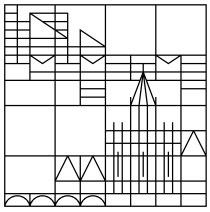
Computer Graphics Support in Head-Mounted Displays for Helicopter Guidance

Dissertation zur Erlangung des akademischen Grades des
Doktors der Naturwissenschaften

vorgelegt von
FERDINAND EISENKEIL

an der



Universität Konstanz

Mathematisch-Naturwissenschaftliche Sektion
Fachbereich Informatik und Informationswissenschaft

Tag der mündlichen Prüfung: 25. April 2016
1. Referent: Prof. Dr. Oliver Deussen
2. Referent: Prof. Dr. Tobias Schreck

Abstract

Providing navigational information from sensors and databases to a rotorcraft pilot on Head-/Helmet-Mounted Displays during *Degraded Visual Environment* situations is a challenging problem in computer graphics and a substantial task in avionics engineering. Data from different sensors such as GPS and LiDAR sensors as well as transponders needs to be processed in order to provide more high-level descriptions and symbols than raw data. Subsequently, the sensor data needs to be not only displayed but also abstracted dependent on the flight situation during a mission according to the pilot's needs for spatio-temporal coherence.

This dissertation investigates the use of a local-to-global method to create digital terrain models for LiDAR data classification as an extension to existing local approaches optimized for avionic hardware. Furthermore, a clustering approach as pre-processing as well as different methods for rendering of non-classified LiDAR data are analyzed and provided within a flight simulator environment. We present an efficient 3D meshing approach that avoids cluttering of rendered elements in the helmet-mounted display. This meshing approach is combined with even more abstract renderings at different Levels-of-Detail. Similarly, we apply a density-based cluster method on traffic information in order to create a spatio-temporal coherent 2D abstraction with minimal impact to the pilot's workload. These methods are completed with an alternative landing site selection as well as a terrain rendering approach motivated by automatically created artistic stylizations.

The results of this dissertation are advantageous for navigation through different flight situations such as landing and flights in low altitude. We extend state-of-the-art methods already applied in flight decks for operational use. The resulting visualizations are prototypes that have been analyzed in a small user study with a professional helicopter pilot. More exhaustive user studies with experienced pilots reflect the subsequent step. which is left for future work.

Zusammenfassung

Eine wesentliche Aufgabe bei der Entwicklung von Avionik-Systemen und ein herausforderndes Problem in der Computergrafik ist die Darstellung navigatorischer Informationen auf Head-/Helmet-Mounted Displays die in *Degraded Visual Environment* Situationen dem Helikopterpiloten als Assistenzsystem dienen. Daten von verschiedenen Sensoren wie zum Beispiel GPS und LIDAR-Sensoren als auch Transponderinformation müssen verarbeitet werden, um Symbole und High-Level-Beschreibungen anstatt von Rohdaten bereitzustellen. Anschließend werden die Sensordaten nicht nur angezeigt, sondern auch abhängig von der Flugsituation während eines Einsatzes in Abhängigkeit von den Bedürfnissen des Piloten raumzeitlich kohärent abstrahiert.

In diesem Zusammenhang wird in dieser Dissertation die Verwendung einer Methode untersucht, mit der anhand von lokalen Merkmalen ein global gültiges, digitales Geländemodell zur Klassifizierung von LiDAR Daten erstellt wird. Dieses Verfahren dient als Erweiterung zu bestehenden lokalen Ansätzen. Zusätzlich werden ein Clustering-Verfahren als Vorverarbeitung sowie verschiedene Methoden zur Darstellung von nicht-klassifizierten LiDAR Daten analysiert und in einem Flugsimulator zur Verfügung gestellt. Um diese 3D Cluster ohne Renderingartefakte und Clutter darzustellen, wird ein effizientes Verfahren zur Erstellung eines Polygonnetzes eingeführt. Dieses Netz wird in verschiedenen Levels-of-Detail mit weiter reduzierten Darstellungsmöglichkeiten kombiniert. Ähnlich zur 3D Darstellung der Cluster wird ein dichtebasierter Clusteralgorithmus auf Verkehrsinformationen angewendet, um eine räumlich-zeitliche kohärenten Abstraktion mit minimalen Auswirkungen auf die Arbeitsbelastung des Piloten zu schaffen. Diese Verfahren werden um eine Darstellung eines alternativen Landeplatzes als auch um eine Terraindarstellung, die von automatisch erstellten, künstlerischen Stilisierungen motiviert ist, erweitert.

Die Ergebnisse dieser Arbeit sind für die Navigation in verschiedenen Flugsituationen nutzbar, wie beispielsweise während eines Landeanfluges und Flüge in geringer Höhe. Damit werden moderne Methoden erweitert, die bereits in Cockpits im Einsatz sind. Die resultierenden Visualisierungen sind Prototypen und wurden in einer Nutzerbefragung mit einem professionellen Hubschrauberpiloten analysiert. Weitere und vollständige Benutzerstudien mit erfahrenen Piloten sind der nächste logische Schritt und werden für weiterführende Arbeit offen gelassen.

Acknowledgments

First and foremost, I would like to thank my advisor, Prof. Dr. Oliver Deussen, for the opportunity to work with his group and providing the freedom to pursue my personal interests in computer graphics and avionics. I also thank my second advisor, Prof. Dr. Tobias Schreck, for reviewing the thesis and the great discussions we had in Konstanz and Graz.

Many thanks go to the project partners at Airbus Defence and Space, Dr. Uwe Kühne, who made this work possible through the cooperation with the University of Konstanz, Dr. Tobias Schafhitzel, who initially invited me to participate at this collaboration and provided me the opportunity to work with his team, and Dr. Ralf Stadelhofer, who had a great impact on this thesis and spent a lot of time with thorough and profound discussions.

Additional thanks go to my colleagues who shared not only work time with discussions and brainstorming but also time used for more comfortable things such as playing Quake III Arena, coffee meetings and summer evenings in beer gardens during recent years. In particular these are my back-to-back colleague Thomas Lindemeier, the next-door colleagues Julian Kratt, Marc Spicker, Till Niese, and the former colleagues Daniel Heck, Sören Pirk and Hendrik Strobelt. Furthermore, many thanks go to Ingrid Baiker, for all her organizing and moral support. Moreover, I would like to thank all the students who had an impact to my work: Sophie Seidel, Markus Friedrich, Dennis Martin, David Haidl and Andrea Capobianco. Also I want to thank Johannes Ernst, who had a great impact on the realization of the traffic visualization topic during his master thesis at Airbus Defence and Space.

Many thanks go to my friends from the neighboring departments, Tanja Waldmann and Lisa Stühler. Thank you for the great time during coffee breaks, beer garden sessions, pub visits and journeys to Forchheim and Eichstätt!

An important impact on this thesis had my family and friends. Therefore, special thanks go to my girlfriend and partner Veronika, who always inspired and supported me with her patience and love. Last but not least, special thanks go to my parents, Eva and Sigurd - for all their love, patience and sustainable support!

Contents

1

Introduction

Rotorcraft missions during *Degraded Visual Environments* (DVE) situations have a significant impact on the workload as well as on the navigational capabilities of a helicopter pilot. Particular situations that cause degraded visual are fog, heavy precipitation, limited sunlight and dispersed sand or snow during brownouts (Figure 1.1(a)) and whiteouts that can cause spatial disorientation. Flying during such situations often results in fatal accidents, especially during low level flights and landings. In order to increase the pilot's situational awareness during low level flights and landings, pilot assistance systems are developed. One particular helicopter equipment system containing a LiDAR sensor system, a digital map generator and a synthetic vision solution is the SFERION® system provided by Airbus Defence and Space. We use the sensing and synthetic vision system that are part of this situational awareness solution as a basis for providing algorithms for data classification and clustering together with rendering approaches for Head-/Helmet-Mounted Displays (HMD).

As a part of the situational awareness solution, the used sensing technology is a modern LiDAR (Light Detection And Ranging) system (Figure 1.1(b)) that emits light pulses to measure the time of flight of the reflected light [WL99]. The data recorded by such a sensor is used to support navigation by providing additional visual cues in the cockpit or the pilot's Head-Up Display (HUD). To make the LiDAR data accessible, the scanned range values are recorded in a regular two-dimensional projection plane and thus interpreted as a depth image seen from the helicopter. This image can directly be displayed on a Head-Up Display as a superimposed out-the-window view. Since the position and orientation of the helicopter is available through GPS and INS sensors, another possibility is to transform the depth image into a global coordinate system. To generate additional visual cues derived from the resulting 3D point cloud, sensor data needs to be classified. This system records a point cloud with up to 20.000 measured points at an approximate frequency of 3Hz. Examples of these two representation forms of LiDAR are given as a depth image in Figure 1.2(a) taken from the scene depicted in Figure 1.2(b)

as well as the corresponding point cloud in Figure 1.2(c), and the classification of a deployed operational ground classification in Figure 1.2(d). Within the flight simulator provided by Airbus Defence and Space, the simulation of such a point cloud is possible and an example of it from the pilot's perspective, projected into a virtual 3D scene with classification hints is given in Figure 1.2(e).

(a) (b)

Figure 1.1: (a) Helicopter in a DVE situation during a brownout [DDP]. (b) The LiDAR sensor is highlighted in a blue ellipse mounted in front of a NH90 helicopter [Air].

In modern helicopter flight decks, Head-/Helmet-Mounted Displays are part of standard equipment to enable the use of *Enhanced Vision Systems* (EVS). For an optimal usage such as the see-through display, these systems are head-tracked [DSL14, MSS+14]. One of the major benefits of such displays in a cockpit is the reduction of the pilot's head-down time by avoiding the need of mentally transferring the information from Head-Down Displays (HDD). Information such as the speed and altitude tapes together with three-dimensional symbology to increase the pilot's situational awareness is presented conformal to the environment. An example of an HMD as well as two-dimensional symbols and 3D conformal terrain renderings and a landing aid is depicted in Figure 1.3. Similar to the LiDAR sensor system, these head-up visualizations as well as head-down renderings are provided through the existing cockpit visualization system.

Melzer [Mel12] describes HMDs as "powerful tools that can unlock the pilot from the interior of the cockpit or the forward line of sight of the Head-Up Display". The displays "can enable the pilots to do their job more effectively while simultaneously decreasing workload" [Mel12]. As a matter of fact, during DVE scenarios the pilot's workload is automatically increased. To avoid a further growth of the workload resulting from display clutter and popping artifacts, the visualization of the surrounding environment as well as traffic renderings have to present the required information in an intuitive way without any unnecessarily distractions to the pilot. This requires all rendered elements within the HMD to appear either directly or smoothly faded - according to the situation. For the different rendered elements - unknown obstacles, terrain lines and traffic symbols - we need to introduce spatio-temporal data structures as well as interpolation techniques.

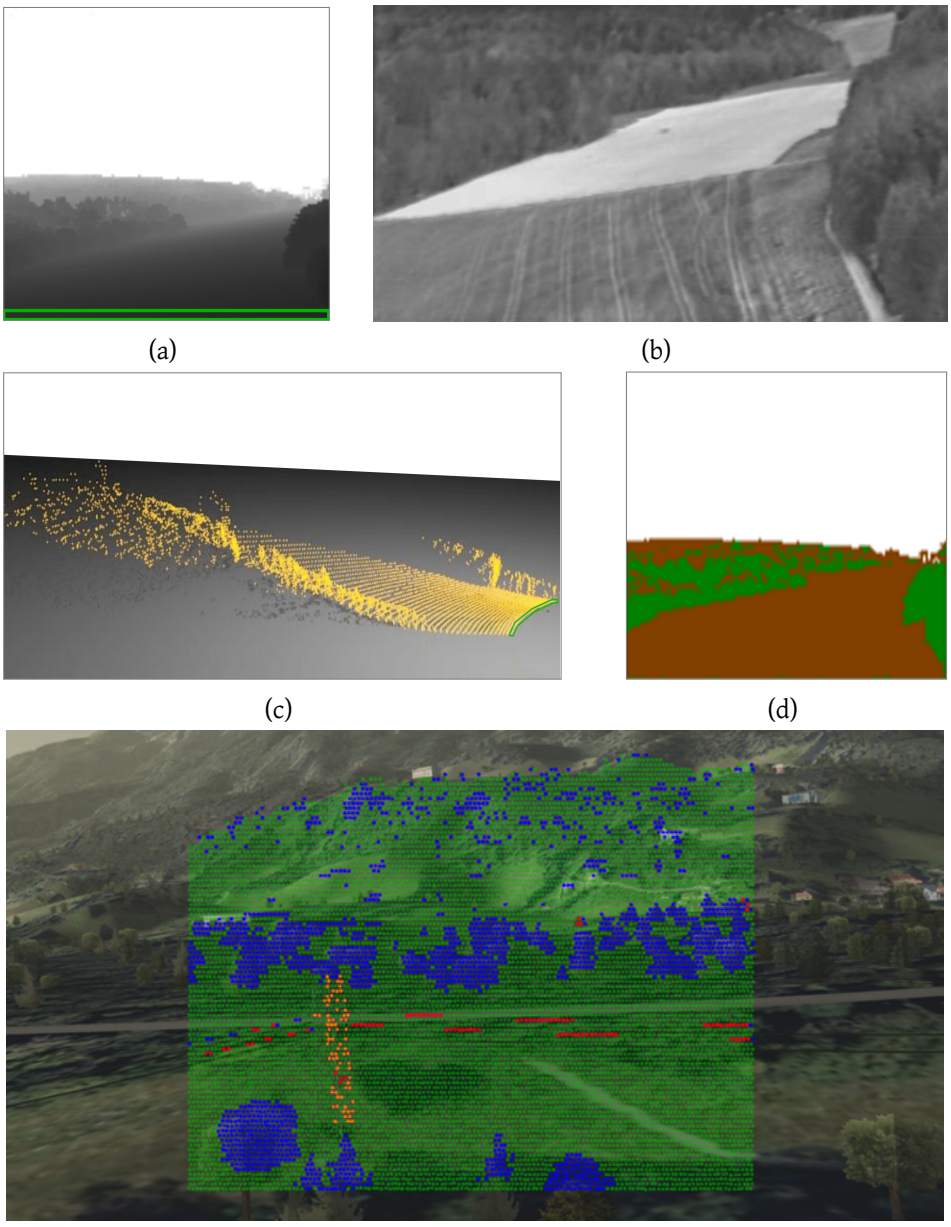
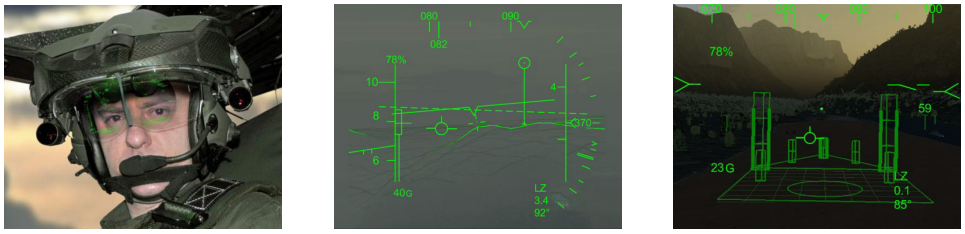


Figure 1.2: (a) LiDAR point cloud represented as a depth image. The range of each point is coded as gray value while the green box indicates corresponding data points in (c). (b) Forward Looking InfraRed (FLIR) camera image from similar perspective as LiDAR range sensor. (c) Range values are transformed to 3D coordinates. Since the camera position in (c) is not corresponding to the helicopter's view, the green box indicates corresponding pixels. (d) Ground and Non-classified pixels as a result from the operational system. Non-classified pixels are green while ground pixels are brown. (e) Classification result from front view of a point cloud from our flight simulation, divided into different classes: red points belong to wires, poles are given in orange, ground is green and blue points have the label "non-classified".



(a) (b) (c)

Figure 1.3: The head-up symbology of the pilot assistance system [MSS+14]. (a) Pilot wearing a TopOwl® helmet with see-through display (HMD) provided by Thales Group (Figure from [LFP13]). (b) 3D conformal terrain grid incorporating contours with 2D symbology overlay. (c) 3D conformal landing symbology.

1.1 Overview of the Thesis

Data classification and processing together with Level-of-Detail rendering and data abstraction have a long history in computer graphics and computer vision. Creating digital terrain and surface models from airborne scanned data is a well-known problem and plenty of algorithms exist for rendering large landscapes or complex objects. Terrain surface analysis has a broad application in archaeological as well as surveillance topics. The most rendering techniques developed in recent years had the scope of providing efficient methods that can be used within computer games or other real-time applications on a modern PC or comparable target systems. Providing rendering techniques on HMDs during flight missions is an upcoming application and for different use cases such as landing site depiction and terrain mesh renderings, a few prototypes exist. These applications have special requirements in terms of code generation, computation efficiency as well as Human-Machine-Interaction (HMI) aspects.

The major goal of this thesis is to provide the pilot a visualization that is as efficient and meaningful as possible on the HMD by making use of state-of-the-art data processing algorithms, data abstractions methods and rendering techniques for different flight situations. This implies the research questions on how LiDAR data can be made accessible to the pilot. This can be answered by different processing steps: the point cloud can be divided into different classes, each of which can be processed separately and a symbolization that is rendered on the HMD can be derived from the LiDAR information. In addition to the symbolization, a spatio-temporal coherent data structure for storing the rendered objects has to be provided. Especially for LiDAR information that remains non-classified, this has to be achieved by providing a coherent cluster approach. After these pre-processing steps, a spatio-temporal representation for unknown obstacles detected from non-

classified LiDAR points is presented, whereby the focus of the thesis is shifted to generating renderings on the HMD. For the four rendering topics - non-classified obstacle rendering, terrain depiction, landing site representation and traffic visualization - prototypes are implemented that fulfill the research question concerning how this information can be presented according to the HMI aspects. The resulting methods and visualizations are accessible in the flight simulator designed for pilot questionnaires and evaluations. The full processing pipeline from the LiDAR sensor up to the final visualization scenarios is given in Figure 1.4.

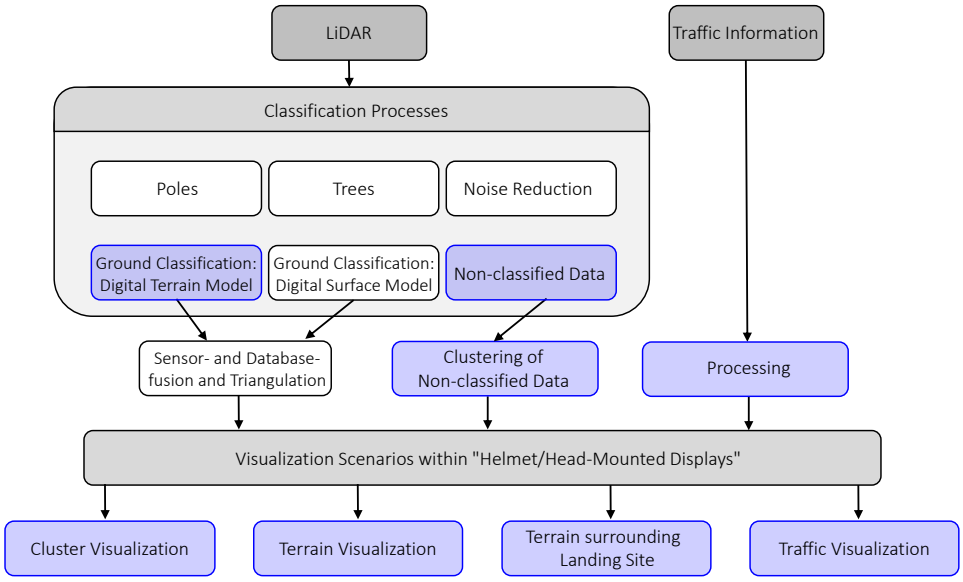


Figure 1.4: Typical overview of a sensing and visualization system for DVE situations: White boxes indicate existing processes while blue boxes depict the contributions of the thesis to increase the pilot's situational awareness.

1.2 Summary of Contributions

The main contributions of this thesis are:

1. An efficient classification algorithm that partitions the LiDAR point cloud into ground and non-ground. This is the first algorithm in such a sensor system that introduces a global and not only local terrain model.

2. A stable, scalable and efficient streaming-like clustering algorithm based upon *DBSCAN*. This approach enables detecting obstacle information in the form of convex hulls from non-classified LiDAR points.

3. A spatio-temporal stable meshing and rendering method for non-classified obstacle clusters at different Levels-of-Detail according to the navigation terms of the pilot.

4. A 2D-based, abstract rendering technique for traffic information that is also based upon the *DBSCAN* algorithm.

5. Additional feature renderings surrounding the landing site with two configurations. The one configuration is designed as a warning system about areas with unstable terrain where landing is not possible, if the landing site is missed, while the second configuration shows alternative areas where landing is possible and the landing aid can be activated.

6. A rendering system for artistic appearing fractal and real-world terrains in different styles.

7. A terrain rendering system derived from the artistic one that is applied on the HMD.

The contributions 1, 3, 5 and 7 were defined as part of the verbal dissertation proposal and are addressed in combination with a spatio-temporal interpolation requirement, which was also defined in the supplement material of the proposal. These contributions have partly been published in the following publications:

F. Eisenkeil, T. Schafhitzel, U. Kühne, and O. Deussen. Real-time Classification of Ground from LiDAR Data for Helicopter Navigation. In Proc. SPIE 8745, Signal Processing, Sensor Fusion, and Target Recognition XXII, 2013. SPIE. [ESKD13]

The author of this thesis was involved in development the classification algorithm, implementing the prototype and the integration into an operational system as well as writing the paper.

F. Eisenkeil, T. Schafhitzel, U. Kühne, and O. Deussen. Clustering and Visualization of Non-classified Points from LiDAR Data for Helicopter Navigation. In Proc. SPIE 9091, Signal Processing, Sensor/Information Fusion, and Target Recognition XXIII, 2014. SPIE. [ESKD14]

The author of this thesis contributed the idea of the clustering algorithm, the development and prototype implementation as well as the integration into the flight simulator setup with simulated LiDAR data and writing the paper.

T. Münsterer, T. Schafhitzel, M. Strobel, P. Völschow, S. Klasen, and F. Eisenkeil. Sensor-enhanced 3D Conformal Cueing for Safe and Reliable HC Operation in DVE in all Flight Phases. In Proc. SPIE 9087, Degraded Visual Environments: Enhanced, Synthetic, and External Vision Solutions, 2014. SPIE. [MSS+14]

The author of this thesis contributed the idea and the prototype implementation of the contour enhancement algorithm.

F. Eisenkeil, J. Ernst, R. Stadelhofer, U. Kühne, and O. Deussen. Traffic Visualization in Helmet-Mounted Displays in Synchronization with Navigation Displays. In *Digital Avionics Systems Conference (DASC)*, 2015 IEEE/AIAA 34th. [EES+15]

The author of this thesis was responsible for the project coordination, requirement definition and algorithm design and was involved in writing the paper.

1.3 Structure of the Thesis

This thesis is organized as follows. In the next chapter we define different navigation tasks in varying classes. According to these classes we introduce requirements for renderings on HMDs regarding to HMI aspects - avoidance of cluttering and providing spatio-temporal coherence - as well as explaining data management issues in flight decks. Additionally, we briefly introduce evaluation methods for HMD renderings. The following chapters are dedicated to the processing pipeline given in Figure 1.4. In Chapter 3 the ground classification procedure is presented and in Chapter 4 the clustering method for non-classified LiDAR points is introduced. Chapter 5 is dedicated to rendering meta-information such as the non-classified LiDAR clusters from Chapter 4 as well as the more abstract traffic information. Chapter 6 describes how terrain can be artistically stylized and the outcome of this method can be used to provide an abstract terrain rendering on the HMD. The thesis concludes with a summary containing a pilot feedback and future work in Chapter 7.

2

Requirements and Evaluation in Avionic Systems

Flying in DVE environments requires input from different systems such as sensors and databases as well as the pilot's input itself such as positions for the landing sites. A representation of this data and input can be given in the form of visual cues on HDDs and/or HMDs. By using head-up or head-mounted cues, information can be provided to the pilot that still allows him to fly manually while minimizing the reduction in the situational awareness resulting from head movement by using an HDD or visual clutter on the HMD. Moreover, an important piece of information is at which flight phase the pilot handles its rotorcraft and which piloting task must be taken into account for an appropriate rendering on the HMD. Padfield [Pad07] divides the piloting tasks into three sub-classes that are extended by Viertler et al. [VKH15] according to their impact to collision avoidance:

- Short-term stabilization: task of controlling the continuous horizontal and vertical movements triggered by natural instabilities of the helicopter.
- Mid-term guidance: task of controlling the trajectory such as maintaining the flight speed and altitude in order to avoid collisions. Within this task the pilot changes its flight speed depending on the situation and the view range to have the possibility to react within less than five seconds.
- Long-term navigation: task of controlling the flight path from the start to the landing destination with the extension of prevent collisions and improve navigation with the use of rendering landmarks.

Each of these tasks results in a feedback loop and have to be handled by the pilot continuously. Our focus is placed upon the mid-term guidance as well as the long-term navigation task. For these two tasks, obstacle and terrain information

are used to increase the situational awareness since both can be used as collision avoidance features as well as landmarks. This results in the need of at least two Levels-of-Detail for the rendering process in order to distinct between these two tasks. In Figure 2.1 a 2D schema for the mid- and long-term task is depicted.

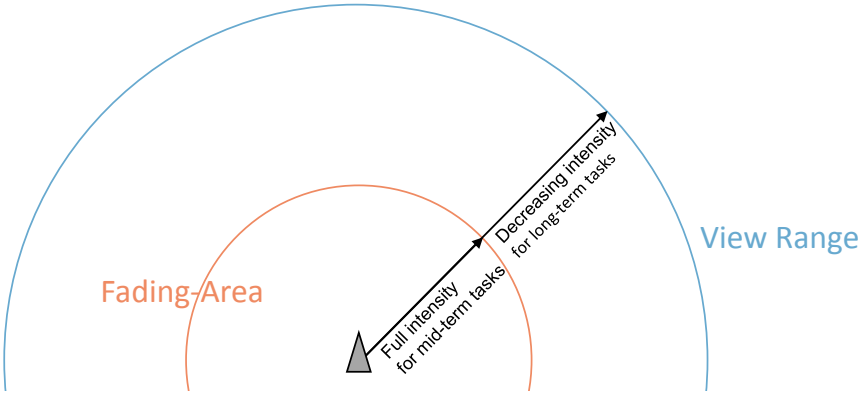


Figure 2.1: Two Levels-of-Detail centered at the helicopter position: the inner circle describes the Level-of-Detail designed for mid-term tasks. Within this area, information is rendered at full intensity for the guidance task to prevent collisions. For the outer circle, the intensity of rendered elements decreases with the distance to reduce the amount of geometry within the pilot's Field-of-View for the long-term task.

2.1 Cluttering

By using different Levels-of-Detail, we reduce the amount of rendered geometry, which directly results in the avoidance of cluttering. For terrain mesh rendering with different Levels-of-Detail, many methods ranging from the method using progressive meshes [Hop96] to the recursive mesh subdivision [LH04] exist. Moreover, methods using hybrid Level-of-Detail approaches exist for full landscapes [CCDH05]. For rendering terrain as well as obstacles on an HMD, it is necessary to extract information from geometry input such as meshes in order to provide a more abstract visualization that conserves the essence of the obstacle and terrain. Rendering all available information - terrain mesh, LiDAR point cloud, traffic information - simultaneously and at full Level-of-Detail can lead to overlapping geometry elements and symbols and thus cluttering. By reducing the number of rendered primitives, the number of active pixels within the see-through HMD is reduced. To reduce the clutter effect, different problem-based Level-of-Detail approaches can be used, depending on the input information. Based upon the LiDAR point cloud, the incoming information can be clustered and represented in the form of wire meshes or other more abstract symbols. This avoids the need of rendering the parts of the input point cloud that cannot be replaced by simple

symbols such as poles or wires. Important details of the terrain can be represented by contourlines as well as ridge and valley lines. For such elements that are 3D conformal with the environment, typical Level-of-Detail approaches can be used such as reducing the detailedness of a rendered object. moreover, the reduced of opaqueness of rendered objects with increasing distance to the helicopter results in a de-cluttered display. For more abstract information, such as traffic, the same simplification techniques as for the LiDAR point cloud can be used to group similar traffic and represent them in 2D. Combined with a threat-measurement this also reduces the number of colored pixels.

2.2 Coherence

In order to provide the pilot a meaningful and smooth visualization it is necessary to rapidly render near sudden changes while the visualization of far away changes in the sensor data that happen far away can be slowly interpolated. The interpolation, intended to guarantee coherence, depends on the data structures used for different elements to be rendered. For terrain abstraction structures, such as ridges and valleys as well as streamlines, a spatial quadtree storage helps to efficiently place these feature lines heuristically. The depth of a quadtree corresponds directly to the rendered Level-of-Detail and can hold a maximal number of feature lines. After this maximum is reached, no further lines can be added. For an increasing Level-of-Detail the number of renderable lines is maximally four times as high as for the previous Level-of-Detail-based upon the quadtree subdivision.

To provide coherence for non-classified LiDAR data we calculate a spatial cluster representation. Each convex hull of a cluster is stored in a doubly linked list that allows - unlike vectors or stacks - adding and removing elements at every position within the memory without losing the linkage information. This enables a fast detection of overlapping clusters and we additionally provide an efficient smooth union of these clusters (see Chapter 4). The visual representation is based upon mapping the LiDAR points within a cluster to a simple geometry that can be interpolated between two sensor updates.

Such a temporal and spatial coherence results in a smooth visualization for especially the long-term navigation area, as well as an effective global scene understanding.

Since the traffic visualization is based upon 2D shapes and the number of traffic is very limited, a temporal and spatial coherent interpolation is reached in a both simple and elegant way. If we detect overlapping traffic symbols by calculating the pairwise minimal distance between all traffic symbols, we detect the corresponding points and subsequently fuse them in discrete steps to one common representation.

2.3 Data Management

A flight deck equipment containing mission computers and display systems as well as network connections between them is limited in its performance. The memory usage is not comparable to modern PCs since the lifetime of such a system needs to be multiple decades and thus the accessible amount of memory - especially the RAM - is often not higher than a few Megabytes. This limitation requires that all algorithms and procedures are implemented with a constant worst-case memory allocation. Moreover, due to the certification of avionic software, only static memory allocation is suitable to reach certain Design Assurance Levels (DAL). This is specified within the *Software Considerations in Airborne Systems and Equipment Certification* guideline DO-178B [RTC82]. The used algorithms for processing sensor data also have the requirement that the delay between acquisition of the data and rendering of the processed information is minimal. Therefore, the algorithms used are either very efficient in their computation time (at least linear for image processing or sensor data processing) or are scalable by simplifying the corresponding data structures or data amount. We implemented all of our proposed methods to be scalable. This yields the advantage that we can adapt the parameters of all algorithms to optimize the trade-off between runtime and result quality as well as reducing the delay between data recording, processing and rendering. A last possibility to increase computation speed and guarantee a certain frequency of data update is to skip parts of the recorded data. For instance, data points that are measured far away from the helicopter and thus do not have a strong impact on the pilot's navigation decisions could be skipped if the data processing takes too much computation time and thus influences the number of frames rendered on the HMD to a non-interactive frame rate. Vice versa, the incoming data can be separated to make the processing algorithms more efficient by using different Levels-of-Detail or other spatial subdivisions. Additionally, for database information offline pre-processing is possible as a preparation for the flight mission. The result of this pre-processing can be adapted according to sensor data during the flight.

2.4 HMI Aspects and Evaluation

For the different rendering topics - terrain, unknown objects derived from non-classified LiDAR points, alternative landing sites and traffic - different HMI aspects are important. In general, it is only necessary to render additional navigation cues in distances higher than the visual range of the pilot. Recent analyses in the form of HMI workshops in flight simulators as well as real flights conducted by Airbus Defence and Space yield the finding for terrain renderings in the form of meshes

could lead to a misunderstanding of the spatial structure of the real terrain and its impact to navigation issues. Since the terrain is represented by a regular grid aligned in two major directions, the orientation of the mesh could be interpreted as a suggested flight direction recommendation. To avoid such a misinterpretation, more organic structures such as ridge and valley and streamlines or even height-based isolines in combination with contourlines could help the pilot to have a more distinct conception of the real terrain.

Furthermore, point cloud rendering was evaluated during the HMI workshops: the outcome was that a full point cloud rendering - especially if multiple successive clouds are accumulated - results in a cluttered and tangling representation of the environment on the HMD. However, for different situations such as takeoffs or landings, a short activation of the full point cloud rendering to guarantee that no obstacle was overlooked is helpful for the pilot.

For a quantitative evaluation of reducing display clutter there exists a method in the form of pixel ratio measurement for occlusion determining. The principle of this approach is to set the opaque pixels within the HMD in a relation to the screen resolution. This evaluation approach is used to gain knowledge in order to determine the source of cluttering, definition and standardization of the maximal number of allowed opaque pixels and introduction of an adaptive information control by rendering only the closest objects [VKH15]. Kaber et al. [KAS+08] tested different HMD configurations (activation of different sensors and 2D information such as flight altitude as well as 3D information such as tunnel in the sky) for having an impact on the navigational abilities to the pilot by cluttering. They also used the number of active pixel in addition to the pilot's feedback as a measure for classifying a distracting or non-distracting feature visualization during a landing approach. Kim et al. provide a similar evaluation system to consider a multi-dimensional measure of display clutter in correlation with pilot performance [KPK+11]. The lack of such evaluations is that they do not consider moving, fading and transparent rendered objects. For a more qualitative evaluation that has an impact on further development according to the needs of the pilot, we prefer HMI workshops that have compared to the image-based evaluation the disadvantage of a cost intensive and time consuming planning. For this work, we only conduct a preliminary study in the form of a pilot questionnaire.

For our proposals of different information renderings, we also have to undertake an evaluation within HMI workshops. Since our systems all imply moving objects trough sensor updates and interpolations, a pure pixel measurement is not sufficient. This measurement can be taken into account to reduce clutter but is insufficient to make a clean statement about the quality of coherence and concrete benefit for the pilot. Furthermore, it cannot be taken into account to answer the question of whether 3D information can be replaced by abstract 2D renderings

for in its essence abstract information such as flight and ground traffic data. The same problem emerges for an evaluation of our system consisting of 3D conformal symbols for presenting an alternative 3D landing zone around the selected landing site as well as its counterpart depicting dangerous areas surrounding the landing site. Both of these systems are new and thus have to be evaluated within HMI workshops.

3

Classification of LiDAR Data

In order to make the point cloud received by the LiDAR sensor usable for a visualization, the sensing and visualization system for DVE situations provides multiple classification steps as introduced in Figure 1.4. Non-classified data points can be used to generate surface and terrain models. Surface models contain all measured data elements such as ground, obstacles, and vegetation. These models are used to determine potential collision points without differentiating between various types of obstacles.

