td>66</td>
</tr>
<tr>
<td>B</td>
<td>Settings</td>
<td>68</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Lightning is a natural phenomenon that has intrigued humans for millennia on, yet still it has not been fully understood. We now know that lightning is very much related to plasma physics and high-voltage electrodynamics, but several aspects of lightning are yet to be studied in detail. One of them is how lightning propagates; before the flash we see as lightning is visible, several branches of ionizing fingers or 'streamers' are propagating through the medium to a grounded material. These streamers in the context of lightning are typically jerky and experience a lot of branching, which is easily visible, but it is not exactly known why these jerky movements and branching phenomena are happening. Likewise, at such branching moments, charge is distributed among the different branches. Streamers that experience branching often have significantly different properties (such as diameter, intensity and velocity) in the branches compared to the pre-branching streamer channel, the original streamer. How this happens exactly is also not known.

Acquiring more knowledge on such subjects regarding electric streamers is not only useful in studying lightning on earth. In many technological applications where high voltages are used, streamer discharges also occur. Since these streamers are actually ionized channels originating from a point with a very high electric potential to a point with a relatively low electric potential, a large current can go through this channel if the streamer reaches this point with a low electric potential. This is what happens in lightning; in technological applications such an event is equal to a short circuit which severely damages components such as transistors and capacitors, especially with high voltages. It is very important to avoid such streamer discharges and thus to know why and how such streamer discharges are created. Furthermore, since the streamer discharge heavily ionizes the surrounding gas, these streamer discharges are also used in applications such as purification, generation of molecules, charging of particles and flow control. For increasing efficiency in these processes it is also important to know how streamer discharges behave.

Electric streamers have been studied for decades already, but the usual strategy is to obtain information from single two-dimensional pictures of a streamer discharge. Since streamer discharges happen in three-dimensional space, a whole dimension of information is lost this way (such as the branching length or branching angle). Surprisingly little initiative has been undertaken to investigate the three-dimensional reconstruction of streamer discharges. Research has been done by Ichiki et al. on this subject, but their results where not very promising and their reconstruction algorithms where quite basic. Furthermore, they miss an important piece of information about the streamer discharge, which is their behaviour in time. The idea is to provide this information using stroboscopic techniques; combining several photos with a very short shutter time in one large picture. In this project, the goal is to create innovative algorithms for the three-dimensional reconstruction of streamer discharges using a structural approach based
on theory on mathematical image analysis and on mathematical physics. In chapters 2 and 3 the required theory on physical and mathematical subjects are given respectively. In chapter 4 the used experimental setup is described and experimental remarks regarding the goal of reconstructing streamer discharges are made. Afterwards, in chapter 5 the created algorithms and MatLab and Mathematica code are explained and compared with existing algorithms. The results on the three-dimensional reconstruction can be seen in chapter 6 as well as data analysis on these streamer discharges. From these results, a conclusion is drawn in chapter 7 and a discussion can be read in chapter 8. In appendix A the Mathematica code for the most advanced algorithm is shown and in appendix B the settings for obtaining the resulting reconstructions shown in chapter 6 can be found.
Chapter 2

Theory on Streamer Discharges

2.1 Definition and Initialization of Electrical Streamers

As this project is based on the phenomenon of electrical streamers, it is useful to first give an exact definition of what is meant by this. Electrical streamers are ionized fingers, mostly in gasses where a large potential difference is present. By creating a streamer channel and ionization front in a wave-like manner, the streamer makes way for a possible discharge. However, during the streamer propagation, the actual electron and ion drift is low or non-existent. In nature, streamer discharges are mainly found during lightning, as they are the precursors of the discharge which happens during lightning, known as a spark or lightning arc. The channels of ionized gas they leave behind thus pave the way for a possible discharge. However, electrical streamers occur on all occasions when large charge differences or large local field enhancements are present and they are widely used in industry. A widely used application is purification of gasses or production of gasses such as ozone, as the streamers have a significant effect on the gas they move through, such as ionization and excitation of nearby molecules.

Since the electrical streamer is a channel of ionized gas, it must be initiated via means of some kind of ionization of this material and will maintain itself via self-organization. For the intents of this research, only ionization via an electrical field will be taken into account, however background effects will influence this ionization process and will also be regarded. Ionization of a specific point in space via strong pulse lasers could also be possible, but this way of ionization will not be used during this project.

Electrons are charged particles and are present in practically every material, so acceleration due to an electric field is most of the times the initiator of an electrical streamer. Only from a certain threshold of kinetic energy a collection of electrons can ionize surrounding atoms. During these processes, more ionization than absorption occurs. The focus of this research will lie on electrical streamers in gas environments. For strong electric fields, the amount of electrons in the material tends to an 'avalanche effect' in which the amount of ionized molecules increases strongly due to the fact that the ionized particles also contribute to the ionization of other molecules. The distinction between an electron avalanche and an electrical streamer is commonly based on whether the ionization process has significant effect on the existing electrical field or not. From the point on where the ionization process has a significant effect on the electrical field, the structure is called an electrical streamer.

However, this definition is too vague for a systematic approach. A classical rule of thumb for the minimum amount of free electrons for the presence of an electric streamer is $10^8$ to $10^9$ free electrons, but this is very dependent on the density and pressure of the gas. This is known
as the Raether-Meek criterion and has been in use since the 1930’s [4]. While this serves as a reasonable approximation, the actual boundary is of course dependent on material parameters and system parameters. An adequate boundary is defined by Montijn and Ebert where the ionization process contributes to 3% of the strength of the initial electrical field, since then this contribution can no longer be neglected [5]. In practice, the distinction between free electron avalanches and electrical streamers in experiments is hard to observe.

## 2.2 Initiation Cloud

In experiments, the easiest method for initiating streamer discharges is using a electrode tip, as shown in figure 2.1; this point-plate set-up has the simplest geometry. A high voltage is applied to this tip for a short period of time (a pulse). Due to this voltage and the local field enhancement, all nearby atoms and molecules ionize, creating an avalanche almost instantaneously. While electric fields have no rise time, the time needed to apply the voltage (due to the inherent capacitive, resistive and inductive properties of the electrical circuit), the presence of jitter and stochastic processes lead to a small time period between setting the voltage and the onset of the avalanche. When the electrical field generated by the electrode tip is still present, this ionization process will occur roughly spherically (based on the geometry). The size of this initiation cloud is of course dependent on the applied voltage and the gas used, since some gasses can be more easily ionized than others.[1]

![Electrode tip used in all the experiments for this project. Such a configuration is an example of a point-plane geometry. Image property of EPG-group at TU/e.](image)

The maximum size of the initialization cloud can easily be obtained via basic electrostatics. Assuming the initialization cloud acts as a conductor, there will be no net electric charge in between the edge of the cloud and the electrode tip, making the initialization cloud a quasi-conductor. Thus, in this approximation the potential is the same throughout the spherical cloud. From theory, we know that outside a conducting sphere the potential $\phi$ is given by:

$$\phi(r) = \frac{q}{4\pi\varepsilon_0 r}.$$  \hspace{1cm} (2.1)
2.3 Behaviour of streamer discharges

with \( r \) is the distance to the charge, \( \phi(r) \) the potential as a function of the distance \( r \), \( q \) is the charge in the sphere and \( \epsilon_0 \) is the vacuum permittivity of \( 8.854187817 \cdot 10^{-12} \) Farad per meter. The charge \( q \) can also be written as follows:

\[
q = C \cdot V, \tag{2.2}
\]

with \( C \) the capacitance and \( V \) the applied voltage. Combining this with equation (2.1) gives:

\[
\phi(r) = \frac{C \cdot V}{4\pi\epsilon_0 r}. \tag{2.3}
\]

Now, for a conducting sphere the capacitance is given by:

\[
C = 4\pi\epsilon_0 R, \tag{2.4}
\]

with \( R \) the radius of the sphere in meter. Filling this in in equation (2.3) then gives:

\[
\phi(r) = \frac{R \cdot V}{r}. \tag{2.5}
\]

The electric field \( \vec{E} \) is given by \( \vec{E} = -\vec{\nabla}\phi \). Since \( \phi \) is only dependent on the distance \( r \) and not on the polar angle and the azimuthal angle this results in:

\[
\vec{E} = -\frac{d\phi(r)}{dr} \hat{\mathbf{r}} = \frac{V \cdot R}{r^2} \hat{\mathbf{r}}. \tag{2.6}
\]

Consequently, the size of \( \vec{E} \) at the edge of the initialization cloud is \( E = \frac{V}{r} \). A certain minimal electric field \( E_{\text{min}} \), differing per gas, should be present to keep the spherical cloud stable. If the voltage at the electrode tip is switched on long enough for the sphere to reach this critical field, the maximum radius of the initialization cloud is \( R_{\text{max}} = \frac{V}{E_{\text{min}}} \) from equation (2.6) and the cloud will fade at this radius. Otherwise, if the electric field is switched off, the initialization cloud will extinguish from that moment on. \[1\] \[6\] It should be noted that the initialization cloud is not a perfect conductor, nor a perfect sphere. Thus, this critical radius serves only as an approximation.

At the moment the critical electrical field \( E_{\text{min}} \) is reached by the initialization cloud, the electric field is too weak to support the spherical structure and electrical streamers begin to form. These first streamers are called primary streamers and they bridge the gap between the initialization cloud and a grounded surface. Several streamers can form this way and due to their charge they repel each other. There is no full comprehension yet on how these streamers begin to form and how the number of streamers can be predicted. \[7\]

2.3 Behaviour of streamer discharges

After a primary streamer from the initialization cloud reaches a grounded surface, the streamer stops extending. This streamer channel from the initialization spot to an electrode is then ‘finished’ and the charge from the initialization spot can be transported to an uncharged object. During this process of streamer-to-spark transition, significantly more light is emitted than during the phase where streamers are extending from the initialization cloud. It can happen that after such a final channel has been made, several other streamers begin to sprout from the initialization cloud, the so-called ‘secondary streamers’. Similar to these, new streamers can also arise after one or more streamers originated from the initialization cloud but have not reached a grounded surface yet. These streamers are called ‘late streamers’, since they start later than
2.3 Behaviour of streamer discharges

the primary streamers. During this project, only primary streamers will be studied, so only the streamers which are the first to extend from the initialization cloud. From now on, when the word 'streamers' is used, only primary streamers are meant. The pulse length, applied voltage and distance between electrodes are determined such that there is not enough current or time to initiate the transition to sparks. [1]

Additionally, streamer discharges can extinguish due to various causes. First of all, while the ionization process due to the streamer head causes an electric field, it is not strong enough to sustain the streamer alone. A background electric field is needed to make the streamer discharge sustain itself; without a background field there is no potential in the streamer head. In most cases of streamer occurrence, this electric field will be the electric field which initiated the streamer discharge to begin with. However, since the streamer channel functions as an electric conductor, the initial electric potential at the electrode tip is roughly the same as the electric field at the streamer head, provided no branching has occurred. This does not hold for the present electric field; due to field enhancements of branches this field can be completely different than the initial electric field. This branching can lead to the distribution of charge among the branched channels, complicating this argument. If the conductivity of the streamer channels is high enough, the electric potential is still roughly equal in all streamer channels.

Similarly to the presence of a minimum value of the electric field magnitude needed to sustain a spherical initialization cloud, another minimum value of the electric field magnitude is needed to sustain a streamer discharge. When the electric field at the streamer tip is below this critical value, the streamer will simply extinguish and transfer its excess charge to the surrounding gas molecules. Since the electric field is dependent on the applied voltage to the electrode tip, decreasing the voltage will also result in extinguishing streamers prematurely. Another way of extinguishing the streamer discharge is to simply switch off the electric field, for instance by regulating the electric field through a pulse.

2.3.1 Positive and negative streamer discharges

There are two main kinds of streamers: positive and negative streamers. They are fundamentally different, but behave in similar ways. In the case of negative streamer discharges, the streamer tip is made of negative ions or free electrons. Due to Coulomb interaction, this will result in repelling free electrons in the gas. Free electrons have to be supplied by the streamer channel itself, through which the free electrons can flow to the streamer tip due to the direction of the electric field (ions and other molecules are too heavy to be transported in this time scale). In the case of positive streamer discharges, the streamer tip is made of positive ions. Here free electrons will not drift in the direction of the streamer tip due to the streamer channel itself. However, since the streamer tip is made out of positive ions, it will attract free electrons from the gas. The result of this attraction of natural occurring free electrons is that the streamer tip glows more brightly and the streamer channels are smaller than is the case for negative streamer discharges. Additionally, more ionization occurs near the streamer heads for positive streamers. A figure indicating this difference between positive and negative streamers can be seen in figure 2.2. [1]

In experiments, both kinds of streamers can easily be produced. Changing from negative streamer discharges to positive streamer discharges is simply done by turning around the direction of the electric field, which is done by switching the polarity. However, since positive streamers tips have a higher photon emission and their channels are thinner and more focused, they are easier to observe optically. Imaging of streamer discharges via suitable cameras is the main method to obtain information about actual, non-simulated streamers and that is why it
is usually preferred to use positive streamer discharges for experiments. Therefore, during this project, positive streamers will be used.

![Visualization of the difference between positive (left) and negative (right) streamers. Plus- and minus signs indicate positive and negative molecules or atoms respectively.](image)

2.3.2 Streamer propagation

After leaving the initialization cloud, the streamer discharge moves in the direction of the electric field; it will follow the electric field lines. The streamers tend to follow the path of least resistance, but their path can be influenced by the local concentration of ionized gas molecules and other particles in between them. The reason why most streamer paths look so irregular is due to another effect, related to branching of streamer paths. This branching effect is discussed in section 2.4.

Since the streamer is positive for this project, the streamer head requires a constant source of free electrons in order to propagate further. This is done by means of ionization of nearby gas molecules. Commonly used gasses for streamer discharge experiments are nitrogen and oxygen. The ionization processes for these gasses are given by the following reactions [1]:

\[
N_2 + e \rightarrow N^+_2 + e + e
\]  

\[
O_2 + e \rightarrow O^+_2 + e + e
\]

2.4 Branching of streamer discharges

A very interesting phenomenon and one of the key points of this project is the branching of streamer discharges. It is a very common process in streamer experiments and streamers in natural processes. With the right parameters, a streamer discharge experiences no branching. However, these streamer channels are not very complex and lack interesting physical phenomena,
2.4 Branching of streamer discharges

such as electromagnetic interaction between streamer channels. For complex streamers with branching, less is known about their behaviour.

However, the process of streamer branching is not yet understood completely. One common reasoning is that the fundamental Laplacian instability of the streamer head results in branching independently of ionization effects or stochastic effects. This would mean that the streamer head is fundamentally unstable and that the streamer channel must branch under certain conditions, like when local charge distributions in the gas are present. Other theories state that (photo-)ionization effects are very important for branching and that streamers will branch when they reach a critical radius. This critical radius would then be dependent on the photo-ionization absorption cross section and the partial pressure. Another approach, which is already quite dated, focuses on stochastic effects in front of streamer head shells to explain branching effects. The problem with this idea is that it has never shown to fully explain the expected distribution of electrons in front of the streamer heads. Experiments have also been performed on the presence of background radiation during streamer discharges and the influence of this on branching. It was found that this background radiation can indeed be a cause for streamer branching.

Still, the phenomenon of branching is not yet fully understood and no explanation has been brought forward that completely explains all effects. Simulations of streamer discharges which model branching usually use stochastic or random behaviour of molecules to simulate branching or create fractal trees to model streamer branching. Even taking rounding effects of computed numbers into effect can lead to branching in simulations. However, these methods do not have a physical basis. Such simulations often give very realistic results, but they cannot implement the actual physical phenomena that influence the branching, since these phenomena are not yet understood.

A first start to understanding branching effects is to see what parameters influence branching of streamer discharges. Three important parameters which influence branching are the pressure of the gas, the type of gas and the voltage on the electrode pin, as well as the electric field homogeneity. All three lead to more branching when increased. However, it has also been observed that point-wire discharges lead to about ten times more branching than plane-plane discharges. Plane-plane discharges occur in an almost homogeneous and uniform electric field and experiments can also be performed on this. Here the electrode tip is replaced for an electrode plate with a small protruding tip in the middle for initialization. It has also been observed that the amount of branching depends on the type of gas used. Oxygen is often used in streamer discharges experiments and it has been seen that decreasing oxygen levels in gas mixtures lead to more branching. Also, the power supply plays a role in streamer branching. During experiments on the influence of the power supply, it was found that the impedance of the used power supply has a considerable effect on the amount of branching, but also on the thickness of the streamer branches. Additionally, high pulse repetition frequencies gives smoother streamer propagation and less branching. This also comes with higher background ionization. Lastly, dust and bubbles in the gas mixture have been observed to increase branching of streamers. However, this is a macroscopic effect, while all other effects can be described microscopically, so this observation is of little help when building a potential model of streamer branching. Most of these observations are not researched any further in this project.

Another observation is that streamers tend to repel each other. This is no surprise, since this is expected from Coulomb forces. However, this can also be seen in streamer branching. There will always be an angle in between two branched streamers which is characteristic for the used parameters. On the other hand, two streamers can also merge together to form one streamer; the streamer channels are then of opposing polarity.
2.4 Branching of streamer discharges

In summary, there are a number of parameters of interest when researching streamer branching:

- Number of branches during experiment
- Distance of branches to electrode tip
- Amount of background radiation or ionization
- Charge of streamer tip at the moment branching occurs
- Distribution of charge among branches of branched streamer
- Radius of streamer tip at the moment branching occurs
- Radius of streamer tips of branches of branched streamer
- Angle between branches of branched streamer
- Pulse frequency of electrode tip
- Impedance of power supply
- Macroscopic effects

Multiple branching

The question arises whether branching of streamer discharges always happens in pairs, thus whether streamer channels always split up in two streamer channels upon branching. Very little research has been done on this specific subject, but some initial work was done by Heijmans et al. [15]; this research was the first systematic investigation of this phenomenon. As noted in their article, the current understanding of streamer branching is incomplete and there is no clear reason why all branching should be into two branches, rather than several. However, little of these multiple branching moments have been observed; as Heijmans et al. write, some of this was documented by Briels, but these observations may be due to misinterpretations [16]. An example of such an observation is shown below in figure 2.3. However, Heijmans et al. managed to do a systematic study and find several clear instances of triple branching in streamer discharges. While these instances have to be found manually, they are unmistakably instances of triple branching.

![Figure 2.3: An example of a streamer discharge with possible multiple branching.][15](image)

The total amount of multiple branching moments was found to be quite low and only happening in about 1 out of 200 instances. However, there was quite some uncertainty present in this
result. A reasonable explanation would be that this triple branching is in reality two double branching moments shortly after each other, such that it cannot be seen that there indeed two separate branching moments. This hypothesis was also given by Heijmans et al. in their article. [15]. They found that this could happen one every thousand branches, so five times less often than found. However, due to uncertainties involved, this explanation could still cover the observations. This hypothesis would explain the relative difference in occurrence in double branching (two streamer channels) and triple branching (three streamer channels), as well as explain why branching in four streamer channels has not been observed; the probability is too low. On the other hand, the hypothesis is unable to explain what makes double branching so special. More research with more data points would help with this question, but as has already been mentioned, this would be very time-consuming as all identification would have to be done manually. Computer algorithms would be needed for such projects.

2.5 Streamer discharges in nature

The most common occurrence of natural streamers is during lightning. Here the initialization of the streamers happens in thunderclouds due to a huge charge difference with respect to the ground surface. It is not completely known what the causes the actual start of the streamers, but the main hypothesis is that macroscopic imperfections such as ice crystals in the cloud trigger a streamer. The typical jerky movement of lightning before it touches the ground is as a matter of fact caused by branching streamers. When such a streamer touches another surface or a lightning leader, a final ionized streamer channel is made and an enormous electrical discharge takes place. This is what we see as actual lightning and what emits the most light during a streamer discharge. The air molecules inside the ionized channel suddenly clash with a huge amount of free electrons, causing the air molecules to heat up tremendously. The air inside of the streamer expands due to this heat and this causes a shock wave, which we hear as thunder. A corollary of this effect is that streamer discharges which have not reached grounded objects yet are not hot at all and do not result in thunder.
Chapter 3

Mathematical Theory

In order to understand the more advanced algorithms in streamer reconstruction which will be discussed in subsection 5.8.4, some mathematical theory is needed on functional analysis, graph and group theory and some basic algorithms. These theoretical aspects will be presented in this chapter.

3.1 Metrics and norms

A metric is a very important concept in functional analysis that will be dealt with in this project. It is a function $d$ defining a distance between two points in a set $X$ which have a few basic properties:

$$d(x, y) \in \mathbb{R} : X \times X \rightarrow \mathbb{R}$$  \hspace{1cm} (3.1)

$$\forall x, y \in X, d(x, y) = 0 \iff x = y$$  \hspace{1cm} (3.2)

$$\forall x, y \in X, d(x, y) = d(y, x)$$  \hspace{1cm} (3.3)

$$\forall x, y, z \in X, d(x, y) \leq d(x, z) + d(z, y)$$  \hspace{1cm} (3.4)

Lastly, $d$ yields real-valued, finite and non-negative elements of $\mathbb{R}$. The third property is known as the symmetry property, the fourth property is known as the triangle inequality. \[17\]

A concrete example of a metric is the well-known Euclidean distance function. This distance function works in one-dimensional space, where we have $s^2 = x_{\text{diff}}^2$ with $s$ the distance and $x_{\text{diff}}$ the difference in x-coordinate, but also in two-dimensional space, where we have $s^2 = x_{\text{diff}}^2 + y_{\text{diff}}^2$, in three-dimensional space, where we have $s^2 = x_{\text{diff}}^2 + y_{\text{diff}}^2 + z_{\text{diff}}^2$ and theoretically also in higher dimensions. Another concrete example is the Manhattan distance, for which it holds that only horizontal and vertical lines can be drawn between points, no diagonal lines. What is meant by this is shown below in figure 3.1.
3.1 Metrics and norms

Figure 3.1: The difference between Euclidean distance and Manhattan distance in the XY-plane \cite{18}.

These metrics are both defined on $\mathbb{R}^m$, with $m$ taken to be 1, 2 or 3, which makes them easy to visualize. However, metrics can also be defined on more abstract sets, such as on the set of all bounded complex numbers, which has sequences as elements, or even on the set of all continuous functions on a given interval \cite{17}. A set with a metric attached to it is said to be a metric space. For such metric spaces, as discussed in this paragraph, it is not ‘trivial’ what the distance between elements should be, as was more the case in Euclidean space. However, if there exists a function which has all properties of a metric as shown above, this function can be used as a distance function. As it turns out, some distance functions are more useful than others in specific cases.

Important generators of metrics are metric tensors, objects which have two (tangent) vectors as input and a real number as output. Thus, the set associated with the metric is a vector space with vectors as elements. For this project, most of the times a set of key points with a given orientation is analyzed. Indeed, they can be seen as vectors and metric tensors prove to be very important for this analysis. Note however that the space of positions and orientations should not be treated as a flat Euclidean space. If we want to include the information of the orientation of a point into the distance function, more attention should be given to the choice of this distance function, as noted in figure 3.2. Euclidean distance is defined in an analytic way, but distances can also be defined via optimization problems or by the minimal length of a curve connecting two points. All possible curves have different shapes and to define lengths of tangent vectors and of curves, a so-called metric tensor is needed, which is explained in section 3.2. In the context of sets consisting of elements, the more general term ‘manifold’ rather than ‘space’ will be used from now on. Metrics need elements of the manifold as input variables and defines distances in a manifold, while metric tensors need vectors as input variables and defines lengths of tangent vectors. These objects consequently require different mathematical tools.

Another important concept is that of a norm. A norm is a real-valued function on a vector space $X$ to $\mathbb{R}$ whose value at an $x \in X$ is usually denoted by:

\[
||x||_p \quad \quad \quad (3.5)
\]

Norms obey the following properties ($x, y \in X, \alpha \in \mathbb{R}$):

\[
||x|| \geq 0 \quad \quad \quad (3.6)
\]

\[
||x|| = 0 \iff x = 0 \quad \quad \quad (3.7)
\]

\[
||\alpha x|| = |\alpha||x|| \quad \quad \quad (3.8)
\]

1The ‘$p$’ indicates that the norm is in the $L^p$ space
\[ ||x + y|| \leq ||x|| + ||y|| \] (3.9)

The first property denotes non-negativity, the second property states that only zero elements have zero norm and vice versa, the third property defines scalar multiplication and the fourth is again the triangle inequality.

Norms are closely related to metrics in the sense that every norm induces a metric:

\[ d(x, y) = ||x - y|| \] (3.10)

This result does not hold the other way around however; not every metric has an associated norm. [17]

## 3.2 Riemannian and sub-Riemannian space

The paper of Bekkers et al. on retinal blood vessel reconstruction using key points forms a major source of information for this research on streamer reconstruction [19]. In this paper, three manifolds are used, namely the Euclidean \( \mathbb{R}^2 \), the Riemannian \( SE(2) \) and the sub-Riemannian \( SE(2) \) as well as some approximations. In this section it is explained what these spaces encompass. A visualization is shown below in figure 3.2.

[Image of figure 3.2: Visualization of the difference in distance functions. The green arrow and the red arrow are both an equal Euclidean distance away from the black arrow, but using a different metric the green arrow is closer to the black arrow. [19]

Measuring distances with sub-Riemannian geometry is done using the so-called Lie group \( SE(2) \) as the base manifold. A Lie group is a differentiable manifold as well as a group, with the additional property that all group operations are smooth. Its elements can be represented by a set of continuous parameters [20]. \( SE(2) \) is constructed by means of a semi-direct product of the group of planar translations and rotations (a semi-direct product is a way of creating a new group out of two other groups involving normal groups) [19]. For this project, the most
3.2 Riemannian and sub-Riemannian space

An important application is how it acts on the space of positions and orientations $\mathbb{R}^2 \times S^1$; for a group element $g \in SE(2)$ and elements $x' \in \mathbb{R}^2, \theta' \in S^1$ we have:

$$g \cdot (x', \theta') = (R_\theta x' + x, \theta + \theta')$$  \hspace{1cm} (3.11)

Here $R_\theta$ denotes the operation of rotation over an angle $\theta$. As can be seen, translations and rotations can not simply be added and consequently these do not form a vector space. This is due to the interaction between the rotation part and the translation part.

Additionally, it is useful to define an associated vector space $\mathfrak{se}(2)$ called the Lie algebra which is equal to $\text{span}\{A_1, A_2, A_3\}$. These generators are equal to the differential frame at the origin:

$$A_1 = \partial_{\theta}|_{(0,0,0)}, A_2 = \partial_x|_{(0,0,0)}, A_3 = \partial_y|_{(0,0,0)}$$ \hspace{1cm} (3.12)

There are three of such generators, since there are also three independent coordinates in the manifold $SE(2)$. Additionally, it is useful to define corresponding left-invariant vector fields using the pushforward of left multiplication to obtain a linear approximation at an arbitrary point $(\theta, x, y)$ in $SE(2)$. These vector fields are defined by:

$$A_1|_{x,y,\theta} = \partial_{\theta}|_{(x,y,\theta)}$$ \hspace{1cm} (3.13)
$$A_2|_{x,y,\theta} = \cos(\theta)\partial_x|_{(x,y,\theta)} + \sin(\theta)\partial_y|_{(x,y,\theta)}$$ \hspace{1cm} (3.14)
$$A_3|_{x,y,\theta} = -\sin(\theta)\partial_x|_{(x,y,\theta)} + \cos(\theta)\partial_y|_{(x,y,\theta)}$$ \hspace{1cm} (3.15)

Now measuring distances on sub-Riemannian geometry is done by measuring lengths of 'shortest horizontal paths'. A horizontal path is a curve $\gamma$ in $SE(2)$ by lifting planar curves in $\mathbb{R}^2$ to $SE(2)$. Since curves in $\mathbb{R}^2$ have only two coordinates, the orientation, which is needed for elements in $SE(2)$, is the slope of the curve. The slopes of the lifted curve $\gamma$ are then always in the subspace of the full tangent space. A horizontal path has tangent vectors $\dot{\gamma}(t) \in \text{span}\{A_1|_{\gamma(t)}, A_2|_{\gamma(t)}\}$, with $A_i$ being the left-invariant vector fields introduced earlier\[19\]. The curve does not have a $u^3$ component. Note that not every curve in $SE(2)$ is a horizontal curve, but every horizontal curve is an element of $SE(2)$.

For this project only horizontal curves are used as tools for streamer reconstruction and thus it is required to use the sub-Riemannian geometry, rather than the general Riemannian geometry. The standard way of measuring horizontal curves $\gamma(t)$ with $\dot{\gamma}(t) = u^1(t)A_1|_{\gamma(t)} + u^2(t)A_2|_{\gamma(t)}$ - $u^1$ and $u^2$ being control parameters - is with the sub-Riemannian metric tensor

$$G_{\gamma(t)}^{\xi,C}(\dot{\gamma}(t), \dot{\gamma}(t)) := C(\gamma(t))^2|u^1(t)|^2 + \xi|u^2(t)|^2$$ \hspace{1cm} (3.16)

Note that the superscript $2$ at $u^2$ denotes the second coordinate, not a square. Squaring from now on will be indicated with additional brackets, as is convention. A cost function $C : SE(2) \rightarrow \mathbb{R}^+$ has been introduced here. It penalizes curves which go through specific regions in $SE(2)$. With this cost function, it can be controlled in which directions reconstruction should occur. The equation \[3.16\] thus has $\gamma$ as one of the input variables. This is quite useful when reconstructing streamer channels, as will be shown in subsection 5.8.4. Additionally, there is the parameter $\xi$ which occurs in the metric tensor. This parameter is present to be able to balance the motion in the angular and the spatial directions by adding another penalty. Lastly, one can see that indeed both control parameters $u^1$ and $u^2$ are used as input variables in equation \[3.16\] being related to $\dot{\gamma}(t)$.

From this metric tensor in equation \[3.16\] it is possible to define a distance function $d_0$, the sub-Riemannian distance, between two points $g_1, g_2 \in SE(2)$:

$$d_0(g_1, g_2) := \inf_{\gamma} \left\{ \int_0^1 \sqrt{G_{\gamma(t)}^{\xi,C}(\dot{\gamma}(t), \dot{\gamma}(t))} \, dt \right\}$$ \hspace{1cm} (3.17)
3.2 Riemannian and sub-Riemannian space

The two points \( g_1 \) and \( g_2 \) are the end points of the trajectory \( \gamma(t) \), so \( \gamma(0) = g_1 \) and \( \gamma(1) = g_2 \). Inf denotes an infimum taken over all possible trajectories \( \gamma \). Due to technical reasons, the trajectories \( \gamma \) should be Lipschitz continuous curves, meaning that the speed of change in the curve is limited. The absolute value of the slope of the line segment connecting two arbitrary points on the trajectory is always smaller than a previously fixed number. In practice, this means that only physically possible curves are taken into account. This is not a strong restriction for the project, as streamer channels are almost always smooth or can at least be reasonably approximated by smooth lines. The infimum rather than the minimum is taken to ensure the distance function is also valid for the case that the trajectories converge to an optimal trajectory that is not physically possible (e.g. it is not Lipschitz-continuous). In the easiest cases, the infimum will be equal to the minimum. [19]

3.2.1 Nilpotent approximation for SE(2)

As it turns out, this distance function \( d_0(g_1, g_2) \) can be reasonably replaced by a distance function based on an approximation. Since equations (3.13) are difficult to calculate and these calculations require much computational work, an approximation is needed. While computational tools can be used to approximate these equations manually, it is more insightful and useful to make an analytical approximation for these equations, which can then be evaluated real-time. This approximation is based on the attribute nilpotency of Lie algebras. In order to explain this, first the exponential map \( \text{Exp} : \mathfrak{se}(2) \to \text{SE}(2) \) needs to be introduced, a more general notion of the exponential function. It defines a mapping from a vector \( X \in \mathfrak{se}(2) \) in the tangent space at the origin \((0,0,0) = g\) to an element in the group \( \text{SE}(2) \) along an integral curve in the left invariant vector field of the pushforward of the vector \( X \). Similarly, the logarithmic map \( \text{Log} : \text{SE}(2) \to \mathfrak{se}(2) \) defines the inverse mapping from a group element in \( \text{SE}(2) \) to the tangent vector at the origin \((0,0,0) = g\). [19]

Next the Lie bracket needs to be defined. For vector fields, the definition is as follows:

\[
[X, Y] := \lim_{t \to 0} \frac{\gamma(t) - e^{t^2}}{t^2} \tag{3.18}
\]

Here, we have

\[
\gamma(t) = \text{Exp}(-tY)\text{Exp}(-tX)\text{Exp}(tY)\text{Exp}(tX) \tag{3.19}
\]

This Lie bracket can be interpreted as describing the infinitesimal displacement when following a path moving back and forth in \( X \) and \( Y \) directions. Note that the Lie bracket produces a new vector and that moving along the vectors field defined by vectors in the Lie algebra does not necessarily commute:

\[
[A_1, A_2] = -[A_2, A_1] = A_3 \tag{3.20}
\]

\[
[A_1, A_3] = -[A_3, A_1] = -A_2 \tag{3.21}
\]

As a result, the Lie algebra \( \mathfrak{se} \) is not truly abelian\(^2\). Another important result is that the Lie algebra \( \mathfrak{se} \) is not nilpotent. A Lie algebra is nilpotent when its lower central series (the descending series of subgroups) converges. The idea behind the approximation of the distance function \( d_0(g_1, g_2) \) is to assume the Lie algebra \( \mathfrak{se} \) is nilpotent by neglecting higher order terms in nested Lie brackets.