The primary focus of the sensing and visualization system for DVE situations lies on digital terrain models used to pre-process the collected data for obstacle segmentation. The received LiDAR points that are classified as part of the ground are removed from the input point cloud. The remaining LiDAR points must be analyzed for being potential dangers such as power lines, man-made objects and trees. As part of the cockpit equipment, multiple separated hardware devices process the obtained sensor data. The first part is to classify the data directly within the sensor hardware. A mission computer that is running the synthetic vision system is not only used for rendering the 3D enhancement of the environment on Head-Down as well as on Head-Mounted Displays, but also for the detection of additional man-made objects such as buildings as a second processing step. As an extension to the existing system, we propose a ground classification approach implemented as part of the visualization component of the synthetic vision system that is specialized in order to provide a digital terrain model for rendering purposes. It is based upon active contours, used to generate energy minimizing splines through certain constraints [KWT88], and allows an adaptive classification in multiple consecutive sensor frames. In order to deal with the real-time requirements during a flight mission, the implementation of active contours is highly improved to reduce the runtime complexity. In Figure 3.1 is depicted, where this classification step is located within the processing pipeline.

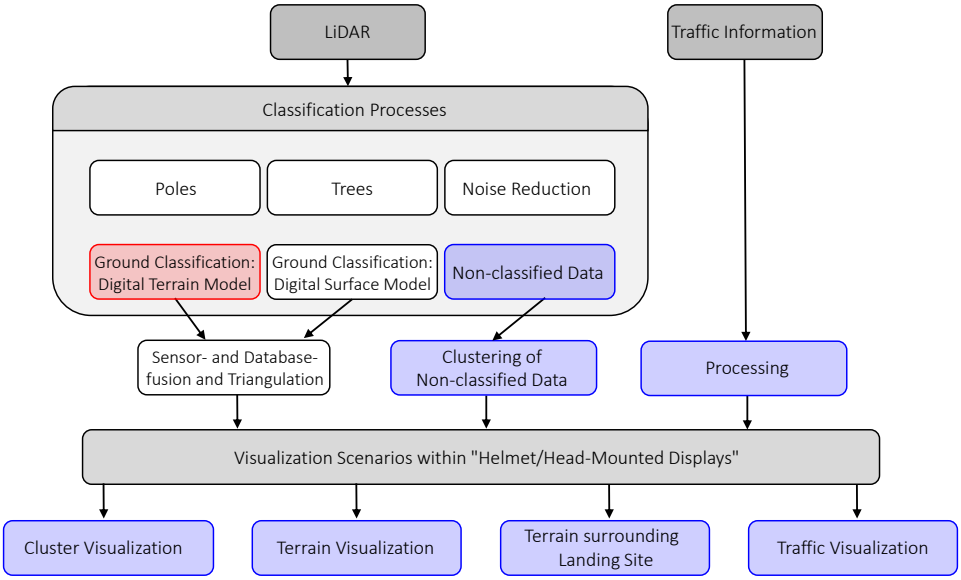


Figure 3.1: Location of the classification step, indicated through the red box within the processing pipeline.

3.1 Related Work

Since classification of LiDAR point clouds has a wide field of applications in land survey a variety of data mining methods for extracting natural vegetation cover from airborne laser data exist. In related papers, triangulation methods are used to construct digital surface models from either airborne or ground-based LiDAR data [CAGZ09]. Furthermore, Meng et al. [MCZ10] provide an overview over a number of filter operators that can be applied to process LiDAR data. In their survey, they distinguish between classification methods based upon segmentation, morphological filters, interpolation, Triangular Irregular Networks (TINs) and active contours.

Segmentation methods:
Rabbani et al. propose a method for arbitrary point clouds segmentation that is based upon a smoothness constraint combined with an initial fitting of a plane onto point clusters. The clusters are detected by using the local surface normals and point connectivity. With this method, they achieve arbitrary oriented shapes in a 3D feature space [RvdHV06]. Using similar features, Tóvári and Pfeifer apply a region growing segmentation to find clusters in unordered airborne LiDAR data [TP05]. In order to find homogeneous regions with similar statistic behavior in LiDAR point clouds, Filin [Fil02] provides a method for clustering points based upon their elevation and slope in a local neighborhood. In order to distinguish

between different classes, Jacobsen et al. provide a method for a global filtering in airborne-scanned LiDAR data with automatic parameter estimation for each class, solely based upon the elevation of the points [JL03].

Morphological filters:
Morphological filters are a common method for determining a digital terrain model. The system proposed by Arefi and Hahn describes a dual rank filter, based upon dilation and erosion. To apply their method to airborne laser data, a mapping from a point cloud to a gray-scale depth image is necessary [LKS00]. In order to detect objects at different scales, Arefi's and Hahn's method makes use of filters with different window sizes [AH05]. To increase the effectiveness of their, it can be extended by changing the dilatation and erosion filters by using progressive window sizes [ZCW+03].

Interpolation methods:
Kraus and Pfeifer introduce a method for generating digital terrain models based upon a iterative linear least-square interpolation of LiDAR points with an adaptive weight function. As part of each iteration, outliers that are not belonging to the ground are removed. This results in a smooth terrain representation [KP01]. As an application, Schickler and Thorpe use a previous version of this filter [KP98] as a basis for their terrain model generation that implies the improvement of using a triangulation, additional curvature and slope constraints as well as break line weighting [ST01]. Other interpolation methods were introduced by Briese et al. [BP01] and Zheng et al. [ZSLZ07] using a facet model.

Deployed ground segmentation during flight:
Airbus Defence and Space already deployed a ground segmentation method for dividing ground points from elevated data during helicopter flights. This method is based upon a Laplacian pyramid [BEA83] and its classification result is shown in Figure 1.2(c). This method is not explicitly used to generate a digital terrain model, since its main objective is to find points in the LiDAR point cloud belonging to obstacles that are clearly elevated compared to the ground, such as power lines or trees. In this method points measured from the top of canopy in forest areas as well as lower vegetation or small rocks might also be classified as ground. This method is used as a comparison for our algorithm.

Iterative Triangulation Methods:
Similar to the interpolation techniques, common methods for classification are based upon iterative triangulation procedures [Axe99, Axe00, HH01, LLSF08]. As a reference to our proposed classification method, we implemented a variant of

Peter Axelson's system [Axe00] using Triangular Irregular Networks. The measured points are sorted according to their height. A first triangle is constructed by the three lowest points; by sorting in new points a triangulation is created using Delaunay triangulation. For each new point the nearest neighbor vertex of the triangulation is found and the angle α between triangle and its normal is calculated. If α is greater than a threshold ($\alpha \in [75°, 90°]$), the point is classified as ground and inserted into the triangulation. A point not classified as ground, is rejected for the TIN generation.

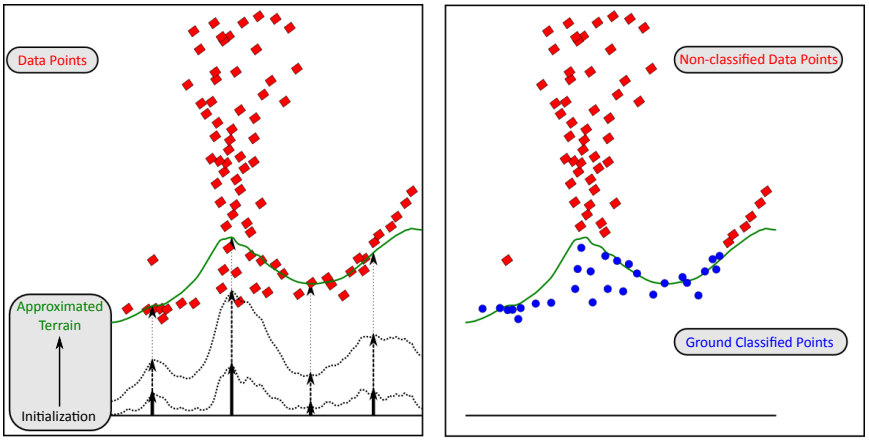
Active Contour Methods:
The approach proposed by Elmqvist [EJL+01, Elm02] for generating terrain models is based upon active contours. Such contours are used to approximate the shape of an object that is only represented by a noisy point cloud [KWT88]. Since the method can be implemented as an iterative process containing a convolution in each step it is not very efficient to generate a terrain-based classifier. Since computation time is the limiting factor in flight decks, we extended this method in order to be more efficient.

3.2 Active Surfaces for Ground Classification

An active contour model is a concept for fitting the shape of objects in 2D images and 3D scenes to parametric curves or surfaces. An active contour or surface is influenced by internal forces, e.g. smoothness, and external forces computed from the underlying data. If the external force is formed by the gradient of an image, the contour is drawn in the direction of lines and edges. When the contour reaches a position where its energy is minimal, it stabilizes and the result is taken. This minimization problem is solved by regularization for internal and external forces [KWT88, XP98]. In order to evolve our method for real-time capability, we use a simplified version of active contours that only regularizes internal forces. We decided to use Elmqvist' method as basis for our algorithm, because most of the related work is used in LiDAR data, obtained in top-down perspective. In addition, none of the compared methods are designed to be used in real-time application where data changes frequently.[1]

Given a data set $d = \{(x_i, y_i, z_i) \in \mathbb{R}^3\}_{i=0}^{N-1}$ of 3D points provided by the sensor, we use the minimization to fit the ground surface from below to the data points. Figure 3.2 shows how such a contour evolves iteratively in 2D on a point distribution as it occurs in LiDAR point clouds and how the classification due to this fitting should be.

[1] The calculations in this Section are based upon the lecture notes "Computer Vision II" by Heiko Neumann, Ulm University, 2008.



(a) (b)

Figure 3.2: (a) Beginning with a horizontal line, the regularized contour is drawn upward until a stable solution is achieved. (b) Points below and slightly above the terrain contour will be classified as ground indicated by blue dots (Figure from [ESKD13]).

Fitting the classifier to scanned points:
The mathematical description of regularization is derived from the formulations of Terzopoulos' and Poggio's equations [GJP95, PTK85, Ter83]. Suppose that the data set $d = \{(x_i, y_i, z_i) \in \mathbb{R}^3\}_{i=0}^{N-1}$ has been obtained by the LiDAR sensor, then each of the data points is derived from a function that describes a surface model or rather the environment. A terrain-based classifier, described by a function $f : \mathbb{R}^2 \to \mathbb{R}$, shall be recovered from this data. This is an ill-posed problem because the number of possible terrain models is infinite.

In order to select one particular solution, *a priori* knowledge such as smoothness of the searched function is needed and is taken into account by using the first derivative $[D_1 f](x, y)$ as proposed for the "membrane" surface model in [Ter83]:

$$[D_1 f](x, y) = \frac{1}{2} \int \int (f_x^2(x, y) + f_y^2(x, y)) dx dy. \tag{3.1}$$

Additionally, the function f should be close to the measured data. Therefore, a distance measure is needed. In our case we chose the squared distance in height, otherwise the function would not be differentiable. Another reason is that the contour should lie in the minimal possible distance below or above to the point. A solution for these two requirements is found by minimizing the following functional $H : C^1[\mathbb{R}^2, \mathbb{R}] \to \mathbb{R}$ for the energy function E:

$$H[f] = \int E[f(x, y), f_x(x, y), f_y(x, y), x, y] dx dy \to min, \tag{3.2}$$

where $H[f] = H_d[f,d] + \lambda H_p[f]$ with the *Data Term*

$$H_d[f,d] = \frac{1}{2}\sum_i \kappa(x_i,y_i)[f(x_i,y_i) - (d_z)_i]^2, \tag{3.3}$$

and the *Model Term*

$$H_p[f] = [D_1 f](x,y). \tag{3.4}$$

The *regularization parameter* λ is a positive number that controls the trade-off between *Data Term* and *Model Term*. The *Data Term* guarantees the closeness to measured data and the *Model Term* the smoothness. κ is a weighting function that indicates whether a data point was measured on position (x_i, y_i).
$H[f]$ is solved with Euler-Lagrange-Equation by calculating the partial derivatives and results in a linear equation system:

$$\kappa(\delta - f) + \lambda\Delta f = 0. \tag{3.5}$$

By interpreting the input data d mapped to a grid δ and f as 1D vectors in the same domain as κ and setting up the *Model Term* and the weighting function κ in the matrices M and K a linear equation system is formulated with a stationary solution instead of an iterative function:

$$(K\vec{\delta} - K\vec{f}) + \lambda M\vec{f} = 0 \Rightarrow \vec{f} = (K - \lambda M)^{-1}K\vec{\delta}. \tag{3.6}$$

where M and K have the following form:

$$M = \begin{pmatrix} -4 & 1 & 0 & \cdots & 1 & \cdots & \\ 1 & -4 & 1 & 0 & \cdots & 1 & \\ 0 & 1 & -4 & 1 & 0 & \cdots & 1 \\ & 0 & 1 & -4 & 1 & 0 & \cdots \\ 1 & \cdots & 0 & 1 & -4 & 1 & 0 \\ & 1 & \cdots & 0 & 1 & -4 & 1 \\ & & 1 & \cdots & 0 & 1 & \ddots \end{pmatrix} \tag{3.7}$$

and

$$K = \begin{pmatrix} \kappa(x_1,y_1) & \cdots & & & 0 \\ 0 & \kappa(x_2,y_2) & \cdots & & 0 \\ 0 & \cdots & \kappa(x_3,y_3) & \cdots & 0 \\ 0 & & \cdots & \ddots & 0 \\ 0 & & & \cdots & \kappa(x_n,y_n) \end{pmatrix} \tag{3.8}$$

Terzopoulos proposes the second derivative as smoothness prior for surface reconstruction. For our classifier both the first and the second derivative yield the same result. The benefit in using the first derivative is the possibility to construct a matrix M for the *Model term* that simplifies the solution of the linear equation system with the conjugate gradient method. This approach is faster and more accurate as the iterative approximation given in Equation 3.9. With the usage of Eulers method for solving ordinary differential equations an iterative solutions for the functional H can be found:

$$f^{k+1} = f^k + \nabla t(\kappa(\delta - f^k)) + \lambda \Delta f, \tag{3.9}$$

where ∇t is a constant value smaller than 1 that influences the intensity of an update. The full derivation of all Equations is given in Appendix A.1

The linear equation system is solved for a regular grid of points. If data points were collected in the vicinity of this grid point, the height is determined (see Figure 3.3) and the weighting function κ is set to 1 for this position, otherwise it is 0. The initialization for the elevation of the grid points is equal to the smallest elevation value $min(\{z_i\}_{i=0}^{N-1})$ in the set d. The grid dimension of the terrain model that is also the domain of a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ is based upon the spatial dimension of the point cloud. If two or more data points of the sensor fall into the same grid cell, the one with the lower elevation is kept for computing the elevation of the cell.

Classification procedure

Having a ground layer available, we are able to classify the measured points as belonging to the ground or to artificial objects. We determine the orthogonal distance of all points to the terrain-based classifier. If the elevation of the grid cell is higher than the elevation of the point, it is classified as ground. Based upon the statistical inaccuracy of height values of each pixel, also points slightly above the classifier are tagged as ground. For the classification of a data point, we identify the grid cell on the classifier membrane matching points' x and y coordinates. The complete process is given in Figure 3.3.

Progressive Runtime Optimization with different Levels of Details

In practice, avionic computers are limited in their performance. Therefore, we propose an optimization to gain real-time behavior of the ground segmentation. Since a sensor frame rate of 3Hz allows for a processing time of at most 333 milliseconds, the regularization procedure exceeds this limitation due to the solving of the linear equation to find the function f for every data set d. In order to in-

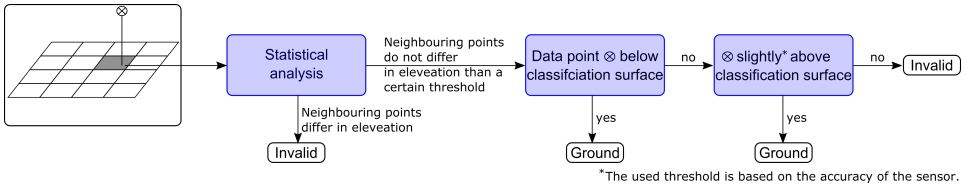


Figure 3.3: Decision chain if a point ⊗ can be classified as ground: The first test is part of the progressive pre-processing. If all the points that match a cell in the classification membrane grid have a large variance in their elevation, the point can impossibly be a ground point. After this step, ⊗ is classified as ground, if ⊗ lies below or slightly above the cell. The maximal distance ⊗ can have above the classifier to be a ground pixel is based upon the sensor inaccuracy (Figure adapted from [ESKD13]).

crease the speed, we have to reduce the accuracy of the grid in dependency to their spatial distribution and quality. The goal is to find the best ratio between Level-of-Detail expansion and computation speed and correctness of classification. Points far away from the helicopter are handled using a coarser grid (see Figure 3.4), points close to the helicopter using a smaller grid. The resulting terrain-based classifier is approximated in a progressive behavior. In the classification of data recorded by a LiDAR sensor mounted on a helicopter, we make a special statement: A spatial and temporal coherence between two sensor frames exists, so that the terrain-based classifier of one frame provides a basis for the following frame. We perform a statistical analysis on all pixels in the area of a grid cell on the terrain-based classifier. This information is kept for the following frames and all pixels that are in the area of this grid cell not taken into account for solving the linear equation. This reduces the number of calculations to be done in the conjugate gradient method. This assumption is valid because of the correlation between the speed of the helicopter and the frequency of recording environmental data. The position and the viewing direction of the helicopter does not change significantly in the short time slot of 300 ms. In the best case, the sensor delivers the same information as in the previous frame and no adaption is necessary. This describes a theoretical optimum if the helicopter does not move. In practice, the sensor is biased by noise and errors in the helicopters positioning system.

3.3 Results and Evaluation

The triangulation method as well as the for operational issues established method deliver terrain classification in real-time, based upon a Laplacian pyramid. In both cases, the results satisfy the requirement to filter LiDAR data for elevated points for further calculations like power line detection. Especially the operational method is usable for this purpose because irregular surfaces like forest areas are classified as ground. Our approach is slower than the two reference implementations, but its

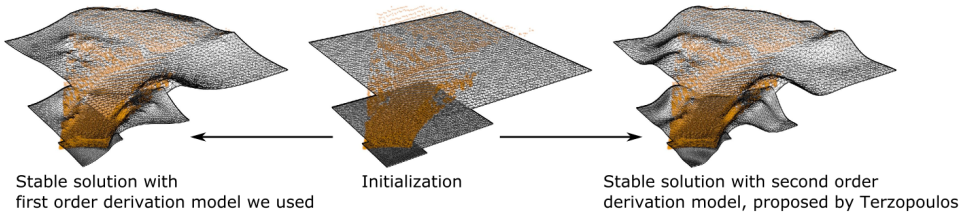


Figure 3.4: The hierarchical approximated terrain-based classifier in different Levels-of-Details is calculated. This is possible in real-time, if the first order derivative model is used (Figure from [ESKD13]).

result is more usable for visualization aims and a candidate for a real-time application in avionic computers. The proposed method is an extension of the process Elmqvist et al. introduced in [EJL+01]. Our method has a few enhancements and differences to their method:

- The generation of the terrain-based classifier is calculated in a fast way, meaning that the calculation can be done between two sensor frames and therefore is fast enough for terrain visualization in real-time.
- Progressive behavior: The spatial coherence between two sensor frames is used to reduce calculation complexity and duration.
- As an additional application to determinate the terrain features, Elmqvist et al. propose change detection, based upon environmental alternation over time. In their paper, they use as an example the building of a wall. In a time slot of 333 ms it will not happen, that the environment changes in a drastic way. The differences between two sensor frames occur in the form of outliers or not yet sampled surfaces.

Evaluation of Classification Procedure

The usage of different Levels-of-Detail and discrete grids for fitting a classifying surface under the point cloud effects not only the time complexity but also the classification result. In Figure 3.5 the classification results for different methods for arbitrary 3D point clouds are shown. In the example with the rift (Figure 3.5(a)-(c)), the operational Laplace-based method classifies points as ground in the almost flat part and not in the valley of the rift. The triangulation and our method detect points at the valley as ground. A similar effect is observed in the arbitrary point cloud in the shape of a tree (Figure 3.5(d)-(f)). In comparison to the triangulation method, our result delivers more ground pixels.

Figure 3.6 shows the classification result for our method in three different real and simulated cases. Cases (a) and (b) are synthetic sensor frames while (c) is

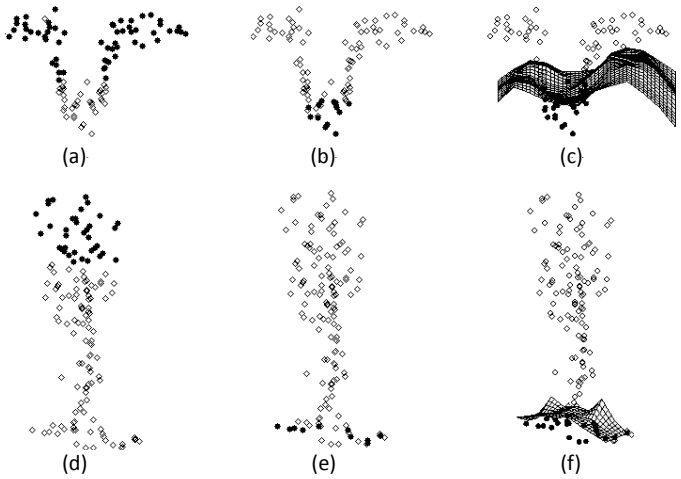


Figure 3.5: Classification results for three different methods arbitrary point clouds in the shape of a rift and a tree (Figure from [ESKD13]). Black dots are ground pixels, boxes are unknown. (a,d) Operational Laplace-based method. (b,e) Triangulation method proposed by Axelson [Axe00]. (c,f) Our method with the classifier shape.

recorded data from a real flight. The first column shows the range image, while the pictures in column 2 indicate the ground truth for (a) and (b). Column 3 shows the operational segmentation result and column 4 the result of the triangulation method. Our result is shown in column 5. The colors depict the class for each pixel. Ground pixels are dark and unknown pixels appear in light grey. In scenario (a) the top side and a part of the cube's side in the foreground is misclassified as ground. Especially for initial sensor frames, misclassifications cannot be excluded. The rest of the image is mostly classified correctly. As illustrated in Table 1, the number of correct classified pixels in scenario (a) is larger than 90% (Correct Ground + Correct Non-ground). More results are given in Appendix A.3.

Table 3.1: Comparison between Ground Truth and our classification result. The amounts of pixels that are correctly and wrongly classified are given in percent.

Classification		Synthetic scene with flat ground (Figure 3.6(a))	Synthetic scene with slope (Figure 3.6(b))
Ground	Correct	66,77 %	34,23 %
	False	4,74 %	1,03 %
Non-ground	Correct	27,44 %	10,34 %
	False	1,05 %	54,40 %

For scenario (b) more than 50% of LiDAR points are misclassified ground pixels (see. Table 3.1). This is due to the fact that only the point with the lowest elevation in a defined neighborhood is used for the construction of the classifier membrane.

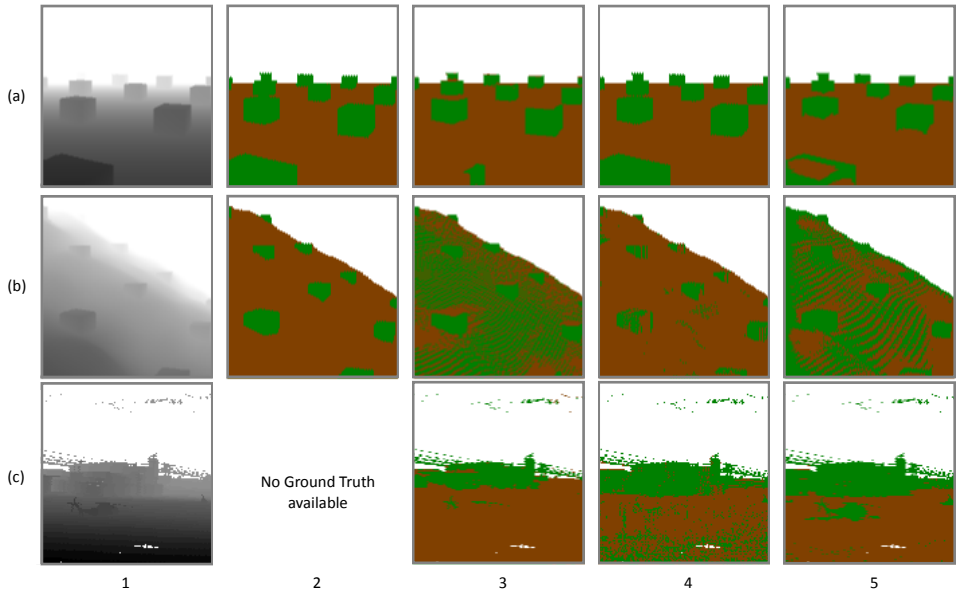
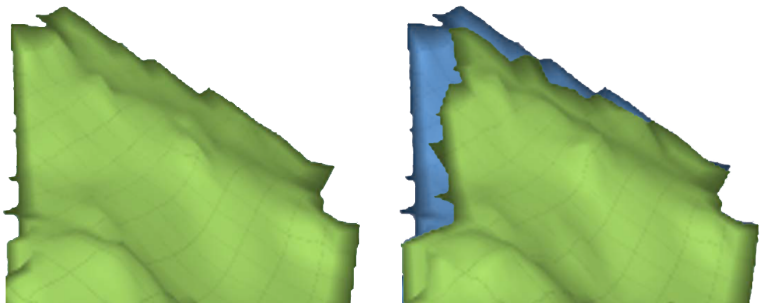


Figure 3.6: Three different scenarios (Figure from [ESKD13]): (a) and (b) show synthetic sensor data with the range image in column 1 and the ground truth classification in column 2 (brown is ground and green is unknown). Column 3 shows the operational segmentation result and column 4 the result of the triangulation method proposed by Axelson [Axe00]. The classification result of our system is given in column 5. Line (c) shows real flight data without ground truth.

Many points lie above the classifier, if the surface is not flat. Because our approach generates a smooth terrain classifier, pixels on the left side on the image lying on a ridge are also not classified as ground. Nevertheless, both the ground truth and our classification result when given as input for the synthetic vision system, result in almost the same terrain visualization (see Figure 3.7).

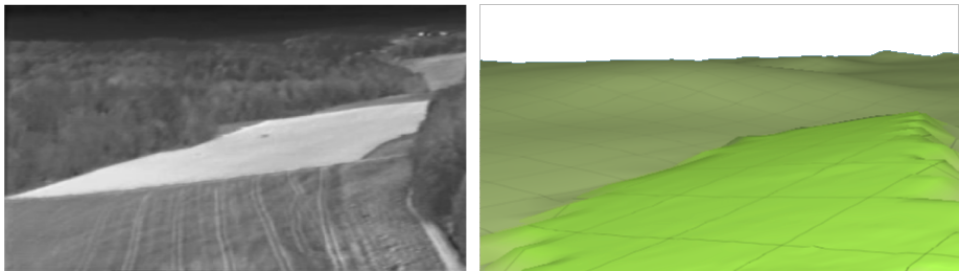
In comparison to the other methods, our method has a few advantages on real data: The operational method misclassifies pixels near the real ground and all flat parts as ground. This behavior is demonstrated on the helicopter pixels in scenario (c, 3) and the top of the cubes in (a, 3). The triangulation method cannot deal with small changes in the slopes. Cubes on the hillside in scenario (b, 4) are partially misclassified. In addition, this method is noise sensitive (cf. Figure 3.6 (c, 4)). Our method increases the classification quality of the operational classification in scenario (c). The differences between Figure 3.6 (c, 3) and Figure 3.6 (c, 5) show that the helicopter as well as the lower parts of the buildings' walls in the background are correctly classified as non-ground. As demonstrated on the surface reconstruction behavior for scenery (b, 1) in Figure 3.6, the noisy misclassification of the ground in the foreground will not influence correct ground visualization in the HDD of the sensing and visualization system for DVE situations. The discretiza-



(a) (b)

Figure 3.7: Terrain generated with a synthetic sensor frame that provides LiDAR data from a hill. This hill is givin as a depth image in Figure 3.6(b): (a) is generated by triangulating the Ground Truth data and (b) by our method with the as ground classified pixels. Both show almost the same terrain. The only exception is the borders of the terrain. In (b) the terrain from (a) is depicted as blue background so the difference between the two triangulation results is visible. This behavior is induced by the numerical solution of the linear equation system (Figure adapted from [ESKD13]).

tion and the resolution of the terrain-based classifier has a negative impact on the classification result. Figure 3.8 shows how a visualization of the environment would look like for a pilot. The light areas are constructed from LiDAR points that are classified as ground. The darker parts of the ground surface seen in the background are from a digital terrain database. LiDAR points that are not classified as ground are not taken into account, so the measured data from the forest is not visible in the synthetic view.



(a) (b)

Figure 3.8: A synthetic view within the sensing and visualization system for DVE situations is created, based upon our new classification method for generating digital terrain models. The artificial generated scene (b) correlates to the image taken with an infrared camera (a) at the same time (Figure adapted from [ESKD13]).

Evaluation of time complexity

To show the reduction of the computation time and complexity of Elmqvist et al. method with our approach with different Levels-of-Detail, we measured the calculation time on a machine with a Quad core i5 CPU@2.4Ghz and 4GB RAM. For Elmqvist's reference implementation, we used a grid with the resolution of the lowest Level-of-Detail in our method. As input served sensor frames in scenarios of different complexity. For this comparison, we analyzed 20 frames for every flight scene. The computation times are shown as box plots in Figure 3.9. In scenario 1, the helicopter flies over a valley with bushes and woods on the sides. Scenario 2 is a flat field, and scenario 3 is an airport with hangars and electrical towers with power lines. Our approach takes less than 333 ms in all three cases. The computation speed of solving the linear equation in Section 3.2 depends on the grid size. It is therefore more efficient to use small dimensions for high Levels-of-Detail. We also tested our implementation on a mission computer with the real-time operating system VxWorks (Version 5.5.2, CPU: 833 MHz, PCI Bus: 33 MHz). The calculation times are slightly higher than on a PC, but with a further reduction of calculations in higher distances, the classification method will be also real-time capable on a mission computer. Based upon the inaccuracy of the sensor in high distances over 500 meters, a ground classification might not be necessary.

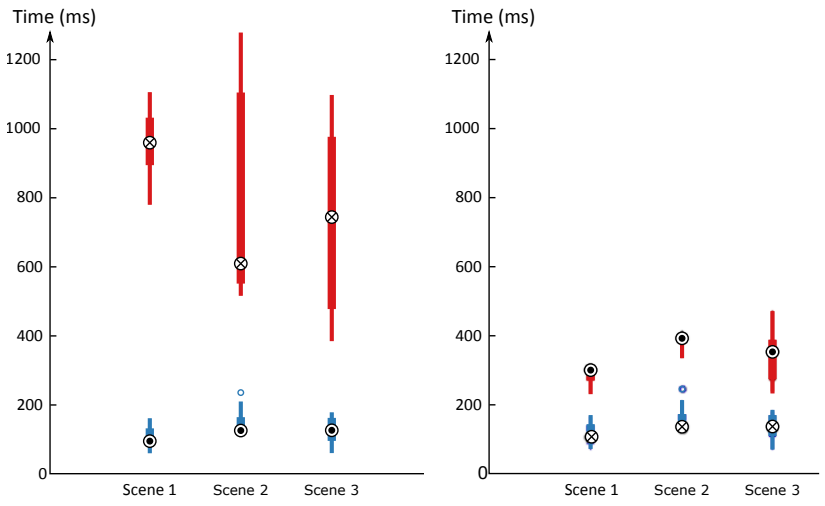


Figure 3.9: The box plots for the computation time of 20 frames for each scenario between two sensor frames (Figure from [ESKD13]). (a) Calculation durations over three scenarios with (Mean: ⊙) and without different Levels-of-Detail (Mean: ⊗) on a PC. (b) Calculation duration with usage of different Levels-of-Detail on a PC (Mean: ⊗) and on a mission computer (Mean: ⊙).

Portability to different sensors

In order to show the portability of our classification system for different sensors, we integrated it to a car-mounted sensor system. The used sensor is the RIEGL VMZ delivered by RIEGL Laser Measurement Systems GmbH. This sensor delivers a 3D point cloud up to 60.000 points per frame with a frequency of ca. 2 Hz with a measurement range of 1.5m to 600m. Beneath these values that have a direct impact to the runtime of our algorithm, it also has differences to the helicopter-mounted LiDAR sensor according the vertical and horizontal scan angle, accuracy and field of view. In Figure 3.10 a mock-up vehicle equipped with such a sensor as well as a classified point cloud is depicted. Since the vehicle does not have any restrictions such as time consumption or memory usage, we used one of the first implementations of our algorithm that had the best performance.

(a) (b)

Figure 3.10: (a) Sensor mounted on the roof of a car. (b) Classified accumulated point cloud. Non-classified points are color-coded from black to red. The color depicts the timestamp when a point is recorded, black points are the oldest while red points are the newest. Blue points are classified as ground.

3.4 Conclusion

The classification process can be motivated for generating terrain models for different purposes. The Laplace-based method, that is deployed within the sensing and visualization system for DVE situations, delivers a model that can be used as basis for obstacle segmentation. This method aims for classification in clearly elevated, but almost homogeneous surfaces like bushes in a reasonable way. In such regions ground is detected, which although allowing for sufficient pre-processing of obstacle classification, was not primarily conceptualized for visualization purposes. For visualizations based upon detected ground by the Laplace method, the pilot himself has to distinguish between ground and vegetation by comparing the real environment with the 3D visualization. In case of the triangulation, our

method contains a major advantage: The cited procedure uses one reference triangle for classification. With the regularized grid, it is possible to classify a point with respect to the neighboring pixels because of the smooth classification membrane. Regularization, used in our approach, improves finding ground in LiDAR data, but does not guarantee to find a completely correct terrain-based classifier. Especially in flights above forests, trees form an almost homogeneous surface and consequently lead to false classifications.

Our approach delivers a more realistic ground classification for visualization purposes. Our classification result has the quality to facilitate a more realistic terrain reconstruction for common scenarios. Additionally, we demonstrated that our technique is real-time capable on mission computers within the time frame of 333 ms. The classified points serve as input for the sensing and visualization system for DVE situations in order to generate a terrain model. We have also shown that it is possible to perform all calculations and visualization in real-time.

Mission computers are limited in their performance, because different applications for pilot assistance run simultaneously and have to work as real-time applications. For the application to be running on the cockpit equipment and sensing infrastructure, together with ground classification and other parts as the obstacle detection and visualization, the requirement is to improve the developed calculation optimization to reduce the need for computation power. Consequently, for some of the calculations, e.g. the very time consuming solving of linear equations, a shader implementation on a GPU could be one possibility for the future.