\(^2\)This means that the elements of the Lie Algebra in general do not commute.
To illustrate this, consider the exponential map from the Lie algebra \( \mathfrak{se}(2) \) to the group \( \text{SE}(2) \):

\[
(c^1, c^2, c^3) \rightarrow (x, y, \theta) = \text{Exp}(c^1 A_1 + c^2 A_2 + c^3 A_3)
\] (3.22)

Note again that these superscripts do not indicate powers of \( c \), but rather a specific coordinate \( c^i \). \( A_i \) are the basis vectors of \( \mathfrak{se}(2) \) as shown previously and \((c^1, c^2, c^3)\) are so-called generalized coordinates of the first kind:

\[
c^1 = \theta
\] (3.23)

\[
c^2 = \begin{cases} 
\frac{1}{2} \theta (y + xcot(\frac{\theta}{2})), & \text{if } \theta \neq 0 \\
x, & \text{if } \theta = 0 
\end{cases}
\] (3.24)

\[
c^3 = \begin{cases} 
\frac{1}{2} \theta (-x + ycot(\frac{\theta}{2})), & \text{if } \theta \neq 0 \\
y, & \text{if } \theta = 0 
\end{cases}
\] (3.25)

Now, we will need the Baker-Campbell-Hausdorff formula, a formula that gives the solution of \( e^X e^Y = e^Z \) for non-commutative \( X, Y \) in \( \mathfrak{se} \). For two general left-invariant vector fields \( X = \sum_{i=1}^{3} x^i A_i, \ Y = \sum_{i=1}^{3} y^i A_i \) the Baker-Campbell-Hausdorff formula gives:

\[
\log(\text{Exp}(X)\text{Exp}(Y)) = X + Y + \frac{1}{2}[X,Y] + \frac{1}{12}([X,[X,Y]] + [Y,[Y,X]]) + O([\cdot,\cdot,[\cdot,\cdot]])
\] (3.26)

Here \( [\cdot,\cdot] \) denotes the Lie bracket as encountered earlier. We also encounter the term \( O([\cdot,\cdot,[\cdot,\cdot]]) \) signifying higher order nested Lie brackets. Here, the approximation introduced by Bekkers et al. comes into play [19]. The higher order terms, including the one with the prefactor \( \frac{1}{12} \), are neglected as if \( X \) and \( Y \) would commute. If \( \mathfrak{se} \) was nilpotent, this approximation would give the exact result, thus the term nilpotent approximation is appropriate. However, the term \( \frac{1}{2}[X,Y] \) is not neglected, as it is known that the commutation relation for this is \( [A_3 = [A_1, A_2]] \).

By again omitting the higher order Lie brackets, we obtain the relation \([A_1, A_3] = 0 \) with \( A_1, A_3 \) the same as in the first commutation relation in equation (3.22). Using the canonical coordinates of the first kind, this yields:

\[
(x^1, x^2, x^3) \cdot (y^1, y^2, y^3) = (x^1 + y^1, x^2 + y^2, x^3 + y^3 + \frac{1}{2}(x^1 y^2 - x^2 y^1))
\] (3.27)

This is an example of a group product, expressed in terms of coordinates of the first kind. However, we have already seen a group product in equation (3.11), defined on the same sets. It turns out this new group product is a local approximation of the true group product in the region where the higher order terms can be neglected. To distinguish the two, we call the group which is the output of the local approximation \( \text{SE}(2) \), while the exact group product gives \( \text{SE}(2) \) as a result [19].

For these elements of the group \( \text{SE}(2) \), a norm \( ||c||_\zeta \) is defined by Bekkers et al.:

\[
||c||_\zeta = ((||c^1||^2 + ||c^2||^2 + \zeta ||c^3||^2)^\frac{1}{2}
\] (3.28)

Here \( c^i \) are the usual canonical coordinates of the first kind and \( \zeta \) is a parameter for penalizing non-horizontal movement in the plane. Via the Ball-Box theorem and the equivalence of homogeneous norms, they make a connection between the distance function in equation (3.17) and the following, slightly different norm:

\[
||c||_{\zeta, \xi} = ((||c^1||^2 + \xi^2 ||c^2||^2 + \zeta \xi^2 ||c^3||^2)^\frac{1}{2}
\] (3.29)

\[\text{Three coordinates; two for the position and one for the orientation}\]
Here $\xi$ is again present for balancing the motion in the angular and the spatial directions.

Lastly, for no data-adaptivity ($C=1$) they justify the use of the following approximation:

$$d_0(g, h) \approx |\log(g^{-1} h)|_{\xi, \zeta}$$

This is the final approximation for the distance function in sub-Riemannian geometry used most prominently in the paper. \[19\]

### 3.3 Perceptual Grouping and Fast Marching Algorithms

In the paper by Bekkers et al., also two algorithms regarding retinal vessel tracking are given to aid the process of tracking \[19\]. These two are the **perceptual grouping algorithm** to sort all key points according to the true pathways of the vessels in the image and the **fast marching algorithm** to compute distances between key points efficiently. Because of their significant role in this project, they are highlighted in this section.

#### 3.3.1 Perceptual Grouping Algorithm

The aim of the perceptual grouping algorithm is to construct a graph representing the true network of key points according to a distance function and the axiom that key points are connected to at most two other key points. Worth noting is that branches are not represented in this approach; one avoids this problem by assuming all parts of a blood vessel network or streamer discharge are separate segments. If the algorithm does its job correctly, it should not be very difficult to reconstruct which segments are actually connected.

The input of the algorithm is a set of key points $S$, the set of all distances between key points $\{d(g_i, g_j)\}$, $g_i, g_j \in S$ and a maximum distance $s_{\text{max}}$ between key points to alleviate computation costs. The variables used are $\tilde{D}_S$, the set of all possible edges, and $\delta_i$, the node degree of a key point $g_i$. The node degree is the number of nodes a key point has. Finally, the output is $D_S$, the final reconstructed set of edges which is believed to be the ground truth \[19\].

The basics of the algorithm are quite simple. During the initialization, all distances $d(g_i, g_j)$, $g_i, g_j \in S$ and its geodesics (optimal trajectories) are computed via a distance function (in most cases the sub-Riemannian distance or its approximation) and the set $\tilde{D}_S$ is initialized with all edges which have a geodesic with spatial arc length smaller than $s_{\text{max}}$. This is done with the idea that the lengths of geodesics between neighbouring key points are approximately known already. Now, while there are still edges in the set $\tilde{D}_S$, each step one edge is taken out of the set. This edge is not random, but it is in fact the edge with a geodesic with the shortest spatial arc length according to the chosen distance function. Following this, the node degree of both involved key points is evaluated. If both are smaller than two and the key points are not already both present in the same subgraph, the edge is added to the set of final edges $D_S$ and one is added to the node degree of both key points. If this is not the case, nothing is done with the edge. This process continues until all edges have been removed from the set $\tilde{D}_S$.

Of course, due to its relative simplicity this is not the first time such an algorithm is used. The main improvements which have been done by Bekkers et al. are that the choice of distance

\[4\]In the case of Bekkers et al., these key points are constructed separately. In the case of this project, these key points are 'streamer dots', defined later in chapter \[5\].
There are some caveats to be noted in this algorithm. Firstly, it assumes that when two points have a very small distance between each other, e.g. when the arc length of the geodesic is small, that this is indeed the optimal geodesic. This poses problems in Euclidean space with crossing lines. Fortunately, sub-Riemannian distances solve this using orientations of key points, such that crossing key points would have a very curved geodesic between them, thus making it not optimal. Secondly, the algorithm assumes that key points with a node degree of zero or one are eligible for connection with another key point. Certainly, this must hold for key points with a node degree of zero; the need to be connected to at least one other key point. However, end points in pictures always have a node degree of one. It needs to be avoided that these end points are connected to each other just because they have a node degree less than two.

3.3.2 Fast Marching Algorithm

Euclidean distances are easy to compute since it is trivial what the geodesic must be; a straight line. However, what the geodesic should be for sub-Riemannian distances is less trivial. One can try some trajectories and using the distance functions proposed earlier determine which gives the better trajectory for the key points, but there are infinitely many of such trajectories possible. Just choosing random trajectories until one with a distance close enough to an optimal distance has been found is far from efficient, though it is effective. For this problem, the fast marching algorithm is used which is an efficient numeric solver of the obtained differential equation. There are two parts of this fast marching algorithm, the first for efficiently solving the distance function and the second for backtracking to find the geodesics themselves. The idea is that the gradient of the distance function corresponds to the direction of the geodesic.[21]

Generally, let \( g_0 \) be a point of interest in a domain \( \mathbb{M} \), let \( \mathcal{G}|_g : T_g(\mathbb{M}) \times T_g(\mathbb{M}) \to \mathbb{R}^+ \) be a metric tensor with associated tangent space \( T_g(\mathbb{M}) \) at \( g \in \mathbb{M} \). In the context of this project, \( g_0 \) is a key point, \( \mathbb{M} \) is \( SE(2) \) or \( (SE(2))_0 \) and the metric tensor is given by equation (3.16). The distance function \( U(g) \) can consequently be defined by:

\[
U(g) := d(g_0, g) = \inf_{\gamma \in \mathcal{T}(g_0,g)} \int_0^1 \sqrt{\mathcal{G}|_{\gamma(t)}(\dot{\gamma}(t), \dot{\gamma}(t))} dt
\]  

(3.31)

Again, we need the infimum to be taken over \( \mathcal{T}(g_0, g) \) with \( \gamma \) being a Lipschitz continuous curve, \( \gamma(0) = g_0, \gamma(1) = g, \dot{\gamma}(t) \in T_{\gamma(t)}(\mathbb{M}) \)[19]. \( \mathcal{T} \) here denotes the set of all possible pathways in \( SE(2) \) connecting \( g_0 \) with \( g \).

As it turns out, the distance function \( U \) is the unique viscosity solution of the so-called eikonal equation:

\[
\begin{align*}
& \sqrt{\mathcal{G}(\nabla G U(g), \nabla G U(g))} = 1 \\
& U(g_0) = 0
\end{align*}
\]  

(3.32)

\[\iff\]

\[
\begin{align*}
& ||\nabla G U(g)||_{\mathcal{G}} = 1 \\
& U(g_0) = 0
\end{align*}
\]  

(3.33)

Here, \( \nabla G := \mathcal{G}^{-1} dU \) is the intrinsic gradient with inverse metric \( \mathcal{G}^{-1} \). \( dU \) is the differential of \( U \) and \( || \cdot ||_{\mathcal{G}} \) is the norm with respect to the chosen metric tensor.

The idea behind the fast marching algorithm is to solve this equation numerically in an efficient manner. This is based upon the gradient descent on \( U \); one can find the direction of greatest
descent in U for various parts of the trajectory and by means of overshooting one can find a trajectory which falls in the limits of the desired accuracy. Mathematically, this comes down to solving the following equation:

\[
\begin{align*}
\dot{\gamma}(t) \propto -G^{-1}dU(\gamma(t)) \\
\gamma(0) = g_0
\end{align*}
\] (3.34)

In the case of concrete choices for G and U, this ordinary differential equation is easy to solve numerically.

3.4 Graph Theory

An alternative way of looking at the problem of streamer reconstruction is by perceiving some key points as vertices which have to be connected via edges, the geodesics between the key points. These key points should in some way be obtained from the image. By assigning a value to each possible vertex, such a key point, in the graph, one can optimize the connection of these streamer dots using various optimality algorithms. The basic algorithm based on this idea is called the minimum-spanning-tree algorithm. In this section some specific cases are highlighted and further explained. The implementation of this theory in the streamer reconstruction is described in subsection 5.8.3.

### 3.4.1 Kruskal’s algorithm

One of the algorithms able to tackle the problem of minimum-spanning-trees is Kruskal’s algorithm, a very basic algorithm which is ‘greedy’: it only looks at the optimal decision at one step in the algorithm, rather than looking at long-term decisions. However, Kruskal’s algorithm for spanning trees in graphs is very cost-efficient and yields actual minimum spanning trees. This makes it a useful candidate for application in a program in which this algorithm needs to be performed numerous times.

There are only a few rules which the algorithm must abide. Firstly, the algorithm needs a set of vertices (streamer dots) and edges (geodesics) with a cost (length or curvature). It then takes the edge with the least cost and puts it in the minimum spanning tree. In case there is more than one edge with the least cost, one edge is chosen randomly. The edge is then removed from the original set and this process of choosing edges continues. Edges which are not allowed in the minimum spanning tree are those which make loops or cycles, since having such loops would not yield additional connections between vertices in the graph. The algorithm stops when the minimum spanning tree is a structure in which all vertices are directly or indirectly connected with each other. It is thus important that the graph is connected to begin with, i.e. that all vertices are connected with each other, directly or indirectly. There are then no separate, disconnected clusters. In the case of streamer discharge reconstruction, this is certainly the case. Due to the inner workings of the algorithm, it is guaranteed that one fully connected network is obtained as a result. It turns out that the result of Kruskal’s algorithm is indeed a minimum-spanning-tree. A proof of this can be found in literature [22]. It turns out that the time complexity of the algorithm is \(O(n\log(n))\), with n the number of vertices. For a sorting algorithm, this is relatively fast, though for a greedy algorithm it is relatively slow. An example of how Kruskal’s algorithm works is shown below in figure 3.2.
3.5 Interpolation

An interesting alternative to obtaining smooth geodesics is by means of interpolation of data points to get a smooth curve through these data points. This technique is commonly used in data fitting and finding experimental relations, but has a surprising application in the context of streamer reconstruction. The goal is namely to make the transition from vertices connected by linear line segments to smooth curves through these vertices. Here the step from graph theory to optimal connections between vectors must be made. While this approach does not give optimal curves in terms of curvature, it serves as a good approximation for such geodesics.

3.5.1 Spline Interpolation

An easy interpolation method using as much relevant information as possible is spline interpolation. Here, the fitted curve is forced to go through all data points and have the properties of continuity and differentiability. Splines have the additional property that the second derivative is also continuous and thus that the curve is very smooth. The power behind splines compared to regular interpolation is that they are found and derived on small intervals on which a simple, low-order polynomial is fitted. The most commonly used spline is the so-called cubic spline,
which consists of polynomials of order three on each interval. In order to have no degrees of freedom, constraints need to be given. The standard constraints are that the values of the splines at the boundaries or vertices need to comply with the data, that the first and second derivatives need to be continuous on the vertices and that the second derivative vanished at the beginning and end points. Since the assumption is that all fitted curves are polynomials of degree three, with this information all splines can be determined.

3.5.2 Hermite Interpolation

The other interpolation method which will be used is the Hermite interpolation, for which the derivatives at certain points need to be known. In the case of streamer discharges and streamer dots, the orientations of the streamer dots fulfill the role of derivative or slope, since they give a direction to the data points. In some situations such as fitting experimental data, having to know the derivatives is a drawback since they cannot be exactly known. Some algorithms thus use numerical approximations for these derivatives. In the context of this project, these derivatives are known quite precisely, depending on the quality of the picture used. In the case of Hermite interpolation, no constraints are given regarding the second derivative of the fitting function. Rather, the information about the function values and the derivatives at each point are used as constraints. A difference in the result of these two interpolation methods can be seen below in figure 3.4.

![Figure 3.4: Difference between the results of cubic splines and Hermite interpolation. The cubic splines give more wave-like curves, while Hermite interpolation gives quite rigid curves](image-url)
Chapter 4

Experimental Setup

In this section, all relevant information about the experimental setup will be given with descriptions and schematic representations of various parts of the setup and the use of these. Furthermore, it will be discussed which experiments use which settings in the setup as well as some problems with the setup.

There are three main parts of the experimental setup, this being the gas part, the electricity part and the photography part. The gas part is about the supply and the regulation of the amount of gas in the vessel used for experimenting. The electricity part is about the supplying of the voltage needed to initiate an electrical streamer and various settings and control systems regulating this electricity supply. The imaging part is about obtaining as much information as possible from the streamer discharge as well as a description of the computer software used.

4.1 Gas-related parts of the setup

In the simplest representation, the vessel has an input tube and an output tube for the gas. Most of the regulating systems are connected to the output tube for the gas. Here a Pfeiffer MVP160-3 bypass and a Pfeiffer TMH261P turbo pump is present for regulating the gas output. These should not be used simultaneously, as the gas will always try to go through the turbo motor as this gives the least resistance. However, this turbo can only be used with pressures lower than 10 mbar, otherwise the bypass way should be used. The pressure in the tubes is measured on two locations, one in the front of the output tube and one at the beginning of the bypass. The first measurement performed by a Balzers TPG-252-A Dual Gauge Vacuum Controller is more accurate for low pressures, the latter performed by a Pfeiffer RVC 300 control unit is more accurate for higher pressures. As a rule of thumb, this distinction is in the interval of 1 mbar to 10 mbar; inside this region the pressure meters give the same value, outside of this interval they give (slightly) different values for the pressure. With this meters, it can be seen whether the turbo can be switched on or not. This turbo is particularly useful when ending an experiment and pumping gas out of the vessel. Note that the turbo pump needs to cool down before use and that pumping the gas off using the bypass needs to be done slowly. On the pressure measure devices the desired or nominal pressure can also be set. If this nominal pressure is different than the actual pressure, valves and pumps are activated or deactivated to alter the pressure to the desired level.

However, the gas related systems also have to work apart from the beginning and the ending of the experiment, where gas has to be put either in or out of the vessel. The vessel is for example not a perfectly closed system as it must interact with its surroundings. The effect is that the
pressure inside the vessel will always approach the atmospheric pressure. Since the pressure inside of the vessel is always lower than the atmospheric pressure for this research, the result is that air will flow into the vessel through unwanted ways. Since the composition of the air is unknown and experiments are not always done on air, this is undesirable and the gas inside of the vessel needs to be 'flushed', i.e. refreshed. There is another reason why this is favourable and that is that the presence of electrical streamers has a significant influence on the gas itself (for example by ionization of particles). This ionization can lead to some memory effect in the streamer discharges, but this ionization is undesirable even if this does not happen, since the experimental conditions would not be the same throughout the experiments. Thus, the gas is flushed to eliminate these effects and have fresh, unionized gas present in the vessel. The supply of the gas can be controlled in a separate control box where the desired pressure from the gas tanks can be set and altered by a Brooks Instrument 0254 mass flow transmitter. Via a series of manual levers the supply can also be quickly changed. These safety mechanisms are present to make sure that not to much force is exerted on the delicate control systems, but rather on the manual levers. Ultimately, the system of gas tubes is connected by a gas canister of 50 liters with pure nitrogen, UN1066 of Linde Gas Benelux.

4.2 Electricity-related parts of the setup

Using various electric circuits with transistors and resistors it is possible to have a large electric potential between the electrode tip in the gas vessel and the grounded plate on the bottom of the ground vessel. For the power supply, a Wallis High Voltage Regulated DC Power Supply with a maximum voltage of 60 kV and a maximum current of 8 mA is used. A high voltage button as well as the power button needs to be pressed in order to let this power supply work. With a rotating know, it is possible to set the voltage with an accuracy of 5 volt.

For monitoring the relevant parameters during the experiments, a WaveJet Touch 354 500 MHz Oscilloscope is used that receives signals from the camera setup, the high voltage supply and the electric circuits inside the gas vessel. Below in figure 4.1 a front view of this oscilloscope as well as a screen with the relevant parameters can be seen.

![Figure 4.1: The front view of the WaveJet Touch Oscilloscope with on the screen relevant parameters.](image)
4.3 Imaging-related parts of the setup

As can be seen in figure 4.1, three coloured lines are visible on the touch screen display (A in figure). The yellow line is the power supplied by the high voltage power supply. This power supply is triggered by a periodic pulse. The green line is the voltage at the electrode tip. Currently, it can be seen that a streamer discharge is happening in the gas vessel. The blue line denotes that the camera is on and working. Here, the shutter time is relatively high and no stroboscopic image is being made. In part B the input/output panel is visible and part C shows the control part.[25].

For the communication between the high voltage power supply and the imaging setup a Keysight 33600A Series Waveform signal generator 50 MHz coupled with a Rigol DG1022 2 channel waveform generator is used. On the waveform signal generator two channels are present; one for the settings of the electric pulse which triggers a streamer discharge and one for the camera settings. Relevant settings which can be set are the frequency of the pulse, the pulse width, the cycle count and the delay time for both the pulse for the streamer discharge as well as the pulse for the camera-related parts. The Rigol DG1022 waveform generator is present to make sure the pulses and triggers are properly synchronized, such that the camera gates are indeed open when the streamer discharge is happening. These pulses need to be synchronized that well that even the cable length is a relevant factor, as signals need time to be transmitted through the cables. That is why a digital delay generator is indeed needed. Note that the time between pulses needs to be at least 8 nanoseconds for the signal generator to work, while the camera can process pulses much faster.[26][27]

4.3 Imaging-related parts of the setup

For the imaging of the streamer discharge, an ICCD camera is used, namely the LaVision PicoStar HR12. Since stereoscopic techniques are used, two images are stored each time a picture is taken. The stored image has a round shape and the configuration of the mirrors and lenses needs to be adapted such that the two images, left and right, of the streamer discharge can both be projected on the round image without too much overlap in order to increase efficiency. Since the shutter time or exposure time of the ICCD camera is usually very low, in this project in the range of tens of nanoseconds, the intensity needs to be amplified in order to have useful images. This is the defining property of an ICCD camera; first the captured photons which have been focused on the entrance window are captured by a phosphorescent plate and converted into electrons and then these electrons are multiplied such that the electron intensity becomes amplified. This is done by an electric field inside the camera with a strength of approximately 1 kV. These electrons are then again converted to photons which reach the CCD-chip of the camera.

1With proper reconstruction techniques, overlap of the images is possible since with reconstruction algorithms the images can still be split. However, this goes beyond the scope of this project.
4.3 Imaging-related parts of the setup

Figure 4.2: The cycle of imaging the streamer discharges. [28]

Above in figure 4.2 a schematic representation of what happens during the imaging process is shown. During a period of time determined by the pulse length, the streamer discharge is present and emits light. At certain time steps determined by the frequency of imaging a trigger signal is send out by the I/I-trigger. When this signal is received by the I/I-gate, the intensifier gate in the camera is opened and light is received on the CCD-chip. The gate is closed again after a fixed time determined by the shutter time per photo. After all photos have been taken of the streamer discharge, the I/I-gate no longer reacts on the I/I-trigger and the CCD takes its time to read out the information. After this, the read out signal is transmitted to the computer and saved so that it can be processed. The results is one stroboscopic picture with the information of all individual photos together. [28]

Coupled with the PicoStar camera is a Kentech High Rate Imager. This device gives the pulses for taking pictures with a very high frequency to the digital delay generator, the camera and the oscilloscope.

Below a schematic figure of the used setup is shown. On the left of the image the vessel with gas can be seen as well as the electrode tip used for streamer initialization. The purple lines are possible light rays emitted from the streamer discharge. They are reflected by mirrors set on an optical railing and afterwards are focused on the ICCD camera by a prism. Between the gas vessel and the rest of the experimental setup a metal grating is present in order to shield the high voltages inside the vessel. The walls, floor and ceiling of the vessel are also grounded for safety reasons.
Two lenses were available for this project; one with a focal length of 50 mm (Nikon UV-50 F1.8) and one with a focal length of 105 mm (Nikon UV-105 F4.5). These lenses are regular photography lenses, but with a special UV-coating specified for the imaging of streamer discharges, which have an emission spectrum mostly lying in the UV-region. With the lens with a focal length of 50 mm the lens could be set closer to the vessel to obtain useful images than using the lens with a focal length of 105 mm. However, during all measurements the lens with a focal length of 105 mm was used. With this focal length, the small-angle approximation is more justifiable than with the 50 mm lens. Furthermore, a larger part of the streamer discharge can be in focus with the 105 mm lens compared with the 50 mm lens. Lastly, the undesirable effect of chromatic aberration in the quartz of the prism and inside the mirrors is way more prevalent using the 50 mm lens compared to using the 105 mm lens.

For the checking of images, the storing of measurements and the camera calibration, the LaVision Davis8 software was used. With this software, connected to the PicoStar camera, the images obtained by the camera could be seen directly. Furthermore, relevant ICCD settings such as intensity amplification and camera mode could be set here. During experiments, this software needs to be used to make sure that the setup is working properly and that the images show structures that are sharp and in focus. When making measurements, the signals originating from the camera software need to be shut off for just a moment before turning on again for the actual data acquisition. Sometimes this gives complications, which are explained later on in section 4.4.3.

4.4 Calibration

All in all, there are a number of relevant parameters that can be changed during the project:

- Focal length of lens used: 105 mm
- Type of gas used: Nitrogen (N₂)
- Voltage between electrode tip and grounded plate: 3 - 9 kV
4.4 Calibration

- Gas pressure: 40 - 150 mbar
- Length of pulse: 600 - 1200 ns
- Frequency of pulses: 1 Hz - 1kHz
- Frequency of image capturing: 1 MHz
- Shutter time per photo: 40 - 80 ns

In section 5 more is written about the choice of certain parameters for getting specific types of streamer discharges and reconstructing them. During the experiments, it was important to keep most parameters constant and only change one or two during a set of measurements. One complication occurred at low voltages close to the so-called 'breakdown voltage', a critical voltage which indicates the boundary between having streamer discharges and having no streamer discharges. However, for the first streamer discharge in a set of measurements, the voltage needs to be somewhat higher. After a first streamer discharge has been created, the gas in the vessel has enough ionization processes to aid the creation of further streamer discharges in the measurement series and the voltage can be set close to the breakdown voltage. Due to the flushing of gas in the vessel, after a certain amount of time where the voltage is below the breakdown voltage the ionization processes in the gas are too rare to aid the initialization of new streamer discharges. Then the voltage needs to be set relatively high again to begin a new set of measurements.

When making a measurement series, the signals originating from the ICCD camera stop for just a moment and this has the same effect as described above. Thus, with voltages close to the breakdown voltage there is a possibility that the streamer discharges stop occurring when the measurements start. In order to still obtain useful results, a UV-lamp has sometimes been used to aid the ionization processes in the gas and thus the initialization of streamer discharges. This use of a UV-lamp does not have a significant effect on the imaging of the streamer discharges. Note that this is a regular UV-lamp with no specific connection to the spectrum of the streamer discharge.

When using the set-up as described in this section for the intentions of this research, a calibration measurement series also needs to be done of a remarkable object in various positions and orientations the streamer discharge is also expected to be. For the experiments, a tripod with a printed checkerboard pattern was used for this. The checkerboard squares had a length of $9.0 \pm 0.05$ mm and $7.2 \pm 0.05$ mm. When the contour between the black and the white parts is high enough, the reconstruction algorithms work well (as will be described in section 5). However, it is not possible to put this tripod inside the gas vessel for every measurement. Thus, a translation needs to be performed where the camera and the mirrors are translated a distance equal to the distance between the end of the optical railing and the electrode tip, such that the distance between the camera and the checkerboard pattern during the calibration is roughly equal to the distance between the camera and the electrode tip during the experiments. Since the tripod with the checkerboard pattern needs to be translated in this region to represent the possible positions of streamer heads inside the gas vessel, the exact distance is not very important. With small vertical, horizontal and angular displacements a calibration session with useful results can be made.
Chapter 5

Creating a 3D Image

A first step in researching the effect of streamer branching is to get as much information as possible on these branches. As stated previously in chapter 2, the most practical way to get information on streamer discharges is to make optical measurements of the streamer discharges, i.e. photos. These photos have to be taken with very small shutter times and can give great insight in the process of streamer discharges when calibrated properly. However, there is one big disadvantage of the usual approach of taking these pictures which is especially important for streamer branching; photos give a two-dimensional view, thus only one plane of three-dimensional space is seen. Much information is lost this way, such as the solid angles in branches. These solid angles are here known as branching angles and are of vital importance in understanding the effects of streamer branching. In this section a number of algorithms are proposed that can aid the transition between several two-dimensional pictures to a three-dimensional model of experimental results. For this, several steps can be made. A simple algorithm can be made for streamer discharges with low complexity, explained in subsection 5.8.1. For more intricate streamer discharges, some improvements are done in order to make sure the streamer channels are recognized in a correct manner, described in subsection 5.8.2. An algorithm which deals with optimizing the connections in the discharge as a whole is described in subsection 5.8.3. Lastly for the reconstruction, a mathematical analysis is made in order to tackle difficult problem of complex crossing streamer channels in subsection 5.8.4. Afterwards, several ways of connecting the two two-dimensional images are described in section 5.5.

However, first an explanation of several parts of this problem needs to be given, namely on the subject of which information can be extracted, identification of streamers, analysis of so called streamer dots, and error analysis.

5.1 Obtainable information

The available information for reconstruction is limited. The data on which analysis can be performed is one stroboscopic picture consisting of several superimposed photos of the streamer discharge. Since the streamer head emits the most light during the experiment, this is what is most visible on the photos. However, since the photos are superimposed, in the final stroboscopic picture a series of these streamer heads are visible. These will be called ‘streamer dots’ from now on. It is not useful to talk about a ‘streamer head’ in the context of stroboscopic pictures; this term will only be used to refer to individual streamer channels and their ends. All streamer dots in a branch are the same streamer head, but on different time steps. Similarly, the term ‘streamer dot’ is not useful in the context of streamer channels and their ends, this term will only
be used in the context of stroboscopic pictures. A typical picture obtained during measurements in shown below in figure 5.1.

![Typical obtained image of a streamer discharge with low complexity during experiments, using stroboscopic and stereoscopic techniques.](image)

**Figure 5.1:** Typical obtained image of a streamer discharge with low complexity during experiments, using stroboscopic and stereoscopic techniques.

An intuitive manner of obtaining more information is to save the individual photos the stroboscopic picture contains separately. A movie is then obtained of the streamer discharge, but unfortunately cameras with more than ten million frames per second would be needed for this and such cameras were not used during this project. However, this is not possible with the used configuration. In the same way, changing the focus of the lenses during the picturing of one specific streamer discharge is not possible. Changing focus is done manually and can’t be performed on these time scales. However, for streamer discharges which branch a lot, it is well possible that some branches are out of focus due to these branches being more in the back or the front than others. In this case, this information can be used to determine which streamer dots are supposed to be connected, i.e. to reconstruct the streamer head trajectories.

For obtaining information, it is interesting to search for specific structures among the streamers. One useful example is searching for extinguished streamers. Since these are easily visible and distinguishable from the other streamer channels, they can serve as a reference point between pictures from different angles and as a reference point for reconstructing streamer pathways. General useful information which can be obtained about these streamer dots are position, orientation, intensity, eccentricity and area.

### 5.2 Noise reduction

Before meaningful analysis can be performed on the streamer discharges, noise has to be removed from the obtained images. In these grey-scale images, the amount of white on a pixel represents the intensity of the radiation on the camera via the process discussed in section 4. As was stated previously, background light has very little influence on the obtained image. However, there actually is some noise present in the image, as will always be the case. An option to reduce this noise is to convert the grey-scale picture to a black-and-white picture, with only the pixels with a certain threshold level of intensity becoming white. This way, practically all noise is removed, as the intensity of the streamer dots is way higher than the intensity of the
noise. However, also information about intensity information such as intensity gradients in the streamer dot is lost and this information can be used for further analysis on these streamer dots and the reconstruction of the trajectories of the streamer heads. Due to this problem, a more useful approach would be to make all pixels with an intensity lower than the threshold level black and keep the shades of grey of pixels with a higher intensity. This way, only the bright part of the image is obtained which can be analyzed. The downside is that the afterglow is also removed that way, but as mentioned previously this afterglow also has very little use in the further analysis. A noise reduction algorithm that is automatically present in the ICCD camera related software is a dark image subtraction method; for every measurement an average dark picture is subtracted from the measured image to eliminate background noise as much as possible.

5.3 Identification of Streamer Dots

As described in section 5.1, the brightest spots of streamer discharge pictures are due to the high field region of the streamer head. This is where the actual ionization processes take place. However, due to the non-zero shutter time and more importantly the radiative lifetime of the gas, which is approximately 50 nanoseconds in nitrogen, these streamer dots are often not visible as shells, but rather as circles or even ellipses. The actual final picture to be analyzed is a total view of the streamer discharge and is a combination of all photos made of that streamer discharge in time. An example of an image with identified streamer dots is visible below in figure 5.2.

After noise reduction, identification of streamer dots is done by looking for spots in the picture with a remarkable high intensity and then extracting the shape out of this high-intensity spot. In case of a very large high-intensity spot in the picture, such as at the electrode tip for complex streamers, intensity maxima are sought for and the high-intensity spot is divided into as many streamer dots as there are intensity maxima. After this first identification process, only the dots with their area and intensity above a certain threshold are stored as actual streamer dots, to avoid detection of possible noise-like streamer dots. This easy method gives reasonable results, as described further on in subsection 5.8.1 and certainly is favourable above manual identification of streamer dots. A neural network for this identification might certainly be useful, but there

Figure 5.2: Images of streamer discharges processed with MatLab. On the left stereographic images of the original streamer discharge, on the right the same images with the streamer dots identified with green circles. In the middle an intensity bar can be seen with 1000 being the maximum intensity the camera can process with the settings used.
are several reasons why this has not been used. Firstly, this neural network would need a lot of training with training data that needs to be manually made. Secondly, if a neural network makes an error in identification, it is almost always impossible to tell why this error has been made and how it can be prevented, as it is fundamentally a black box model. Lastly, creating a neural network for this task goes beyond the scope of this project.