Furthermore, the use of an offline processed terrain database as initialization for the classification shape could be usable for the iterative variant for the equation solving. A good initialization of the shape grid reduces the number of iterations for the first sensor frame. For the following sensor frames, such an initialization is not needed because the iterative regularization automatically makes use of the results of the previous sensor frames.

4

Clustering of Non-Classified LiDAR Data

After the input data obtained by the LiDAR sensor is classified, obstacles such as trees and power lines can be visualized as symbols. Ground information from the operational method as obtained with our method from Chapter 3 can be rendered as a mesh. Additional information can be calculated on terrain shape such as landing zone symbols and terrain awareness information [Avi11]. Since the sensor also obtains data points that are not labeled by one of the classification algorithms, they are tagged as non-classified. Since this information represents potential dangerous obstacles and may not be deprived to the pilot, it has to be presented in the form of appropriate symbols or visual structures. We divide such non-classified points into clusters, their visualization is based upon convex hulls that are abstracted in order to achieve an appropriate visualization for the pilot. Since the number of such clusters should not exceed the pilot's attention capability, clusters have to be merged and split dynamically during flight. In Figure 4.1 is depicted, where this clustering approach is located within the full processing pipeline.

Based upon *DBSCAN* we propose a dynamic method to cluster spatial superpositioned LiDAR data obtained in short time intervals. A density-based algorithm yields the advantage to detect outliers in the input data that are dispensable for visualization purposes. We use clustering to generate a spatio-temporal stable representation of the underlying data and later temporal stable visualizations of the clusters. In order to merge and split clusters dynamically we make use of Boolean functions. A novel feature of our cluster fusion algorithm is that clusters can be merged differently depending on their context. Clusters on the Field-of-View's borders of the sensor are handled as a logical union while clusters that have completely new sampled information containing non-classified data from the sensor are adapted to existing clusters from previous sensor frames. The practical aspect

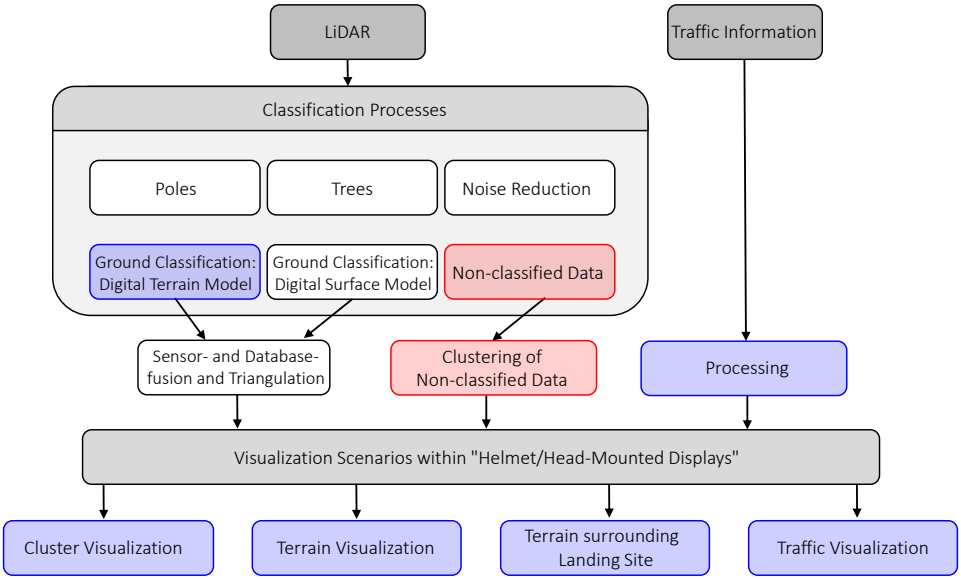


Figure 4.1: Location of the clustering procedure of the non-classified data points, indicated through the red box within the processing pipeline.

for the pilot is the avoidance of drawing all LiDAR points that result in a noisy representation. Our approach is scalable in order to be able to run with very limited resources on avionic computers. In Figure 4.2 we provide a more detailed overview over our clustering system.

4.1 Related Work

Cluster analysis as part of the data mining research field has a large variety of applications and is an important tool to discover knowledge in databases [FPSS96]. It found application in supporting different fields of science such as geology or biology by providing methods for image segmentation, graph mining and other high-dimensional data analysis. These different applications yield varying prerequisites and requirements and therefore numerous algorithms for the identification of clusters in large data sets that have been developed in the last decades [Coe11].

For clustering arbitrary data several methods exist. One of the first and also most common methods is *k-means* [Mac67], a cluster algorithm that requires a specified number of clusters as input. An extension to this algorithm is *x-means* [PM00], which is able to determine the optimal number of clusters for the given data set. As we do not have a priori knowledge about the number of clusters in our data set from the LiDAR sensor, *x-means* would be a suitable option but is limited in its performance. Even more promising approaches are density-based algorithms, which

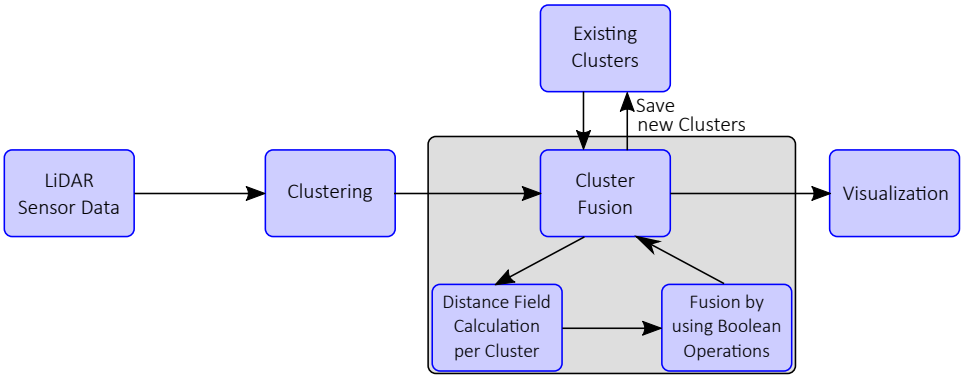


Figure 4.2: System overview. Data from the LiDAR is clustered. The resulting clusters are fused with existing ones by calculating the distance field and using Boolean operations. The result needs to be visualized to the pilot (Figure adapted from [ESKD14]).

became very important in the recent years. The first density-based algorithm *DBSCAN* (Density Based Spatial Clustering of Applications with Noise) [EKSX96] is advantageous of being efficient with an average time complexity of $O(n \log n)$ and being independent from pre-defined number of clusters. An extension of this algorithm called *OPTICS* yields the possibility to detect clusters in data of varying density [ABKS99]. Achtert et al. [ABK06] provide an optimization of *OPTICS* based upon single linkage clustering called *DeliClu* with the possibility to mark noise without using a density estimator.

Following the fact, that our data is not a single point cloud but the sensor delivers multiple point clouds consecutively, we analyzed possibilities for clustering streaming data. One of the first algorithms for stream data ever mentioned is the non-density-based algorithm *BIRCH* (Balanced Iterative Reducing and Clustering using Hierarchies [ZRL96]). This algorithm has the ability to cluster incrementally within constraints like worst case allocation of memory and maximal computation time. Another k-means-based algorithm on data streams optimized for a fixed number of clusters is *StreamLS* introduced by O'Callaghan et al. [OMM+02]. In comparison to *BIRCH*, it needs a lower number of parameters and it performs faster. Additionally, an effective *k-means* algorithm for streams was developed by Ackermann et al. [AMR+12] called *StreamKM++* that improves the clustering quality of *BIRCH* but is significantly slower. In comparison to *StreamLS* the quality is similar but it scales better with respect to increasing number of cluster centers. Aggerwal et al. [AHWY03] present a very effective framework for clustering evolving data streams called *CluStream* by providing an online and offline part for data streams clusterings. The online part is for storing statistics about the clusters periodically while the offline component stores a summary about these statistics. With this method, spherical clusters are generated.

DenStream is the first density-based cluster algorithm for stream data developed by Cao et al. [CEQZ06]. It makes use of micro-clusters of different types within the data set. Core-micro-clusters summarize clusters, potential core-microclusters and outlier micro-clusters are proposed to adapt data to existing clusters or remove clusters. Another density-based approach is *D-Stream* [CT07] that merges data to a grid. By taking advantage of the indexing of grid structure and avoiding neighborhood calculations like with *DBSCAN*, density-based clusters are generated in linear time. Kranen et al. [KABS11] provide with *ClusTree* a clustering algorithm for streams that delivers a clustering model at any time of the streaming. This model is improved whenever computation time allows so and is adapted with new incoming data. Like *CluStream*, *ClusTree* is also divided into an offline and an online component.

Wang et al. [WYWW12] proposed *OPCluStream* to discover overlapping clustering for arbitrary cluster shapes. Their density-based algorithm uses a tree topology to index points that depicts the clustering structure. Another approach using tree structures is the Online Divisive-Agglomerative Clustering (*ODAC*) [RGP08], which uses a correlation-based dissimilarity measure between already analyzed data and incoming data. In contrast to *OPCluStream*, *ODAC* is not density-based. To be able to cluster from different sources like sensor networks, Rodrigues et al. [GRL11] introduce *DGClust*. Within this system, continuous cluster structures are adapted for data of an entire network.

In Table 4.1 an overview of the clustering algorithms of stream data with their underlying cluster method and resulting clustering shape is given.

Table 4.1: Stream analyzing methods with their underlying cluster algorithm and resulting cluster shapes and outlier detection.

Algorithm Name	Cluster Algorithm	Cluster Shape
BIRCH	*k-means*	spherical
CluStream	*k-means*	spherical
DenStream	density-based	arbitrary
ClusTree	*k-means*/density-based	arbitrary
DGClust	*k-means*	spherical
StreamLS	*k-means*	spherical
StreamKM++	*k-means*	spherical
D-Stream	density-based	arbitrary
OPCluStream	density-based	arbitrary
ODAC	*k-means*	spherical/elliptical

For our data from the LiDAR sensor it is important to deal with varying densities in the data set: The density of data points decreases with the increasing distance to the helicopter. As *OPTICS* yields this possibility it is a part of our processing pipeline. The result of *DenStream* is the motivation for our approach.

4.2 Clustering and Convex Hull Calculation

The data we used contains for each data point a classification tag. Tags like “tree”, “wire” or “ground” can be visualized within the synthetic vision system with symbols for obstacles or, in case of ground, as a mesh. The non-classified data also needs to be shown to the pilot. For each of the following steps we use a 2D projection of the sensor data in order to increase the calculation speed. In order to generate a stable visualization of clusters by fusion of clusters, we avoid classical streaming approaches with micro-clustering.

Clustering with *DBSCAN*

With *DBSCAN* a clustering method is proposed by Ester et al. [EKSX96] that divides the input data into different clusters based upon spatial density characteristics. Since it works on densities, it is able to detect clusters with arbitrary, not necessarily convex, shapes. An advantage of this method is that it requires only two single input parameters and no *a priori* knowledge about the number of clusters. The first parameter is the ϵ*-neighborhood* while the second is the minimal number *minPts* of points a cluster has to consist of. With these parameters, the algorithm detects outliers, that have a minimal distance ϵ to the next cluster and the number of points within this ϵ*-neighborhood* is smaller than *minPts*. This functional principle of *DBSCAN* algorithm is depicted in Figure 4.3.

For each point p as part of the database D an ϵ*-neighborhood* is defined as $N_\epsilon(p) = \{q \in D \mid ||p - q||_2 \leq \epsilon\}$. These areas are given by the circles in Figure 4.3(b). The ϵ-neighborhood of a core point ● contains more points q than the ϵ-neighborhood of a border point ●. To add such points as the orange one to an existing cluster, the definition of a **density-reachable** point is introduced. A point q is **directly density-reachable** from a core point p, if it is inside the ϵ-neighborhood of p. In addition, a point q is **density-reachable** from a core point p trough the connection with other core points and these points are **density-connected**.

Combining these definitions, a cluster C is a subset of the database D that satisfies two conditions:

1. $\forall\, p, q$: if $p \in C$ and q **density-reachable** from p, then $q \in$ C.
2. $\forall\, p, q \in C$: p is **density-connected** to q.

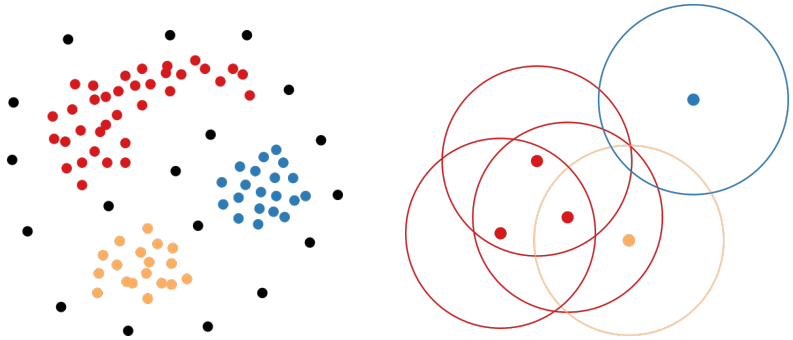


Figure 4.3: Functional principle and result of the *DBSCAN* algorithm on exemplary 2D data. (a) Data points with different colorization of identified clusters (●, ●, ●) and outliers (●). (b) A set of core (●), border (●) and noise points (●) with their circular ϵ-neighborhoods and *minPts* is set to 3.

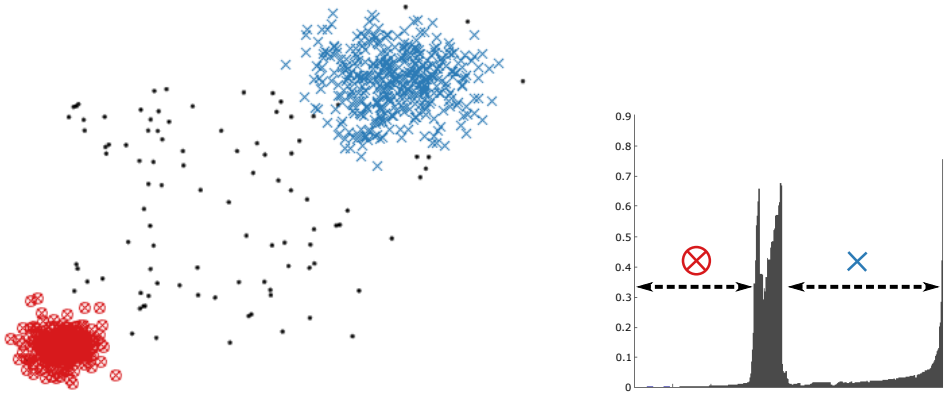
All points of a cluster are **density-connected** and the cluster contains all points that are pairwise **density-reachable** from its members. All data points not being part of any cluster are declared as noise.

Since each point of a cluster is **density-reachable** from all of the cluster's core points, the central idea of the algorithm is to iterate through all non-classified points in the database D and to test if the current point fulfills the core point condition, if the number of points in its ϵ-neighborhood is greater than or equal to the threshold *minPts*. If this test succeeds, the cluster is expanded by iteratively finding the points that are **directly density-reachable**.

Clustering with *OPTICS*

Clustering within non-classified LiDAR data points is necessary, since these points are scattered over the complete data range within a sensor frame. For clustering we use the algorithm *OPTICS*. As density of data decreases in LiDAR data according to the measured distance to the helicopter, *OPTICS* is an optimal choice for clustering because it is able to detect clusters of different densities. To calculate clusters with *OPTICS*, for each data point the core distance is calculated. The core distance is the maximal distance value to the n neighbor points, where n is the minimal number of points a cluster needs to have. Based upon the core distances, for each point the reachability distance to every other point is calculated. This calculation can be improved by binning the data points in a grid. This results in a constant complexity for searching the nearest neighbors. The reachability distance between two points p_1 and p_2 is the maximum between the actual distance and the core distance of p_2 and can be visualized in a 2D reachability plot with the ordered points on the x-axis and the reachability on the y-axis. Parts of that plot that only consist of

low reachability values are called valleys. Since points of the same cluster have a low reachability distance to each other, the clusters appear as valleys in the reachability plot. The density of a cluster increases with the depth of the valley. A sample data set with its classification based upon the reachability plot is given in Figure 4.4. The synthetic data set in 4.4(a) consists of two clusters denoted by ⊗ and ×. In 4.4(b) the clusters are depicted in the reachability plot also by ⊗ and ×.



(a) (b)

Figure 4.4: (a) Synthetic data set with different densities, ⊗ and × denote clusters, while simple dots are not clustered noise. (b) shows a reachability plot, the right side is the valley for cluster labeled with ⊗, the left valley is labeled with × for the second cluster (Figure adapted from [ABKS99]).

Convex Hull Calculation

As a pre-processing step for visualization the 2D convex hull of each cluster is calculated. This shape is used for two purposes. On the one hand it is used to find identical or neighboring clusters in successive sensor frames. On the other hand an abstract visualization for the pilot's Head-Mounted Display is generated on basis of the convex hull. For calculation of the convex hulls we use Graham Scan [Gra72] with a runtime complexity of $O(n \log n)$. The first step in this algorithm is to find the point p_1 with the lowest y-coordinate. Next, the set of points must be sorted in increasing order of the angle they and p_1 make with the x-axis. p_1 and the point p_2 with the smallest angle to the x-axis are points of the convex hull. This has to be repeated with p_2 until the beginning point is reached. The process is depicted in Figure 4.5.

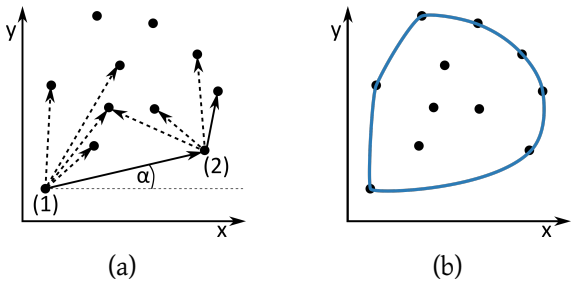


Figure 4.5: Convex hull calculation with Graham Scan (Figure adapted from [ESKD14]). (a) Point (1) is the initial point p_1 with the lowest y-coordinate. The connection to point p_2 (2) is based upon the smallest angle to the x-axis. From (2) the process is repeated until the beginning point is reached. (b) The final convex hull is depicted in blue.

4.3 Overlap Detection and Fusion

The temporal coherence is introduced in two distinct steps: The overlap detection of two clusters and the subsequent fusion step. To find overlapping clusters in an efficient way, we make benefit of using a doubly-linked list as data structure. During the fusion, two overlapping clusters are combined to a new one under certain constraints such as smoothness of the border contour and preserving former shapes of the cluster from previous sensor frames.

Overlap Detection

To find the clusters that have to be fused with our binary union/intersection method, we provide a merging method, based upon a doubly-linked list data structure. Within this list the 2D bounding box is stored. As a pre-processing for the actual cluster fusion with the binary functions, the bounding boxes are pairwise tested for an overlap. If two or more clusters have an overlap, they are merged and the resulting new cluster is pushed to the end of the list. One of the old clusters is erased from the list while the other cluster is replaced by the new one. This reduces the pairwise overlapping test from an $O(n^2)$ runtime complexity to an $O(n \log n)$ complexity, since the comparison of clusters can be continued from the position of the replaced cluster and has not to be started all over again from the beginning of the list. Even if the access to the elements stored in the doubly-linked list is slower than in other data structures that allow data access in constant time such as vectors, the time consumption of the fusion process is decreased. The detection of overlapping clusters as well as the cluster storage is depicted in Figure 4.6. In addition, this approach yields the advantage that even clusters are merged that only have a small spatial distance to each other, if the bounding box are overlapping and the cluster itself do not have any intersections. Therefore, the probability of overseeing an occluding obstacle as well as cluttering is reduced.

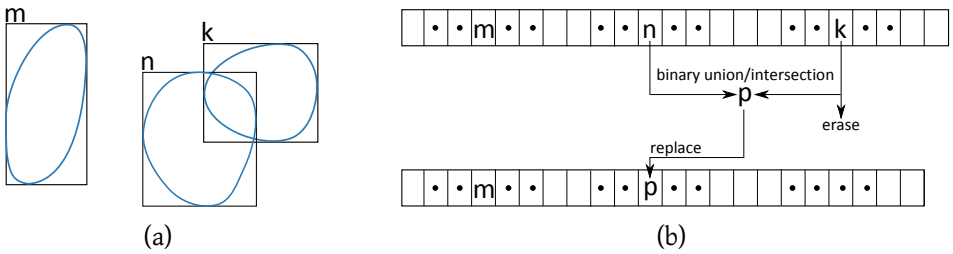


Figure 4.6: Storing and merging of overlapping clusters. (a) Overlapping cluster n and k and non-overlapping cluster m. (b) Storage of cluster information in doubly-linked list. Cluster n and k are fused and the new cluster p replaces n while k is erased. The pairwise comparison of bounding boxes is continued from the position of p.

Cluster Fusion

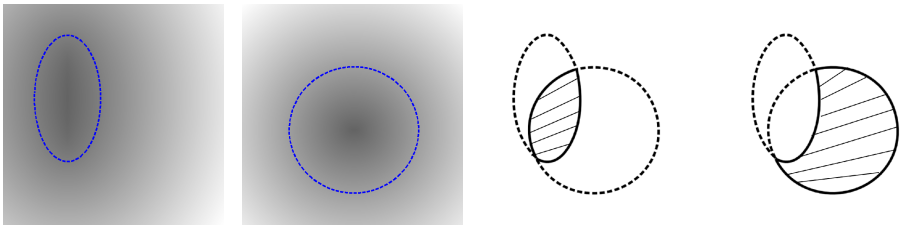
Between two time steps the clustering of the incoming sensor image has to be fused with the existing clustering for which different constraints are necessary. Clusters outside of the sensor's field of view have to be stable. This is important in case that a cluster is only partially within the new sensor frame and the part of the cluster outside the reachability of the sensor should not be touched. The opposite is the case, if both the cluster from the previous frame and the current cluster are within the field of view of the sensor. Then the old cluster has to be adapted to the new one. We achieve that by using R-functions [Rva63] (also called Rvachev functions). With R-functions it is possible to apply Boolean operations to implicit functions. A real-valued function $f(x_1, x_2, \ldots, x_n)$ is a R-function, if its sign is determined by the signs of its arguments x_i. If the sign of a function is considered to be a logical property, negative values of a function correspond with logical false, while positive values correspond with logical true. A R-function works as a Boolean switching function, changing its sign only if its arguments change their signs [Sha91, Sha07]. For example, $min(x_1, x_2)$ is a R-function whose analogue Boolean function is logical and ($\wedge$), and $max(x_1, x_2)$ is an R-function whose analogue Boolean function is logical or ($\vee$). The following equations are examples for the Boolean operators for the Boolean functions *not*, *and* and *or*. The derivation of these equations and the proof that they are equivalent to $min(x_1, x_2)$ and $max(x_1, x_2)$ are given in [Sha91].

$$\text{Logical } not : \overline{x} \equiv -x \tag{4.1}$$

$$\text{Logical } and : x_1 \wedge x_2 \equiv x_1 + x_2 - \sqrt{x_1^2 + x_2^2} \tag{4.2}$$

$$\text{Logical } or : x_1 \vee x_2 \equiv x_1 + x_2 + \sqrt{x_1^2 + x_2^2} \tag{4.3}$$

A convex hull can be converted to an implicit function by calculating the distance field of the discretization of the hull. For the calculation of the distance field we use Felzenszwalb's algorithm [FH04]. The distance field is negative inside of the convex hull and positive on its outside. The resolution of the distance field can be adapted in order to achieve the optimal trade-off between time consumption of the algorithm and clustering quality. Each convex hull can now be interpreted as a function $g : \mathbb{R}^2 \rightarrow \mathbb{R}$. In this way the two functions 4.7(a) and 4.7(b), can be handled as input to a R-function equivalent to the values x_1 and x_2 in the example equations 4.2 and 4.3. In Figure 4.7 Boolean operations for two convex hulls are depicted, including the calculation of the distance field.



(a) Distance field x_1 (b) Distance field x_2 (c) (d)

Figure 4.7: (a) and (b) are distance fields for convex hulls, depicted as dashed line. The distance inside the convex hull is negative and appears black. (c) The union (Equation 4.2) is given as dashed line, the intersection (Equation 4.3) is depicted as solid line. (d) The dashed line indicates the result of union of the ellipsoidal convex hull with the negotiation of the circular convex hull. The solid line shows the other way round. These operations are equivalent to binary operations (Figure adapted from [ESKD14]).

The R-functions allow us to do an initial conservative estimation of the convex hull for a new detected cluster. This makes R-function more flexible than binary operations. For an exact union for example, binary operations would be enough. We use this property to adjust the cluster visualization over time. The adjustment of a cluster representation is an offset to the distance field. To increase the size of the convex hull, a constant offset is subtracted from each value of the field and an offset is added to decrease the convex hull's size. The resulting equation for a logical union is:

$$(x_1+o_1)\vee(x_2+o_2) \equiv (x_1+o_1)+(x_2+o_2)+\sqrt{(x_1+o_1)^2+(x_2+o_2)^2} \quad (4.4)$$

where o_1 is a constant offset used for the older convex hull and o_2 is the constant offset for the newer hull. For fusion of two clusters we apply a conservative Boolean operation and adapt the cluster for multiple sensor frames. This results in a stable representation of the clusters and therefore in a stable visualization. In Figure 4.8 a conservative union ($\vee$) is shown.

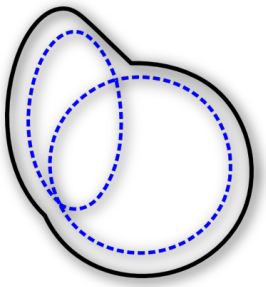


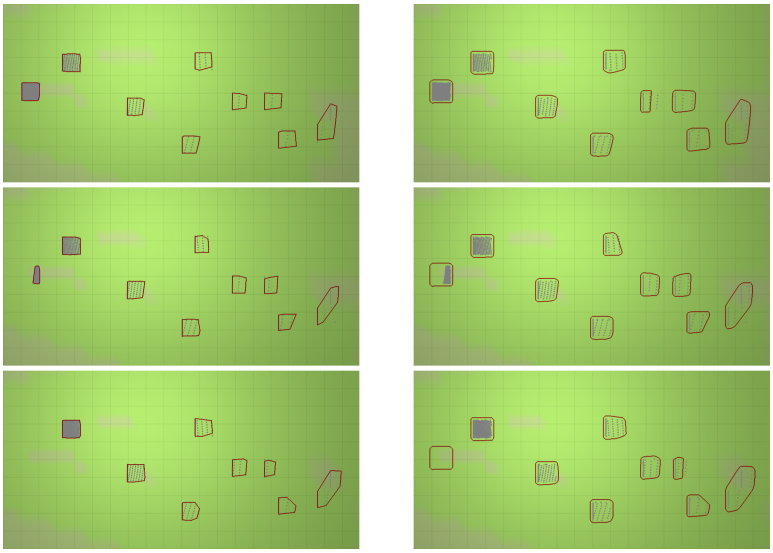
Figure 4.8: Conservative union ($\vee$) of the convex hulls from Figure 4.7. The dashed lines are the initial convex hulls, the solid line is the union of both clusters (Figure adapted from [ESKD14]).

4.4 Results

Our approach with clustering non-classified data points of the sensor frames with convex hull calculation followed by merging clusters is a suitable approach in order to visualize potential dangers to the pilot. *OPTICS* is a fast initial clustering method and provides good results for our purposes with respect to outliers. With the *Graham Scan* algorithm we are able to find a shape of clusters for further analysis and fusion of clusters over time. The fusion of clusters, depending on their spatial position within the sensor's field of view, is through scalability of the distance fields very efficient and yields the possibility to apply different merging approaches, based upon Boolean operations. We considered two different cases where we apply different merging techniques. The first case is that a cluster lying completely within the sensor's field of view appears where we already detected a cluster. Then we merge by applying a Boolean union while reducing the size of the older cluster and a conservative increasing size of the new cluster. If a cluster appears at the border of the sensors' field of view, also a union takes place but the size of the older cluster is not increased. We compared *OPTICS* to our approach with synthetic data that has the same characteristics as the sensor data. The synthetic data is derived from a simulated LiDAR sensor under perfect conditions on a stable helicopter position and without any noise bias in the recorded data. Furthermore, the data has the same spatial and temporal resolution as the real sensor. In Figure 4.9 we applied *OPTICS* and our R-function-based approach to such data.

4.5 Conclusion

In this chapter we presented a memory stable, scalable cluster streaming method optimized for LiDAR data. An initial implementation of *DBSCAN* turned out to be too slow and instable through time. Therefore, we improved the first solution by using



(a) (b)

Figure 4.9: Top-down perspective on not classified data in three consecutive sensor frames. (a) Clustering with *OPTICS* per frame. (b) Clustering with our approach that results in smooth contours and a stable clustering with small changes over time (Figure from [ESKD14]).

OPTICS resulting in a faster implementation but also with the lack of being instable for visualization. With our approach we generated a stable clustering for a data stream from a LiDAR sensor. Additionally, we are able to adapt our technique in such a way that it is real-time capable on mission computers within the time frame of 333 ms, based upon the update time of the LiDAR sensor. All introduced methods, the clustering and the fusion, are scalable in their computation time by reducing the amount of data. Data points in high distance to the helicopter are optionally and not necessary to be clustered. Furthermore, the resolution of the grid used to apply the Boolean operations for the cluster fusion is definable by the user and can be adapted through different platforms. Furthermore, the full cluster process can be designed in such a way that it runs in parallel to the visualization system. The clustering as well as the fusion can be divided into several steps that are executed only if the system can provide enough computation time. The clustering might be calculated during multiple rendered frames on the HMD/HDD by subdividing the point cloud received by the LiDAR sensor. In the same way, the cluster merging can take place only for a specific number of clusters per rendered frame. With our system we provide a new approach in the direction of cluster streaming. Compared to state-of-the-art cluster streaming techniques, we provide a mechanism that allows to extend or shrink cluster shapes according to the R-function used for temporal fusion.

5

Visualization of Meta-Information in the HMD

Information from different sensors such as the LiDAR sensor but also from other information systems such as traffic systems in the form of transponder or ADS-B must be visualized on HDDs or on a HMD. To provide information about non-classified obstacles from LiDAR information, we use cluster information that can be accessible with the clustering approach given in Chapter 4. Based upon this cluster information, we implemented and tested different meshing algorithms as well as other spatial representations. We also applied a modified *DBSCAN* cluster system on traffic information in order to render aircraft and grounded vehicle movement on the HMD in combination with a typical Head-Down Display. Additionally, we provide an abstract terrain information system for areas surrounding the landing symbology with the use to either improve the search for an alternative landing site or avoid uneven terrain during the approach procedure. Since renderings on the HMD need to fulfill requirements for not increasing the pilot's workload and de-clutter the display, we provide interpolation techniques in order to provide spatial and temporal stability. In Figure 5.1 is depicted, where these rendering approaches are located within the full processing pipeline.

5.1 Related Work

To increase situational awareness during flight, multiple visualization systems have been developed for different operational aspects since the late 1970s [LJAM00]. These systems provide 3D representations of the outside world, based upon terrain and obstacle databases and extend them by rendering runways and symbolize flight management systems such as tunnel in the sky or landing aid symbologies. Existing systems for such information mainly for Head-Down Displays are pro-

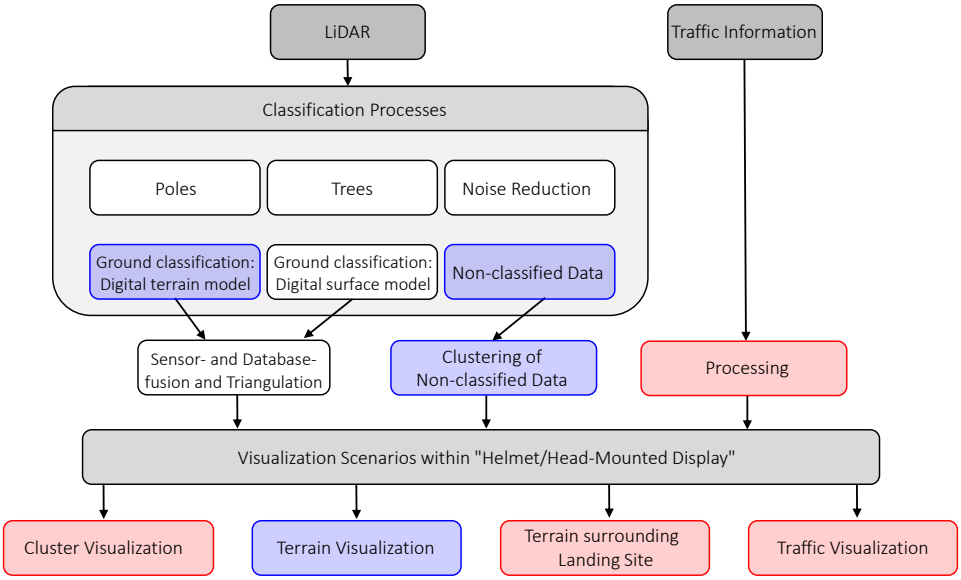


Figure 5.1: Location of the visualization of the non-classified data clusters, traffic information and surroundings of landing areas, indicated through the red box within the processing pipeline.

vided by Garmin with the G1000 system [Garb, Gara], by Rockwell Collins' Pro Line Fusion [Rocb], with the Cobham 3D Synthetic Vision EFIS [Cob] and Honeywell's SmartView Synthetic Vision that allows aircraft the advantage of updated lower landing minimums, going from 200 to 150 feet [Fey15]. An example of an augmented reality systems in the form of Head-Mounted Displays is the Aero Glass system that provides features such as navigation aids, flight plan route and waypoints as well as airways [Aer].

To provide more accurate information about the surrounding environment of the helicopter, these systems use different sensors such as Forward Looking InfraRed (FLIR), RADAR as well as LiDAR systems. The information gathered with such sensors is usually provided in the form of an overlay to the synthetic vision systems and are called *Enhanced Vision Systems* (EVS). For instance, the EVS system provided by Gulfstream is based upon an image, recorded by a FLIR camera, as a 3D conformal overlay to the outside scenery onto the Head-Up Display [Gul]. For other sensors such as a millimeter wave radar, DARPA provided funding in 2007 for the development of an EVS optimized for helicopter landing approaches in deserts called "Sandblaster" [Nat]. For fixed-wing aircraft, Rockwell Collins provides the infrared-based system EVS 3000 [Roca] and Astronics the Max-Vis system [Ast]. Since the data delivered by these sensors is not geo-referenced, it is only possible to provide a head-up visualization of the EVS system. Geo-referenced information accessible in 3D can be provided in the form of a head-mounted visualization.