### 5.4 Analysis of Streamer Dots

Using the locations of the streamer dots that have been found and stored, relevant information can be extracted. Among the most relevant information for the scope of this project is the shape of the streamer dot, which is related to the orientation of the streamer dot, the minimum, mean and average intensity of the dot, the area of the dot and the eccentricity.

Since the goal is to make logical connections between the streamer dots, this information should be used to aid this process. The area and intensity of a streamer dot can be used to filter streamer dots from noise, while the eccentricity and the orientation of the streamer dot can be used to make statements about the likelihood of certain connections between streamer dots. Note that since the shutter time for each photo is short but finite, every streamer dot will either look like an ellipse or like a shape with three or more ends, like in the event of branching.

Furthermore it is important to notice that variables like intensity and area of the streamer dot might depend slightly on the streamer dot being in focus or not. The distance between the electrode tip and the camera is very large compared to the dimensions of the streamer discharge. Streamer dots in the back of the streamer discharge hence do not look smaller than streamer dots in the front. On the other hand, both streamer dots in the front and the back are more out of focus than the streamer dots in the middle of the streamer discharge and thus can have a significantly lower intensity than the dots which are in focus. This can help with making a distinction between streamer dots.

Research on blood vessels in the retina gives a surprising application in this research, as in this research advanced mathematical methods are used to identify blood vessels which are similar to streamer paths in quite a few ways [29]. Similar methods can be used for streamer paths to see which streamer dots belong to each other, which also aids the ultimate transition to a 3D picture of the streamers. This research will be particularly relevant for intricate streamer networks. This subject will be discussed in more detail in subsection 5.8.4.

### 5.5 Connection between two two-dimensional images

Lastly, a very important and crucial step needs to be made, that is, the connection between the two different pictures. It has to be identified which parts of the streamer discharge in one picture are actually the same as parts of the streamer discharge in the other picture, but just seen from a different angle. If the reconstruction in the individual two-dimensional pictures has already been done, it is possible to both use information about the streamer dots as well as the information about streamer channels for this connection. The crucial step is to find as much unique information as possible for the connection and then gradually connect segments and streamer dots with each other. As it turns out, useful properties for this connection are:

- Number of streamer dots in the channel. This is a useful property when looking for extinguished streamers or when the channel is cut into segments at every branching points.
5.5 Connection between two two-dimensional images

- Distance between the starting dot and the end dot. Of course, there will be differences in the pictures regarding this property. However, in the small-angle-approximation, this property can still help with the connection.

- Relative position of streamer channel. Similarly to the point above, if the streamer channels lie in a certain sequence in the left image, in the small-angle-approximation they will have a similar sequence in the right image.

- Length of streamer channel. When an appropriate curve through the streamer dots has been found, evaluating the length of this curve could give useful results for the connection between images.

- Curvature of streamer channel. Similarly to the point above, this obtained curve can also be evaluated in terms of maximum and minimum derivative and second derivative, thus in terms of curvature. Again, the small-angle-approximation is needed to make this property useful.

- Velocity inside streamer channel. When the reconstruction of the streamer discharge in the two-dimensional image has been finished, the dots in this image already have a connection with each other. Thus, by evaluating the distances between connected streamer dots it is possible to say something about the velocity of the streamer channel.

- Vertical position of streamer dots. Practically all streamer channels will go from the electrode tip to the grounded plate, as fast as possible. Only when the streamer is so complex that the channels start to repel each other, this is not the case. In any way, while for the horizontal position of the streamer dots the small-angle-approximation has to be used again, both pictures are taken on the same height, thus the dots in the left image all should have the same height in the right image. Not all streamer dots will have a vertical position that is unique enough to make the connection between images, but there will be streamer dots where that is the case.

In theory, the following two properties should also be useful for making the connection between images. In practice, they give too unreliable results;

- Diameter of the streamer channels. It is possible to obtain the diameter of the streamer channels by using the minor axis length of the streamer dots and experimental results have proven that the radius of the streamer channel can indeed change upon branching, but experimental procedures performed for this project have not yielded useful results.

- Intensity of the streamer dots. Again, experimental results have proven that the intensity of a streamer channel can change upon branching. However, possibly due to focal issues, measurements have not given useful results regarding streamer reconstruction.

After this, the three-dimensional model should be made from these two two-dimensional connected images. This last step is far from trivial, but fortunately MatLab has an extensive package for exactly this step, namely the stereo vision package. This package focuses on depth estimation from two-dimensional images and also on three-dimensional image reconstruction from several two-dimensional pictures. This coincides neatly with the aim of this project and with slight alterations this MatLab package and its functions are used extensively during this research project. Of course, some kind of calibration needs to be done in order for this three-dimensional image reconstruction to work properly. For this, the checkerboard pattern is used, as described in section 4.4.

32
5.6 Experimental Remarks

Some experimental issues regarding the goal of 3D streamer reconstruction need to be made. There are some limitations that have as result that not in all cases a full 3D reconstruction is possible, no matter the algorithm. Also, a few choices regarding the setup had to be made for optimal data acquisition.

First of all, while two lenses were available for measurements as described in chapter 4, only the lens with a focal length of 105 mm has been used for actual measurements. The lens with a focal length of 50 mm gives a wider image and the lens can be put closer to the streamer discharge, but it is also more difficult to put the streamer discharge in focus. Parts in the back or in the front of the streamer discharge become blurry very easily and most of the times too blurry to obtain information. Furthermore, the effect of chromatic aberration is way stronger for the 50 mm lens than for the 105 mm lens. Thus, the 105 mm lens was used during experiments. Using only one lens also resolved the issue of choosing the optimum measurement angle; the distance between the camera and the streamer discharge was chosen such that the measurement area on the CCD was used most efficiently, depending on the size of the streamer discharge.

Apart from the voltage and the gas pressure, only two parameters have really been changed during the experiments and those are the frequency of taking photos taken per stroboscopic picture and the shutter time per photo. The main influence of these parameters is the shape of the streamer dots, which is more stretched for large shutter times, and the amount of streamer dots in the picture (if the pulse width is kept constant). This concept is visualized in figure 4.2 in chapter 4. For more complex streamers, it is useful to have less streamer dots to have a clearer overview of what was going on. For streamer discharges with many crossing streamer channels, large shutter times were needed to have more confidence in the orientation of streamer channels.

Lastly, there will always be instances where exact reconstruction is impossible. Due to orientation of the configuration, streamer channels might be strongly overlapping in one image but not in the other image. If this overlapping is so distinct that significantly less information about the streamer discharge can be obtained in one image compared to the other image, there is a fundamental problem, as this information is needed for accurate reconstruction and can’t be obtained in another way. It seems likely that for some measurements performed this way, no accurate 3D reconstruction can be executed. An example of such a situation is shown below in figure 5.3.
5.7 Error Analysis

Of vital importance in all these methods and algorithms is to have knowledge of the errors in the results about things like streamer dot position or charges in streamers. These can be given by a relative error following from the mathematical derivation of the answer. However, another problem may arise further on, and that’s when objects in the image are falsely identified as a streamer dot (type I error) or when streamer dots are not identified as such (type II error). For different configurations the probabilities that these errors occur should be known. Furthermore, another important error can occur in the further analysis. When identifying streamer paths which are overlapping, the wrong path can be found (a path that is not present in the actual experiment). Additionally, when comparing the two 2D images, the goal is to find which streamer dot in one image corresponds to which streamer dot in the other image. Errors can easily be made here as well. It is vital to know what the probability of an error occurring here is.

Apart from knowing probabilities of errors and thus knowing what the probability is that everything goes right, it would also be very useful to minimize the error probability. When a mathematical error analysis has been done, it is also known what parameters influence the error and thus how to minimize it. However, when error probabilities are done by means of indicator variables, it would be the most useful to just compare different methods that achieve the same goal on their error probabilities to see which is favourable. During this project, the method most commonly used was to calibrate the MatLab programs for each image to be analyzed and manually remove all type-1 and type-2 errors individually for each image analysis.

5.8 Two-dimensional reconstruction and basic image analysis

Using a basic image analysis script as the foundation of the algorithm, two big steps have to be made in order to obtain a final three-dimensional reconstruction of the streamer discharge. These steps are the two-dimensional reconstruction in both two-dimensional images and the connection between images. In this section, the basic program and the two-dimensional reconstruction is explained with various ways of sequencing.
5.8 Two-dimensional reconstruction and basic image analysis

5.8.1 Starting algorithm for creating three-dimensional images

The first approach to the problem is the most basic one and the aim of this approach is to make the connection between two 2D images with a specific viewing angle and no branching in the streamer discharge. Thus, the streamer dots of both pictures need to be identified using the techniques described in section 5.3 and then it needs to be found which streamer dot in one 2D image corresponds to which streamer dot in the other image. Sequencing the dots in the image themselves isn’t needed, as they are already in the right sequence (top to bottom).

To make a first working algorithm, the most basic streamers were used. Parameters for such streamers are low gas pressure and low voltage, as described in chapter 4. The resulting streamer experiences no branching and is approximately straight. In this section, the ideas and principles of the code are explained.

First, we have the file 'LoadImages.m' which is used to load a collection of images in an efficient manner in order to let the main program work more quickly. This is a program which was previously provided by the EPG staff of the TU/e, no additions have been made on it. In order to import the image, a data folder needs to be given. Inside this folder, one can filter the files by means of ‘tags’. The program checks whether the tag is in the file name for each file in the folder and if this is the case, the file is loaded. Files considered for this are .ome.tif files, image file types in which multiple images are stored in one file and in which one can scroll through these as if it where multiple files. While the files become large this way, it is a useful way of information presenting.

The other important file used is '3DPlotter.m'. This file too has been provided by the EPG staff of the TU/e, but numerous additions have been made to this file, indicated in the code. Using this file, it is possible to make a 3D image of the streamer using photos from two different angles. In the beginning of the file, there is an option to choose which image number or numbers in the .ome.tif file should be analyzed. The first plot which is made is just of the image itself combined with a colour map based on intensity to get more contrast compared to the standard black-and-white images. The second plot made is the first plot combined with a Gaussian blur to make the objects appearing in the figure smoother and blurred. The aim is to have one intensity maximum in each streamer dot, but technical difficulties sometimes result in pixelated images which have to be smoothed with this Gaussian blur in order to let the algorithm work.

The third step made in the file is to have noise reduction and cancellation in the image. For this, a band-pass filter is used that suppresses noise from pixels and small variations in the image. This function needs a typical size of noise and a typical size of objects. Since objects (streamer dots and afterglow) are usually much larger than noise (pixel variations) this method works well in this application. With this information, local and global minima and maxima can be found using the standard MatLab function FastPeakFind via means of Gaussian 2D fits. After this, a useful step is to also store a binarized version of the image. This binarized version also has a lot of afterglow parts with full intensity, but combined with the information on the local maxima it is possible to divide the image into parts with full intensity in which every part only has one local maximum in the original image. These division stripes are then saved and ’pasted’ upon the original image to make sure every streamer dot is indeed separate from the rest.

Next the very useful standard MatLab function 'regionprops.m' is used, a function that finds a lot of relevant information about the objects found earlier, namely the streamer dots. These streamer dots are supposed to look like ellipses and with this function one can abstract the area, the center of mass, the eccentricity, the major axis length, the minor axis length, the orientation, the perimeter, the maximum intensity, the minimum intensity, the mean intensity, the pixel values and the center of intensity. Of these, mostly the area, the position, the eccentricity and the orientation are useful. With the area, one can filter the mismatches out, since the
streamer dot area should be fairly constant. With the orientation and position of the dots, the reconstruction of the streamer pathways becomes much easier. Lastly, with the eccentricity of the dot one can give a confidence level to the orientation value.

For the streamer tracking part, each found object from the function regionprops is tested for a few criteria, namely area, minor axis length, major axis length and maximum intensity. The values for this testing come from calibration; most of the times the values of an area of 60 pixels, a minor and major axis length of 4 pixels and a maximum intensity of 30% of the intensity scale is used. This was, all 'objects' that are actually not streamer dots are removed from consideration. After this operation, all found objects should be streamer dots and all streamer dots should have an associated found object. To check this, an object called 'Streamer' is made which stores the position and orientation of the found objects and these positions are plotted on the image of the streamer which has had a Gaussian blur and noise cancellation. The testing whether the algorithm did the work correctly is usually done manually.

So far, the MatLab functions described just did general pre-analytic work and all algorithms described later on in the report have these functions as a basis for the program. The simplest algorithm, the one of this section, assumes that the streamer dots are already sorted when the program 'reads' these dots from downwards starting from the top. Thus, the program only works if there is no branching at all and if the way the program reads the streamer dots already gives the correct streamer propagation. In listing 5.1 below the code is shown for this small section of the algorithm

Listing 5.1: The method of tracking in the most basic algorithm where no branching is allowed.

```matlab
j=1;
k=1;
clear Streamer
for i=idx
    plot(image.BWinfo(i).Centroid(1),image.BWinfo(i).Centroid(2),'og')
    if image.BWinfo(i).Centroid(1) > 50
        if image.BWinfo(i).Centroid(2) > 310
            Streamer.L(j,:) = [image.BWinfo(i).Centroid(1),image.BWinfo(i).Centroid(2)];
            j=j+1;
        else
            Streamer.R(k,:) = [image.BWinfo(i).Centroid(1),image.BWinfo(i).Centroid(2)];
            k=k+1;
        end
    end
end
```

The if-statements are only present to separate the left image from the right image in stereoscopy. It is assumed they have no overlapping and the values are obtained from calibration. Using the counting variables j and k, it is possible to see how long the streamer channels are. Note that j must equal k as the same streamer discharge is viewed. Note also that in principle j and k must be known beforehand, as it is just the amount of photos the camera made in the stroboscopic image. In image.BWinfo.Centroid, the information about the positions of the found objects has been stored and idx is a matrix in which all corrected objects are stored, i.e. here the positions of all streamer dots are present.

Now the Stereo Camera Calibrator MatLab applet needs to be used, a very useful built-in applet that can be used to create a 3D image out of two 2D images only if the connection between the images already has been made. Since no branching is allowed in this first basic algorithm,
this connection is trivial; the algorithm just sorts both images the same way and assumes the connection is made already. In order for the applet to work, a calibration session needs to be performed first. In the calibration session, images of a checkerboard pattern need to be made where the checkerboard pattern is placed approximately in the position where the streamer discharges are expected. If pictures are taken in which the spatial position and orientation of the checkerboard pattern are varied slightly each time, the applet can extract the relevant information from these images alone if the lengths of the checkerboard pattern squares are given. The angle between the main axis and the cameras and the distance between the cameras thus doesn’t have to be provided, but can be extracted from these images using the applet. Of course, measuring these parameters manually is always useful to check the accuracy of the applet. On the other hand, the parameter that mainly depends on this information is the distance between the cameras and the streamer discharge, which is not very important for the analysis of the streamer.

The last part of the program does some analysis on the positions of the streamer dots in the reconstructed 3D image via the function triangulate, which undistorts matching points in stereo images using the extracted stereo parameters from the Stereo Camera Calibrator applet. This applet needs to be run before the script 3DPlotter.m to avoid manual calibration problems.

The main disadvantage of this approach is immediately obvious; the algorithm only works for very basic streamer discharges which show no complexity at all. When branching is present, additional functions are needed for correct sequencing of the streamer dots. Solving such problems is possible with other, more advanced algorithms and these will be discussed in section 5.8.2.

5.8.2 Predictive algorithm for streamer discharges with simple branching

The first algorithm served as a basic start for the problem of creating a 3D image. The main problem which occurred was that sometimes multiple connections were possible and that some kind of preference needed to be built in. This section will explain solutions to this problem.

As was discussed, it is possible to obtain an orientation of streamer dots using the function 'regionprops.m'. With the velocity of the streamer head, related to the time between photos for the stroboscopic photo, for each streamer dot a prediction can be made for the position of the streamer head at the next time step, thus the next streamer dot in the channel. When there are several possibilities for the next dot, additional checks need to be used. This is the case when there are several streamer branches or when the streamer channel has a surprising direction.

In the MatLab program, these algorithms are implemented in the functions 'ReconstructionFromFront.m', 'ReconstructionFromBack.m' and 'ReconstructionFromBackVelocity.m', each being an improvement on the previous. The basic principles of these programs is the following: based on the position and the orientation of one streamer dot on a time step, predict the position of the streamer head on the next time step. The assumption is that these dots move linear in time, while they are actually following the field lines. With small time steps, this assumption is justified. In all cases, the general program '3DPlotter.m' is used like described in section 5.8.1 but the piece of code shown in listing 5.1 is changed for either one of the reconstruction functions mentioned above, which are described below.

The function 'ReconstructionFromFront.m' first identifies the starting point, then makes a prediction for the next streamer dot by creating a line segment with as slope the orientation of the dot and as length some calibration velocity. Then the streamer dot which is the closest to this prediction is identified as the next dot in line. Problems occur at branching and these problems are tackled by identifying for each predicted position whether there is more than one candidate for the next streamer dot in line. If either one of the orientations match reasonably well with
the slope of the line segment and the positions of the predicted dot and the actual dots are reasonably close (in some error margin), a new branch is identified. The algorithm then first finishes the original branch and then continues with the new branch. This process continues until all dots have been listed in the SequencedDots matrix.

The main disadvantage of this approach is that identifying branching moments is tricky and possibly inaccurate and that much calibration is needed in order to make this work. Furthermore, it is unnecessarily difficult to write working code in case of exceptions. Actually, starting from end points and then reconstructing the channels to the starting point has some advantages. Since merging is not considered, we don’t see 'backwards branching' and only 'backwards merging', which is actually branching when watched from the front. Merging is more easily detectable than branching and no additional exceptions have to be considered. This has been implemented in 'ReconstructionFromBack.m'. End points are found by making a prediction for the streamer head at the next time step for each streamer dot, and if no streamer dot is reasonably close to the prediction or when the orientation of the nearest streamer dots don’t match the expectation, the streamer dot is assigned as an end dot. An additional advantage is that an improvement regarding velocity detection can be implemented more easily. If for the calibration length for line segments between dots just the length of the previous length segment is used, more accurate results can be obtained than when only one fixed velocity is used. This is also possible when considering front end reconstruction, but is more difficult due to information storage of branching. This improvement regarding velocity has been implemented in 'ReconstructionFromBackVelocity.m'. A pseudo-code for the back-end predictive algorithm with automatic velocity calibration is shown below in listing 5.2. The crucial part of this algorithm is that no check has to be performed for branching, since the reconstruction starts at the back.

Listing 5.2: Pseudocode for the Euclidean predictive algorithm, back-end velocity based

```
input: Streamer dot coordinates, Streamer dot orientations, Velocity, Neighbour distance checker, Dot distance checker
variables: Number of Neighbours, Predicted Position, Sequenced Dots
output: Spanning Tree, Number of Branches

Beginning Point = First point in streamer object
Number of Branches = 0

for all streamer dots
    Number of Neighbours = 0
    Create predicted neighbour in front based on orientation and velocity
    Create predicted neighbour in back based on orientation and velocity
    if the predicted neighbours are closer than a calibrated value to a streamer dot
        Number of Neighbours = Number of Neighbours + 1
    end
    if Number of Neighbours of dot i = 1
        Number of Branches = Number of Branches + 1
        Starting Dot of Branch = i
        Sequenced Dots (Number of Branches,1) = Dot i
    end
end

while t< Number of Dots
    for each branch
        Start with second dot in branch
        Obtain velocity from line segment of dot i and dot i-1 in branch
        Predict next dot in time based on orientation of dot i and obtained velocity
        Find dot nearest to predicted dot by evaluating all dots in a circle of radius
```
The advantage of these methods is that much more complex streamers can be analyzed compared to the most basic algorithm and furthermore that crossing streamer channels can already be identified, since they have different orientation. In most cases, two crossing streamer channels in a two-dimensional image don’t actually cross in three dimensions, so indeed such channels have to be separated. The main disadvantage of this method is the huge dependence on the orientations of the streamer dots without knowing much about the uncertainty in this orientation value. The eccentricity of the streamer dot tells something about the uncertainty of the orientation, but sometimes the streamer dot does not have a clear orientation and making the algorithm depend on this orientation can give bad results. This is for example the case for streamer dots which are right at branching points; these dots often have three or more appendices. Additionally, these functions do not necessarily result in the most optimal network, since they do not have a strict ending point and make connections based on the individual streamer dots, not on the network as a whole. One improvement on this is described in subsection 5.8.3.

5.8.3 Graph-based algorithm for streamer discharges with moderate branching

The algorithm described in this section is based on the optimality principle; rather than making connections which are optimal for a point, connections are made which are optimal for the network as a whole. This methods yields an optimal connected network of streamer dots which should have as result that the network looks more like the actual streamer pathways than in previous sections. This algorithm is based on theory described in section 3.4. Also, rather than having straight line segments between streamer dots, smooth differentiable curves are made, based on cubic splines and Hermite interpolation, described in section 3.5.

The algorithm in 3DPlotter.m uses the same basic functions as described in subsection 5.8.1, but a separate function for the sequencing of the dots is made in ‘MinimumSpanningTree.m’. First a cost matrix is made with information about the cost of pathways between every combination of streamer dots. For this cost, length or curvature can be used and for the pathways, Euclidean pathways or Sub-Riemannian geodesics can be used. Streamer dots can not have a pathway leading to themselves; an infinite cost is set between equal dots.

Next, a recursive while-loop is used to obtain a minimum spanning tree. Here, the function ‘kruskal.m’ is used, written by Georgios Papachristoudis [30]. It implements the Kruskal algorithm described in subsection 3.4.1 efficiently. However, the basic algorithm based on Euclidean distance and straight lines does not use any information on orientation of streamer dots. Thus, the recursive while loop is used to check whether the slopes of the optimal lines are reasonably close to the orientations of the streamer dots. If this is not the case, the distance between these streamer dots is set to infinity and Kruskal’s algorithm is performed again until all optimal lines have reasonable slopes. A pseudo-code of this algorithm is shown below in listing 5.3. The crux of the algorithm is that the spanning tree must be calculated again each time a distance is set to infinity.
5.8.2 Two-dimensional reconstruction and basic image analysis

Listing 5.3: Pseudocode for the minimum spanning tree algorithm

```plaintext
input: Errorangle, List of streamer coordinates, Cost Matrix
variables: Angle, Orientation, Total Checker, Checker, Merging Checker
output: Spanning Tree

Streamer x coordinate = Streamer[1]
Streamer y coordinate = Streamer[2]
Streamer orientation = Streamer[3]
checker = 1
while Checker == 1
  Total Checker = 0
  Merging Checker = 0
  Spanning Tree = Kruskal(Cost Matrix)
  Angle of dot i = Slope of line segment connecting dot i with dot i-1 in spanning tree
  if Angle of dot i - Orientation of dot i < Errorangle OR Angle of dot i - Orientation of dot i-1 < Errorangle
  Total Checker = Total Checker + 1
  else
  Cost of path between dot i and i-1 = Infinity
  end
  if Total Checker = Amount of streamer dots
  Check whether a dot is present twice in the second row of the Spanning Tree
  if this the case
  Merging Checker = Merging Checker + 1
  Make the highest of these connections Infinity
  end
  end
  if Merging Checker = 0
  Checker = 0
  end
end
```

With this minimum spanning tree, it is straightforward to obtain the end points of the streamer discharges if the program is calibrated correctly; these points do not have a successor in the minimum spanning tree. Now, by recursively going through the spanning tree the dots are sequenced and by means of (spline) interpolation, smooth and differentiable curves are obtained of each section between streamer dots. These curves are then merged which yields a smooth, differentiable curve from each end point to the starting point.

5.8.4 Curvature-based algorithm for streamer discharges with intricate branching

The theory needed for this algorithm is largely based on the research of Bekkers et al. on retinal blood vessel tracking [29]. This retinal blood vessel tracking and the reconstruction of streamer discharges have a surprising similarity, both relying on a set of points with a certain orientation. While in the case of streamer discharges this set of points is the set of streamer dots, the algorithm of Bekkers makes use of a generation of so called 'key-points' on important parts of the retinal vessels [19]. These key points are given an orientation score which indicates a direction, and these oriented key points are then connected in an optimal way using the distance metric suggested by Bekkers et al. [29]. There is no need to generate key points in the case of 3D streamer reconstruction, as there already are key points in the form of streamer dots. Since
these streamer dots also have an orientation, they fully fulfill the role of these key points and this algorithm can just be used in the same way. Note that when using this approach, the orientation of the key points must be known with a fair amount of certainty. To improve the accuracy of obtaining the orientation, the shutter time of the camera will be such that the streamer dots are observed as elongated ellipses, rather than perfect ellipses. This is related to the eccentricity of the streamer dots.

The main file for this is '3DPlotter.m', but this file also makes use of several other functions. The file is built upon the basis as explained in subsection 5.8.1. The starting point is again assumed to be the uppermost streamer dot in the image, which is always the case with correct calibration during the experiment. For the reconstruction of the streamer discharges in two dimensions, the algorithm of Bekkers is used. For this, a Mathematica notebook is used which solely uses the pictures of the streamer discharge and the positions and orientations of the 'key points', thus the streamer dots. The output of the notebook is every connection made between the streamer dots, corresponding to the output of for example Kruskal’s algorithm for minimum spanning trees.

Not every function and algorithm described in Bekkers’ work is needed for this algorithm of connecting streamer dots; only the perceptual grouping algorithm is vital for the purpose of this project. Generation of key points or connecting dots in three-dimensional space is not needed, and functions for these purpose can be ignored. However, several important definitions and equations still need to be used. Among these are the group product and the group inverse, the Lie algebra, Exponential map and Logarithmic map and the Folland-Kaplan-Koranyi gauge on the coordinates of the first kind. Additionally, the exact formulas and metrics don’t need to be used, since we know that the approximation works well enough for the intention of this project. Thus also the approximation for the sub-Riemannian distance needs to be given.

After these definitions, the picture data and streamer dot data is loaded, the approximation for the sub-Riemannian distances is used for all pairs of streamer dots and then the perceptual grouping algorithm is applied to the geodesics and the key points as described in subsection 3.3.1. The code for this algorithm is shown in appendix A. Obtained is a table with all connections made between the streamer dots. Note that since every streamer dot can only be connected with at most two other dots, no branching is considered. This makes the algorithm more robust in this case, but it also has as a result that only segments are obtained and not one spanning tree. A pseudo-code for this algorithm, written by Bekkers et al., is shown below in figure 5.4.
### 5.9 Connection between two-dimensional images

With the two-dimensional reconstruction finished, making the connection between the images becomes much easier. Now, not every dot in one image can correspond to every dot in the other image, since there must be some kind of grouping present. In this section, various ways of making this connection using MatLab algorithms are described, based on the listing in section 5.5.

First, a choice has to be made how to refer to the streamer channels, since the sequence can very well be different in the left and the right image. Since the end points of the streamer channels are identified quite early in the script and the variables are based on these end points, the connection between the images is also based on the end points. The idea is to find which end points in the left streamer image correspond to which end points in the right streamer image. These end points can be found using the predictive algorithm described earlier, or by means of the spanning tree matrix. A correspondence matrix is then initialized with zeros and dimensions.

#### Figure 5.4: Pseudo-code written by Bekkers et al. on the perceptual grouping algorithm using the sub-Riemannian distance

```plaintext
input : S: a set of key points;
d(g_i, g_j): distances between g_i, g_j ∈ S;
s_{max}: max spatial length of geodesics;
variables: D_S: set of possible edges;
δ_i: node degree of x_i;
output : D_S: final set of edges;

Initialization:
Compute the distances d(g_i, g_j) (and corresponding geodesics) between all key points g_i, g_j ∈ S.
Initialize D_S with the set of all edges between each g_i, g_j ∈ S whose connecting geodesic has spatial arc length smaller than s_{max}, and set D_S = Ø.

Main algorithm:
while D_S ≠ Ø do
  1. Select edge and remove it from D_S:
      (g_i, g_j) ← argmin_{(g,h)∈D_S} d(g, h);
      D_S = D_S - (x_i, x_j);
  2. Check topology and update network:
      if δ_i < 2 and δ_j < 2 and g_i, g_j are not already in the same subgraph in D_S
      then D_S = D_S + (g_i, g_j);
      δ_i = δ_i + 1;
end
```

5.9 **Connection between two-dimensional images**

With the two-dimensional reconstruction finished, making the connection between the images becomes much easier. Now, not every dot in one image can correspond to every dot in the other image, since there must be some kind of grouping present. In this section, various ways of making this connection using MatLab algorithms are described, based on the listing in section 5.5.

First, a choice has to be made how to refer to the streamer channels, since the sequence can very well be different in the left and the right image. Since the end points of the streamer channels are identified quite early in the script and the variables are based on these end points, the connection between the images is also based on the end points. The idea is to find which end points in the left streamer image correspond to which end points in the right streamer image. These end points can be found using the predictive algorithm described earlier, or by means of the spanning tree matrix. A correspondence matrix is then initialized with zeros and dimensions.
equal to the number of branches or end points in the images. These need to be equal for the algorithm to work.

Next, the parameters in the listing in section 5.5 are checked for uniqueness. For the velocity throughout the streamer channel and the vertical positions of the streamer dots in the streamer channel, for each combination of branches in the left and the right image the accumulated difference of the velocities and the vertical positions is calculated. If one of these accumulated differences for a certain combination of branches is significantly lower (usually this factor is chosen to be a factor two lower than all other accumulated differences), a connection between these branches is made and a ‘1’ is set in the correspondence matrix. These checks are done using the functions ‘VelocityConnection’ and ‘VerticalPositionConnection’ respectively.

For the horizontal position of end points and the curve length of the smooth curves in the branches something similar is done, but not with accumulated numbers. Rather, the individual values per branch are compared and here too if the difference between a branch from the left image and one from the right image is significantly lower (usually a factor two), a connection between these branches is made by setting a ‘1’ in the correspondence matrix with as row number the number of the left branch and with as column number the number of the right branch. Note that for these methods the small angle approximation is applied; it is assumed that the positions of the streamer discharges don’t differ significantly in the images and the sequence of the branches is not very different. If this actually is the case, simply no connections can be made with this method and other parameters need to be used. These check are performed using the functions ‘HorizontalPositionConnection.m’ and ‘CurveLengthConnection.m’ respectively.

If there are many branches present, it is very useful to also perform all these tests from a branching point on to the end point of the streamer discharge. Using this method, it is much easier to distinguish two streamer channels with many shared streamer dots. Another check can be done when performing this method which is to find the angle between branching points and other branching points and branching points and end points. Especially when streamer channels share a large amount of streamer dots, this angle between branching point and end point is one of the most unique properties that makes it possible to distinguish these two streamer branches. Note that approximately linear streamer channels are needed for this method, as well as the small-angle-approximation. This check is done using the function ‘AngleFromBranchingConnection.m’.

The last parameter used is the number of dots in the streamer branch. Here it is checked whether the number of dots in a certain branch is unique. If no other branch in this image has the same amount of dots and there is one branch in the other image with the same unique number of dots, a connection is made between these branches by adapting the correspondence matrix. This is done with the function ‘NumberOfDotConnection.m’.

After the connection between the image has been completed, it is useful to redefine a streamer variable in which the dot in row ‘i’ in the streamer variable of the left image corresponds to the dot in the same row in the streamer variable of the right image. Previously, only the connection of which dot corresponds to which had been made, but with this addition the variables are better organized for the last step, the transition to the three-dimensional model. This sorting is done with the use of the function ‘StreamerDotCorresponder.m’.

5.10 Transition to three-dimensional model

As mentioned previously in section 4.4 for the last step of the reconstruction the applet ‘Stereo Camera Calibrator’ is used. This applet can introduce a depth coordinate to a series of points
in two images when a calibration session has been done, in this case on checkerboard images. Pictures need to be taken of these checkerboard images in positions and orientations where the streamer discharge is also expected. After calibration, a file consisting of the stereo parameters and their errors can be exported to the MatLab work space and separately stored in a file. This file is then used by the standard MatLab function 'triangulate' to fully merge the information of the two images into one three-dimensional model of the streamer discharge. To have a better overview of the structure of the streamer discharge, this function is also applied to the found spline variables, which contain much more points than only the streamer dots.

After this three-dimensional object has been plotted, the goal of this project has been achieved. Next, a basis can be laid for further analysis on the three-dimensional reconstructed streamer discharges. For this, the branching points need to be identified. These can easily be extracted from the spanning tree matrix, which contains all information on which streamer dot is connected to which. The spanning tree matrix has two columns; the first column contains the beginning points of a connection and the second column contains the end points of a connection. Thus, if a certain streamer dot is twice or more times present in the first column, it is a branching point. It does not matter in which image the branching points are found, since it is known at this point which dot in the left image corresponds to which dot in the right image. Since the analysis is performed on the splines rather than the streamer dots (the splines have a finer discretization), actually the spline point closest to the branching dot is looked for. When these branching points are identified, as well as the starting and end points, properties like velocity, branching length and branching angle can be extracted from the three-dimensional streamer discharge reconstruction.