To render the full information recorded by such sensors yields the disadvantage of providing more information than actually needed. This does not fulfill the requirements of head-up see-through symbologies. With the data analysis methods provided in Chapter 4 we are able to reduce the amount of data given by the LiDAR sensor and air traffic systems. These pre-processing steps are used as a basis for presenting 3D conformal representations on a HMD for non-classified obstacles as well as abstract representations of traffic and terrain features that fulfill the requirements for an improved reduction of workload.

5.2 HMD-based Cluster Visualization

For the generation of meshes or other spatial representations for point clouds there exist several methods. We compared 2D convex hull representations generated with the Graham Scan [Gra72] as introduced in 3.2 as well as 3D representation techniques such as skeletons [HWCO+13] and meshing techniques. The meshing techniques we evaluated are *Thin-Plate Splines* (TPS), *Algebraic Point Set Surfaces* (APSS) and *Robust Implicit Moving Least Squares* (RIMLS) [DB02, GG07, OGG09]. Our final solution is based upon the voxelization proposed by Mehra et al. [MZL+09] combined with interpolation and an additional simplification technique, based upon an Eigenvector representation. We show the behavior and results of the different approaches on the sample cluster scenery given in Figure 5.2.

Representation techniques

2D Convex Hull:
To create a convex shape representation of a cluster, the LiDAR points are projected into the 2D screen space and the convex hull is generated by the Graham Scan [Gra72] as described in Chapter 4. Since this approach is efficient and the number of LiDAR points per cluster is limited, the generation of the hull is fast and can be calculated at interactive frame rates. Even if this leads to a smooth visualization, this representation has the disadvantage that perspectively overlapping hulls with large spatial distances cannot be divided into foreground and background and thus culling is not possible. Moreover, the 2D rendering is not a 3D conformal representation of the LiDAR data, since it yields no depth information about the real object. The convex hull or in a simpler case an ellipsoidal form is more useful for abstract data such as traffic information in the surroundings of the helicopter. An example of what this representation looks like is given in Figure 5.6(a).

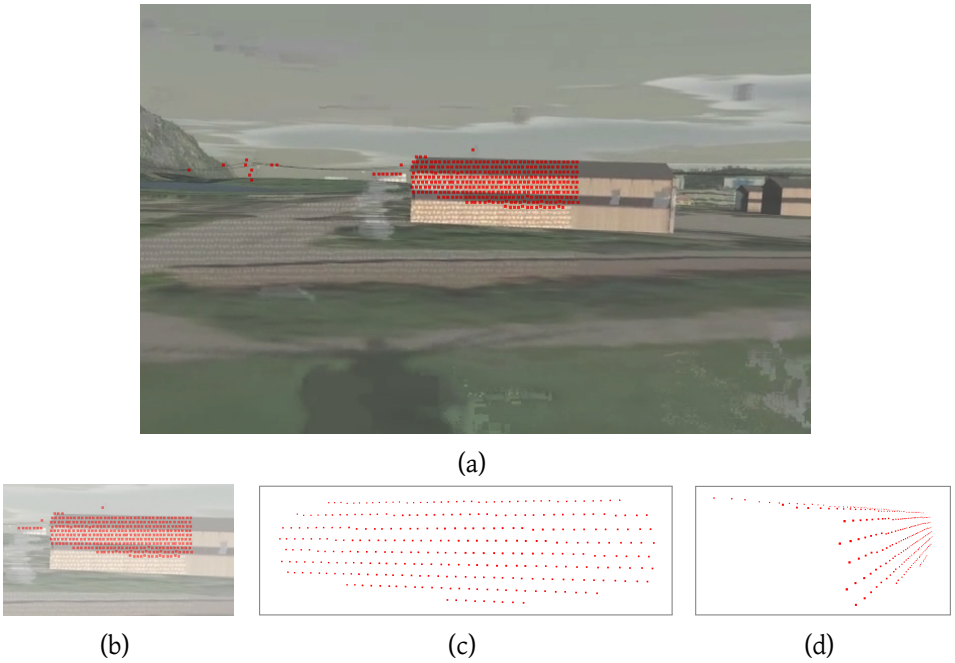
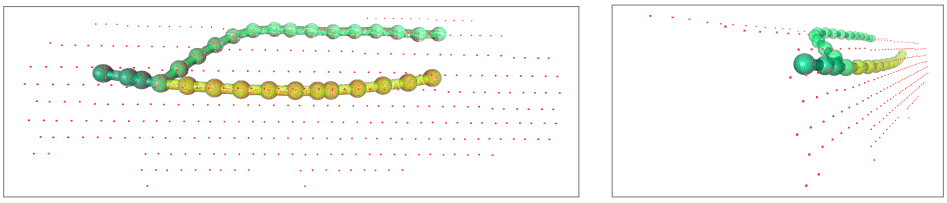


Figure 5.2: (a) This scenery is used for the analysis of each of the cluster visualization techniques. The red points sampled on the house surface are used as input for all meshing methods. (b) Screen capture from the flight simulator with sensor overlay. (c) Clustered data sampled from the house surface rendered from front view. (d) Same data rendered from side view.

Eigenvector representation:
In order to generate a representation with minimal drawing effort, we stylize the LiDAR data in a cluster by performing a principal component analysis (PCA) and rendering the two horizontal eigenvectors representing the maximal spatial dimension at the top of a cluster with an additional vertical line representing the height of a cluster. Since this representation only comprises three lines, it yields the possibility to calculate a linear interpolation between two cluster updates at interactive frame rates. This results in a smooth representation without sudden changes when a cluster is updated. A sketch of how this representation appears on a HMD is given in Figure 5.6(b).

Skeletonization:
The idea behind skeletons for representing LiDAR data clusters is to render only one specific 3D line that encodes the spatial dimension of the object. As can be seen in Figure 5.3, the skeleton does not fulfill this requirement. Moreover, it emerged that the skeleton is instable due to small changes in the point cloud that evolve in our data automatically over time due to inaccuracies of the sensor itself as well

as the changing position of the helicopter. This makes it difficult to interpolate between two successive skeletons in order to generate a smooth visualization. An example how this representation is rendered on a HMD is given in Figure 5.6(c).

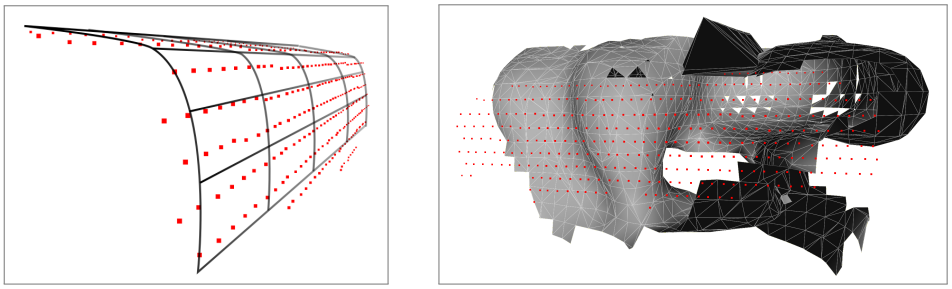


(a) (b)

Figure 5.3: 3D skeleton representation on clustered LiDAR data. (a) Front view. (b) Same skeleton from side view.

Thin-Plate-Splines:
Since the LiDAR points are sampled from a surface, a mesh representation can be derived by generating an interpolation between them. To generate such a mesh, we use *Thin-Plate-Splines* which are similar to the regularization on elevation data described in Chapter 3 [DB02]. This optimization results in a shape approximation of a given input point cloud. A result of applying TPS to a LiDAR cluster is given in Figure 5.4(a). Since the calculation of the TPS is not usable in our setup due to time consumption, depending on the resolution of the output mesh, we improved this method by combining it with the skeletonization process: every point from the cluster is assigned to a joint of the skeleton to form a new sub-cluster. For each of these sub-clusters the TPS is calculated. This results in an approximated shape representation with multiple patches as given in Figure 5.6(d). Similar to the skeleton representation, the TPS is also instable to small changes in the point cloud, but this effect can be reduced by accumulating new patches over time. Even with a culling optimization this results in a time-consuming and cluttered rendering with a lot of geometry. The resulting meshes for multiple sub-clusters for one LiDAR cluster are depicted in Figure 5.6(c).

APSS/RIMLS:
Efficient methods for meshing are *Algebraic Point Set Surfaces* [GG07] and *Robust Implicit Moving Least Squares* [OGG09]. Both of them have a lower time complexity compared to TPS, but they cannot deal with low sampled input. Given that the output APSS and RIMLS often look like the shape given in Figure 5.4(b) that is not representing the objects' essence properly. An example of how this mesh representation looks is provided in Figure 5.6(e)

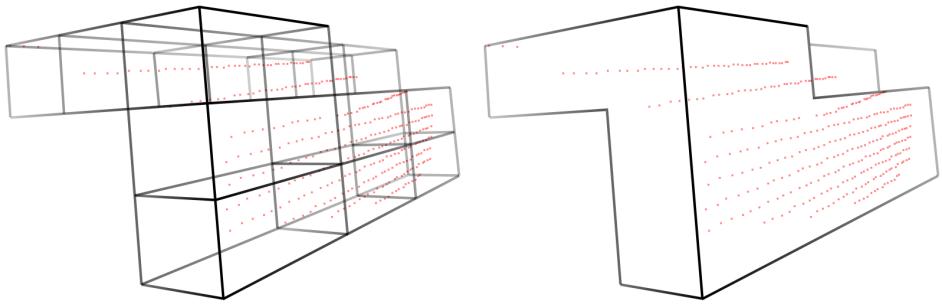


(a) (b)

Figure 5.4: (a) TPS applied to one LiDAR cluster. We used one TPS curve for the entire cluster in order to create the surface. (b) APSS technique applied to one LiDAR cluster, which results in a shape that is not representing the sampled surface of the object properly. We used a higher resolution than in the flight simulator to gain a more detailed result.

Box Mesh:

We introduce a meshing and rendering technique optimized for HMDs in order to compensate the disadvantages of the methods mentioned above in terms of the amount of geometry and time consumption, while keeping their advantages of representing the characteristic shape of a sampled object with our LiDAR sensor. This method is based upon the voxelization process, which is part of the man-made shape abstraction technique proposed by Mehra et al. [MZL+09]. Every point of a cluster is mapped to a $n \times n \times n$ sized cell grid. Each of the cells is rendered in the form of a wire box, if the cell contains at least m points of the cluster. A schematic principle of the mapping of the LiDAR points to the cells in the grid as well as the reduction of rendered lines is given in Figure 5.5 while its exemplary result is compared to the other representation techniques in Figure 5.6(f).



(a) (b)

Figure 5.5: Principle of mesh generation, based upon LiDAR point clusters: (a) Sensor data is mapped to a $3 \times 3 \times 3$ box grid. Every box that contains at least a user-defined number of LiDAR points (here 1) is rendered. (b) Common box faces of neighboring cells are removed and the backface is culled.

The reduction of lines is based upon two steps:

- If two neighboring, rendered cells have a common box face, the edges of this face will not be rendered.
- The rendering process is divided into two steps: first, the solid boxes are rendered to the depth buffer. In the second render step, the edges of the faces are rendered, if they have the same depth as in the depth buffer. Accordingly, we provide a backface culling on non-solid primitives.

Our approach yields the advantage of using a constant amount of geometry that can be defined by the user. The only thresholds are the resolution of the grid and the number of points, which each grid cell has to contain to become visible. Additionally, this data structure allows a temporal interpolation in a similar way as it is possible with the eigenvector representation mentioned above.

Representation Results and Comparison

We applied all of the described algorithms within our simulator environment to the point cluster given in Figure 5.2. The resulting renderings are depicted in Figure 5.6. The advantages and disadvantages are summarized in Table 5.1.

Table 5.1: Advantages and disadvantages of different cluster representations.

Type	Advantage	Disadvantage
2D Convex Hull	interactive frame rate	not 3D conformal
3D Eigenvectors	interactive frame rate, low amount of geometry	difficult for object recognition
Skeletonization	useful as pre-processing	instable result
TPS	smooth shape representation	high computation duration
APSS/RIMLS	interactive frame rate	instable result
Box Mesh	interactive frame rate, low amount of geometry	

For our final representation on the HMD as well as on the HDD, we render the eigenvectors for high distances as well as box meshes for a more accurate representation in the direct surrounding of the helicopter. Examples of result renderings in the HMD and HDD are given in Figure 5.7. Within this Figure the meshing procedure for real LiDAR data is depicted as well as the application of this rendering system within our flight simulator. We implemented a combination with the eigenvector representation by using the distance-based Level-of-Detail approach for different navigational terms as described in Chapter 2. For Figure 5.7(c) and

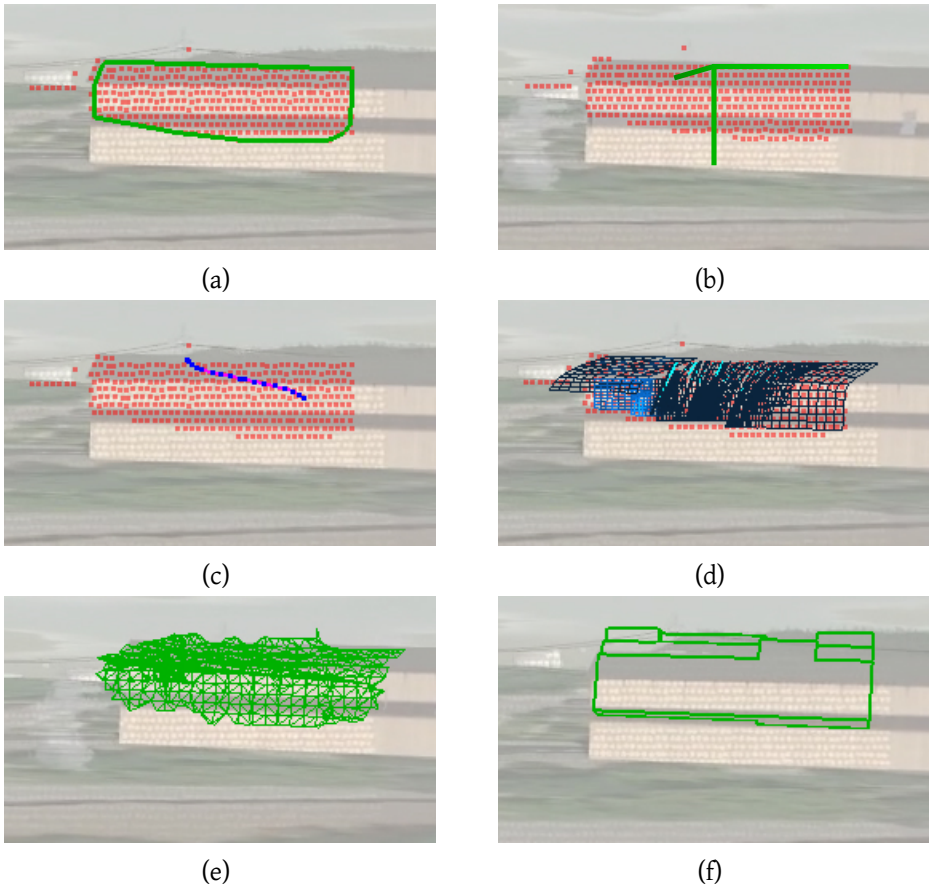
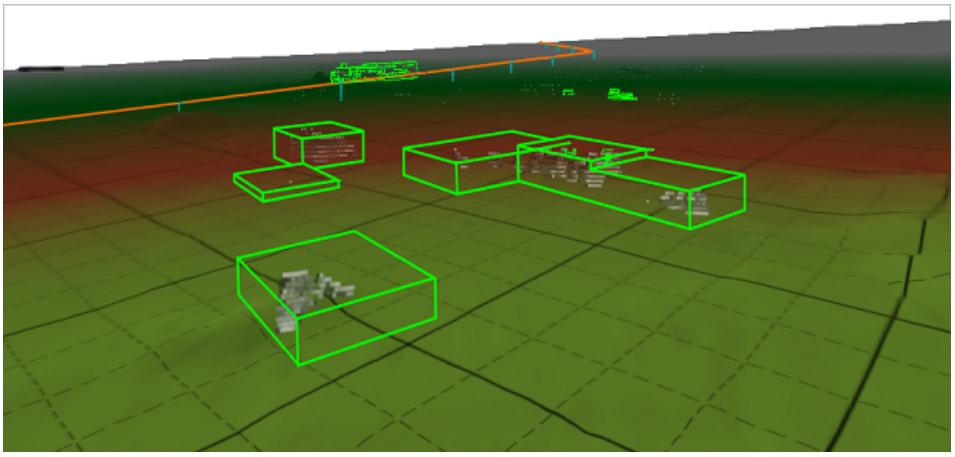
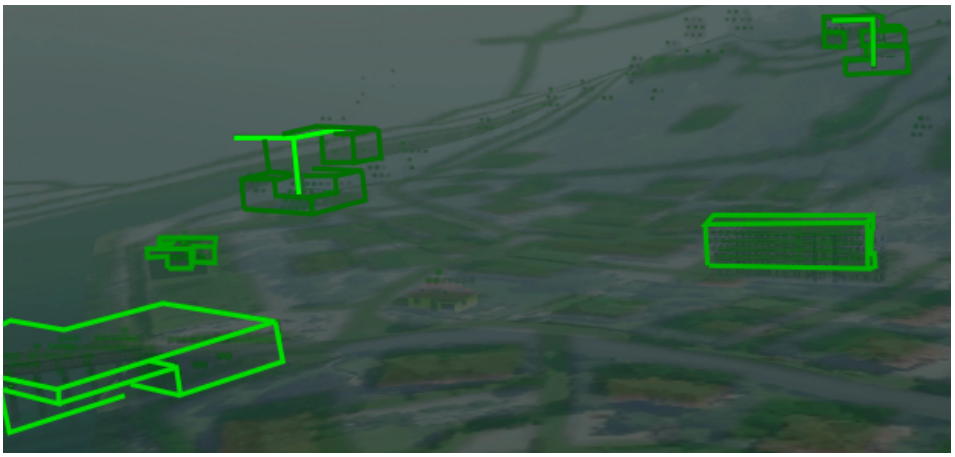


Figure 5.6: Intermediate results for generation of 2D and 3D representations of clusters: (a) 2D Convex Hull, (b) Eigenvectors, (c) Skeletonization, (d) Thin-Plate-Splines applied on sub-clusters, (e) APSS, (f) Box Mesh.

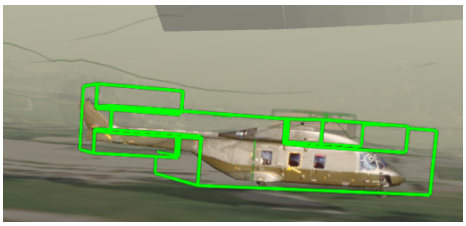
5.7(d) the impact and quality of such a meshing method becomes visible since a representation of the shape of a helicopter can be achieved by such a basic meshing and remains recognizable during DVE conditions such as brownouts. Additional results provided in our flight simulator are given in Appendix B.1. We also considered rendering techniques used for uncertainty areas used in the field of geovisualization in order to provide more detailed hints about the LiDAR data that is the basis information for the resulting mesh representation. Therefore, usual techniques that can be used are contouring, adjacent maps and symbols for georeferenced information rendering but also there exist visualization techniques for statistical information such as error bars [SMJS14]. Such renderings are not sufficient for see-through displays because of the danger of cluttering. The rendering possibilities for statistical information also yield the disadvantage of providing too abstract information that must be reinterpreted by the pilot for spatial interpretation of



(a)



(b)



(c)

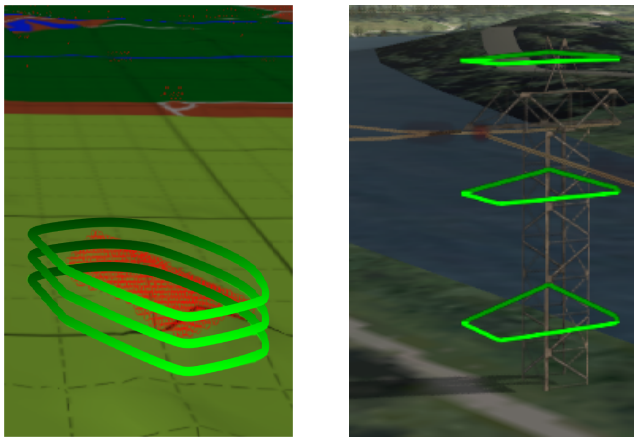
(d)

Figure 5.7: Box meshing on real data and in simulator environment. (a) Head-down view with only boxes on real non-classified LiDAR. (b) Box renderings with additional distance cue on a HMD: Objects far away are faded out and replaced by a symbol representing the principal components of the cluster in a simulated DVE scene. (c) Data sampled from the surface of a helicopter are meshed during full visibility. (d) Same helicopter from (c) during a simulated brownout condition.

obstacles. Techniques that make use of transparent renderings and also have the potential to represent spatial information are Bollinger Bands that are common in financial data analysis [SSW+12]. These Bollinger Bands can also be extended for usage in 3D environments such as Synthetic Vision Systems. Volume renderings can also be used for depicting uncertainty in 3D applications and could also be used for HMD visualizations [DKLP01]. For both, the Bollinger Bands and volume renderings, the computation complexity might be too high for the use on helicopter hardware and also need to be evaluated according to the pilots' needs and expectations.

Spatial and Temporal Coherence

In order to achieve smooth transitions and guarantee temporal and spatial coherence between different representation forms as well as adapting a representation to new LiDAR data, interpolations have to take place. To achieve the best result according to temporal and spatial coherence, we use the LiDAR points in combination with points sampled from the cluster shape at different height levels resulting from Chapter 4. Since our final selection of representation forms are the box meshes in combination with the eigenvectors, we provide an interpolation procedure designed for these two types.



(a) (b)

Figure 5.8: Convex hull shapes from our clustering approach introduced in Chapter 4. (a) On real LiDAR data. (b) Within our simulator environment.

If the distance decreases from the helicopter to a cluster represented by its eigenvectors, it is faded out while the box mesh is faded in. Since it is only one distance threshold that triggers the fading process, the pilot is able to select a proper value for this threshold. The transition between two representations is given in Figure 5.7(b) (cluster in the background).

Due to the rotation of the principal axis between two cluster updates, the interpolation can take place in the same way as for the eigenvectors. During the rotation process, the cells are filled with different LiDAR points because the point cloud itself is not rotated. This may result in popping artifacts, if a box is rendered during one frame and is not rendered in the following frame. This can be avoided by recalculating the mesh at every interpolation step. During this recalculation, an lifespan for every box is defined, counting how often a box is rendered as well as a box is not rendered, beginning from the first interpolation step. This frame-wise interpolation is possible, because the calculation of the box mesh is very efficient. With increasing lifespan, it becomes more opaque and with decreasing lifespan, it becomes more transparent. This lifespan regulates the transparency of the box edges in order to avoid popping artifacts. The concept of this procedure is given in Figure 5.9.

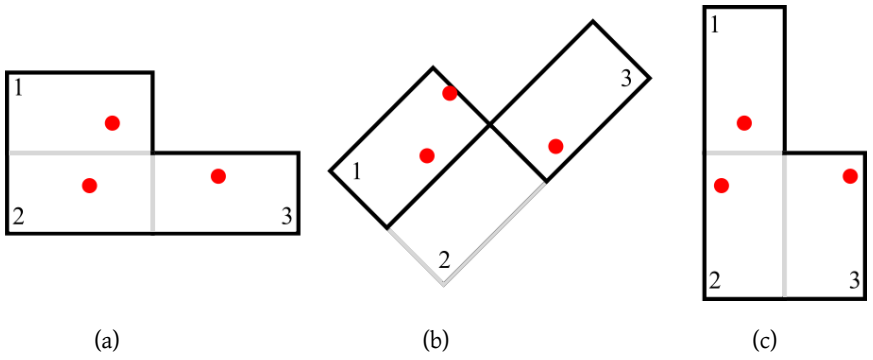


Figure 5.9: Principle of a box mesh rotation around 90° in 2D with smooth edge fading. (a) Initial situation. (b) Box 2 is faded out, since it does not contain any data point. (c) For the final rotation step, box 2 is faded in again, since it contains again a data point.

We introduce a quality measurement in order to keep the number of non-classified shapes at a reasonable amount to avoid cluttering. With this measurement we are also able to avoid an increase of computation speed and memory consumption resulting from the data structure for storing the clusters presented in Section 4.3. This quality measurement can be used for two application cases:

- Change of the cluster appearance: if the clustered LiDAR information is unstable, it makes more sense to render the accumulated point cloud rather than rendering a permanently rotating and changing mesh.

- Reduce of computation time and memory usage: we reduce the number of clusters that have to be compared as pre-processing of the cluster fusion by eliminating clusters with low quality outside of the range or Field-of-View of the LiDAR sensor.

Table 5.2: Parameters for box shape detection and rendering.

Parameter	Value description
Box resolution	For the number of voxels per box shape, we use a grid with the size of at least $2 \times 2 \times 2$ up to $4 \times 4 \times 4$
Fading thresholds	This threshold regulates at which distance, the axis lines are faded out and the box shape is faded in (and other way round). We use 300 Meters as an example value.
Alternative styles	We also allow the possibility to render all styles given in Figure 5.6.
Interpolation speed	Threshold that regulates the interpolation speed between two sensor frames. This threshold is scaled with the distance to the helicopter and its flight speed in order to have a faster interpolation for shapes next to the helicopter.
Visibility	The necessary number of points that needs to be within a voxel. We set this value to 1 (minimal value).
Accumulation	Binary value that is set to *true*, if the LiDAR points are accumulated over multiple sensor frames within the box voxels. If the accumulation value is set to *false*, the box is only updated with the newest sensor points.
Quality threshold	If the measured change of a box representation between multiple sensor frames is higher than the quality threshold, it will not be saved for visualization, if it lies outside of the sensor range. If the accumulation threshold is set to *true*, the error is estimated with Equation 5.2. Otherwise Equation 5.1 is used without taking previous errors into account.

The average quality measure e per box voxel $P_k^t = \{p_0^t, p_1^t, \ldots, p_7^t\}$ is defined as

$$e(P_k^t) = \frac{\sum_{i=0}^{7} ||p_i^{t+1} - p_i^t|| \cdot \kappa_i^t}{8}, \tag{5.1}$$

with k as index of a voxel within the box shape and t as the time stamp within a temporal sliding window. The definition of a voxel is based upon its corner positions $p_{\{0,\ldots,7\}}^t$ in Euclidean coordinates. The transparency at which a voxel is rendered is given with $\kappa_i^t \in [0, 1]$. Using Equation 5.1, the full quality measure E over all box voxels within a time window is defined as

$$E = \sum_{t=0}^{T-1} \sum_{k=0}^{K} e(P_k^t), \tag{5.2}$$

where $t \in \{0, \ldots T\}$ and $k \in \{0, \ldots K\}$ have the same meaning as in Equation 5.1 and represent the T considered time frames as well as the K voxels within a

box shape. At initialization of the box mesh, the error is defined as *infinity* and optionally, the LiDAR point cloud is rendered. A full set of parameters used for the box representation is given in Table 5.2. It is important to mention, that these parameters should only be accessible during pilot workshops to evaluate, which parameters are most reasonable for an operational use. Only the interpolation speed parameter might be accessible in order to provide an optimal look and feel for the pilot.

5.3 HMD-based Traffic Visualization

With "Next Generation Air Transportation System" (NextGen) and the "Single European Sky ATM Research" (SESAR), a new aeronautical surveillance system called Automatic Dependent Surveillance-Broadcast (ADS-B) has been introduced [Fed14, SES]. As part of this system the precise determination of the position, speed and other supplemental information of aircraft, ground vehicles or ships is possible, which is then broadcast to be received by air traffic controllers and other vehicles in the surrounding such as the helicopter that we use as a target system for a traffic rendering system.

We make use of the benefits of these two technologies by visualizing received ADS-B data of traffic surrounding the helicopter on an HMD. Such a system will assist a helicopter pilot flying in offshore windparks or taking part in search and rescue operations where many other vehicles operate within a small range. Example situations are given in Figure 5.10.

(a) (b)

Figure 5.10: Exemplary scenarios where a Head-Mounted Traffic Display increases the situational awareness. (a) A close formation of Coast Guard helicopters [Lag]. (b) Two helicopters operating inside an offshore windpark [Rad].

To conform with the requirements of head-mounted see-through symbologies, we develop an integrated traffic visualization concept: a Head-Down and a Head-Mounted Display complement each other in their roles so as to combine the advantages of both representations. For the synthetic ADS-B input data, we adapt

DBSCAN described in Section 4.2 to analyze the high-dimensional traffic data in order to identify groups of similar traffic that can be visualized by a single symbol in the HMD. This cluster information is used to avoid display clutter by decreasing the number of rendered symbols. In addition, we present a special symbology that offers supplemental information about the traffic group. This includes parameters such as the number and type of the contained vehicles as well as a measure of the formation's threat potential, based upon the predicted closest point of approach. To further decrease the pilot's workload by avoiding distracting changes in the visual representation of clusters, a smooth transition visualization algorithm is developed for splitting and merging clusters in 2D.

Traffic Data System

Automatic Dependent Surveillance-Broadcast:
According to the ICAO Doc 9924 Standard [ICA10], an Aeronautical Surveillance System "provides the aircraft position and other related information to Air Traffic Management (ATM) and/or airborne users". In its basic realization, it provides only the aircraft's position within defined time intervals. More sophisticated designs provide additional information on the identification such as speed, the intent and other characteristics of the aircraft. Depending on the technologies used, it can be distinguished between three major types of aeronautical surveillance systems: Independent Non-Cooperative Surveillance, Independent Cooperative Surveillance and Dependent Cooperative Surveillance [EUR].

While Independent Non-Cooperative Surveillance systems do not rely on the co-operation of the aircraft, which means that an aircraft does not have to be equipped with any on-board surveillance equipment to be detectable, Independent Cooperative Surveillance systems communicate with the aircraft to be identified. Therefore, it is possible to request supplemental data such as the identification or the current airspeed. To do so, each aircraft must be equipped with a radio receiver and transmitter, referred to as transponder. Examples of independent cooperative surveillance technologies are Secondary Surveillance Radar (SSR), Wide Area Multilateration (WAM) and Multi-LATeration (MLAT) [EUR], [Vab]. Table 5.3 gives an overview of state-of-the-art aeronautical surveillance systems.

The ADS-B system we require for our traffic visualization approach is a Dependent Cooperative Surveillance system. This kind of surveillance systems depend on a system that enables the on-board determination of the vehicles position such as a GPS receiver. This spatial information is then extended with additional data for broadcast. The data provided by the ADS-B system and used for our visualization is the identification ID, position, velocity, air/ground state and heading of the vehicle.

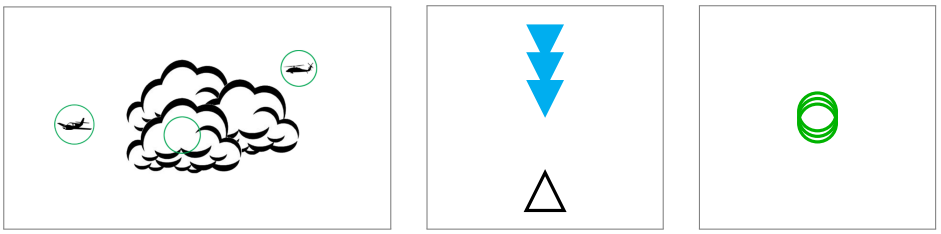
Table 5.3: Overview of aeronautical surveillance systems [EUR].

Category		Technology
Independent	Non-Cooperative	Primary Surveillance Radar (PSR) Multi-Static Primary Surveillance Radar (MSPSR)
	Cooperative	Secondary Surveillance Radar (SSR) (Mode A, C, S) Wide Area Multilateration (WAM) Multi-LATeration (MLAT)
Dependent	Cooperative	Automatic-Dependent Surveillance Broadcast (ADS-B)

Traffic Alert and Collision Avoidance System:
TCAS is an Airborne Collision Avoidance System that works independently of airborne navigation equipment and ground-based sensor systems, solely based upon the direct communication of aircraft via their transponders. The central TCAS unit processes all responses and applies a special logic to the classification of each intruder into three categories: *Proximate Traffic*, *Traffic Advisory (TA)* and *Resolution Advisory (RA)*. The remaining aircraft are declared as *Other Traffic*. According to this classification, an intruder is highlighted on the HDD, triggers an aural annunciation (TA) or even initiates to display instructions on how to solve the conflict (RA). The TCAS threat detection is based upon the warning time τ, which is an approximation of the time to the Closest Point of Approach (CPA) between ownship and intruder. The lower this measure, the higher is the threat potential of the target aircraft. The full TCAS description as well as the definition of the different dimensions of the spaces for RA and TA and the threat resolution logic is given in the ACAS Guide [EUR14].

Overview

For the visualization of our cluster approach, we use combined renderings on a Head-Mounted Display as well as on a Head-Down Display. Vehicles in the surrounding of the helicopter are displayed in the form of a symbol superimposed on the real aircraft in the outside view. In Figure 5.11(a) such a visual-conformal symbology is sketched that highlights other aircraft with a framing circle. Since it is sufficient for the HDD to provide one symbol for every vehicle (Figure 5.11(b)), such a visualization could lead to cluttering effects in the HMD (Figure 5.11(c)).



(a) (b) (c)

Figure 5.11: Illustration of a possible traffic situation causing overlapping symbols on the HMD (Figure adapted from [EES+15]). (a) A basic traffic visualization on the HMD - visible and invisible traffic highlighted with circles. (b) Top down view on the HDD. (c) Overlapping symbols on the HMD.

We use the HMD as well as the HDD in order to make use of their complementary roles for developing an integrated traffic visualization concept:

Head-Down Display:

- Provides an overview of the traffic situation, from which the pilot can estimate horizontal distances and tracks of other aircraft.
- Incorporates an interface through which the pilot can select nearby aircraft so as to receive additional information and highlight them in the head-down and in the head-mounted visualization.
- Adapts the aircraft symbol according to the threat potential of the intruder.

Head-Mounted Display:

- Supports locating of relevant traffic.
- Includes as much important information as possible in order to decrease the pilot's head-down time, but not more than needed to avoid cluttering.
- Allows the pilot to de-clutter the display by excluding irrelevant traffic.
- Applies a clustering algorithm and an advanced symbology in order to avoid overlapping traffic symbols and to densely pack available traffic information.