5.11 Comparison with existing methods

It is insightful to also take a look at other existing methods for observing and reconstructing three-dimensional streamer discharges, even more since parts of the algorithms used are based on existing research. However, not a whole lot has been researched on this specific topic; a way more tackled problem is the one of streamer discharge simulation. Practically all existing research on observation of three-dimensional streamer discharges make use of the stereoscopic approach, but not all researchers use two cameras. Ichiki et al. for example use three cameras with one having a 0° angle with the vertical axis, one having a 90° angle and one having a 225° angle, such that the dilemma between measurement angle and complexity of problem mostly vanishes [2]. They also study the morphology of underwater streamers, but the most relevant result is that the histogram of the branching angle have a bell-shape which will be discussed more in section 6, the results section.

For the actual reconstruction of three-dimensional streamer discharges there is an important difference between the approach of this research and the approaches in most researches, namely the use of stroboscopic techniques. In this research, several photos of streamer heads are combined to form one stroboscopic picture. There is a clear distinction between the bright streamer dots and the vague trajectories of the streamer heads, the afterglow. In most existing research, just one picture of the whole streamer discharge is taken where the whole of the trajectories of the streamer heads have the same intensity. Something similar can be seen if in the current configuration a photo is taken of the afterglow when all streamer heads are gone. While the stroboscopic approach gives great advantages, some pre-analysis must be done on the stroboscopic pictures before comparison with other research can be done, as the trajectories of the streamer heads can not be seen adequately in the stroboscopic pictures and these need to be extracted using algorithms described in subsection 5.8.2. Thus, some of the used techniques
5.11 Comparison with existing methods

regarding streamer dot analysis can not be compared with existing techniques on analysis of the streamer discharge, but some research done by Ichiki et al. has been used in further analysis of the streamer discharge. Namely, when the trajectories have been found and noise has been removed, one of the steps leading towards a three-dimensional model of the streamer discharge is the identification of branching nodes. Ichiki et al. propose an elegant technique for that \[3\]. Their idea is to create a circle around each pixel belonging to the streamer head trajectories. Then the amount of pixels in each direction inside this circle is counted and plotted inside a histogram. For pixels inside a straight streamer channel without branching, two peaks should occur with an angle of 180° between them. For pixels near branching nodes, one of the peaks should become lower and broader. For pixels on top of branching nodes, three or more peaks should be visible with equal heights. In this way, the branching nodes can be identified, as well as the branching angles. This approach poses problems when there are lots of seemingly intersecting streamer channels, but this can be overcome by careful administration of streamer channels and the algorithms of subsection 5.8.4.
Chapter 6

Results

The various algorithms described in chapter 5 all work for different kind of streamer discharges and each have their advantages and drawbacks. In this section, the results per algorithm or working method are described and visualized, as well as what problems occur for each algorithm. The settings used to obtain all these pictures are listed in appendix B.

6.1 Most basic algorithm results

The most basic algorithm as described in subsection 5.8.1 does not do any sequencing step, but assumes that the dots as the program reads them (from top to bottom) are already in the right sequence. Branching streamer cannot be analyzed using this method. As a result, the sequencing step is very straightforward and little goes wrong, with the right streamer discharges. Streamer discharges which do not go straight down can not be analyzed. Additionally, the amount of streamer dots should be the same in both images in order for the algorithm to work. Since there is no occurrence of overlapping streamer channels or branching, this can simply be made possible by proper calibration.

The second important step, namely connection between images, is also trivial since nothing really happens; the sequence is already assumed to be right. All in all, three-dimensional reconstruction is quite easy using this algorithm, but it only works with the most basic streamer discharges possible.
6.2 Predictive Algorithm results

The predictive algorithm is the most basic algorithm in which a sequencing step is actually performed. The basic principle is easy; making a prediction of the next dot in time using information of the dot at the current time step. In practise, there are a lot of problems, mostly regarding the uncertainty in the orientation value and at branching points.

Several options were suggested for this reconstruction in this project, boiling down to two main methods; a front-end reconstruction and a back-end reconstruction. Results for the reconstruc-

Above in figure 6.1 a typical reconstruction result can be seen of a streamer discharge which can be reconstructed using the most basic algorithm. The streamer approximately goes straight down, experiences no branching and has no overlapping streamer channels. Other examples in which this algorithm works are very similar and are thus now shown.

6.2 Predictive Algorithm results

The predictive algorithm is the most basic algorithm in which a sequencing step is actually performed. The basic principle is easy; making a prediction of the next dot in time using information of the dot at the current time step. In practise, there are a lot of problems, mostly regarding the uncertainty in the orientation value and at branching points.

Several options were suggested for this reconstruction in this project, boiling down to two main methods; a front-end reconstruction and a back-end reconstruction. Results for the reconstruc-
6.2 Predictive Algorithm results

Two types of circles can be seen in every picture; the prediction made for each dots and the actual streamer dots (which already have been identified). As can be seen, most predictive dots are nearby the actual next or previous dots in time. In the front-end reconstruction method visible in subfigure 6.2a, there are less predictive dots, as at branching points only one prediction can be made per dot. This is solved by also searching for possible branches at every streamer dot, which is quite computation expensive. As a results, the backwards reconstruction method gives nicer results in general.

**Figure 6.2**: Comparison of predictive algorithm reconstruction methods for a streamer discharge.
Thus, further reconstructions have made use of back-end reconstruction, specifically back-end reconstruction with additional automatic velocity calibration. Results of such two-dimensional reconstructions are shown below in figure 6.3.

As can be seen, the limits of the algorithm are tested here. Some predictions are far off and some connections are almost wrong, as the predictive points are somewhere in the middle of two streamer dots. Additionally, it is clearly visible that the algorithm works less well on non-straight parts of the streamer discharge.

**Figure 6.3:** Comparison of predictive algorithm reconstruction methods for a streamer discharge. Axis values are in pixels.
A total reconstruction of a streamer discharge using the predictive algorithm is shown below in figure 6.4. This is a typical streamer discharge which can be reconstructed using this algorithm.

![Image of a streamer discharge as imported in MatLab.](image)

![Spanning tree of a streamer discharge.](image)

![Interpolation splines through streamer dots of a streamer discharge.](image)

![Three-dimensional reconstruction of a streamer discharge.](image)

Figure 6.4: Typical reconstruction results for a streamer discharge with branching using the predictive algorithm. Dimensions are in pixels for the first three pictures and in mm for the last picture.

This streamer discharge shown in figure 6.4 is a typical streamer which can be reconstructed easily with this predictive algorithm. If the dots are too close to each other, if they are not intense enough, if there is too much branching or if the orientation is not clear, this predictive algorithm will not work.

### 6.3 Minimum Spanning Tree results

The minimum spanning tree fixes most problems of the predictive algorithm and it needs much less calibration in order to work. Furthermore, it is built upon optimized already existent MatLab functions, which has as a result that it is quite efficient. In order to let it work, a cost matrix
6.3 Minimum Spanning Tree results

needs to be made based on distances between streamer dots. If Euclidean distance is chosen for this cost matrix, the cost matrix is easily and quickly found. If instead sub-Riemannian distance is chosen, the cost matrix needs to be imported via the Mathematica notebook regarding the nilpotent approximation. However, information on orientation is already implemented this way, while an additional orientation check needs to be performed afterwards in a recursive algorithm if the Euclidean distance is chosen instead.

A typical streamer discharge for which the reconstruction method with the minimum spanning tree works well is shown below in figure 6.5.

![Image of a streamer discharge as imported in MatLab.](a)

![Spanning tree of a streamer discharge.](b)

![Interpolation splines through streamer dots of a streamer discharge.](c)

![Three-dimensional reconstruction of a streamer discharge](d)

**Figure 6.5:** Typical reconstruction results for a streamer discharge with moderate branching using the minimum spanning tree algorithm. Dimensions are in pixels for the first three pictures and in mm for the last picture.

As can be seen, the algorithm can work with streamer discharges with some degree of branching and also quite well with streamer discharges where branches are quite close together, as it optimizes the streamer discharge as a whole, rather than making optimal connections based on the information of streamer dots separately. The main problem is that one streamer dot with a misleading orientation (i.e. due to its roundness) can this way make the whole spanning tree
6.4 Nilpotent Approximation results

The last algorithm discussed implements all theory on sub-Riemannian distances and geodesics and in this algorithm curvature can be penalized. The value of the orientation of the streamer dots is used much more effectively in this method and distances between streamer dots can be given in much more ways than just Euclidean in this way. Results for a typical streamer discharge which can be analyzed this way are shown below in figure 6.7.

Figure 6.6: An example of an instance where the minimum spanning tree algorithm does not work or needs quite some calibration. The left image is the spanning tree as made by the Kruskal algorithm, the right image is the correct spanning tree.

6.4 Nilpotent Approximation results

a wrong, since the algorithm values each streamer dots evenly. Furthermore, the check used for the orientation value is quite weak and prone to errors. An example of this is visible below in figure 6.6. The Kruskal algorithm combined with the orientation check yields two incorrect connections (the green line in the left and the blue line in the right image) and thus omits two important connections (between the large dots in the left and the large dots in the right of the image).
6.4 Nilpotent Approximation results

Figure 6.7: Typical reconstruction results for a streamer discharge with moderate complexity using the nilpotent approximation algorithm. Dimensions are in pixels for the first three pictures and in mm for the last picture.

As can be seen, this streamer discharge has some nearby channels and quite a lot of branching. It is important that all connections are made from the top of the streamer discharge to the bottom of the streamer discharge, since this is the natural flow of the streamer.

For overlapping streamers some problems arise. This is mostly due to the fact that with overlapping streamers, almost always the streamer dot of one of the streamers can not be identified, since another channel is lying on top of it. The algorithms used require an equal number of streamer dots, however this can be circumvented by only analyzing the three-dimensional model of the interpolated curves, rather than the one of the streamer dots. Worth noting is that this algorithm is the only algorithm that can reconstruct streamer discharges where overlapping streamers are present, since it uses the orientation value the best and since it only considers a maximum of two connections per dot, such that no unrealistic branches are made. An example of a successful two-dimensional streamer discharge reconstruction when crossing streamer channels are present can be seen below in figure 6.8. For this reconstruction, the algorithm based on nilpotent approximation has been used. Other algorithms like the predictive algorithm or
the algorithm based on a minimum spanning tree built upon a cost matrix are not able to do this reconstruction, since the missing key point in the underlying streamer channel is crucial for these algorithms to work.

![Figure 6.8](image.png)

**Figure 6.8:** An example of a streamer discharge in which the right image shows an example of crossing streamer channels. With the algorithm based on nilpotent approximation such a streamer discharge can still be reconstructed.

This example above in figure 6.8 also shows the strength of the algorithm based on sub-Riemannian distances; it also works well when the streamer dots to be connected lie far away from each other. The orientations and the possible curves between the key points is the most important piece of information for reconstruction in general and this algorithm based on sub-Riemannian distances uses this information most effectively.

### 6.5 Interpolation results

As mentioned before, different interpolation methods have been used during this project; cubic spline interpolation, Hermite spline interpolation and Hermite spline interpolation with extra derivative information. Underneath in figure 6.9 the differences can be seen in the case of streamer discharges.
6.5 Interpolation results

The differences are quite small, but it can be seen that the Hermite Spline with derivative information yields smooth curves with more curvature than the other two interpolation methods. It can also be seen that the standard cubic spline actually does use information on the derivative at the end points, as this method sets the derivative equal to zero there. These pictures also give a justification why the standard Hermite Spline interpolation method is preferred to the Hermite Spline interpolation method with extra derivative information, while the latter uses more relevant information. The curves in the former case give more smoother and realistic curves.

A reason why the Hermite Spline derivative information interpolation method is sometimes preferred over the standard Hermite Spline interpolation method is shown below in figure 6.10. Before branching the orange and the blue line should be overlapping, since they indicate the same streamer channel. Due to the extra derivative information, the Hermite Spline derivative information interpolation method stays closer to the truth in this case.
6.6 Miscellaneous Results

Regarding the information which is available for the reconstruction algorithms, there is one thing that is especially important; orientation. The orientation of the streamer dots is obtained by plotting a 2D Gaussian fit on top of the found streamer dots, which should be elliptical. The center coordinates of the streamer dots are found without problems, but in some cases the orientation of the streamer dots is misleading. This is mostly the case for streamer dots on branching points and streamer dots with low intensity. In the first case, the streamer dots are actually not ellipses, but are shapes with three appendices. In the latter case, the streamer dots are often identified as circle-like, thus having a very unreliable orientation. This unreliability is strongly correlated to the eccentricity of the streamer dots. If this eccentricity is high, the streamer dots are strongly elliptical and have a reliable orientation. Since almost every algorithm described and used in this project relies on this orientation, it is very important to have a reliable orientation. Relevant experimental parameters for this are the pulse length for making pictures and the intensity of the dots. Higher pulse lengths give more elongated streamer dots, which consequently have higher intensities. Additionally, gases like pure nitrogen result in more intense streamer discharges than gases like pure carbon-dioxide.

Alas, something still goes wrong in the last reconstruction step, namely finding the depth coordinate. Using the calibration with checkerboards, the cameras make pictures of easily identified objects in the space where the streamer discharge is also expected with different typical orientations. Using this, the Stereo Camera Calibrator applet in MatLab finds spatial distances of connected points in two images. While exact distances can not be measured using this experimental setup due to the presence of the gas vessel and the fact that the streamer discharge takes up a full space rather than one specific point, it is not possible to check exactly whether the found distances are correct. However, they are very close to reality: a distance of $55 \pm 3 \text{ cm}$ between the mirrors and a distance of $120 \pm 10 \text{ cm}$ between the mirrors and the streamer discharge. This has been consistently obtained with different calibration sessions. However, the major problem is that for all measurement series of streamer discharges and all calibration sessions, the three-
6.6 Miscellaneous Results

Dimensional reconstructed streamer dots lie in the same plane, see figure 6.11. This plane is not horizontal or vertical, but diagonal with realistic depth coordinates. Clearly, something is going wrong in the three-dimensional reconstruction. Different pictures of checkerboard were used, one with squares of $9.0 \pm 0.05$ mm and one with squares of $7.2 \pm 0.05$ mm. They were put into different regions of space, made on different measurement days, made with different intensity intensifiers and all possible streamer discharges were used for three-dimensional reconstruction. In every single case, the dots laid in the same plane with similar angles with the basis axes. The assumption is that something is going wrong internally in the Stereo Camera Calibrator in MatLab. Due to this problem, no data analysis was performed on the found three-dimensional reconstructed streamer discharges, since this would yield meaningless data anyway. A possible reason for this malfunction is the small angle between the pictures; the applet is not able to distinguish the angular separation between the pictures. However, it was not possible to test larger angles between pictures in the experimental setup used.

![Figure 6.11: Visualization of the streamer dots all lying in the same plane in the three-dimensional reconstructed streamer discharge. As it turns out, this is the case for all streamer discharges and all calibration sessions.](image)

From the experiences obtained in this project there are a number of things which are important for three-dimensional reconstruction of streamer discharges. In theory, every streamer discharge can be reconstructed when the right information is extracted from the discharge. Depending on the experimental setup and the camera settings, the streamer discharge can be reconstructed or not. Most importantly, the shutter time of the ICCD camera needs to be chosen correctly. When working with stroboscopic techniques, several pulses for the opening and closing of the gates in the ICCD camera are sent, see section 4.3. When the time between the pulses is too short, the streamer dots can’t be properly distinguished from each other and the identification of streamer dots becomes very difficult. However, having a long time between pulses results in a too high information loss such that the channels can’t be properly identified. From experience, the gap between the streamer dots should be a half to two times the width of the streamer dots itself. This corresponds to a pulse camera frequency of 8 to 15 MHz in the experiments done for this project. Since the streamer dots are usually remarkably bright, identifying the streamer dot from the afterglow poses few problems. However, the shutter times themselves are more important. Too short shutter times yield streamer dots that are too round and have a less reliable orientation value. The orientation of the streamer dots is a very important piece of
information for the reconstruction, thus the shutter times should be rather high. Additionally, the streamer dots become brighter when they are more elongated, since more light is captured from the streamer dots. Ultimately, this corresponds to a camera pulse width of 20 to 50 ns for the experiments in this project.

The pulse width and frequency of the high-voltage pulse are less important for the actual streamer, but they should be chosen such that the streamer discharge does not reach the grounded plate in the gas vessel, a distance corresponding to about 15 cm. The frequency of the high-voltage pulse just needs to be chosen such that it corresponds with the image taking of the camera. However, regarding the problem talked about in section 4.4 about the voltage being too low to initiate a streamer discharge when doing measurements, it helps when this frequency is rather high, in the order of 10 Hz to 1 kHz. The pulse width for these experiments should be below 1 µs in order to avoid short circuit. Shorter pulses give shorter streamer discharges and these need less stroboscopic photos. Optimal settings are maximally 30 streamer dots for the longest streamer discharges and 10 to 15 for most streamer discharges.