System Architecture:
The ADS-B traffic visualization software is designed as an add-on to the synthetic vision system [SHV13]. The ADS-B traffic visualization implementation comprises five major modules: Traffic Input, Traffic Processing, Head-Down visualization,

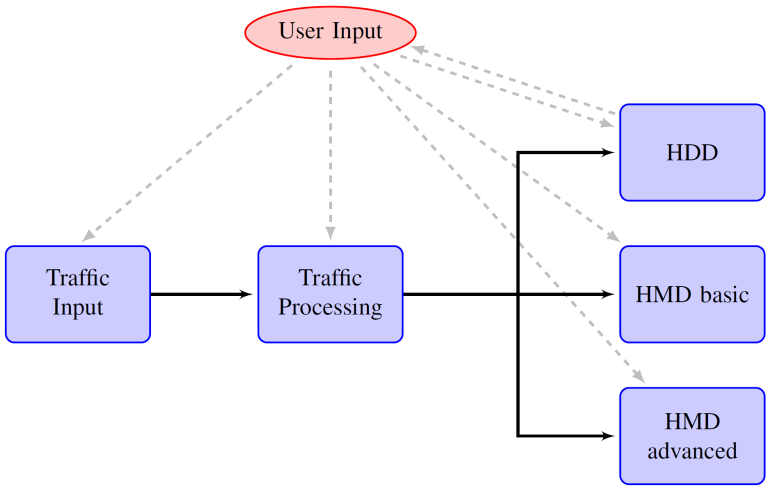


Figure 5.12: The system architecture of the ADS-B traffic visualization software (Figure from [EES+15]).

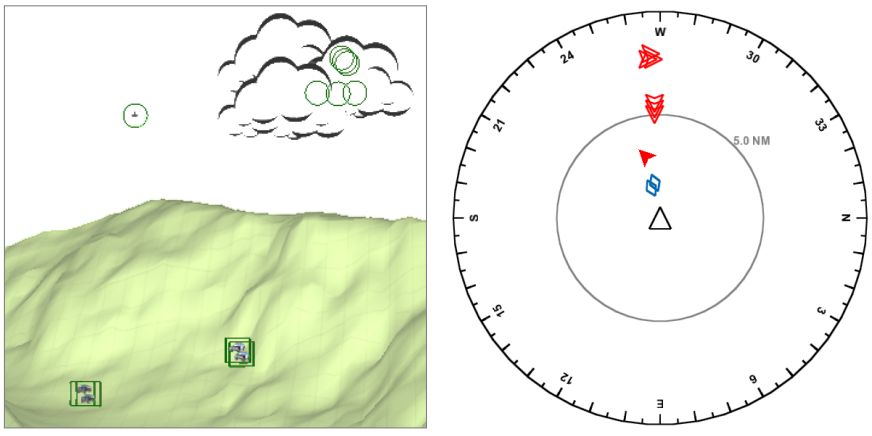
basic HMD rendering and advanced HMD visualization. The relations between these subsystems as well as an additional user input are depicted in Figure 5.12.

The Traffic Input module receives and converts ADS-B messages broadcast by surrounding aircraft and ground vehicles. One current ADS-B message for each traffic participant is generated and sent to the subsequent module. Within the Traffic Processing subsystem, the received ADS-B messages are further processed and the parameters required by the developed traffic displays are derived. The output comprises all quantities included in the synthetic ADS-B data as well as additionally computed parameters such as the distance to the ownship. The values can either be based directly on last message received or be derived from an interpolation between the last two stored messages. An interpolation of these values yields the possibility of a smooth visualization. For different displays types, varying visualizations are provided: The HDD traffic visualization is inspired by current Cockpit Display of Traffic Information designs, with additional features so as to support the concept of the integrated head-mounted and head-down traffic visualization. For the HMD we implemented a straight-forward traffic visualization approach as given in Figure 5.11(a) and 5.11(c) as reference for our more sophisticated visualization. This more advanced HMD subsystem is the key part of our work. This advanced HMD subsystem incorporates all processing required to realize a head-mounted traffic symbology that presents valuable information to the pilot. This includes the clustering as well as the generation of a de-cluttered, smooth traffic representation. The user is able to adapt the parameters for the Traffic Processing as well as the configuration for the visualizations.

Advanced Traffic Visualization in Head-Mounted Displays

This section describes the development and implementation of the advanced traffic visualization in HMDs. As stated above, this module is integrated with the head-down traffic display. Recalling the role definition of the head-mounted module within the overall traffic visualization concept, the central requirement is that the advanced traffic visualization in Head-Mounted Displays should include as much important information as possible in order to decrease the pilot's head-down time, but not more than needed to avoid cluttering. Thus, the definition of an advanced visualization strategy is started with the identification of methods for de-cluttering the HMD while presenting all relevant information in a way that conforms with the requirements of head-mounted see-through symbologies.

De-Cluttering of the Display:
Rendering the received position data of all surrounding ADS-B-equipped vehicles can lead to many overlapping traffic symbols and unnecessary information on the HMD. Figure 5.13 shows such a traffic situation, where 11 other ADS-B equipped vehicles are visible for the pilot: two groups of ground vehicles below, three aircraft flying towards the ownship, three intruders crossing the own flightpath from left to right and one single aircraft departing from the ownship on the left side. For head-mounted renderings, we propose to group intruders that operate close to each other and in a similar way and represent them by a single HMD traffic symbol instead of overlapping icons.



(a) (b)

Figure 5.13: Example of a traffic situation resulting in cluttering (Figure adapted from [EES+15]). (a) Cluttered visualization of two groups of ground vehicles (squares), two groups of airborne traffic and one single aircraft (circles). (b) Head-Down Traffic Display showing a top-down view of the same situation as in (a).

Due to the movement of the own and other aircraft, groups of similar traffic will be split up into new clusters, others will fuse as well as various groups will overlap on the screen occasionally. This emphasizes the need for a strategy for **merging and dividing** different clusters and visualizing that process smoothly for the pilot not to get distracted.

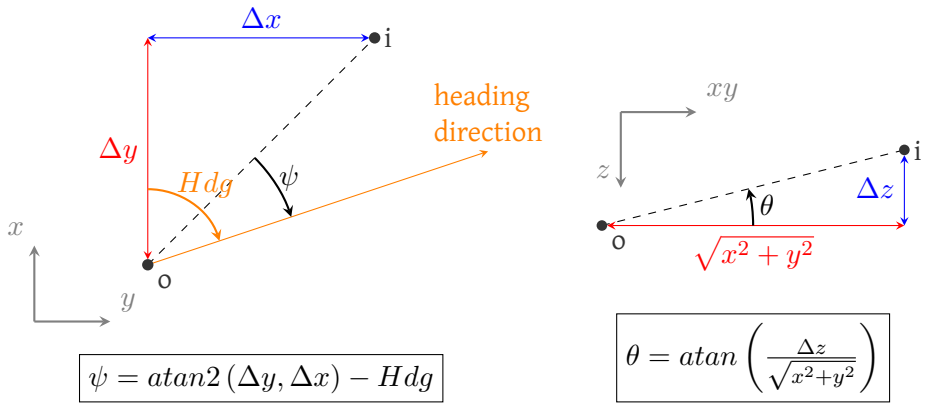
In summary, the realization of a visualization method for groups of similar traffic can be divided into three steps:

1. Identification of traffic groups
2. Graphical representation of these clusters
3. Smooth visualization of cluster fusions and splittings

Clustering to Identify Groups of Similar Traffic:
In general, clustering algorithms try to group a set of patterns, here the feature vector of an aircraft, into different clusters, based upon their similarity. The result is that objects which are similar to each other belong to one cluster while objects that are dissimilar are assigned to different clusters. It can be stated that a group of similar traffic should contain vehicles that operate **close to each other** and **in a flight formation**. Additionally, further investigations revealed that these vehicles must also appear **close to each other on the HMD**. The features we use can be derived from the basic received ADS-B data:

- **Spatial proximity:**
  Geometric position
- **Similar flight formation:**
  Track angle
  Speed
  Air/Ground state
- **Closeness on the HMD:**
  Relative three-dimensional bearing

All aircraft belonging to one group operate nearby the other members of their group. However, this must not be the only feature to be compared because different aircraft formations fly close by each other, for example when their routes cross. Thus, solely spatial proximity does not form a group of similar traffic. Similar behavior of the group members is ensured by clustering vehicles, based upon their track angle and speed. Moreover, the air/ground state broadcast via ADS-B is a suitable feature to separate airborne from surface traffic. Finally, the proximity of the cluster members on the HMD is covered through the relative three-dimensional



(a) Azimuth ψ. (top-down view) (b) Elevation θ. (side view)

Figure 5.14: The calculation of the three-dimensional bearing (ψ, θ) of an intruder i relative to ownship o, based upon their positions (x, y, z) in a world-fixed North-East-Down frame of reference (Figure from [EES+15]).

bearing (ψ, θ). The calculation of this feature, which represents the azimuth and elevation angle of a vehicle as seen from the ownship, is illustrated in Figure 5.14. The features used for clustering can individually changed.

Clustering Procedure:
For the clustering of the vehicles into groups of similar traffic the *DBSCAN* algorithm is adapted. The selected features comprise a binary value, the air/ground state, and additional values with continuous data types measured on different scales, for instance the speed or the track angle. To derive a composite dissimilarity measure, one can normalize all continuous values to a range from 0 to 1 and combine the results, based upon a chosen weighting to one single dissimilarity measure. The Gower similarity coefficient is a well-known example of such a composite measure [Gow71]. Nevertheless, such an approach implicates the additional challenges of how to weight the different features and how to choose a threshold for this complex distance measure.

To avoid these problems, a different idea is pursued by our work. Instead of a single distance, the region query checks the dissimilarity of all selected continuous features i separately against an associated threshold ϵ_i. This means that for the assessment of the spatial proximity, the Euclidean distance between the three-dimensional ownship and intruder positions is compared with the chosen threshold $\epsilon_{Position}$. Likewise, the differences between the track angles, the speed and the bearing angles are checked. Only if every test succeeds, the respective vehicle belongs to the ϵ-neighborhood. To separate airborne from ground traffic, both types of vehicles are clustered individually.

The `minPts` parameter of the *DBSCAN* is set to two, implying that only single vehicles without any other vehicle in their ϵ-neighborhood are labeled as noise. In a post-processing step each of these vehicles receives a unique cluster ID which is a requirement for the input of the smooth transition visualization.

Traffic Group Symbology

In order to represent the identified groups of similar traffic on the HMD screen, we developed a visualization with two-dimensional symbols. These symbols are used to support the pilot in the detection of surrounding traffic by indicating the position of the cluster. Figure 5.15 shows a few approaches which were developed to fulfill this task. Each sub-figure sketches the same traffic situation in window coordinates (X, Y), while the projected positions of the six vehicles, represented by red dots, are framed by a different symbol in each case.

The straight-forward approach in Figure 5.15(a) applies a rectangle enclosing all vehicles. However, the box, which is aligned with the axes of the window coordinate frame, also covers areas where no vehicles reside. The convex hull, sketched in Figure 5.15(b), overcomes this drawback by finding the surrounding polygon, for instance by means of the gift wrapping or Jarvis march algorithm [Jar73]. This results in a great many symbols of various, continually changing shapes, which depend on the number and arrangement of the involved vehicles. Therefore, defined symbols like the ellipse and the rectangle presented in Figure 5.15(c) are favored for the visualization of traffic in HMDs. In order to resolve the problems encountered in the first approach, the symbol is turned in the direction of the principal axes X', Y' of the traffic group. To clearly differentiate airborne from ground traffic, we decided to visualize the former by an ellipsoidal and the latter by a rectangular symbol. As can be seen in Figure 5.15(c), the parameters of the ellipse and the rectangle are defined in the principal coordinate system. This direction, where the data shows the largest variance, can be found by performing a PCA [AW10].

Smooth Merging and Splitting of Traffic Groups:
With the knowledge about the clustering and the group symbology, all methods needed for the realization of an advanced traffic visualization that decreases the occurrence of overlapping symbols are available. By rendering the ellipse or rectangle for each cluster every time, new ADS-B information is retrieved, yields abrupt changes of the traffic symbols, if two or more clusters merge to one single cluster or if a cluster splits up into several smaller traffic groups. This poses the risk of distracting the pilot by unnecessarily drawing his attention to these events. To avoid these undesired sudden changes of the cluster representation, we present

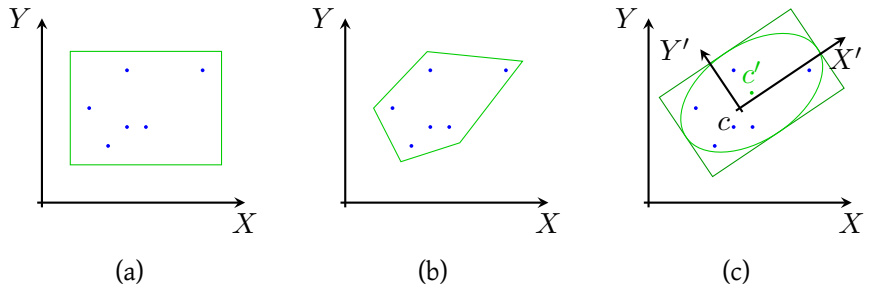


Figure 5.15: Different approaches to the visualization of a traffic group (Figure from [EES+15]). (a) A framing rectangle aligned with the window coordinate frame. (b) The convex hull of a traffic group. (c) An ellipse and a rectangle oriented to the principal axes of the traffic group.

a procedure for the smooth visualization of the transitions between partitioned and merged traffic. Compared to the cluster fusion process described in Chapter 4, the approach we describe here is less complicated. Since the traffic information is not as noisy as the LiDAR data and a lower number of data points are used as input for clustering, we do not have the need of using elaborated methods such as the distance field calculation or the application of Rvachev functions. The system we propose also has the advantage that the memory usage and the calculation time is reduced compared to the cluster fusion process for non-classified LiDAR data.

The smooth visualization is conducted in intervals of a certain length or number of iterations N. In the first iteration of each interval, when $i = 0$, the current traffic is clustered and the fusion and splitting processes to be visualized within this interval are determined. During all following iterations ($0 < i < N$) until the end of the interval no clustering is performed again. Instead, the unchanged group symbols are rendered and the identified transitions between the traffic groups are visualized incrementally. This stepwise change from the initial to the final symbols of the particular fusion or splitting is controlled by the Smooth Transition Progress, which represents a ratio of elapsed to total time of the interval.

The scenario comprises three aircraft represented by red dots (Figure 5.16), which constituted two clusters after the previous visualization interval. The clustering performed in the first iteration of the current interval delivered a new cluster comprising all three aircraft. The corresponding symbol, which shall be the final result of the transition process, is illustrated by the black ellipse in the left image of Figure 5.16. The standard visualization of the clustering results would instantaneously draw this new cluster, the smooth visualization proposed here, however, visualizes a successive transition from the initial to the final traffic symbols.

Figure 5.16 sketches a timeline of the step-wise transformation of the traffic symbols, from the initial ellipse and circle on the left side to the final ellipse on the right side. At the beginning of the visualization interval the two input symbols are opened up. Afterwards, during the visualization iterations of the current interval, the resulting curves are adapted incrementally until the final symbol shape is reached. Figure 5.17 shows an excerpt of the same merging process as above, however with additional information on important details of the smooth transition algorithm.

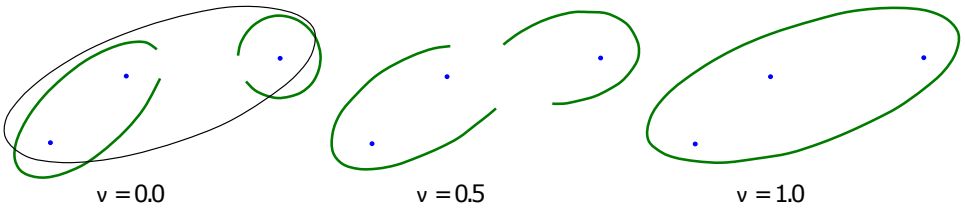


Figure 5.16: The step-wise transformation of the traffic symbols beginning with a clustering iteration ($v = 0.0$) followed by several visualization iterations ($v = 0.5$ and $v = 1.0$). The convex hull of the detected cluster is depicted as a black line (Figure from [EES+15]).

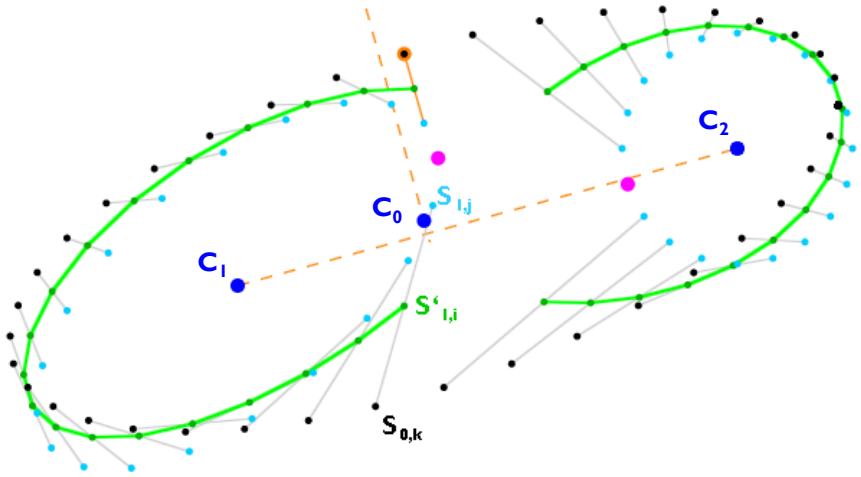


Figure 5.17: Details of the smooth transition algorithm (Figure from [EES+15]).

Each traffic symbol is composed of several line segments connecting two adjacent vertices. During the creation of the traffic group symbols, these corner points are determined in the local principal coordinates (X', Y') of each symbol and afterwards transformed back to window coordinates (X, Y) for drawing. In Figure 5.17, the vertices of the two input symbols, $\mathbf{S}_{1,j}$ and $\mathbf{S}_{2,j}$, an ellipse and a circle, are illustrated by the light blue dots, while black dots show the vertices of the output ellipse, $\mathbf{S}_{0,k}$. Furthermore, the corner points of the currently drawn

symbols, $\mathbf{S}'_{1,i}$ and $\mathbf{S}'_{2,i}$, are illustrated by the dark green dots on the green lines. Finally, the geometric centers C_1, C_2 and C_0 of the symbols S_1, S_2 and S_0 are depicted as dark blue dots.

The algorithm for the computation of the current vertices to draw can be divided into three tasks:

1. Find the points where the two input symbols are opened up (magenta dots).
2. Determine a starting point for the mapping of the input to the output vertices (orange dot) and assign each point of the input symbols to a vertex of the output symbol (gray lines).
3. Interpolate the resulting points of the currently drawn shape (green dots), based upon the previously determined mapping.

The fusion and splitting algorithm presented above has two constraints that need to be considered for the determination of the transitions to be visualized in the subsequent visualization interval. First, only two symbols can be merged at once and a cluster can only be split up into two distinct parts during a single visualization interval. Second, a symbol can only be involved in one transformation process simultaneously; it cannot be part of a fusion and a splitting in the same interval. The method we propose to generate such transitions between the previous and the current clusters is shown by the example scenarios in Figure 5.18.

By comparing the initial constellation with the desired final result, three separate transition processes can be identified:

- fusion of the previous cluster p_0 and the left part of the previous cluster p_1 to the current cluster c_0.
- splitting of the previous cluster p_1.
- merging of the previous clusters p_2, p_3 and the right part of p_1 to the current cluster c_1.

The required symbol transformations conflict with both constraints of the smooth transition visualization stated above. First, the final cluster c_1 is a result of the fusion of more than two input symbols. Second, the previous cluster p_1 is part of two fusions and one splitting process at the same time. Consequently, the final traffic group symbols detected by the clustering cannot be reached within this visualization interval. Instead, all fusions and splittings which are permitted simultaneously will be determined and forwarded for drawing. This status is saved as initial constellation for the following visualization interval, which implies that all transformations that are not performed now will be made up later if the clustering

does not change in the meantime. In conclusion, the whole transition process of the symbols in this example spans over two visualization intervals with an intermediate step as depicted in the right half of Figure 5.18.

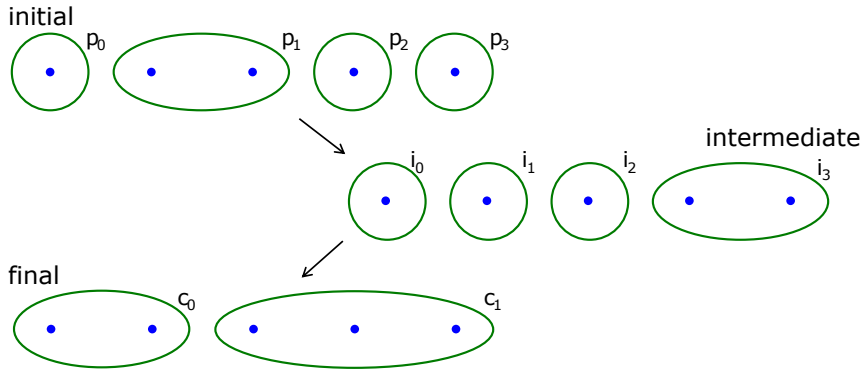


Figure 5.18: Example of a multiple fusion and splitting process with intermediate visualization step (Figure from [EES+15]).

The whole process is predicated on the comparison between the resulting situation of the previous visualization interval and the new desired constellation determined by the current clustering. These initial and final situations are contrasted by means of the assignment of the vehicles to the clusters. For the implementation it must be noted that the clustering does not ensure that the same traffic groups receive the same Cluster ID in two consecutive visualization intervals, which means that even vehicles that remain in the same cluster can have different Cluster IDs in the previous and the current situation.

With this information at hand, the algorithm can be divided into two consecutive parts for the identification of splittings and fusions. Splittings are processed first since this decreases the number of required intermediate steps in many situations. The smooth visualization of the situation in Figure 5.18, for instance, would necessitate two additional intermediate steps, if the order was reversed.

For the detection of splittings, each previous cluster with more than one member is checked. A splitting occurs, if the current cluster IDs are not the same for all vehicles of the respective previous cluster. In other words, the vehicles of this traffic group are now part of at least two different clusters, which requires the visualization of a splitting. To avoid that this previous cluster will be used for a fusion later, it is marked as processed before the two output clusters are identified. Finally, the whole splitting process is stored for visualization.

In the example scenario this part of the algorithm recognizes that vehicle 1 and 2 in the previous cluster p_1 received different cluster IDs c_0 and c_1. Thus, it triggers the splitting of the cluster into i_1 and i_2 and marks p_1 as processed.

The consecutive identification of fusions works in a similar way by checking which final cluster is constructed of at least two previous symbols that were not part of a splitting before. However, the determination of the input and output symbols is different as the algorithm must account for situations were the current cluster is composed of three or more previous clusters. In this case, not all incoming symbols can be fused at once but pairs of two clusters can iteratively be merged until fewer than two are remaining. The same problem is coming up, if a cluster splits up in more than two new clusters as given in Figure 5.19.

Since only one of the three occurring splittings can be executed in the first interval, it is important to choose a proper partitioning of the four involved vehicles into the two intermediate clusters. The favorable subdivision for this exemplary case is depicted on the right side of Figure 5.19. First, the algorithm avoids overlapping symbols because it keeps the adjacent vehicles 0 and 1 as well as 2 and 3 merged. Second, the procedure allows for a simultaneous execution of the two remaining splittings as it distributes the four vehicles evenly over the two intermediate clusters. If three aircraft were grouped into one intermediate cluster, two instead of one additional visualization interval would be needed to reach the desired final state. Without going into the exact details, the basic idea of the algorithm is to first find the two final clusters with the greatest distance in between, here c_0 and c_3. Subsequently, the chosen clusters are used as seeds for constructing the two intermediate symbols as the respective nearest remaining final clusters, here c_1 and c_2, are alternately distributed between them.

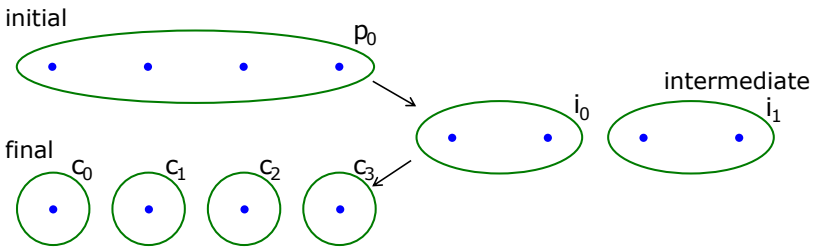


Figure 5.19: Splitting a cluster into more than two distinct traffic groups (Figure from [EES+15]).

Visual Threat Potential

The second major part of the advanced traffic visualization in HMDs besides the de-cluttering is the visual coding of important traffic parameters which are not directly visible from a simple perspective projection of the traffic positions. A suitable traffic visualization on the HMD must employ additional techniques to present the required information to the pilot: The threat potential measure.

One could label every traffic symbol with additional information such as altitude or vertical speed trend as it is done in the Head-Down Traffic Display. Such an approach was chosen by Wong et al. [WKN04]. Moreover, the size of the symbol could be increased as the intruder comes closer to the ownship or the flight direction could be visualized by using arrows as traffic symbols. However, for our approach it was decided to condense all this information to one single measure, the threat potential, so as to reduce display clutter. Supplemental data on every target aircraft shall be retrieved from the HDD. This approach follows the maxim that only the most important information is visualized head-mounted.

Threat Potential Measure:
The main question for a pilot facing an intruder in the vicinity of his aircraft is, if the other aircraft poses a potential threat and if he is required to take any corrective action or if he can continue his flight as planned. This is influenced by many parameters such as positions, speeds and tracks of both the ownship and the intruder. Only certain combinations of these parameters make a nearby aircraft a potential threat. However, the pilot needs to decide in short time, if he must react or if he can ignore a target aircraft. Therefore, TCAS provides a Threat Potential Measure calculation method that is based upon the Closest Point of Approach (CPA). The approach taken by the TCAS is highly affected by the restricted knowledge of the encounter geometry. Because of ADS-B providing complete position and velocity vectors, a three-dimensional vector analysis of the scenario can be performed instead of using the TCAS approach of dividing the range by the range rate.

To simplify the computations, the positions and velocities of the involved aircraft (ownship o, intruder i) are transformed to a relative frame of reference, whose static origin is located at the intruder position (Figure 5.20). The point of the predicted closest approach of intruder and ownship is represented by the position vector $\mathbf{p}$. In terms of the known relative position $\mathbf{x} = \mathbf{x}_o - \mathbf{x}_i$ and the relative velocity $\mathbf{v} = \mathbf{v}_o - \mathbf{v}_i$, it can be expressed as

$$\mathbf{p} = \mathbf{x} + t_{cpa} \cdot \mathbf{v} \tag{5.3}$$

where t_{cpa} is the unknown time at which the closest point of approach is reached.

In order to derive an equation for the determination of this parameter, the vector $\mathbf{w}$ is introduced. As can be seen in Figure 5.20, t_{cpa} can now be described as the ratio between the lengths of the vectors $\mathbf{w}$ and $\mathbf{v}$

$$t_{cpa} = \frac{|\mathbf{w}|}{|\mathbf{v}|}. \tag{5.4}$$

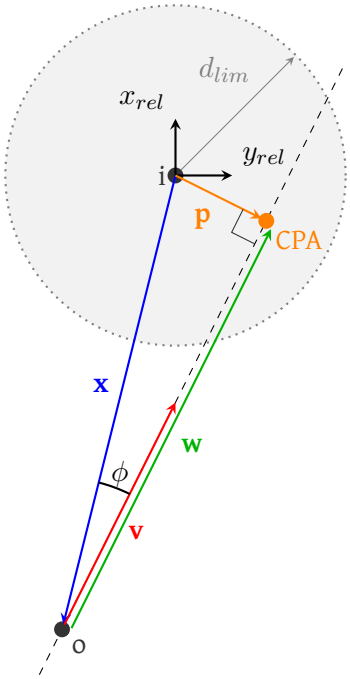


Figure 5.20: Definition of the parameters used by the Euclidean vector algebra method (Figure from [EES+15]).

Furthermore, $|\mathbf{w}|$ is related to $|\mathbf{x}|$ via basic trigonometry $|\mathbf{w}| = |\mathbf{x}| \cdot \cos\phi$, where $\cos\phi$ can be expressed through the scalar product of the adjacent vectors as

$$\cos\phi = \frac{-\mathbf{x} \circ \mathbf{v}}{|-\mathbf{x}| \cdot |\mathbf{v}|}. \tag{5.5}$$

Combining these two expressions, substituting the result into Equation 5.4 and reducing the resulting fraction yields a simple equation for the time of the closest point of approach:

$$t_{cpa} = -\frac{\mathbf{x} \circ \mathbf{v}}{|\mathbf{v}|^2} \tag{5.6}$$

As it turns out, this definition of t_{cpa} is similar to the one derived by [MNC13], however via a different approach. Finally, the calculated t_{cpa} is substituted into 5.3 in order to compute the miss distance of both aircraft at CPA

$$d_{cpa} = |\mathbf{p}| = |\mathbf{x} + t_{cpa} \cdot \mathbf{v}|. \tag{5.7}$$

The final Threat Potential Measure is the normalized value of t_{cpa}, that is color-coded for visualization purposes.

Results

In summary, the preceding chapter presented several methods for the de-cluttering of the HMD and for the graphical representation of important information in a way that fulfills the head-mounted see-through requirements. The former includes a range and altitude filter function, a relevance filter via the threat potential and a method for identifying groups of similar traffic which can be visualized by a single symbol. The latter includes reference lines to support the qualitative assessment of an intruder's relative position as well as the graphical representation of a threat potential, which constitutes an intuitive way to quickly recognize the most relevant intruders. This section illustrates how all these features are combined to the final prototype version of the advanced traffic visualization where the HMD symbology is connected to the head-down visualization.

Figure 5.21: The traffic visualization as seen by the pilot wearing an HMD (Figure from [EES+15]).

Figure 5.21 shows a screenshot of the developed symbology. The image was recorded in a flight simulator, into which the software is integrated. For demonstration purposes, the outside view is overlaid with the head-mounted symbology in order to simulate the pilot's view of the scene through the combiner of the HMD. In general, airborne traffic is highlighted by ellipsoidal shapes whereas ground traffic is represented by rectangular symbols. The head-mounted visualization of both vehicle types can be switched off separately on the implemented HDD. In this exemplary situation a group of airborne traffic with a high threat potential is visible in the upper left corner. To the right of this cluster, a single aircraft with a medium threat potential measure can be seen. Finally, the fusion of two ground

vehicles is captured on the right side of Figure 5.21.

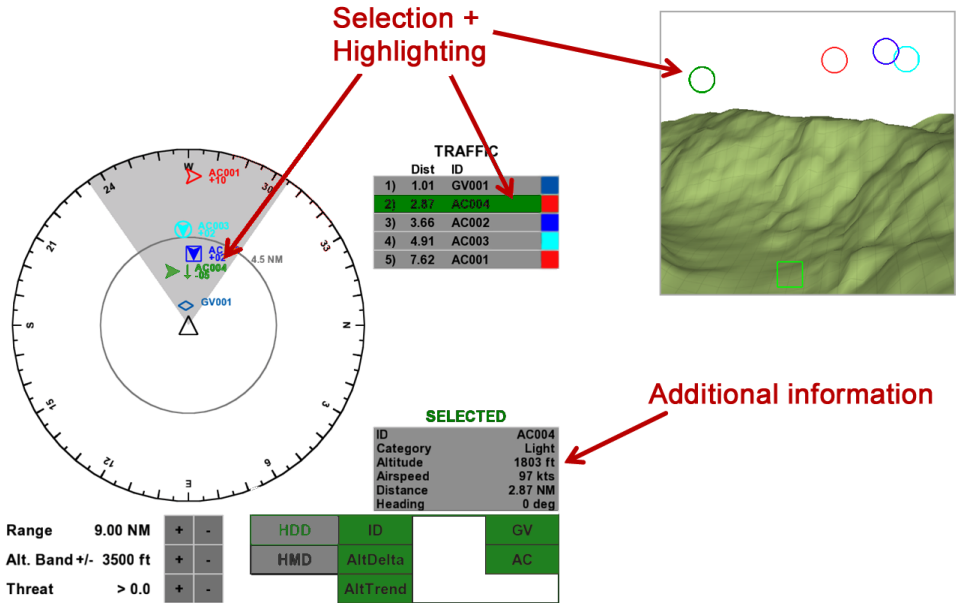
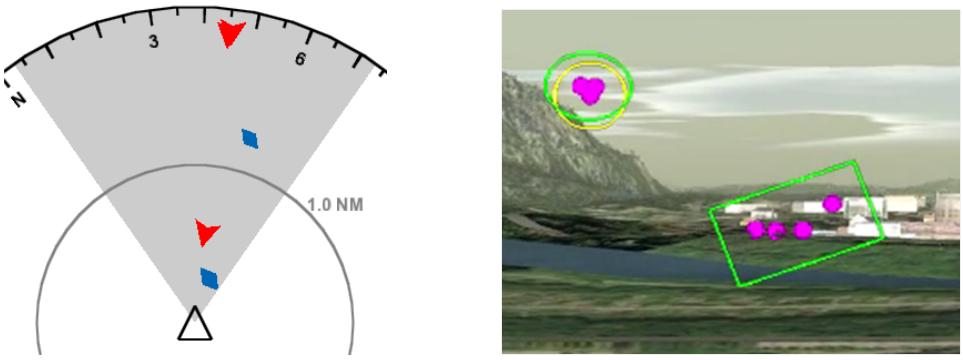


Figure 5.22: The connection between HMD and HDD (Figure adapted from [EES+15]).

As mentioned above, the head-mounted traffic representation is connected to the implemented Head-Down Traffic Display. Figure 5.22 illustrates the realization of this connection via a selection mechanism. A selected vehicle is visualized by a magenta HMD symbol. At the same time, it is highlighted in green within the traffic list and the heading rose on the HDD. Supplemental information about the selected vehicle can be retrieved from the head-down info box.

Figure 5.23 presents the effects of different clustering strategies for ground and airborne traffic. The features on which the clustering is based can be selected easily and combined in many ways. In the depicted case, the ground traffic is grouped subject to its position on the HMD only. Thus, the three surface vehicles in the foreground and the vehicle in the background, which appear clearly separated on the HDD, are represented by one rectangular head-mounted symbol. By contrast, the three visible aircraft are visualized by two distinct icons on the HMD, despite their overlapping. The reason for this behavior is the clustering strategy checking not only the screen position but also the behavior (track angle, speed) as well as the spatial proximity, which is not given in this case.

For this Figure 5.23, the symbology designed for color HMDs is illustrated. The monochrome visualization applies a decreased transparency to aircraft with a lower threat potential. Furthermore, the number of vehicles represented by a group symbol can be coded visually via the line width.



(a) (b)

Figure 5.23: An exemplary traffic situation showing the effects of different clustering strategies for ground and airborne traffic (Figure adapted from [EES+15]). (a) The HDD with ground vehicles (tan diamonds) and airborne traffic (cyan arrows). (b) The situation seen through the HMD. (traffic positions indicated by magenta dots for demonstration purposes only).

Evaluation

The number of primitives, that is the amount of traffic symbols, drawn by the advanced traffic visualization can never exceed the quantity of primitives in the basic approach, which indicates every vehicle by a single symbol. Again, the extent to which the primitive count is reduced depends on the specific situation. For example, displaying a cluster of two aircraft with one group symbol instead of two single symbols decreases the number of primitives by 50% while in case of four *similar* aircraft the quantity can be reduced by 75%. For each cluster comprising N vehicles, the relation is given by

$$n_{Adv} = n_{Bas} \cdot \frac{1}{N} \tag{5.8}$$

where n_{Adv} and n_{Bas} are the number of primitives drawn by the advanced and the basic traffic visualization respectively. Figure 5.13 shows a scenario where the number of rendered icons is lowered by 55%, from 11 to 5. Similarily, the number of rendered pixels can be reduced by our advanced approach. Additionally, the computation time measurements revealed that the proposed algorithms are capable of running in real-time.

5.4 Landing Site Depiction

An evaluated and elaborated feature of the synthetic vision system is the rendering of a selected landing site, that is usually used for DVE landings during brownouts (see Figure 1.3(c)). For varying distances to the landing site different symbols at different Levels-of-Detail are rendered. For large distances a simple symbol depicting only the 3D conformal landing position in the HMD is rendered, while for medium distances the simple symbol is supplemented with a grid shape that provides an impression about the ground quality as can be seen in white in Figure 5.24. For the landing procedure, the doghouse is rendered as a detailed mesh of the ground with indicators for the relative flight height of the helicopter as well as slope indicators. An detailed description of the landing approach and the used symbols is provided by Münsterer et al.[MKK13].

As an additional feature for non-photorealistic and abstract depiction of elements surrounding the landing site, we propose ellipsoidal shapes for providing the pilot a hint about potential dangers. Alternatively, the ellipsoidal shape can be used to depict potential alternative landing sites, if the selected landing site is not accessible for the helicopter due to suddenly detected obstacles. The usage depends only on the configuration of the system.

(a) (b)

Figure 5.24: (a) Landing site with potential dangers, (b) Landing site with potential alternative landing sites.

In Figure 5.25, the features used to calculate an ellipse that is rendered on the ground mesh are depicted. The ellipse is described by a feature vector $\{p, \sigma_1, \sigma_2, \alpha\}$, where p is the position in a 3D geo-referenced, Cartesian coordinate system within a certain sector next to the landing site, σ_1 is the horizontal variance of terrain features and σ_2 is the variance of the terrain heights within the sector. α is the transparency of the rendered ellipse and corresponds to σ_1. σ_1 and σ_2 are used for the dimension of the ellipse.

The information used to calculate the ellipse feature vector $\{p, \sigma_1, \sigma_2, \alpha\}$ is derived from sectors defined around the landing site. These sectors have a fixed size on the screen in order to adapt the number of the rendered ellipses according

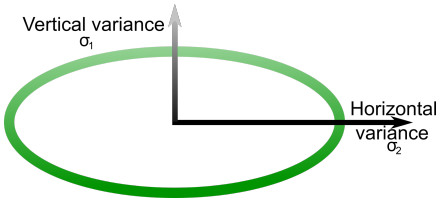


Figure 5.25: Feature ellipse with σ_1 and σ_2 as dimension vectors.

to the 3D distance to the landing site. For each of the 2D sectors, the corresponding height information is retrieved by back-projecting the pixel information within the sector to 3D and rendering the height off-screen from a top-down orthogonal perspective to a depth buffer. The resulting depth image is analyzed by using the FAST feature detector [RD05]. The sectors as well as the resulting points from the analysis with FAST are depicted in Figure 5.26(a). The FAST points are used to calculate the fixed center position p of the ellipse by calculating the mean value. The dimension indicators of the ellipse are the eigenvalues $\sigma_1 \in \mathbb{R}$ (largest horizontal eigenvalue) and $\sigma_2 \in \mathbb{R}$ (vertical variance) and are calculated by applying a principle component analysis to the FAST points. σ_1 and σ_2 are the non-normalized eigenvalues from the PCA and its 95% quintile is used for estimating the dimension of the rendered ellipse. The 95% quintile is derived from the expectation that the feature points are normal distributed and can be calculated by the inverse of the chi-square cumulative distribution for the 95% confidence interval:

$$f(\sigma) = 0.24477 \cdot \sqrt{\sigma}. \tag{5.9}$$

The constant value 0.24477 is derived from the chi-square tabular. The evaluation of this function $f(\sigma)$ for σ_1 and σ_2 results in the final dimension of the ellipse. The normalized eigenvalue

$$\hat{\sigma}_1 = \frac{\sigma_1}{\sqrt{\sigma_1 \cdot \sigma_1 + \sigma_2 \cdot \sigma_2}} \in [0, 1], \tag{5.10}$$

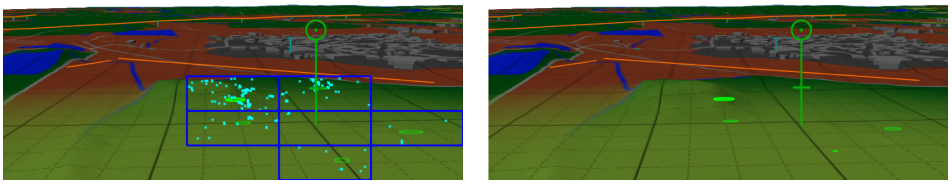
is used for the calculation of the transparency parameter α in Equations 5.11 and 5.12. α is estimated by the multiplication of the normalized distance to the helicopter $d \in [0, 1]$ within the local coordinate system of the sector and $\hat{\sigma}_1$. If the ellipses are used to depict a potential danger, the transparency value is calculated with

$$\alpha = d \cdot \hat{\sigma}_1, \tag{5.11}$$

while the transparency for depicting the potential alternative landing sites is

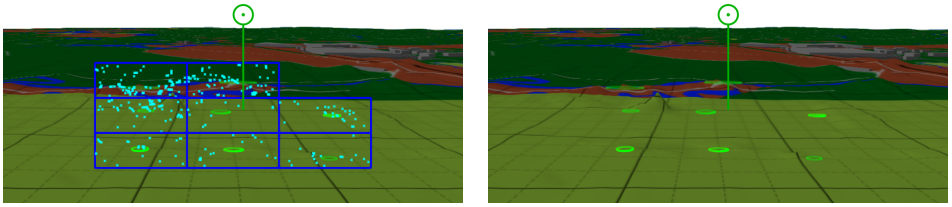
$$\alpha = d \cdot (1 - \hat{\sigma}_1). \tag{5.12}$$

Once the initial ellipse is calculated, it is only updated by new FAST points, if a sensor update changes the terrain mesh. To avoid sudden changes in the ellipse rendering, the old parameters are interpolated to the new ones. In Figure 5.26(b) the landing site is depicted as 3D pole symbol and final ellipses are given for the configuration to avoid potential dangers.



(a) (b)

Figure 5.26: Potential danger depiction in surrounding of the selected landing zone (green pole): (a) Landing site area in the synthetic vision system with sectors (blue rectangle) and features (light blue points), (b) resulting ellipses depicting potential dangers.



(a) (b)

Figure 5.27: Alternatives to the selected landing zone (green pole): (a) Landing site area in the synthetic vision system with sectors (blue rectangle) and features (light blue points), (b) resulting ellipses depicting potential alternative landing sites. Large ellipses in the background are less opaque than foreground ellipses.

5.5 Conclusion

We present a prototype implementation of a visualization concept for non-classified LiDAR clusters, traffic and alternative landing sites on both head-down and head-mounted displays. The main focus of our work lies on the development of methods for de-cluttering the HMD and increasing the information content of the head-mounted symbology by coding important parameters visually. The realized functions are integrated into our flight simulator.

With the cluster visualization, we provide a very efficient and scalable system for rendering non-classified areas within the accumulated 3D LiDAR sensor point cloud. The use of two symbols types in the form of a box-based shape as well as the eigenvector representation is the most promising variant through all the possible meshing methods. In addition to the implemented variants of cluster shape methods even more meshing types such as alpha shapes need to be evaluated. Due to the more organic appearance of an alpha shape a faster mental mapping to a natural occurring objects by the pilot might be possible. Box shapes have the potential to be used for urban sceneries, since they suggest that they are representing man-made objects. Since this method is depended from a variety of parameters, most of them must be analyzed during workshops with pilots and do not have to be adjustable during mission in order to keep the pilot's workload at a minimum. Only the interpolation speed should be accessible to the pilot.

In order to provide abstract information such as traffic to the pilot we developed an additional visualization system. One central idea of the developed traffic visualization concept is to combine the strengths of both involved display types – head-mounted and head-down – while simultaneously diminishing their individual weaknesses. The role of the HMD is to support the pilot in *seeing and avoiding* other aircraft, even if the view is degraded. Additionally, it shall help to keep the *head up* and the *eyes out* during the flight. The Head-Down Traffic Display complements the role of the HMD by providing an overview of the traffic situation and presenting additional information that cannot be displayed head-mounted.

In order to automatically de-clutter the HMD, a technique for clustering the nearby vehicles into groups of similar traffic is developed. For this purpose, the existing *DBSCAN* algorithm is expanded to deal with non-metric scales. Furthermore, a graphical representation of the identified traffic clusters on the HMD is suggested. To achieve time coherency and avoid distracting, sudden changes of the symbology, the cluster representation is dynamically adapted to the varying clustering results. The developed algorithm generates a smooth visualization of the symbol transitions when clusters merge or split. Furthermore, our work deals with the visual coding of important parameters on Head-Mounted Displays. The relevance of a proximate aircraft is expressed by one single measure that is intuitive to use, namely the threat potential. Similar to the non-classified cluster renderings, the threat potential is dependent from different parameters. These parameters should only be adaptable during HMI workshops or right in the beginning or during the preparation of a flight mission, but not during flight in order to avoid mis-interpretations of potential dangers. Another valid strategy would be to introduce a norm similar to the TCAS system that specifies non-adaptable parameters.

In addition to the non-classified cluster visualization and the traffic rendering system, we introduce an additional landing aid extension. Therefore, we also provide an abstract visualization of the environment surrounding the area selected by the pilot where the helicopter is supposed to land. This system allows two configurations, that either show potential dangerous areas, based upon the terrain information or alternative landing sites. Apart from these alternative configurations, this system is not dependent on any parameters, in contrast to the non-classified object rendering system.

Our rendering approaches provide full prototypes of an obstacle, traffic and landing aid visualization display systems, which are adaptable to the pilot's needs. Future work will focus on the evaluation of the traffic system with real ADS-B data as well as the full evaluation within HMI workshops of each of the proposed techniques.

6

Non-photorealistic Terrain Rendering

Terrain illustrations are necessary for a wide variety of applications ranging from map production to geo-visualization and geographic information systems, interactive computer games [Döl07], flight and driving simulators and planning systems for navigational tasks. For navigational issues, methods have to be developed to feed Head-Mounted Displays when only the most important features of a terrain need to be shown [FHT08]. We introduce two methods for artistic terrain renderings as well as terrain renderings optimized for HMD usage in a flight deck. Both of the methods are related to each other. The terrain rendering in the HMD is inspired by the artistic rendering and adapts the used methods according to the memory and computation time requirements on a mission computer. In Figure 6.1 is depicted, where the terrain rendering methods within HMDs are located within the full processing pipeline.

6.1 Related Work

Terrain depiction is not only focused on rendering methods but also on mesh processing approaches to detect immanent features such as streamlines as well as on Level-of-Detail approaches.

Terrain rendering:
Terrain rendering covers a variety of methods ranging from the generation of artistic panorama maps towards realistic rendering [BST09]. Kennelly and Kimerling describe NPR techniques for terrain rendering in a cartographic context [KK06]. The work most relevant for our approach is proposed by Buchin et al. [BSD+04] and by Way and Shih [WS06]. Buchin et al. create terrain illustrations by using light

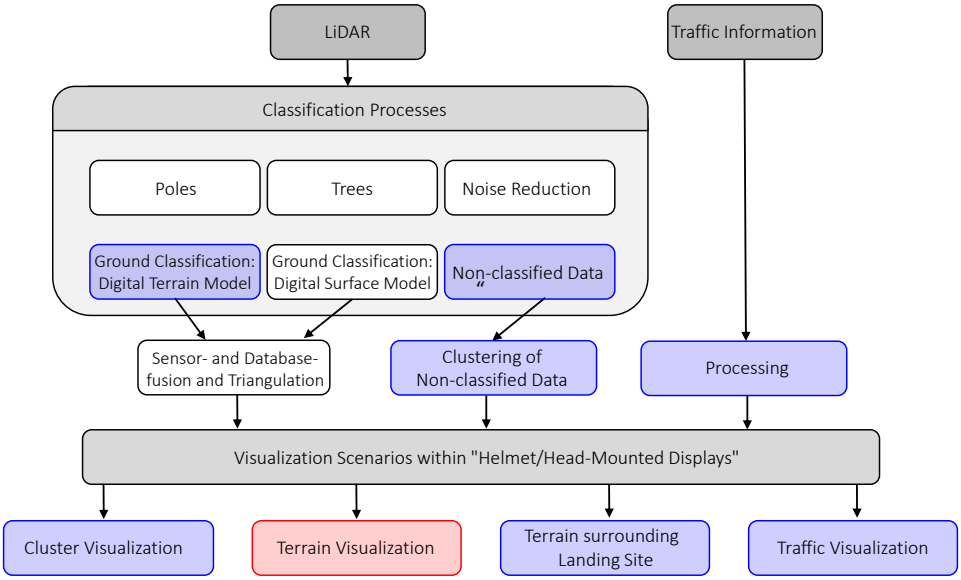


Figure 6.1: Location of the visualization of terrain on HMDs. This processing step is indicated through the red box in the processing pipeline.

intensities on the terrain surface for creating stroke-based renderings. Strokes start at uniformly sampled points within the terrain and flow downhill, the selection of strokes to be rendered is based upon light variation. In contrast to them, our work combines state-of-the-art streamline placement with rendering of ridge and valley lines and furthermore uses the streamlines as guiding paths for our hatching lines to create different kinds of strokes.

Way and Shih [WS06] provide an approach that uses streamlines to represent terrains inspired by Chinese landscape paintings. While their approach is limited to rendering streamlines that are generated for each triangle in the terrain mesh and are textured with a fixed type of Chinese drawing style, we extend their approach by modifying our streamline-based hatches according to the terrain type, by representing important terrain features. We also use lighting information for streamline rendering and a more sophisticated streamline placement as well as an adaptive stroke parametrization that allows us to render plenty of different styles. We also use the method proposed by Whelan et al. [WV03] to enhance our terrain renderings with silhouettes from depth images. Mat and Visvalingam [MV02] provide an evaluation showing that silhouettes are important for terrain rendering. A fast approach for such renderings with a GPU implementation is given by Mower [Mow14].

Streamline placement and rendering:
Streamlines are most common in flow visualization for depicting the directions within a flow field [WE05]. Illustrative rendering of streamlines can be useful in scientific visualization [BCP+12]. Liya et al. [LHS08] present a seeding strategy for a minimum set of streamlines that ensure visual clarity, based upon distance fields of streamlines. They also provide a rendering strategy for streamlines in 3D that avoids cluttering by minimizing overlapping and intersection [LwS07]. Xu et al. [XLS10] present a seeding algorithm in areas with high entropy and additional seeding in areas where the conditional entropy is high. Inspired by this work, we place more streamlines around ridge and valley lines to capture the local structure near salient regions.

Levels-of-Detail approaches for terrain rendering:
Since terrains typically contain huge amount of data, Levels-of-Detail approaches have to be introduced. Hoppe [Hop96] provides a method to represent terrains using progressive meshes that enables viewers to smoothly change Levels-of-Detail, based upon the camera position. Cignoni et al. [CGG+03] introduce the BDAM technique, that simplifies terrain meshes by optimizing their triangulation. Another approach is to recursively subdivide triangle meshes using longest-edge bisection for a view-dependent refinement, following smooth blending of geometry using geomorphing [LP02]. Losasso and Hoppe [LH04] use geometry clipmaps to provide visual continuity by caching terrain data in a set of nested regular grids. The grids are incrementally refilled as the view point changes.

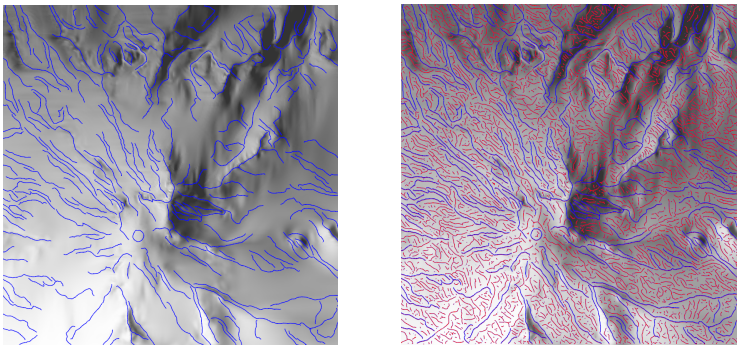
6.2 Stylized Terrain Representation

Input for our system can be meshes generated by fractal functions as well as real-world terrains obtained from earth observation techniques such as airborne LiDAR [Axe99]. These terrain meshes are analyzed for ridge and valley lines in order to create the final feature lines in the form of stream lines, that usually lie orthogonal to the ridge and valley lines, where an artist would draw a line.

Ridge and Valley Detection

Given an input terrain in the form of a height field H, we detect ridge and valley lines by using the method by Yoshizawa et al. [YBS05]. Their library delivers ridge and valley lines with a number of quality measures such as *ridgeness*, *sphericalness* and *cyclideness*. We use the *ridgeness* value for creating different sets of ridge and valley lines at different Levels-of-Detail and allow the user to interactively adjust the thresholds for exploring different Levels-of-Detail renderings. Based upon

this, we create a hierarchy of ridge and valley lines that are then assigned to the different terrain levels: a set of lines on a coarse level is part of a set for a fine level. Only very important ridge and valley lines are shown for the coarsest level. Figure 6.2 depicts two different levels of ridge and valley lines.



(a) (b)

Figure 6.2: Two sets of ridge and valley lines with varying degree of detailedness. Lines of (a) are entirely contained in (b).

Streamline Placement

To produce spatially coherent terrain illustrations in different terrain levels, we create a hierarchy of streamlines that we use in a next step to create strokes or hatchings. The flow field that is used as input for the streamline generation is derived with a structure tensor T [HH07] at each position p of the height field H:

$$T(p) = \begin{bmatrix} (H_x(p))^2 & H_x(p)H_y(p) \\ H_x(p)H_y(p) & (H_y(p))^2 \end{bmatrix}. \tag{6.1}$$

Since ridge and valley lines are the most dominant features in our terrain, we use them for guiding streamline seeding. Similar to Li et al. [LHS08], we use the distance fields from the ridge and valley lines as density distribution to sample seed points for the generation of streamlines within the terrain. For different Levels-of-Detail these seed points are sampled independently. Since we produce a hierarchy of streamlines starting from the coarsest Level-of-Detail, we only add streamlines at seed positions where no streamline has been produced so far. The seeding strategy is illustrated in Figure 6.3.

Starting from the points sampled in the neighborhood of ridge and valley lines, we place long streamlines next to such lines, since they have a large visual impact on the terrain structure. The production of a streamline is done by sampling along the directions of the flow field created in the previous step. We use the method

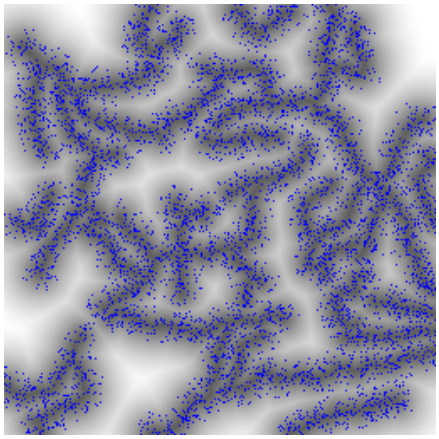


Figure 6.3: Distance field used as density distribution. The sampled seed points are given in blue.

of Chen et al. [CCK07] to determine the elongation of the streamlines, where we specify a minimum and maximum length l_{min}, l_{max} for the streamlines. These parameters depend on the Level-of-Detail, in coarser Levels-of-Detail long streamlines are used in order to represent the terrain with a small number of lines (typical streamline length values: $[3, 12]$ for level 0 to $[54, 64]$ for level 4). In detailed levels, short streamlines are allowed since more dense details are necessary for the visual representation.

Chen et al. [CCK07] calculate the proximity $\rho(s_i, s_j)$ between two streamlines $s_i = \{p, p_0, ..., p_{m-1}\}$ with a central point p within a sliding window over the streamline and sample points $p_0, ..., p_m$ and $s_j = \{q, q_0, ..., q_{m-1}\}$ also within the sliding window by using the following equations:

$$\rho(s_i, s_j) = (t_1 + t_2) \tag{6.2}$$

t_1 is the distance between the two center points

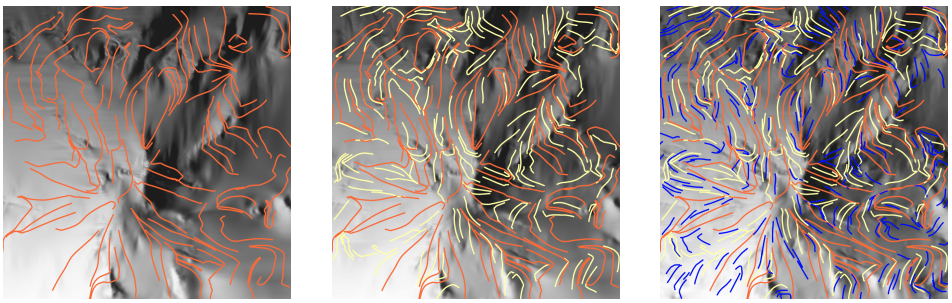
$$t_1 = ||p - q||, \tag{6.3}$$

and a shape and orientation term t_2

$$t_2 = \alpha \frac{\sum_{k=0}^{m-1} ||p_k - q_k|| - ||p - q||}{m}. \tag{6.4}$$

The value for the shape coefficient α and the size of the sliding window m is used according to Chen et al. [CCK07]. Line integration is stopped if $\rho(s_i, s_j)$ is lower than a user-defined threshold $dSep$, indicating that the streamline is too close to

an already existing one. If we change $dSep$ we can influence the visual appearance of the final result ranging from very dense lines to a sparser representation.



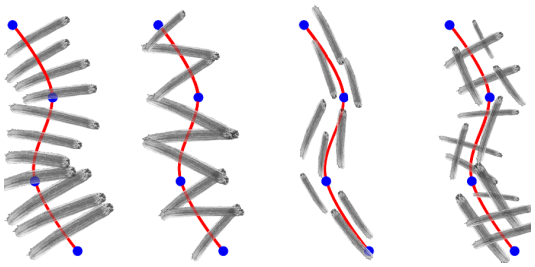
(a) (b) (c)

Figure 6.4: Streamline placement: (a) Lines of highest order in orange. (b) and (c) Additional streamlines of lower order in yellow and then in blue. Based upon the seeding, streamlines are not generated in areas without slopes.

Figure 6.4 shows the hierarchical placement of streamlines. In Figure 6.4(a), streamlines of highest order are shown. All of them emerge from the ridge and valley lines. In Figure 6.4(b), new streamlines are added, however only at places where the distance to the existing streamlines is large enough.

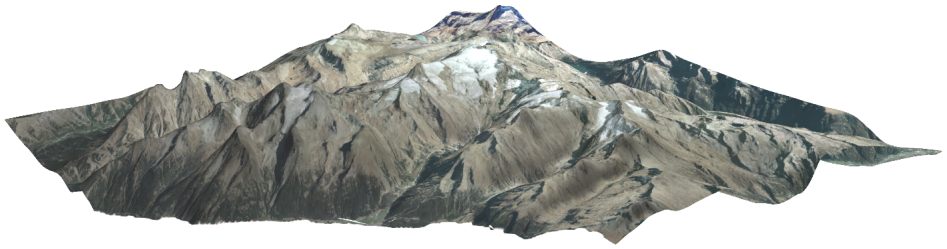
Rendering Results

After computing ridge and valley lines and streamlines, we render the streamlines by using different hatching styles. Our method can resemble a variety of drawing techniques. Figure 6.5 depicts different hatching styles that can be used. Based upon the ridge and valley detection and the streamline placement, we are able to produce different renderings by using different hatching styles as given in Figure 6.6.

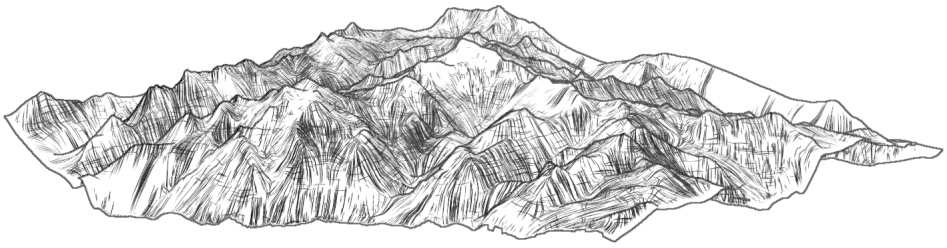


(a) (b) (c) (d)

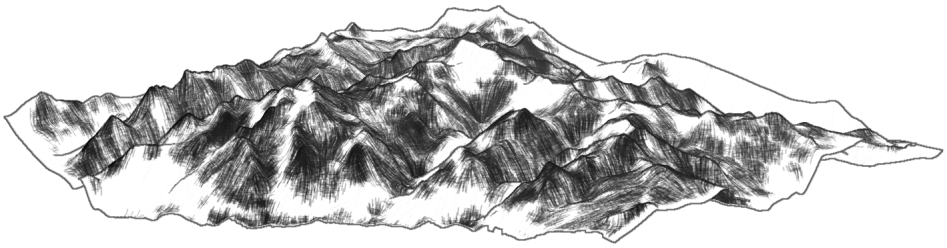
Figure 6.5: Hatching types used to achieve different abstraction styles: (a) Strokes orthogonal to the streamline for pencil hatching. (b) Zick-zack strokes along the streamline. (c) Parallel strokes along the streamline for Pen-and-Ink. (d) Cross-hatching for charcoal style.



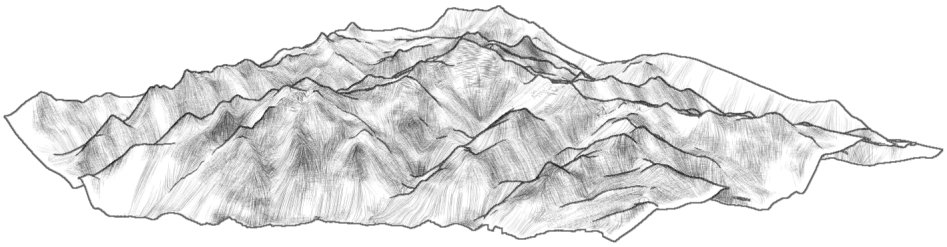
(a)



(b)



(c)

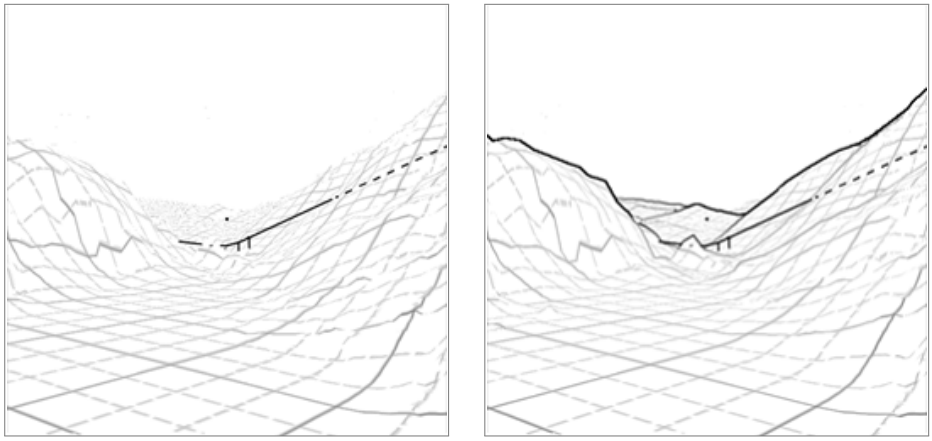


(d)

Figure 6.6: Different rendering techniques: (a) Given input terrain with color texture. (b) Pen-and-Ink hatching style. (c) Charcoal style. (d) Pencil style.

6.3 Terrain Abstraction in HMD

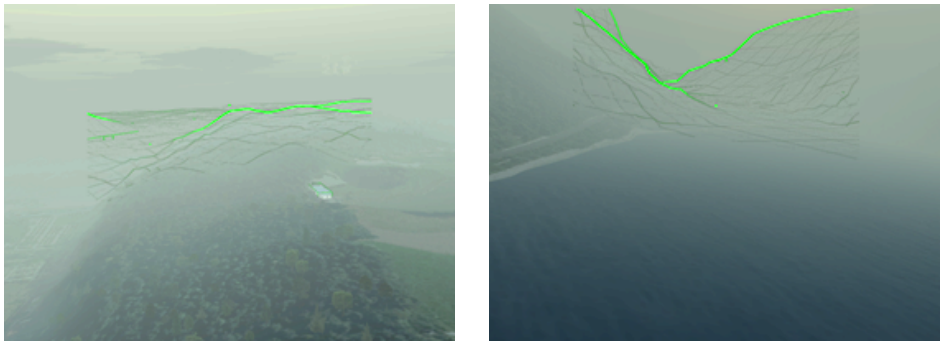
During DVE flights the perception of terrain and landscape shape can be directly augmented though and even beyond the sensor range. This is supported by the sensing and visualization system for DVE situations by optionally displaying a terrain grid combined with or independent from contourlines. The terrain grid is a depiction of the fused terrain information derived from terrain databases and classified LiDAR sensor terrain data. Because the grid information tends to be quite dominant it is typically displayed more transparent than the other symbology on the HMD to avoid distraction from important features such as obstacles or waypoints. The contourlines are based upon the fused information and thus can also be applied for structures in closer ranges with higher resolution thanks to the active sensor data that results in more detailed terrain information. In Figure 6.7(a) an example of the grid representation in combination with obstacle information is given, while in Figure 6.7(b) an extension with contourlines is depicted.



(a) (b)

Figure 6.7: (a) Display of terrain grid and (b) terrain grid with contourlines (Figure from [MSS+14]).

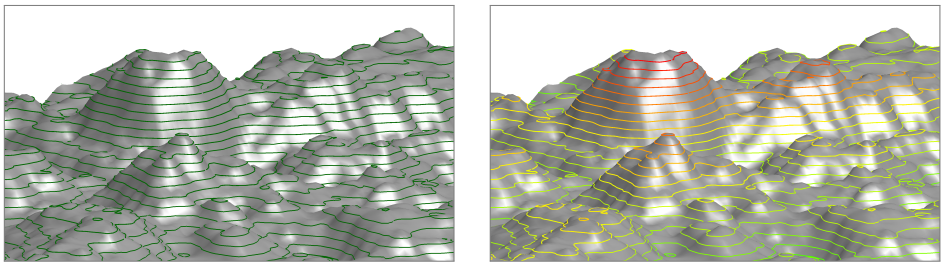
Displaying a conventional grid in DVE flights with limited visibility results in cluttering of the pilot's view on remaining terrain features and textures that may be visible through the window. By contrast, the grid is helpful in distances where the visual perception is obstructed. This led to the concept of a Punch-out-Line, i.e. the pilot's possibility to shift the beginning of the grid and contourline display further away from the helicopter position. An example of the contourline with a reduced grid where the pilot is still able to see potential dangers on its own is given in Figure 6.8.



(a) (b)

Figure 6.8: Two examples of terrain grid with contourlines with a shifted start of display overlaid on outside view at a visibility of 1200 m (Figure from [MSS+14]).

Lenhart and Kaiser evaluated additionally to grid-based meshes an isoline visualization of terrain [Len06, Kai04]. Examples for such rendering is given in Figure 6.9. An outcome of Lenhart's work is that such renderings without additional cues for HDDs as well as for HMDs should be avoided, since it remains unknown whether and how high the maximal height of a peak lies above the highest isoline. Therefore, the pilot gains an incorrect imagination of the surrounding that may lead to hazardous situations. Using such a visualization for terrain that is received during a flight by the fusion of database information with sensor data, yields in addition the problem that changing height information in the terrain results in the need of adapting the isolines. This consequently leads to changes that may not be directly reasonable to the pilot and thus increase its workload.



(a) (b)

Figure 6.9: Isolines with input terrain as evaluated by Lenhart [Len06]. (a) Isolines with one color. (b) Coloring relative to the flight altitude motivated by the *Helicopter Terrain Awareness and Warning System* (HTAWS) technical standard [Cab08].

To extend these existing methods with a more abstract terrain rendering in the HMD we use methods, based upon those described for the artistic terrain stylization in Section 6.2. We use two methods for generating a line rendering: ridge and valley lines and streamlines. Since these lines have the same requirements for

temporal and spatial coherence as the cluster visualization described in Section 5.2, we also provide an interpolation technique in order to avoid an increase of the pilot's workload.

Similar to the renderings of the artistic terrain, it is possible to map textures to the ridge and valleys and streamlines. This yields the additional feature to differ the size of render targets. This could result in large line renderings in the background that are important for the overall situational awareness while in the direct surrounding of the helicopter more detailed structures in the form of the streamlines can be rendered as tiny structures. Moreover, the quality of the texture renderings differs from back to front. Textures in the foreground are rendered in high detail while textures in the background are smoothed and thus lose details.

Placement Strategy and Feature Line Data Structure

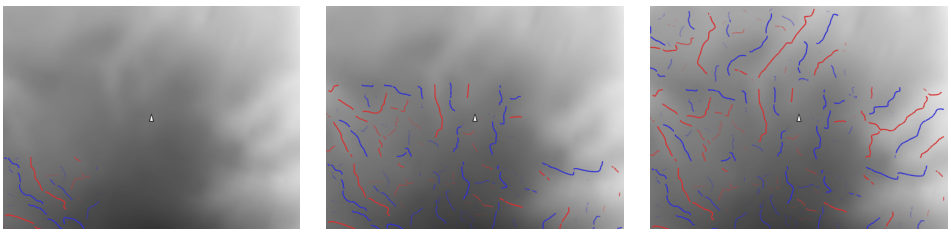
The seeding strategy as well as the placement algorithm, which is provided by Li et al. [LHS08] and is described in Section 6.2, is not usable within the sensing and visualization system for DVE situations due to its runtime complexity of $O(n^2)$. For our system we simplified the placement strategy by using a quadtree: each point of a developed streamline as well as the points of the ridge and valley lines are stored in a quadtree. How many points can be stored within a node regulates the number of lines that can be rendered in a certain neighborhood and replaces the usage of the $dSep$ parameter explained in Section 6.2. The size of the neighborhood is given by the depth of the quadtree. The usage of a quadtree also influences the Level-of-Detail rendering, in high distances a lower number of lines is rendered while in the foreground with respect to the Punch-out-Line all available lines are drawn. The parameters for the quadtree are described in detail in Table 6.1.