Lastly, the ideal voltage and pressure settings are strongly coupled. Higher pressure give thinner and brighter streamers, but the streamers get complex more easily when the voltage increases compared to lower pressures. For higher pressures, the breakdown voltage is also higher. Thus, the voltage automatically needs to be higher for higher pressures in order to make streamer discharges. For streamer discharges low in complexity, it is better to use lower pressures with voltages close to the breakdown voltage. However, the images become more noisy and less clear this way. Experiments where performed in the range of 3.5 to 9 kV and pressures were taken in the range of 40 to 150 mbar. These fitted the experimental setup well and gave useful results.
Chapter 7

Conclusion

The aim of this project was to accurately automate 3D reconstruction in streamer discharges, specifically using algorithms based on mathematical theory for this. Furthermore, in this project much emphasis has been laid in researching the process which leads to this and the problems found on the way. This report describes the process of getting to these algorithms and which improvements have been done as a result of problems found during the experiments. The first algorithm discussed in subsection 5.8.1, the most basic algorithm which assumes trivial sequencing, only gave results for the most basic, straight streamer discharges. The results can be seen in figure 6.1. The first algorithm which actually sequences the dots differently is the predictive algorithm, described in subsection 5.8.2. This algorithm was based on the stroboscopic information of the image and the elliptic shape of the streamer dots to make predictions for neighbours for each streamer dot. From the results in figures 6.2 and 6.2, it turned out that back-end reconstruction worked the best, specifically with automatic velocity calibration. Three-dimensional reconstructions of typical streamer discharges fit for this algorithm are shown in figure 6.4.

An algorithm that optimizes the whole network rather than individual connections was described in subsection 5.8.3. It computes the distance between every point and every other point with various possibilities and then uses Kruskal’s algorithm to make a minimum spanning tree. Results on three-dimensional reconstruction are shown in figure 6.5. However, the classical and easiest way of using this algorithm uses an Euclidean cost matrix and information on orientation needs to be used still. The orientation check is not very robust and thus a last algorithm was introduced in subsection 5.8.4. This algorithm is heavily based on theory in chapter 3 and it computes the distance between every dot and every other dot in sub-Riemannian fashion, where penalization based on curvature is used in order to obtain geodesics between streamer dots. After this, a spanning tree can be obtained via perceptual grouping or Kruskal’s algorithm, which works similarly. Results on this algorithm are shown in figure 6.7. Due to technical limitations and calibration issues, streamer discharges like these are the most complex streamer discharges which can automatically be reconstructed with good confidence.

The largest problem right now is the fact that there is a problem with the depth coordinate in the Stereo Camera Calibrator applet in MatLab, as was discussed in section 6.6. While the values of the coordinates is reliable and has a good correspondence with reality, all reconstructed streamer discharges actually lie in a flat diagonal plane rather than in three-dimensional space. Since this is the case for all streamer discharges, all calibration sessions and even for measurements performed outside this project, the assumption is that the problem lies internally in the Stereo Camera Calibrator applet or that this problem is due to the angle between the pictures being too small.
However, while every streamer discharge can be reconstructed, proper experimental settings need to be used in order to obtain a good reconstruction. From experience, the pulse length of the camera is the most important; it should be in the range of 20 to 50 ns to ensure a proper orientation value of the streamer dots. Furthermore, a pulse camera frequency of 8 to 15 MHz, a voltage pulse frequency of 10 Hz to 1 kHz, a pulse width of below 1 μs, 10 to 30 streamer dots per picture and a voltage closely above the breakdown voltage, depending on the type of gas and the pressure are advised settings for three-dimensional reconstruction of streamer discharges.
Chapter 8

Discussion

The aims of this project have certainly been fulfilled. Structural work has been done on creating algorithms for three-dimensional reconstruction of streamer discharges and unique information about velocity, branching length and branching angles can now be obtained in a way that was not possible before with the addition of a new dimension in the streamer discharge models. It is now important to make these algorithms more robust than they currently are and by obtaining as much information as possible from the reconstructed streamer discharges. This can give quite some new insight in the physics of streamer discharges and this would aid the connection between simulations of streamer discharges and experiments done on streamer discharges.

The project has been more difficult than imagined at the start. A lot of small problems have arisen, mostly related to programming issues and calibration of the algorithms to specific measurement series. Especially this problem of constant calibration makes it difficult to use the made algorithms in a general case of reconstruction of streamer discharges. The amount of manual work per streamer reconstruction is too high right now to make the algorithms efficient. It would be advisable for further work to make use of error margins in important results or variables, such as the orientation of the streamer dots or the probability that a connection between streamer dots can be made.

In particular for pictures of complex streamer discharges, it turned out to be very difficult to accurately and consistently identify all streamer dots. Since the current algorithms are based on the fact that the amount of streamer dots in both pictures were equal, it was very important to make sure that this streamer dot identification happened well. For a lot of overlapping streamer channels, this identification could not be done. Further research should find other methods to make connections that are not based on streamer dots. It should be noted that the images used were quite noisy. This problem was partly overcome by applying a Gaussian blur to the pictures, but it is something that could be improved. Furthermore, quite some streamer dots turned out to have a unreliable orientation since they were not elongated enough. While this was also noted earlier in the report, still some images could not be analyzed properly or there had to be manual work done since the streamer dots where not elongated enough. Additionally, streamer dots on spots where branching occurs most of the times had unreliable orientations. However, these spots are easily identified and thus these spots should be taken into account in the reconstruction in further work.

While it would be advisable to introduce some kind of variable to quantify the concept of complexity for further research, it is somewhat difficult to give such a definition before analyzing a picture. For example, if a streamer discharge has a lot of branching or seemingly crossing streamer channels, it is intuitively more complex since reconstruction is more difficult. However, it is not possible before analyzing the picture to say something about the amount of branches.
or overlapping channels in the streamer discharge. Thus, it is not possible to give an exact
range for the complexity in which specific algorithms will work. However, it is possible to give
a complexity number to a successfully reconstructed streamer discharge. Such a number would
help with comparing different algorithms regarding usefulness. From experience, a streamer
discharge or a picture of a streamer discharge should be considered more complex if it has more
crossing streamer channels, if it has more streamer channels lying close to each other, if it has
more branching, if it has more merging, if it is more closely packed and if is has more short
branches (rather than long branches, where more information can be extracted from).

There are several other suggestions for further research, as well as some point of improvements.
These are listed below:

• First of all, the calibration of the reconstruction programs costs quite a lot of time and
as a result there is still quite a lot of manual work per image of a streamer discharge in
order to get the full three-dimensional reconstruction of the streamer discharge. There is
quite some information missing about a possible pattern in this calibration and a machine
learning algorithm or neural network could really help automatizing this process.

• For complex streamers with lots of overlapping channels, currently it is sometimes too
difficult to accurately obtain the positions of all streamer dots using the function 'region-
props’. Here too a neural network can work out if it is trained with proper training data
of complex streamers where the streamer dots have been properly identified manually.
All algorithms used in this project heavily rely on the information obtained from these
streamer dots and it is very important that this information is indeed reliable.

• For a more serious approach to this interesting problem of three-dimensional reconstruction
of streamer discharges a more professional setup needs to be used, for example using an
optical table and freely translatable components. Due to spatial limitations of the setup
used in this projects, some measurements could not be performed optimally (such as being
able to use only one kind of lens).

• Related to this is the matter of the number of cameras; with this setup information of only
two angles could really be used during the experiment, but the reconstruction becomes
way easier if another camera or angle is used, for example one which is above the other two.
Not only becomes the reconstruction easier since way more information can be obtained,
but also much more complex streamer discharges can be reconstructed when more cameras
or angles are available.

• This has already been mentioned in section 5 but being able to store the information of
all stroboscopic photos separately instead of combining them in one stroboscopic picture
would help tremendously as much more information is obtained and can be used this way.

• Several important parameters have not been altered during this project, such as the type
of gas used and the total dimensions of the streamer discharge. Since the idea of these
algorithms is that all streamer discharges can be reconstructed and analyzed, experiments
with other setups and other gases should be performed.

• The emission spectrum of the streamer discharges has not been accurately measured during
this project. If such spectra are known for different kinds of streamers, the ICCD camera
settings can be altered according to this information and assistive UV-lamps can be used
specified for these spectra.

• The most important result of these reconstruction algorithms is being able to obtain
information on the three-dimensional streamer discharges, rather than from the two-
dimensional pictures. Much information should be acquired and analyzed on streamer
discharges in order to understand these streamers better.
• In this project, no overlap was present for the left and the right image on the CCD chip in order to not overcomplicate the project. However, when the pictures overlap slightly, they can also be slightly larger in order to fill the whole CCD chip space. Larger images give more accurate information and are more useful. Further projects could work on this by rotating one of the images or by giving one a different colour than the other, such that more space in the CCD chip can be used. Furthermore, if in later researches the reconstruction algorithms have been proven to work really well, the left and right images can actually overlap a lot, since they can be parted afterwards using these algorithms.

• A start has been made with error analysis in this project, but this should be looked into with more detail. For every connection the algorithms make between streamer dots, a probability of a correct connection should be given and thus a probability of a correct eventual network should be possible to give. If this is the case, it is also easier to spot possible improvements in the algorithms in a more quantitative manner.
Bibliography


Appendix A

Mathematica Code

In this section relevant code for the algorithms used in this project is shown. The complete code can be retrieved from the data center of the EPG group at TU/e.

Listing A.1: The main code for the perceptual grouping algorithm and the calculation of sub-Riemannian distances.

```mathematica
Options[PerceptualGrouping] = {
  "MaxSpatialArcLength" -> \[Infinity],
  "MaxSRArcLength" -> \[Infinity],
  "ProgressIndicator" -> False
};
PerceptualGrouping[keyPoints_, distanceMatrix_, geodesicsMatrix_,
  OptionsPattern[]] := Module[
  (*Options:*)
  maxSpatialArcLength = OptionValue["MaxSpatialArcLength"],
  maxSRArcLength = OptionValue["MaxSRArcLength"],
  (*Initial edge list:*)
  \[CapitalPi] = DeleteCases[
    Flatten[Outer[List, Range[Length[keyPoints]],
      Range[Length[keyPoints]]], 1], {i_, i_}],
  (*Degree vector:*)
  degreeVector = ConstantArray[0, Dimensions[distanceMatrix][[1]]],
  (*Connectiviy matrix from which we can extract the sub-graph DsTilde:*)
  DsFinalMatrix = ConstantArray[0, Dimensions[distanceMatrix]],
  (*The corresponding list of edges*)
  DsFinal,
  (*some checks:*)
  okQ, geos,
  (*Dimension of base domain*)
  n = 2
  ],

  (*Sort \[CapitalPi] on distances*)
  \[CapitalPi] = \[CapitalPi][[Ordering[distanceMatrix[[Sequence @@ #]] & /@ \[CapitalPi]]]];

  If[OptionValue["ProgressIndicator"],
    PrintTemporary[
      ProgressIndicator[Dynamic[edgeIndex/Length[\[CapitalPi]]]]]];

  Do[
    (*Select edge with shortest distance*)
    {i, j} = \[CapitalPi][[edgeIndex]];
(*Check topology*)
\[\Delta\]i = degreeVector[[i]];  
\[\Delta\]j = degreeVector[[j]]; 

(*-----Check if edge can be added considering the restricted \ connectivity*) 
(*If[\[\Delta\]i<2&&\[\Delta\]j<2,*) 
If[\[\Delta\]i < maxNodeDegree[[i]] && \[\Delta\]j < maxNodeDegree[[j]]], 
If[ 
With[{connectedGraphComponents =  
    ConnectedGraphComponents[
        Graph[Position[DsFinalMatrix, 1]], {i}]}, (*Avoid closed loops*) 
    If[Length[connectedGraphComponents] < 1, 
        True, 
        Not[AllTrue[connectedGraphComponents, 
            ContainsAny[VertexList[#], {j}]]]], 
    okQ = True; 
If[okQ && 
    EuclideanArcLength[geodesicsMatrix[[i, j]]] <= 
    maxSpatialArcLength && 
    distanceMatrix[[i, j]] < maxSRArcLength, 
    DsFinalMatrix[[i, j]] = 1; 
    degreeVector[[i]] += 1; 
    degreeVector[[j]] += 1; ]; ]; ]; 
, {edgeIndex, 1, Length[\[CapitalPi]], 1} ]; ]; 

(*Return the resulting sub-Graph*)
DsFinal = Position[DsFinalMatrix, 1]; 
Return[DsFinal]
Appendix B

Settings

This appendix contains the settings for the experiment and the code for each figure in the results chapter.

Figure 6.1

Picture settings:
Pressure: 50 mbar
Voltage: 3 kV
Frequency of pulse: 1 kHz
Length of pulse: 800 ns
Frequency of image capturing: 8 MHz
Shutter time per photo: 40 ns

MatLab code settings:
LNoise: 2
LObject: 17
image.LocalMaxInfo.STD: 1.5
GaussianBlur: 2.5

Figure 6.2

Picture settings:
Pressure: 40 mbar
Voltage: 3.5 kV
Frequency of pulse: 1 kHz
Length of pulse: 1000 ns
Frequency of image capturing: 1 MHz
Shutter time per photo: 60 ns

MatLab code settings:
LNoise: 2
LObject: 17
Figure 6.3

Picture settings:
Pressure: 49.7 mbar
Voltage: 4.5 kV
Frequency of pulse: 1 kHz
Length of pulse: 1000 ns
Frequency of image capturing: 1 Mhz
Shutter time per photo: 80 ns

MatLab code settings:
LNoise: 2
LObject: 17
image.LocalMaxInfo.STD: 1.5
GaussianBlur: 2.5
CalibratedVelocity: 60
BranchCheckDistance: 50
ConnectionCheckDistance: 125
VelocityUniqueness: 1
HorizontalPositionUniqueness: 1
VerticalPositionUniqueness: 1

Figure 6.4

Picture settings:
Pressure: 49.7 mbar
Voltage: 3.5 kV
Frequency of pulse: 1 kHz
Length of pulse: 1000 ns
Frequency of image capturing: 1 Mhz
Shutter time per photo: 40 ns

MatLab code settings:
LNoise: 2
LObject: 17
image.LocalMaxInfo.STD: 1.5
GaussianBlur: 2.5  
CalibratedVelocity: 60  
BranchCheckDistance: 50  
ConnectionCheckDistance: 125  
VelocityUniqueness: 1  
HorizontalPositionUniqueness: 1  
VerticalPositionUniqueness: 1  
CurveLengthUniqueness: 1

**Figure 6.5**

Picture settings:
Pressure: 40 mbar  
Voltage: 4.5 kV  
Frequency of pulse: 1 kHz  
Length of pulse: 800 ns  
Frequency of image capturing: 1 Mhz  
Shutter time per photo: 40 ns

MatLab code settings:
LNoise: 3  
LObject: 17  
image.LocalMaxInfo.STD: 1.5  
GaussianBlur: 3.5  
ErrorAngle: 25  
VelocityUniqueness: 1  
HorizontalPositionUniqueness: 2  
VerticalPositionUniqueness: 3.5  
CurveLengthUniqueness: 5

**Figure 6.6**

Picture settings:
Pressure: 100 mbar  
Voltage: 7.2 kV  
Frequency of pulse: 5 Hz  
Length of pulse: 1000 ns  
Frequency of image capturing: 8 Mhz  
Shutter time per photo: 50 ns

MatLab code settings:
LNoise: 2  
LObject: 17  
image.LocalMaxInfo.STD: 1.5  
GaussianBlur: 3
Figure 6.7

Picture settings:
Pressure: 60 mbar
Voltage: 6.1 kV
Frequency of pulse: 5 Hz
Length of pulse: 800 ns
Frequency of image capturing: 10 Mhz
Shutter time per photo: 40 ns

MatLab code settings:
LNoise: 2
LObject: 19
image.LocalMaxInfo.STD: 1.5
GaussianBlur: 2.8
VelocityUniqueness: 1
HorizontalPositionUniqueness: 2.4
VerticalPositionUniqueness: 1
CurveLengthUniqueness: 2

Mathematica Code settings:
MaxSpatialArcLength: 380
MaxSRArcLength: 1.65

Figure 6.8

Picture settings:
Pressure: 40 mbar
Voltage: 3.6 kV
Frequency of pulse: 1 kHz
Length of pulse: 800 ns
Frequency of image capturing: 5 Mhz
Shutter time per photo: 60 ns

MatLab code settings:
LNoise: 4
LObject: 17
image.LocalMaxInfo.STD: 2
GaussianBlur: 2.5

Mathematica Code settings:
MaxSpatialArcLength: 400
MaxSRArcLength: 1.5