Table 6.1: Parameters for storing lines in a quadtree.

Parameter	Value description
Depth	The maximal depth of the quadtree. For memory reasons we used values not higher than 8.
Size/Resolution	The size of the rendered terrain. The position of the quadtree is centered at the helicopter's position.
Number of lines per leaf	The maximal number of lines per leaf is directly dependent on the quadtree's depth. We allow only 1 line for streamlines and 2 for ridge and valley lines at a quadtree depth of 8.

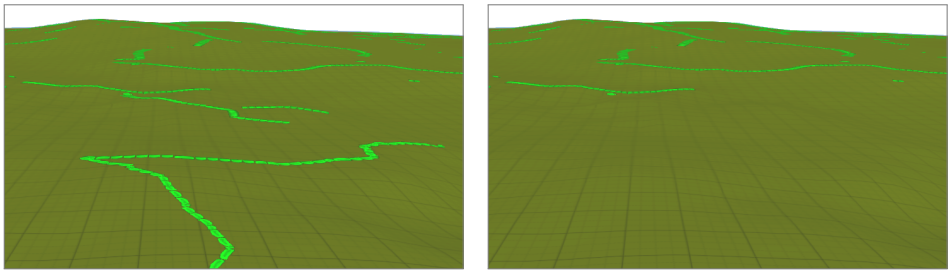
Ridge and Valley Detection

Since the ridge and valley detection technique is time consuming, we use two different approaches to generate these feature lines: either at initialization or at runtime of the sensing and visualization system for DVE situations. At initialization the ridge and valley lines are detected for the entire terrain database without updates from the LiDAR sensor. This results in a higher initialization time but a stable coverage of ridge and valley lines is provided. The second approach is to subdivide the full terrain mesh and detect the ridge and valley lines parallel to every incoming LiDAR sensor frame. This yields the advantage that also parts of the mesh are taken into account, which are updated with sensor information. The successive ridge and valley detection for different parts of the terrain mesh is given in Figure 6.10. For the existing render methods for HDDs as well as for HMDs in the form of textured meshes and contour detection, a stable Level-of-Detail approach is existing in the synthetic vision system that is very similar to the progressive meshes method proposed by Hoppe [Hop96]. This approach results in different sized rectangles for every Level-of-Detail. For the feature line generation, it has the advantage that we are able to select the Level-of-Detail by the size of the rectangles that form the input mesh for the ridge and valley detection technique. In Figure 6.11 ridge and valley lines are provided that are calculated at two different Punch-out-Line configurations. This results in the possibility to provide the pilot a parameter for the detailedness of ridge and valley line presentation beneath the ability to adapt the maximal and minimal values for the *ridgeness*, *sphericalness* and *cyclideness* parameters. These parameters need to be adaptable during evaluation in the form of workshops with pilots. It must be mentioned, that for a final application the parameters should not be adaptable by the pilot during a mission, since they have no impact on the quality of the rendering and its application during a DVE situation. This would only result in a higher workload. A full list of parameters is given in Table 6.2.



(a) (b) (c)

Figure 6.10: Sub-divided ridge and valley detection: (a) ridge and valley lines are detected for a part of the terrain. (b) Parallel to a LiDAR sensor updates, additional parts of the terrain are analyzed. Red lines mark ridges while blue lines represent valleys.



(a) (b)

Figure 6.11: Ridge and valley detection at different Punch-out-Lines: (a) ridge and valley lines are detected for the entire terrain. (b) ridge and valley detection takes only place on geometry far away from the helicopter. Transparent lines are those with a low *ridgeness* value.

Table 6.2: Parameters for ridge and valley detection.

Parameter	Value description
$Ridgeness$, & $cyclideness$, & $sphericalness$	Minimal quality measures that must be reached at least that a ridge and valley line is inserted to a quadtree leaf and thus is taken into account for the placement strategy. For normalized terrain, we use 0.5 for $ridgeness$ and 0 for $cyclideness$ and $sphericalness$.
Pre-processing	Boolean value, if the ridge and valleys are either pre-processed or calculated during runtime.
Terrain subdivision	Number of terrain sub-sectors, where ridges/valleys are detected. For every sensor update one of the sectors is analyzed. A good trade-off between quality and speed is achieved with 25 sectors.
Punch-out-Line	Threshold how far terrain needs to be away from the helicopter position to be analyzed. Depends on the view range of the pilot during operation (weather, fog, brownout, ...)
Resolution	The resolution of the terrain height map can be reduced in order to speed up the calculation. We used an eighth of the visible terrain.
Ridges/Valleys	Binary values, if ridges and/or valleys have to be calculated.

Streamlines

To detect terrain describing features in a depth image, we evaluated different computer vision approaches. We used the depth information, since its analysis results in more stable features than using RGB images. Moreover, depth data provides otherwise absent geometric data that can be simply derived by using the

reverse projection system that is applied to generate the synthetic vision result image. The most reasonable state-of-the-art feature detector with a small computation time is a structure tensor for corner detection. Additionally, we evaluated SIFT-features [Low04] as well as the FAST corner detection [RD05]. Since the SIFT-features have a way higher computation time, the FAST features result in a similar outcome as the structure tensor. All these methods, applied on the depth buffer of the rendered scene within the synthetic vision system, results in very salient and unstable feature points. Therefore, we implemented a multi-step procedure containing edge detection, component-labeling, feature point selection and linear time clustering over multiple rendered frames in order to provide a more stable method for this specific application on depth images. The first step is to apply a basic edge detection with quadratic runtime such as the Sobel-Operator [SF68] or Canny-Edge-Detector [Can86] on the input depth image (see Figure 6.12(a)). For every edge, we store the depth information as given in Figure 6.12(b) and apply a component-labeling algorithm [Hor86] that we extend in such a way that connected regions with different depth information are handled as separated regions (see Figure 6.12(c)). For each of these segmented line regions, we define a feature point for all those parts of the line, where the direction of the line changes. Accordingly, the features are all local maxima or minima within a line as depicted in Figure 6.12(d). The feature points are back-projected to 3D and accumulated over time in bins, which is equivalent to a linear time consuming clustering. If a detected feature point is recurrently re-detected within successive depth buffer images, it becomes a stable feature point. To avoid cluttering, only one feature point per bin is allowed.

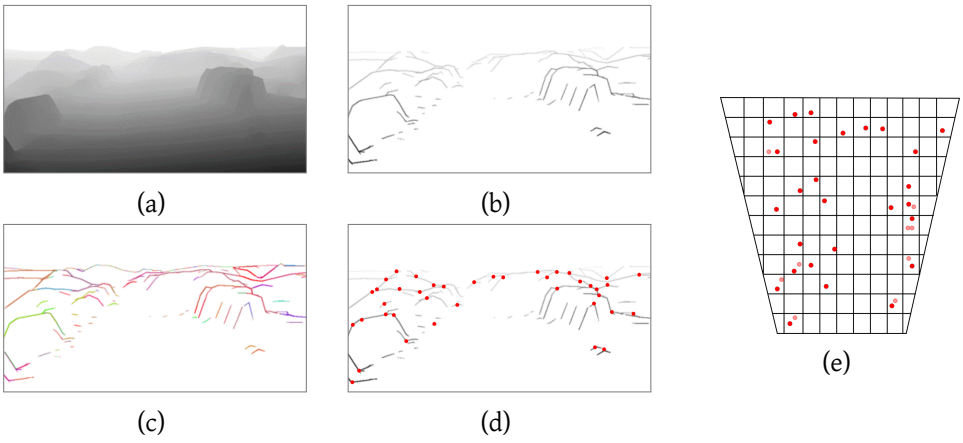
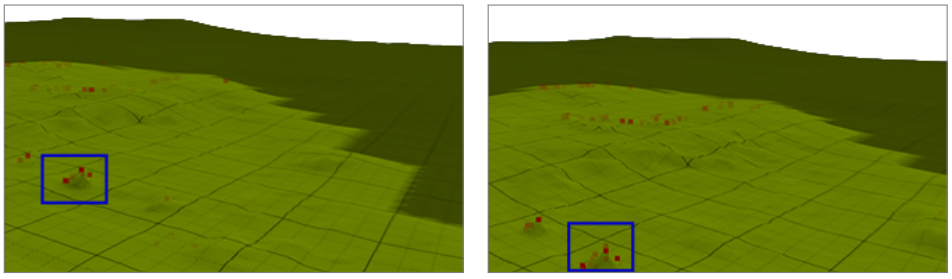


Figure 6.12: Feature detection pipeline on depth images: (a) Input depth buffer image. (b) Contour detection with depth information. (c) Labeled contours. (d) Detected feature points. (e) Top-view on 3D points with binning. Non-feature points are transparent, if a bin contains multiple points.

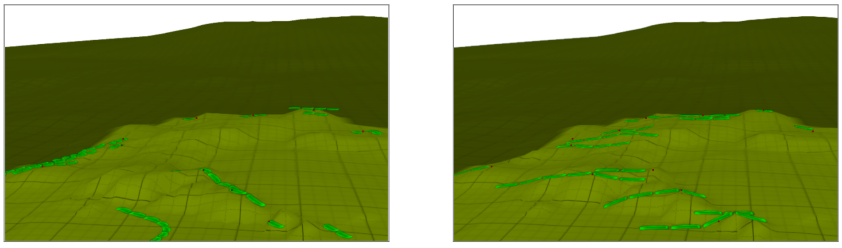
By using each detected feature point for the streamline generation would directly lead to cluttering effects. Therefore, we use the depth information to reproject them into the 3D world space and compare the feature points to the actual terrain geometry. If they match to the terrain, these points were stored for further temporal analysis if they obtain their quality over time. This way spatial stability of the features is not solely based upon pixel information within the depth image but also on the terrain surface on which the 3D points are located. These 3D feature points are then used for feature line generation. In Figure 6.13 the evolving of those feature lines is depicted. To avoid unreasonable changes in the line generation, we use the 3D terrain information rather than gradient information of the depth image to create streamlines. This information is derived every time a terrain update with the LiDAR data is performed by applying the same structure tensor as for the flow detection in Section 6.2. The only difference is the generation of the height field H. Since the terrain mesh is optimized for Level-of-Detail rendering, it is no londer regular and for efficient flow detection we need a regular grid as data structure for the height values. Therefore, we produce the needed height field by rendering the terrain from a top-down perspective with a orthographic projection to the depth buffer. This yields also the possibility to produce height fields at different resolutions in order to guarantee that the flow estimation is real-time capable on every target hardware that is used on flight decks.



(a) (b)

Figure 6.13: Feature tracking over multiple frames: (a) Feature points have been tracked over multiple frames and became opaque. (b) Feature points in the lower part of the image are the same as in (a) and are still handled as stable features. Non-stable features are rendered transparent. An example of stable features is highlighted with a blue box in (a) and (b).

The streamlines itself are produced by following the flow direction starting from the feature point until the slope of the terrain is below a certain user-defined threshold. Additionally, it is possible to integrate the streamlines orthogonal to the flow. The result of using different integration directions are depicted in Figure 6.14. A list of used parameters is given in Table 6.3.



(a) (b)

Figure 6.14: Streamline generation: (a) Starting from a feature point, the streamlines are generated along the terrain flow field. (b) The streamlines are generated orthogonal to the flow field. The red dots indicate the stable feature points.

Table 6.3: Parameters for streamline development.

Parameter	Value description
Binning size	The size of the 3D bins given in Figure 6.12(e). We use a value, based upon the accuracy of the LiDAR sensor.
Quality of feature points	The number of frames, a detected point needs to be stable in comparison to the updated 3D terrain information. A low value is useful for fast changes, while for higher quality a higher value is used. The threshold is scaled with the distance to the helicopter and with the flight speed. For high relative distances to the helicopter, streamlines can be detected later than in the direct surrounding. If the helicopter flies with high speed, feature points must be detected faster than during hovering.
Integration direction	Binary value, if the streamline is integrated along or orthogonal within the flow field.

Interpolation and Parameterization

Since the temporal coherence is given through the generation of streamlines, based upon temporal stable features, the spatial coherence is achieved by interpolating the height of each streamline segment for every terrain update. We use the same height field generation as for the ridge and valley detection by rendering the terrain from a top-down perspective with an orthographic projection to the depth buffer. In this way we determine for every position of a line segment $p_n = (x, y, z)$ within the streamlines and ridges/valleys its new position with a new height $p_{n+1} = (x, y, z')$. Between these two positions p_n and p_{n+1} a linear interpolation is applied. Additionally, lines are not rendered directly after their detection but are faded in over multiple frames that reduces the number of popping artifacts. This number of frames is dependent on the distance to the helicopter and can be adapted by the pilot.

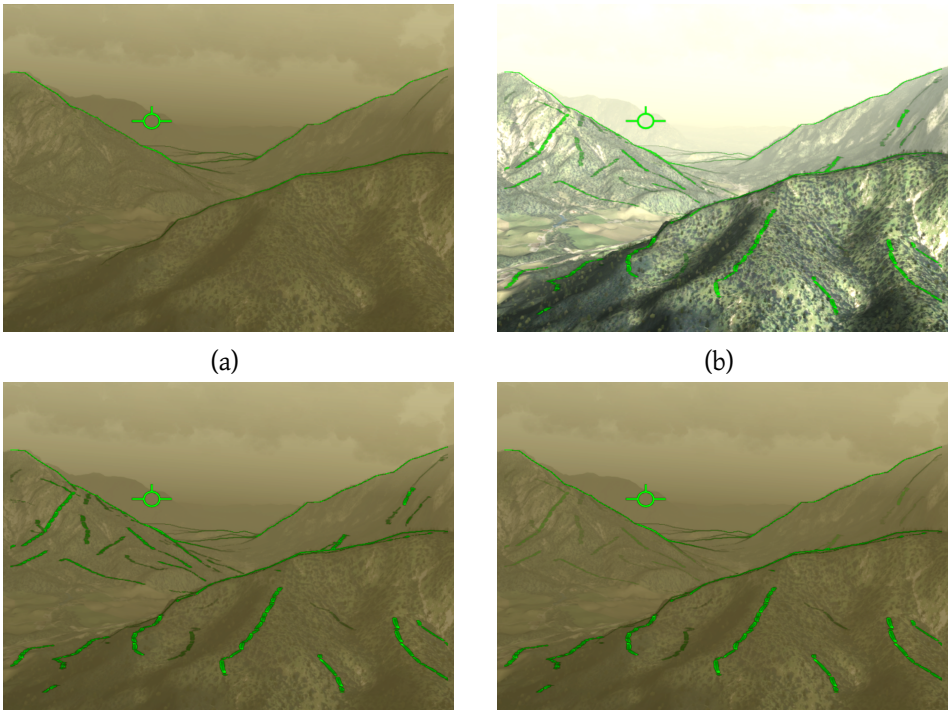
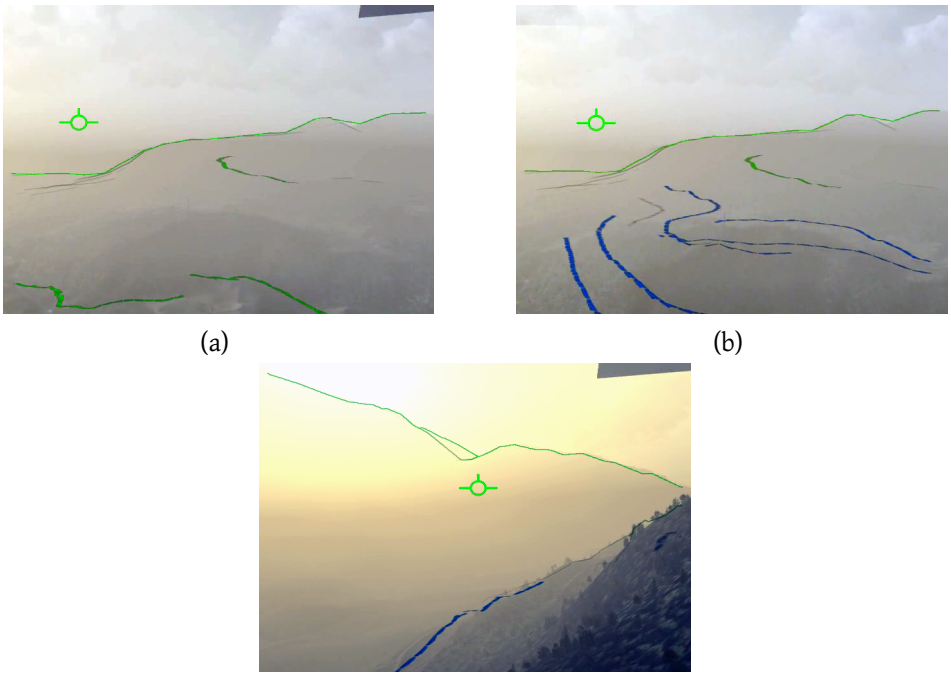


Figure 6.15: Ridge and valley line representation: (a) Terrain representation with only contourlines in DVE. (b) Clear view to the same mountainous scene with ridges and valleys complement the contour rendering. (c) Same rendering in DVE. A mental image of the underlying terrain can be derived without rendering to much information. (d) Background can be reduced due to importance of foreground structures. Contourlines are sufficient as landmarks.

Results

The ridge and valley lines are useful as an additional feature in the HMD without adding streamlines in order to provide a more natural representation of terrain to the pilot than rendering a mesh with a contour enhancement. The renderings from the flight simulator in Figure 6.15 support the conclusion that enhancing the contourline representation with ridge and valley lines provides a more natural depiction of the terrain. In order to merge the advantages of the ridge and valley detection mechanism and the streamline generation, we apply both to the terrain information provided by the synthetic vision system. This results in a rough terrain depiction by ridge and valley lines, depending on the used height field resolution as well as the algorithms parametrization. This is extended by streamlines that are independent from the terrain resolution since their seed points are detected on the depth buffer information of the rendered frame from the pilot's perspective after a terrain update. A final result of a terrain rendering in the HMD

is provided in Figure 6.16. In Figure 6.16(a) ridge and valley rendering with textures is depicted, in the Figures 6.16(b) and 6.16(c) the extension with streamlines is provided. Additional results provided in our flight simulator are given in Appendix B.2.



(a) (b)

(c)

Figure 6.16: Feature line representation: (a) Pre-processed ridge and valley lines. Details in the terrain representation are missing. (b) and (c) Extension with streamlines in order to visualize features in the terrain that are not detected by the ridge and valley processing as well as cannot be depicted by a view-dependent contour tracing mechanism [MSS+14].

6.4 Conclusion

We proposed a new approach for automatically generating a variety of stylized terrain illustrations by rendering hierarchical representations of streamline-based hatches. The outcome of this work could be partly used to extend a contourline-based rendering for DVE environments on HMDs during flight missions. We introduced a visualization technique of terrain for the pilot's HMD by rendering ridge and valley lines as well as streamlines that now can be evaluated within a flight simulator. Similar to the previous chapters, we introduced data structures for a terrain depicting system on HMDs that have the potential to be used on operational hardware for flight decks. This is reasonable due to the scalability of every data

structure. Accordingly, the height maps can be used for the streamline as well as for the ridge and valley detection on different resolutions, depending on the accessible amount of memory on the board computers. Additionally, every algorithm used for the HMD terrain visualization is implemented by using constant memory. Furthermore, our system allows for the user studies plenty of parameters, that have an intuitive influence on the final visualization. It is important to mention that these parameters are only supposed to be evaluated during such a user study and should not be accessible during a flight mission to avoid an increase of the pilot's workload. This needs a similar evaluation than the parameters provided in Section 5.2 for the non-classified cluster visualization.

7

Conclusion and Future Work

Efficiently providing information for DVE and avoiding hazardous situations on the HMD as well as on the HDD is one of the major problems within aircraft cockpits. The results of this thesis show that the combination of state-of-the-art computer graphics and computer vision techniques with requirements in modern flight decks are reasonable for future applications. For all our proposed techniques, we already set up a basis for an integration to an operational system by avoiding software libraries and providing a software implementation from scratch. Except for the techniques that were implemented as references, we also integrated our findings in a flight simulator environment as preliminary modules of the operational system. Since most of our approaches are experimental studies and not fully engineered solutions for operational use, the findings of our work have to be refactored according to software specifications and technical requirements in order to receive a certification and to integrate them into a specific flight deck.

We introduced a new Level-of-Detail-based ground classification algorithm that was already integrated to an operational system. Compared to existing operational methods that allow a local classification in flight systems, it yields the advantage that it allows a more global understanding of ground within multiple accumulated sensor frames. The proposed system is real-time capable and transferable to other sensor systems that deliver any kind of 3D point cloud.

Since the LiDAR point cloud is divided into different classes that are processed and rendered individually by meshing approaches or symbolical representations, the non-classified information that also need to be accessible to the pilot via rendering have to be analyzed and a shape has to be presented on the HMD. We provide for this problem a cluster fusion approach for introducing temporal coherence to such non-classified data points in order to divide them into spatial distinct groups. This approach is designed for 2D data but can simply be extended to work in higher dimensions. The differences in higher dimensions are the detection of the clusters'

convex hull and the calculation of the distance field. The cluster fusion is combined with a meshing method in combination with a more abstract representation in the form of only three lines as a Level-of-Detail approach. Both methods improve the state-of-the-art handling of non-classified obstacle depiction in flight decks. As an additional HMD-based rendering method we introduced a traffic visualization technique and a terrain representation along the landing zone. Both methods are a step towards a reduced and even more abstracted depiction. The main question for a future evaluation will be whether such abstract renderings with only indirect spatial cues are accepted by pilots for likewise abstract information such as traffic data.

As a subsequent question after the ground classification, is how terrain can be depicted in an HMD. To answer this question, we follow the question of how an artist would draw terrain in a non-photorealistic way. We transfer the resulting methods from the generation of these artistic images on arbitrary mountainous terrain meshes to the data structure used for terrain storing on a mission computer within a cockpit.

All rendering techniques have the need of a smooth interpolation to new received sensor information in order to provide spatio-temporal stability. To achieve this, interpolation capabilities are included in the data structure of the terrain representing lines and clusters shapes as well as for the ellipsoids used for terrain rendering along the landing site itself, so all rendered primitives can be handled independently.

7.1 Pilot feedback

As a preliminary study we conducted a questionnaire within a simulated test flight with a professional helicopter pilot. In this workshop we showed three different applications, namely the cluster meshing procedure on non-classified LiDAR data, the traffic visualization and the abstract terrain rendering method. The results are summarized in the following:

Cluster rendering:
The cluster rendering is mentioned to have its main application in urban sceneries with clear distinct clusters. In land side areas with large number of clusters with woods or slopes such as in mountainous terrains, such a representation is not sufficient. The boxes either become too large, if the cluster parameters or the rendering parameters are not set correctly, or lead to cluttering. A more organic representation based upon different meshing procedures such as alpha shapes as well as direct rendering of LiDAR points makes more sense. The Level-of-Detail

approach by rendering objects next to the helicopter in high quality and the use of the eigenvector representation for objects in high distances as well as the fading process between these two representations according to the navigational terms described in Chapter 2 found a high acceptance. One open problem is the acceptance of detailed object meshes for non-classified obstacles with low complexity such as houses. The pilot did not need to know whether such an object has a balcony. In such situations axis-aligned boxes are sufficient for abstraction. By contrast, a higher degree of detailedness for objects such as the helicopter in Figure 5.7 helps the pilot to avoid obstacles with a certain operational meaning. Another HMI aspect is that the distance to the helicopter alone is not a measure for endangerment. If the pilot's focus lies on a power line, the representation of the building next to him does not need to be spatio-temporal interpolated. Therefore, exhaustive studies to find decision rules for Point-of-Interest identification and processing for rendering optimization have to be achieved by using methods such as eye-tracking or psycho-physiological experiments.

Traffic Visualization:
Similar to the cluster rendering, the fading together with the smooth interpolation found acceptance by the pilot. The clustering based upon different proximal and nominal features allows abstraction of the information given by the traffic system in a way that is most comfortable for the pilot. Within our test setup, the clustering was too sensitive to the Euclidean distance. This issue is addressed by using an adapted parameterization set. The avoidance of interpolation by changing the ground state to an air traffic state symbol was very appreciated since such a popping effect leads the pilot's attention to an important event for its mid-term guidance task. The pilot also mentioned, that additional information could increase the usability of such a system. Symbologies such as the NATO-cross for helicopters could be introduced to extend the ellipsoidal shape. Such symbols increase the intuitive usage of such a system. As an example for a use case the following situation could be imagined: multiple helicopters and ground vehicles close to each other in the surrounding of the helicopter is an implication for a landing site. With additional aids such as information about equipment or function such as weaponry and ambulance, an implication about the situation and possibility for a landing approach can be derived. This is obviously highly dependent on the traffic system. ADS-B as a civilian system is not supposed to provide such information apart form emergency information in the future.

Terrain Rendering:
According to the pilot, the usage of additional cues to the contour rendering that is already deployed with the sensing and visualization system for DVE situations

yields a few advantages: even if the contours derived from the depth buffer image of the synthetic vision system are a feature that increases the pilot's situational awareness it does not depict all terrain features. This is the main reason why the contourlines are mostly extended with a decluttered mesh. It must be mentioned that the mesh sometimes has the disadvantage that it implicates the suggestion that the terrain follows a certain direction induced by the two main directions of the mesh. The rendering of the streamlines together with the ridge and valley lines are a step in the right direction by preserving important features of the terrain as well as following a more natural direction. This helps the pilot to create a correct mental image of the terrain. Nonetheless, the decluttering remains an open issue, induced by the use of textures. Furthermore, the orientation of the textures is an additional aid, since these structures are not necessarily detected by the contour detector.

A common finding for all three approaches is that the spatio-temporal coherence and the goal of rendering in a certain kind of abstraction as well as supporting the creation of a mental image of the environment yield promising techniques for operational and certified applications in future.

7.2 Future Work

With this thesis, different methods are proposed that increase the pilot's situational awareness, based upon different data sources. Our first contribution is the ground classification method presented in Chapter 3 that is highly specific to the used LiDAR sensor and thus can be extended to work on data from different sensors. This requires specific adaption in order to meet the sensor specifications according to the frequency and number of data points. We have already shown that the ground classification is transferable to any kind of point cloud and provided an implementation for a car-mounted sensor system. There might also be other application cases for this system on certain sensors that are used for landing approaches to avoid a rotor strike, for instance.

The clustering approach presented in Chapter 4 together with their visualization described in Section 5.2 yield the opportunity for additional research. Further analysis according the visualization in alternative styles such as alpha shapes for non-urban sceneries as well as the structure of the clusters, based upon the approximated cluster shapes, and their position in combination with distinction between different flight phases will help to derive information about the impact of a non-classified obstacle to the pilot. For instance, it is not necessary to render obstacles directly below the helicopter during a cruise phase while obstacles that are usable as landmarks might be important in higher distances during DVE. For landing

approaches, the exact opposite is the case. This results in the requirement of providing a flight analysis approach for automatically classifying the current flight state such as hovering, nap-on-the-earth flights, landing approaches or following a given trajectory during cruise phase. Furthermore, different flight phases need a different handling of clusters and thus it is necessary to derive additional fusion and split rules in the form of Rvachev functions for different kinds of clusters. Also it is important to provide more exhaustive research about and how uncertainty visualizations inspired from geovisualization techniques could improve renderings in the Head-Mounted Display. The implementation of renderings that are based upon such geovisualizations and meet the requirements according computation speed and memory consumption as well as its evaluation are left as future work.

For the terrain depiction we have a similar future issue: in which kind of situations are systems, with contourlines, ridge and valley lines and streamlines being an example, combined in an optimal way? Until now, we provided answers on the more high-level question how we can achieve temporal and spatial stability for terrain as well as cluster representation. One step in the direction of situation analysis and function development for special flight phases is the possibility to remove feature lines within the visible range of the pilot.

Since the traffic visualization described in Section 5.3 is a well-defined problem statement with a clear input data specification, its use needs to be evaluated in comparison to other deployed applications such as the head-down approach and TCAS II. This encounters a detailed study concerning the size of the gap between abstraction and depiction of how the reality can be. Furthermore, this inherits whether such a basic representation with only ellipses and rectangles can be used for not only depicting complex situations like formation flights but also for recognition of landing sites through a group of helicopters that are stationary on the ground. In addition, this method is in a state that can be tested in under real conditions during flight.

One step in the direction of creating additional aids during specific flight phases is the landing approach with the virtual landing aid. In Section 5.4 we showed how such a rendering can be enhanced with further information in order to avoid potential dangers or depicting alternative landing sites. In this context it has to be evaluated which configuration of additional landing cues is more intuitive and provides a real help to the pilot.

In general, future work will be focused on the different flight situations from takeoff to landing with all possible intermediate phases such as climbing, descending and missed approaches. For each of these situations, it must be analyzed which information is necessary and how our data processing chain and visualizations can be applied. This is an analogy to the landing aid that is already deployed for operational use as given in Figure 1.3. To generate an optimal rendering of

this symbology, multiple workshops with pilots were performed. In each of the workshops, improvements were derived and integrated for subsequent workshops. Moreover, there are different kinds of sceneries that need to be analyzed in the same way with different combinations of our provided data processing and rendering methods together with their different parameterizations in order to be able to generate optimal results to reduce the pilot's workload and increase its situational awareness.

Appendix A

Remarks on Regularization

This appendix provides a detailed solution of the equation systems provided in Chapter3. Also additional results are provided in Appendix A.3.

A.1 Derivation of the Regularization Solution

In order to solve the minimization problem of the functional given in Equation 3.2:

$$H[f] = \int E[f(x,y), f_x(x,y), f_y(x,y), x, y]dxdy, \tag{A.1}$$

we use the Euler-Lagrange Equation for the following conditions:

- One dependent variable: the function $f(x,y)$.
- Two independent variables: the coordinates x and y.
- First order derivatives used for regularization: f_x and f_y.

According to Courant and Hilbert [CH62], the derived Euler-Lagrange Equation from these conditions is:

$$\frac{\partial E}{\partial f} - (\frac{\partial}{\partial x}\frac{\partial E}{\partial f_x} + \frac{\partial}{\partial x}\frac{\partial E}{\partial f_y}) = 0. \tag{A.2}$$

Since the Euler-Lagrange equation is calculated only with the integration variables, E is defined as

$$E = \frac{1}{2}\kappa(x_i, y_i)[f(x_i, y_i) - (d_z)_i]^2 + \frac{1}{2}\lambda(f_x^2(x,y) + f_y^2(x,y)). \tag{A.3}$$

In this Equation, the two terms *Data Term* and *Model Term* are handled as equivalent elements within the optimization problem statement. In addition, it helps us

to achieve the partial derivatives for the Euler-Lagrange equations without any additional coefficients. The partial derivatives are:

$$\frac{\partial E}{\partial f} = \kappa(x_i, y_i)(f(x_i, y_i) - (d_z)_i), \tag{A.4}$$

$$\frac{\partial E}{\partial f_x} = \lambda f_x(x, y) \rightarrow \frac{\partial}{\partial x}\frac{\partial E}{\partial f_x} = \lambda f_{xx}(x, y), \tag{A.5}$$

$$\frac{\partial E}{\partial f_y} = \lambda f_y(x, y) \rightarrow \frac{\partial}{\partial x}\frac{\partial E}{\partial f_y} = \lambda f_{yy}(x, y). \tag{A.6}$$

This results in the following equation system:

$$\kappa(x_i, y_i)(f(x_i, y_i) - (d_z)_i) - \lambda(f_{xx}(x, y) + f_{yy}(x, y)) = 0. \tag{A.7}$$

By providing a function δ that has the same domain as f and κ, the parameters can be removed and the resulting Equation is

$$\kappa(f - \delta) - \lambda(f_{xx} + f_{yy}) = 0. \tag{A.8}$$

Since $f_{xx} + f_{yy}$ is equivalent to the Laplace operator, these values can be replaced by the differential operator Δ. After the sign of this Equation is turned for numerical solution, the Equation 3.5 is fully derived. Since Δ is equivalent to the *Model Term*, the Laplace operator can be accessible for the linear equation system in the form of the Matrix M in Equation 3.7.

A.2 Derivation of the Iterative Regularization Solution

By using the Euler method for solving ordinary differential equations, an iterative calculation specification can be provided [But03]. For solving of the initial value problem

$$y'(t) = f(t, y(t)) \text{ and } y(t_0) = y_0. \tag{A.9}$$

By choosing a value ∇t for the step size and by setting $t^k = t^0 + \nabla t k$, one step of the Euler method from t^k to $t^{k+1} = t^k + \nabla t$ is

$$y^{k+1} = y^k + \nabla t f(t^k, y^k). \tag{A.10}$$

By inserting Equation 3.5 into this equation, we receive as result Equation 3.9.

A.3 Additional Classification Results

In addition to Figure 3.6, we provide some more results without ground truth information for different classification approaches.

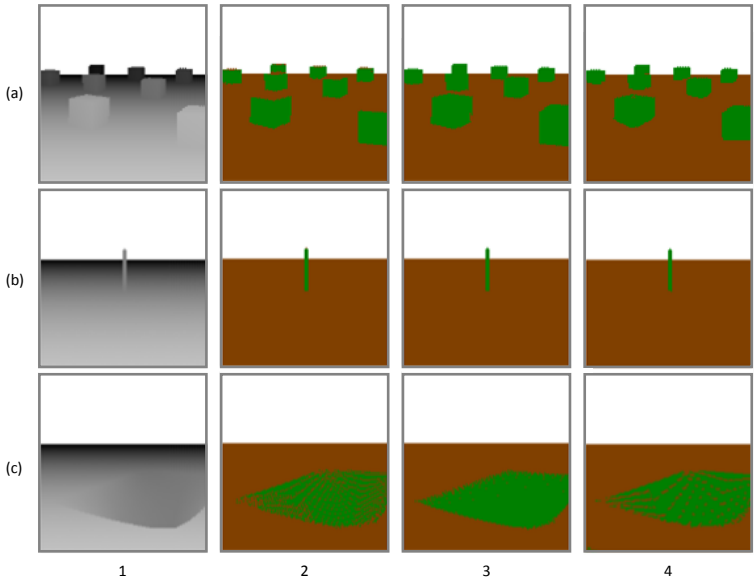


Figure A.1: Classification results on perfect synthetic sensor images. Column 1 shows range images. Columns 2-4 show the classification result for operational, triangulation, and our method.

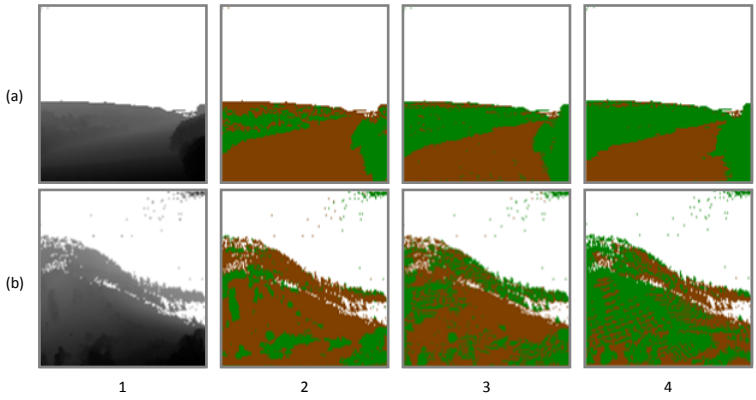


Figure A.2: Classification results on real sensor images. Column 1 shows range images. Columns 2-4 show the classification result for operational, triangulation, and our method.

Appendix B

Remarks on Rendering Results

In this appendix additional results for the visualization of non-classified LiDAR clusters and terrain renderings within the flight simulator are provided.

B.1 Additional Cluster Visualization Results

In addition to Figure 5.7, we provide some more results within our flight simulator.

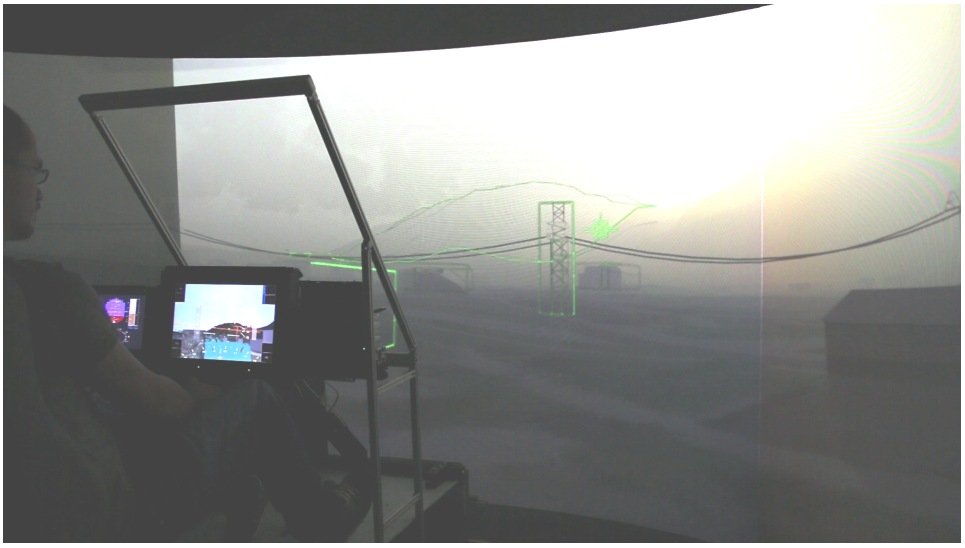


Figure B.1: Box meshing on a HMD in simulator environment: Objects far away are faded out while the most important shape due to the distance is rendered in full opacity.

Figure B.2: Box meshing with additional distance cue on a HMD in simulator environment: Objects far away are faded out and replaced by a symbol representing the principal components of the cluster in a simulated DVE scene.

B.2 Additional Terrain Visualization Results

In addition to Figure 6.15 and 6.16, we provide some more results within our flight simulator.

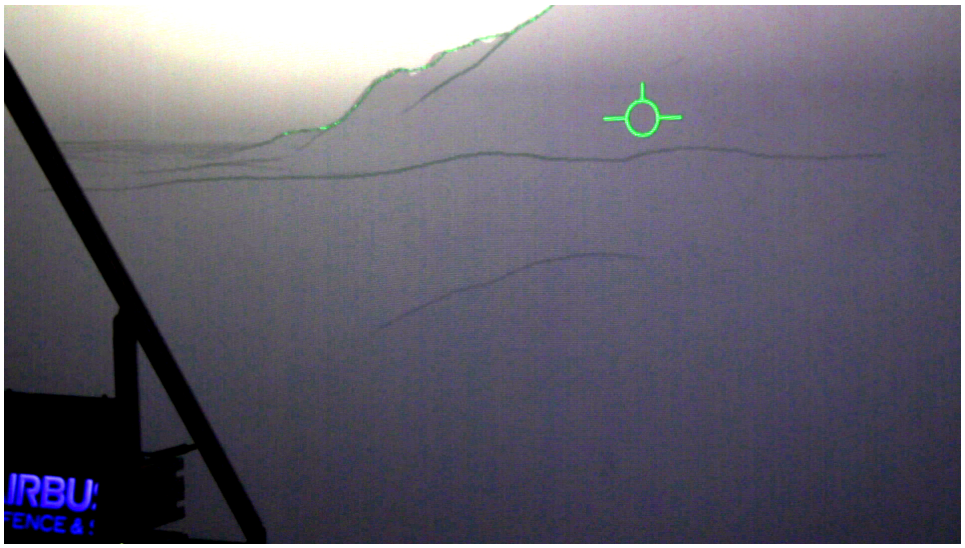


Figure B.3: DVE situation with only contour enhancement.

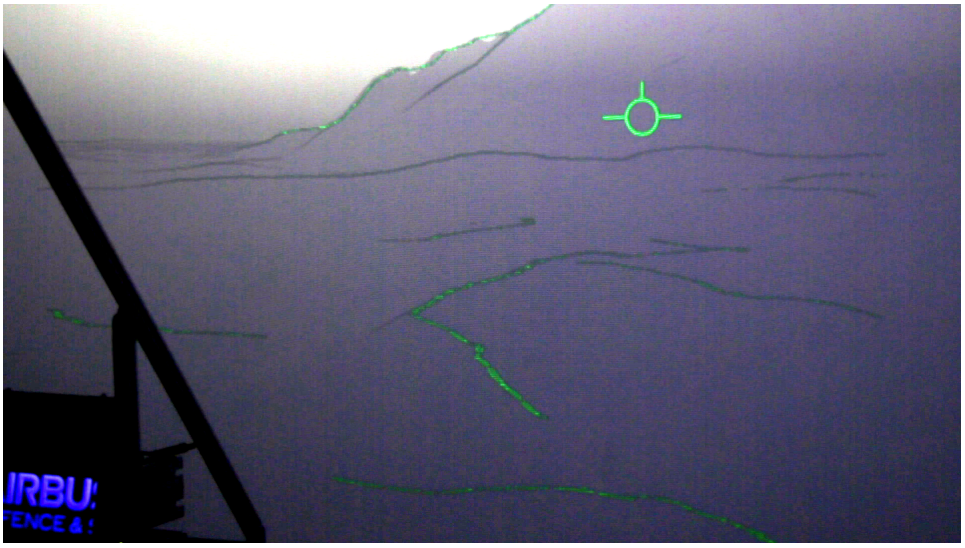


Figure B.4: Contour enhancement extended with only ridge lines.

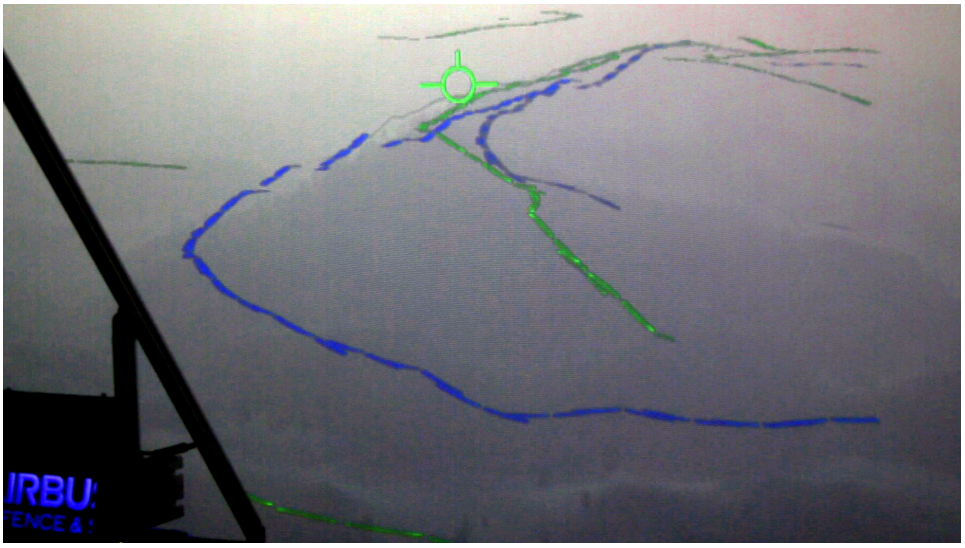


Figure B.5: Contour enhancement with only ridge lines extended with additional streamlines after a few seconds of flight.

Bibliography

[ABK06] E. Achtert, C. Böhm, and P. Kröger. Deli-clu: Boosting robustness, completeness, usability, and efficiency of hierarchical clustering by a closest pair ranking. In *Advances in Knowledge Discovery and Data Mining*. 2006. (Cited on page 33.)

[ABKS99] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. *OPTICS*: Ordering Points to Identify the Clustering Structure. *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, 1999. (Cited on pages 33 and 37.)

[Aer] Aero Glass. Aero Glass. `http://glass.aero`. 08.10.2015. (Cited on page 44.)

[AH05] H. Arefi and M. Hahn. A Morphological Reconstruction Algorithm for Separating Off-terrain Points from Terrain points in Laser Scanning Data. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume 36 (3/W19*, 2005. (Cited on page 17.)

[AHWY03] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A Framework for Clustering Evolving Data Streams. *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, 2003. (Cited on page 33.)

[Air] Airbus Defence and Space. Cassidian's Obstacle Warning System Certified for NH90 Helicopters. `http://www.defenceandsecurity-airbusds.com/pl_PL/web/guest/obstacle-warning-system-certified-for-nh90-helicopters`. 17.10.2015. (Cited on page 2.)

[AMR+12] M. R. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler. StreamKM++: A Clustering Algorithm for Data Streams. *J. Exp. Algorithmics*, 2012. (Cited on page 33.)

[Ast] Astronics Company. Max-Viz. `http://www.max-viz.com`. 07.10.2015. (Cited on page 44.)

[Avi11] U. Aviation. *Installation of Terrain Awareness and Warning System Approved for Part 23 Airplanes*. General Books, 2011. (Cited on page 31.)

[AW10] H. Abdi and L. J. Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2010. (Cited on page 63.)

[Axe99] P. Axelsson. Processing of Laser Scanner Data -Algorithms and Applications. *ISPRS Journal of Photogrammetry & Remote Sensing*, 1999. (Cited on pages 17 and 81.)

[Axe00] P. Axelsson. DEM Generation from Laser Scanner Data using Adaptive TIN Models. *International Archives of Photogrammetry and Remote Sensing*, 2000. (Cited on pages 17, 18, 24, and 25.)

[BCP+12] A. Brambilla, R. Carnecky, R. Peikert, I. Viola, and H. Hauser. Illustrative Flow Visualization: State of the Art, Trends and Challenges. *EG 2012 - State of the Art Reports*, 2012. (Cited on page 81.)

[BEA83] P. J. Burt, Edward, and E. H. Adelson. The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communications*, 1983. (Cited on page 17.)

[BP01] C. Briese and N. Pfeifer. Airborne Laser Scanning and Derivation of Digital Terrain Models. *5th Conference on Optical 3D Measurement Techniques*, 2001. (Cited on page 17.)

[BSD+04] K. Buchin, M. C. Sousa, J. Döllner, F. Samavati, and M. Walther. Illustrating Terrains using Direction of Slope and Lighting. *4th ICA Mountain Cartography Workshop*, 2004. (Cited on page 79.)

[BST09] M. Bratkova, P. Shirley, and W. B. Thompson. Artistic Rendering of Mountainous Terrain. *ACM Trans. Graph.*, 2009. (Cited on page 79.)

[But03] J. Butcher. *Numerical Methods for Ordinary Differential Equations*. Springer, 2003. (Cited on page 104.)

[Cab08] S. J. Cabler. Technical Standard Order. Subject: Helicopter Terrain Awareness and Warning System. Technical report, Department of Transportation, 2008. (Cited on page 87.)

[CAGZ09] M. Carlberg, J. Andrews, P. Gao, and A. Zakhor. Fast Surface Reconstruction and Segmentation with Ground-Based and Airborne LiDAR Range Data. Technical Report UCB/EECS-2009-5, 2009. (Cited on page 16.)

[Can86] J. Canny. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1986. (Cited on page 91.)

[CCDH05] C. Colditz, L. Coconu, O. Deussen, and H.-C. Hege. Real-time rendering of complex photorealistic landscapes using hybrid level-of-detail approaches. *Trends in Real-Time Landscape Visualization and Participation*, 2005. (Cited on page 10.)

[CCK07] Y. Chen, J. Cohen, and J. Krolik. Similarity-Guided Streamline Placement with Error Evaluation. *Visualization and Computer Graphics, IEEE Transactions on*, 2007. (Cited on page 83.)

[CEQZ06] F. Cao, M. Ester, W. Qian, and A. Zhou. Density-Based Clustering over an Evolving Data Stream with Noise. *Proceedings of the 2006 SIAM International Conference on Data Mining*, 2006. (Cited on page 34.)

[CGG+03] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno. BDAM - Batched Dynamic Adaptive Meshes for High Performance Terrain Visualization. *Computer Graphics Forum*, 2003. (Cited on page 81.)

[CH62] R. Courant and D. Hilbert. *Methods of Mathematical Physics*. Interscience Publishers, 1962. (Cited on page 103.)

[Cob] Cobham Avionics. 3D Synthetic Vision EFIS. `https://www.cobham.com/media/103150/efis_for_fixedwing_2009.pdf`. 07.10.2015. (Cited on page 44.)

[Coe11] F. Coenen. Data mining: past, present and future. *The Knowledge Engineering Review*, 2011. (Cited on page 32.)

[CT07] Y. Chen and L. Tu. Density-based Clustering for Real-time Stream Data. *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007. (Cited on page 34.)

[DB02] G. Donato and S. Belongie. Approximation Methods for Thin Plate Spline Mappings and Principal Warps. *ECCV 2002: 7th European Conference on Computer Vision*, 2002. (Cited on pages 45 and 47.)

[DDP] Dutch Defence Press. `http://www.dutchdefencepress.com/gezonde-spanning-aan-de-deur-en/`. 02.11.2015. (Cited on page 2.)

[DKLP01] S. Djurcilov, K. Kim, P. F. J. Lermusiaux, and A. Pang. Volume Rendering Data with Uncertainty Information. *Data Visualization 2001: Proceedings of the Joint Eurographics - IEEE TCVG Symposium on Visualization*, 2001. (Cited on page 52.)

[Döl07] J. Döllner. Non-Photorealistic 3D Geovisualization. In *Multimedia Cartography*. 2007. (Cited on page 79.)

[DSL14] H.-U. Doehler, S. Schmerwitz, and T. Lueken. Visual-conformal Display Format for Helicopter Guidance. *Proc. SPIE 9087, Degraded Visual Environments: Enhanced, Synthetic, and External Vision Solutions*, 2014. (Cited on page 2.)

[EES+15] F. Eisenkeil, J. Ernst, R. Stadelhofer, U. Kühne, and O. Deussen. Traffic Visualization in Helmet-Mounted Displays in Synchronization with Navigation Displays. *Digital Avionics Systems Conference (DASC), 2015 IEEE/AIAA 34th*, 2015. (Cited on pages 7, 58, 59, 60, 62, 64, 65, 67, 68, 70, 71, 72, and 73.)

[EJL+01] M. Elmqvist, E. Jungert, F. Lantz, Å. Persson, and U. Söderman. Terrain Modelling and Analysis using Laser Scanner Data. *International Archives of Photogrammetry and Remote*, 2001. (Cited on pages 18 and 23.)

[EKSX96] M. Ester, H.-P. Kriegel, J. S, and X. Xu. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, 1996. (Cited on pages 33 and 35.)

[Elm02] M. Elmqvist. Surface Estimation from Airborne Laser Scanner Data using Active Shape Models. *Photogrammetric Computer Vision-ISPRS Commission III Symposium, vol. XXXIV, part A*, 2002. (Cited on page 18.)

[ESKD13] F. Eisenkeil, T. Schafhitzel, U. Kühne, and O. Deussen. Real-time Classification of Ground from LiDAR Data for Helicopter Navigation. *Proc. SPIE 8745, Signal Processing, Sensor Fusion, and Target Recognition XXII*, 2013. (Cited on pages 6, 19, 22, 23, 24, 25, 26, and 27.)

[ESKD14] F. Eisenkeil, T. Schafhitzel, U. Kühne, and O. Deussen. Clustering and Visualization of Non-classified Points from LiDAR Data for Helicopter Navigation. *Proc. SPIE 9091, Signal Processing, Sensor/Information Fusion, and Target Recognition XXIII*, 2014. (Cited on pages 6, 33, 38, 40, 41, and 42.)

[EUR] EUROCONTROL. Surveillance: Summary of Terminology. `http://www.eurocontrol.int/articles/summary-terminology`. 20.04.2015. (Cited on pages 56 and 57.)

[EUR14] EUROCONTROL. *ACAS II Guide: Airborne Collision Avoidance System II (incorporating version 7.1)*. 2014. (Cited on page 57.)

[Fed14] Federal Aviation Administration. NextGen Implementation Plan, 2014. (Cited on page 55.)

[Fey15] T. Feyereisen. SmartView Lower minimums: A Synthetic Vision Guidance System. *Digital Avionics Systems Conference (DASC), 2015 IEEE/AIAA 34th*, 2015. (Cited on page 44.)

[FH04] P. F. Felzenszwalb and D. P. Huttenlocher. Distance Transforms of Sampled Functions. Technical report, Cornell Computing and Information Science, 2004. (Cited on page 40.)

[FHT08] J. Fischer, M. Haller, and B. Thomas. Stylized Depiction in Mixed Reality. *International Journal of Virtual Reality*, 2008. (Cited on page 79.)

[Fil02] S. Filin. Surface Clustering from Airborne Laser Scanning Data. *In International Archives of Photogrammetry and Remote Sensing, Vol. XXXII, 3A*, 2002. (Cited on page 16.)

[FPSS96] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 1996. (Cited on page 32.)

[Gara] Garmin. G1000. `https://buy.garmin.com/en-GB/GB/aviation/flight-decks/g1000-/prod6420.html`. 07.10.2015. (Cited on page 44.)

[Garb] Garmin. SVT: Synthetic Vision Technology: A virtual revolution in visual flight reference. `www8.garmin.com/aviation/brochures/02577SVTBrochure/02577SVTBrochure.pdf`. 21.01.2015. (Cited on page 44.)

[GG07] G. Guennebaud and M. Gross. Algebraic Point Set Surfaces. *ACM SIGGRAPH 2007 Papers*, 2007. (Cited on pages 45 and 47.)

[GJP95] F. Girosi, M. Jones, and T. Poggio. Regularization Theory and Neural Networks Architectures. *Neural Computation*, 1995. (Cited on page 19.)

[Gow71] J. C. Gower. A General Coefficient of Similarity and Some of Its Properties. *Biometrics*, 1971. (Cited on page 62.)

[Gra72] R. L. Graham. An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set. *Inf. Process. Lett.*, 1972. (Cited on pages 37 and 45.)

[GRL11] J. a. Gama, P. P. Rodrigues, and L. Lopes. Clustering Distributed Sensor Data Streams Using Local Processing and Reduced Communication. *Intell. Data Anal.*, 2011. (Cited on page 34.)

[Gul] Gulfstream. Enhanced Vision System. `http://www.gulfstream.com/product-support/product-enhancements/enhanced-vision-system`. 07.10.2015. (Cited on page 44.)

[HH01] R. A. Haugerud and D. J. Harding. Some Algorithms for Virtual Deforestation (VDF) of LiDAR Topographic Survey Data. *International Archives of Photogrammetry and Remote Sensing*, 2001. (Cited on page 17.)

[HH07] J. Hubbard and B. Hubbard. *Vector Calculus, Linear Algebra, and Differential Forms: A Unified Approach*. Prentice-Hall, 2007. (Cited on page 82.)

[Hop96] H. Hoppe. Progressive Meshes. *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, 1996. (Cited on pages 10, 81, and 89.)

[Hor86] B. Horn. *Robot Vision*. McGraw-Hill Higher Education, 1986. (Cited on page 91.)

[HWCO+13] H. Huang, S. Wu, D. Cohen-Or, M. Gong, H. Zhang, G. Li, and B.Chen. L1-Medial Skeleton of Point Cloud. *ACM Transactions on Graphics*, 2013. (Cited on page 45.)

[ICA10] ICAO. *Doc 9924: Aeronautical Surveillance Manual*. 2010. (Cited on page 56.)

[Jar73] R. A. Jarvis. On the Identification of the Convex Hull of a Finite Set of Points in the Plane. *Information Processing Letters*, 1973. (Cited on page 63.)

[JL03] K. Jacobsen and P. Lohmann. Segmented Filtering of Laser Scanner DSMS, 2003. (Cited on page 17.)

[KABS11] P. Kranen, I. Assent, C. Baldauf, and T. Seidl. The ClusTree: Indexing Micro-clusters for Anytime Stream Mining. *Knowledge and Information Systems*, 2011. (Cited on page 34.)

[Kai04] J. Kaiser. *Verwendung stereoskopischer Informationsdarstellung in durchsichtfähigen Anzeigen am Beispiel eines Head-Up Displays*. Ergonomia Verlag, 2004. (Cited on page 87.)

[KAS+08] D. B. Kaber, A. L. Alexander, E. M. Stelzer, S. H. Kim, K. Kaufmann, and S. Hsiang. Perceived Clutter in Advanced Cockpit Displays: Measurement and Modeling with Experienced Pilots. *Aviation, space, and environmental medicine*, 2008. (Cited on page 13.)

[KK06] P. Kennelly and A. Kimerling. Non-Photorealistic Rendering and Terrain Representation. *Cartographic Perspectives*, 2006. (Cited on page 79.)

[KP98] K. Kraus and N. Pfeifer. Determination of terrain models in wooded areas with airborne laser scanner data. *ISPRS Journal of Photogrammetry & Remote Sensing*, 1998. (Cited on page 17.)

[KP01] K. Kraus and N. Pfeifer. Advanced DTM Generation from LiDAR Data. *In International Archives of Photogrammetry and Remote Sensing, Vol XXXIV, 3/W4*, 2001. (Cited on page 17.)

[KPK+11] S.-H. Kim, L. J. Prinzel, D. B. Kaber, A. L. Alexander, E. M. Stelzer, K. Kaufmann, and T. Veil. Multidimensional Measure of Display Clutter and Pilot Performance for Advanced Head-up Display. *Aviation, space, and environmental medicine*, 2011. (Cited on page 13.)

[KWT88] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *INTERNATIONAL JOURNAL OF COMPUTER VISION*, 1988. (Cited on pages 15 and 18.)

[Lag] C. Lagan. 3 Coast Guardsmen lost in helicopter crash off Washington. `http://coastguard.dodlive.mil/2010/07/3-coast-guardsmen-lost-in-helicopter-crash-off-washington/`. 24.04.2015. (Cited on page 55.)

[Len06] P. M. Lenhart. *Räumliche Darstellung von Flugführungsinformationen in Head-Mounted Displays*. ergonomia Verlag, 2006. (Cited on page 87.)

[LFP13] O. Lemoine, J.-M. François, and P. Point. Contribution of TopOwl head mounted display system in degraded visual environments. *Proc. SPIE*, 2013. (Cited on page 4.)

[LH04] F. Losasso and H. Hoppe. Geometry Clipmaps: Terrain Rendering Using Nested Regular Grids. *ACM SIGGRAPH 2004 Papers*, 2004. (Cited on pages 10 and 81.)

[LHS08] L. Li, H.-H. Hsieh, and H.-W. Shen. Illustrative Streamline Placement and Visualization. *Visualization Symposium, 2008. PacificVIS '08. IEEE Pacific*, 2008. (Cited on pages 81, 82, and 88.)

[LJAM00] K. Lynda J. and B. Anthony M. Comparison of Pilots" Situational Awareness While Monitoring Autoland Approaches Using Conventional and Advanced Flight Display Formats. Technical report, NASA, 2000. (Cited on page 43.)

[LKS00] P. Lohmann, A. Koch, and M. Schaeffer. Approaches to the Filtering of Laser Scanner Data. *IAPRS*, 2000. (Cited on page 17.)

[LLSF08] W.-L. Lu, J. Little, A. Sheffer, and H. Fu. Deforestation: Extracting 3D Bare-Earth Surface from Airborne LiDAR Data. *Computer and Robot Vision, 2008. CRV '08. Canadian Conference on*, 2008. (Cited on page 17.)

[Low04] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision*, 2004. (Cited on page 91.)

[LP02] P. Lindstrom and V. Pascucci. Terrain Simplification Simplified: A General Framework for View-Dependent Out-of-Core Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2002. (Cited on page 81.)

[LwS07] L. Li and H. wei Shen. Image-based streamline generation and rendering. *IEEE Trans. Visualization and Computer Graphics*, 2007. (Cited on page 81.)

[Mac67] J. Macqueen. Some Methods for Classification and Analysis of Multivariate Observations. *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 1967. (Cited on page 32.)

[MCZ10] X. Meng, N. Currit, and K. Zhao. Ground Filtering Algorithms for Airborne LiDAR Data: A Review of Critical Issues. *Remote Sensing*, 2010. (Cited on page 16.)

[Mel12] J. E. Melzer. HMDs as Enablers of Situation Awareness, the OODA Loop and Sensemaking. *Proc. SPIE 8383, Head- and Helmet-Mounted Displays XVII; and Display Technologies and Applications for Defense, Security, and Avionics VI*, 2012. (Cited on page 2.)

[MKK13] T. Münsterer, M. Kress, and S. Klasen. Sensor based 3D Conformal Cueing for Safe and Reliable HC Operation Specifically for Landing in DVE. *Proc. SPIE*, 2013. (Cited on page 74.)

[MNC13] C. Munoz, A. Narkawicz, and J. Chamberlain. A TCAS-II Resolution Advisory Detection Algorithm. *AIAA Guidance, Navigation, and Control (GNC) Conference*, 2013. (Cited on page 70.)

[Mow14] J. E. Mower. Fast Image-Space Silhouette Extraction for Non-Photorealistic Landscape Rendering. *Transactions in GIS*, 2014. (Cited on page 80.)

[MSS+14] T. Münsterer, T. Schafhitzel, M. Strobel, P. Völschow, S. Klasen, and F. Eisenkeil. Sensor-enhanced 3D Conformal Cueing for Safe and Reliable HC Operation in DVE in all Flight Phases. *Proc. SPIE 9087, Degraded Visual Environments: Enhanced, Synthetic, and External Vision Solutions*, 2014. (Cited on pages 2, 4, 7, 86, 87, and 95.)

[MV02] R. C. Mat and M. Visvalingam. Effectiveness of Silhouette Rendering Algorithms in Terrain Visualisation. *Proceeding National Conference on Computer Graphics and Multimedia (CoGRAMM; Melaka, October 2002)*, 2002. (Cited on page 80.)

[MZL+09] R. Mehra, Q. Zhou, J. Long, A. Sheffer, A. Gooch, and N. J. Mitra. Abstraction of Man-Made Shapes. *ACM Transactions on Graphics*, 2009. (Cited on pages 45 and 48.)

[Nat] National Defense Magazine. Sandblaster2555. `http://www.nationaldefensemagazine.org/archive/2007/August/Pages/Sandblaster2555.aspx`. 07.10.2015. (Cited on page 44.)

[OGG09] C. Oztireli, G. Guennebaud, and M. Gross. Feature Preserving Point Set Surfaces based on Non-Linear Kernel Regression. *Computer Graphics Forum*, 2009. (Cited on pages 45 and 47.)

[OMM+02] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. Streaming-data Algorithms for High-quality Clustering. *Data Engineering, 2002. Proceedings. 18th International Conference on*, 2002. (Cited on page 33.)

[Pad07] G. Padfield. *Helicopter Flight Dynamics: The Theory and Application of Flying Qualities and Simulation Modelling*. Blackwell Publishing Ltd, 2007. (Cited on page 9.)

[PM00] D. Pelleg and A. W. Moore. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000. (Cited on page 32.)

[PTK85] T. Poggio, V. Torre, and C. Koch. Computational Vision and Regularization Theory. *Nature*, 1985. (Cited on page 19.)

[Rad] Radio Bremen. Notfälle auf hoher See: Bergung mit dem Heli. `http://www.radiobremen.de/wissen/themen/rettungsleitstelle-offshore100.html`. 24.04.2015. (Cited on page 55.)

[RD05] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. *IEEE International Conference on Computer Vision*, 2005. (Cited on pages 75 and 91.)

[RGP08] P. Rodrigues, J. Gama, and J. Pedroso. Hierarchical Clustering of Time-Series Data Streams. *Knowledge and Data Engineering, IEEE Transactions on*, 2008. (Cited on page 34.)

[Roca] Rockwell Collins. EVS 3000. `https://www.rockwellcollins.com/Data/Products/Displays/Head-Up_Displays-HUD/EVS-3000_Enhanced_Vision_System.aspx`. 07.10.2015. (Cited on page 44.)

[Rocb] Rockwell Collins. Pro Line Fusion. `http://www.rockwellcollins.com/Data/Products/Integrated_Systems/Flight_Deck/Pro_Line_Fusion.aspx`. 07.10.2015. (Cited on page 44.)

[RTC82] DO-178B: Software Considerations in Airborne Systems and Equipment Certification. *Radio Technical Commission for Aeronautics (RTCA)*, 1982. (Cited on page 12.)

[Rva63] V. Rvachev. On the Analytical Description of Some Geometric Objects. *Reports of Ukrainian Academy of Sciences, vol. 153, no. 4, 1963, pp. 765–767 (in Russian)*, 1963. (Cited on page 39.)

[RvdHV06] T. Rabbani, F. A. van den Heuvelb, and G. Vosselman. Segmentation of Point Clouds using Smoothness Constraint. *ISPRS Image Engineering and Vision Metrology*, 2006. (Cited on page 16.)

[SES] SESAR. `http://www.sesarju.eu`. 29.10.2014. (Cited on page 55.)

[SF68] I. Sobel and G. Feldman. A 3x3 Isotropic Gradient Operator for Image Processing. 1968. (Cited on page 91.)

[Sha91] V. Shapiro. *Theory of R-functions and Applications: A Primer*. Cornell Univ., Department of Computer Science, 1991. (Cited on page 39.)

[Sha07] V. Shapiro. Semi-analytic Geometry with R-functions. *Acta Numerica*, 2007. (Cited on page 39.)

[SHV13] T. Schafhitzel, M. Hoyer, and P. Völschow. Increasing Situational Awareness in DVE with Advanced Synthetic Vision. *Proc. SPIE 8737, Degraded Visual Environments: Enhanced, Synthetic, and External Vision Solutions*, 2013. (Cited on page 58.)

[SMJS14] H. Senaratne, S. Mittelstädt, C. Jacob, and T. Schreck. Uncertainty Visualization for Crisis Management in Smart Grid Environments. In *Eighth International Conference on Geographic Information Science (GIScience 2014) - Workshop on Visually Supported Reasoning with Uncertainty*, 2014. (Cited on page 50.)

[SSW+12] T. Schreck, L. Sharalieva, F. Wanner, J. Bernard, T. Ruppert, T. von Landesberger, and B. Bustos. Visual exploration of local interest points in sets of time series. *2012 IEEE Conference on Visual Analytics Science and Technology, VAST 2012, Seattle, WA, USA, October 14-19, 2012*, 2012. (Cited on page 52.)

[ST01] W. Schickler and A. Thorpe. Surface Estimation based on LiDAR. *Proceedings of the ASPRS Annual Conference St*, 2001. (Cited on page 17.)

[Ter83] D. Terzopoulos. Multilevel computational processes for visual surface reconstruction. *Computer Vision, Graphics, and Image Processing*, 1983. (Cited on page 19.)

[TP05] D. Tóvári and N. Pfeifer. Segmentation based Robust Interpolation- A new Approach to Laser Data Filtering. *ISPRS WG III/3, III/4, V/3 Workshop Laser scanning 2005*, 2005. (Cited on page 16.)

[Vab] P. Vabre. Air Traffic Services Surveillance Systems. `http://www.airwaysmuseum.com/Surveillance.htm`. 20.04.2015. (Cited on page 56.)

[VKH15] F. Viertler, C. Krammer, and M. Hajek. Analyzing Visual Clutter of 3D-Conformal HMD Solutions for Rotorcraft Pilots in Degraded Visual Environment. *41st European Rotorcraft Forum*, 2015. (Cited on pages 9 and 13.)

[WE05] D. Weiskopf and G. Erlebacher. Overview of Flow Visualization. *The Visualization Handbook*, 2005. (Cited on page 81.)

[WKN04] D. T. Wong, L. J. Kramer, and R. M. Norman. Two Aircraft Head-Up Traffic Surveillance Symbology Issues: Range Filter and Inboard Field-of-View Symbology. 2004. (Cited on page 69.)

[WL99] A. Wehr and U. Lohr. Airborne laser scanning—an introduction and overview. *ISPRS Journal of Photogrammetry and Remote Sensing*, 1999. (Cited on page 1.)

[WS06] D.-L. Way and Z.-C. Shih. Wrinkle Rendering of Terrain Models in Chinese Landscape Painting. *IEICE Transactions*, 2006. (Cited on pages 79 and 80.)

[WV03] J. C. Whelan and M. Visvalingam. Formulated Silhouettes for Sketching Terrain. *Proceedings of the Theory and Practice of Computer Graphics Conference*, 2003. (Cited on page 80.)

[WYWW12] H. Wang, Y. Yu, Q. Wang, and Y. Wan. A Density-based Clustering Structure Mining Algorithm for Data Streams. *Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, 2012. (Cited on page 34.)

[XLS10] L. Xu, T.-Y. Lee, and H.-W. Shen. An Information-Theoretic Framework for Flow Visualization. *IEEE Trans. Vis. Comput. Graph.*, 2010. (Cited on page 81.)

[XP98] C. Xu and J. Prince. Snakes, Shapes, and Gradient Vector Flow. *Image Processing, IEEE Transactions on*, 1998. (Cited on page 18.)

[YBS05] S. Yoshizawa, A. Belyaev, and H.-P. Seidel. Fast and Robust Detection of Crest Lines on Meshes. *Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling*, 2005. (Cited on page 81.)

[ZCW+03] K. Zhang, S.-C. Chen, D. Whitman, M.-L. Shyu, J. Yan, and C. Zhang. A Progressive Morphological Filter for Removing Nonground Measurements from Airborne LiDAR Data. *Geoscience and Remote Sensing, IEEE Transactions on*, 2003. (Cited on page 17.)

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, 1996. (Cited on page 33.)

[ZSLZ07] S. Zheng, W. Shi, J. Liu, and G. Zhu. Facet-based Airborne Light Detection and Ranging Data Filtering Method. *Optical Engineering*, 2007. (Cited on page 17.)