Clustering and visualization of non-classified points from LiDAR data for helicopter navigation

Ferdinand Eisenkeil*[a], Tobias Schafhitzel[b], Uwe Kühne[b], Oliver Deussen[a]
[a]University of Konstanz, Universitätsstraße 10, 78464 Konstanz, Germany +497531880; [b]Airbus Defence and Space, Claude-Dornier-Straße, 88090 Immenstaad, Germany +49 7545800

ABSTRACT

In this paper we propose a dynamic *DBSCAN*-based method to cluster and visualize unclassified and potential dangerous obstacles in data sets recorded by a LiDAR sensor. The sensor delivers data sets in a short time interval, so a spatial superposition of multiple data sets is created. We use this superposition to create clusters incrementally. Knowledge about the position and size of each cluster is used to fuse clusters and the stabilization of clusters within multiple time frames. Cluster stability is a key feature to provide a smooth and un-distracting visualization for the pilot. Only a few lines are indicating the position of threatening unclassified points, where a hazardous situation for the helicopter could happen, if it comes too close. Clustering and visualization form a part of an entire synthetic vision processing chain, in which the LiDAR points support the generation of a real-time synthetic view of the environment.

Keywords: LiDAR, Density-based Clustering, Head-mounted Displays, Visualization, R-Functions, Synthetic Vision

1. INTRODUCTION

Helicopter pilots encounter numerous perceptional problems during a flight mission. Landings in environments such as desserts or snowfields are problematic due to dispersed dust or snow when the helicopter approaches the ground. Other hazardous situations are night flights, which allow only for degraded visibility. Not recognized obstacles such as wires could result in helicopter crashes especially while landing approaches. In this situations not classified and potential dangerous objects detected by sensor have to be analyzed and visualized. Modern sensor systems support navigation by providing additional visual cues in the cockpit or the pilot’s head-up display. State of the art sensors for improving the situational awareness of the pilot are LiDAR (Light detection and ranging) systems that emit light pulses to measure the time of flight of reflected light [1] and result in a 3D point cloud in a global coordinate system. To generate cues for the pilot, classes like “noise”, “cloud” and “drop-in” are removed from the data and the remaining data is classified into the object classes “ground”, “tree”, “pole” and “wire”. In addition, navigational values such as flight altitude, speed, and direction are captured and can also be taken into account. A typical system delivers a point cloud with approximately 10-20.000 measured points every 300 milliseconds. An exemplary point cloud is shown in Figure 1 with its classification depicted as pixel color.

For visualizing such additional visual aids *Advanced Synthetic Vision Systems* are used for head-down and head-up displays [2]. The input data is classified, classes like “tree”, “wire” or “pole” can be visualized as symbols, “ground” as a mesh and additional information can be calculated on sensor information such as landing zone symbols and terrain awareness information [3].

In all these approaches there exists unclassified point data that also needs to be shown to the pilot in form of appropriate symbols or visual structures, since it represents potential dangerous obstacles. We divide such unclassified points into clusters, their visualization is based on 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.

*ferdinand.eisenkeil@uni-konstanz.de; phone +49 (0) 7531 88-4749; fax +49 (0) 7531 88-4715; www.uni-konstanz.de

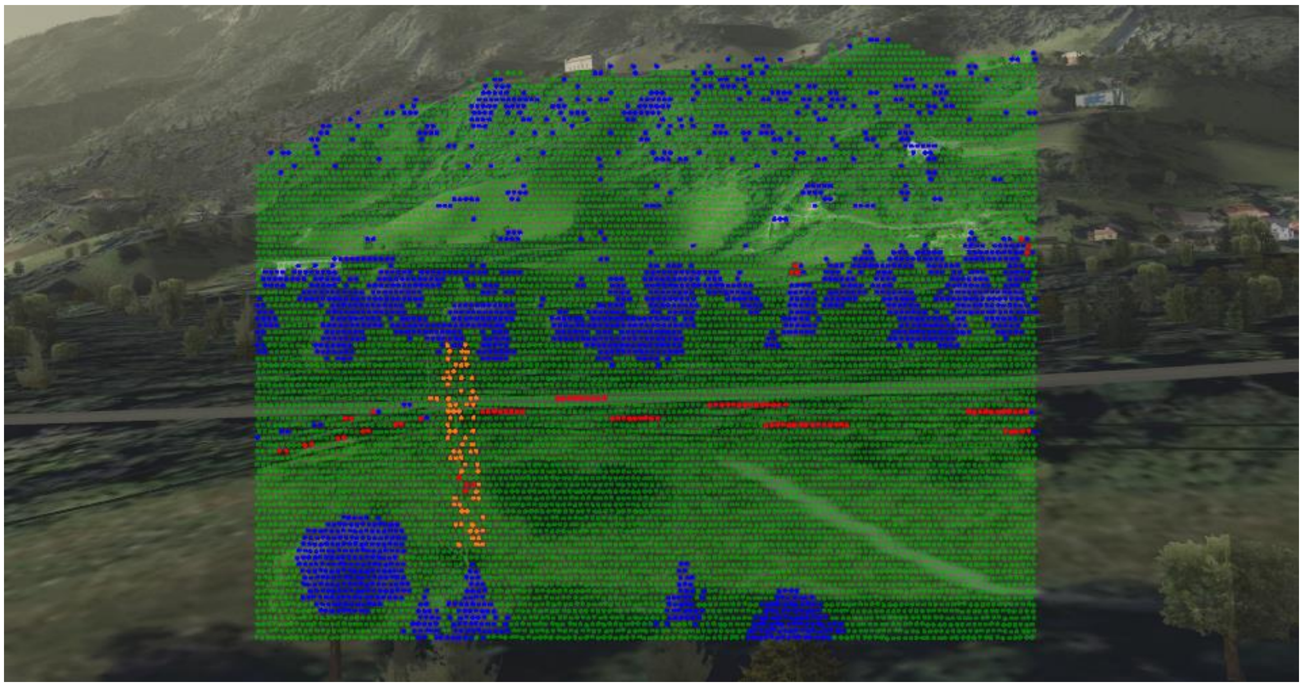

Figure 1. A typical classification of point cloud from our LiDAR sensor. Points are divided into different classes: red points belong to wires, poles are given in orange, ground is green and blue points have the label "unclassified".

We propose a dynamic *DBSCAN*-based method to cluster LiDAR sensor data recorded in short time intervals with spatial superposition. Using density based algorithms like *DBSCAN* we are able to detect outliers in the LiDAR data that are dispensable for convex hull detection. We use clustering to generate first of all a stable representation of convex hulls and later temporal stable visualizations of the clusters. To fuse convex hulls, Boolean functions are applied. Within a head-mounted display, a very simple visualization of the clustered is essential, due to keep the pilot's workload low, and highlighted lines indicate the position of threatening unclassified points, where a hazardous situation for the helicopter could happen, if it comes too close. 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 unclassified data from the sensor are adapted to existing clusters from previous sensor frames. The practical aspect 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 2 we give an overview of our system.

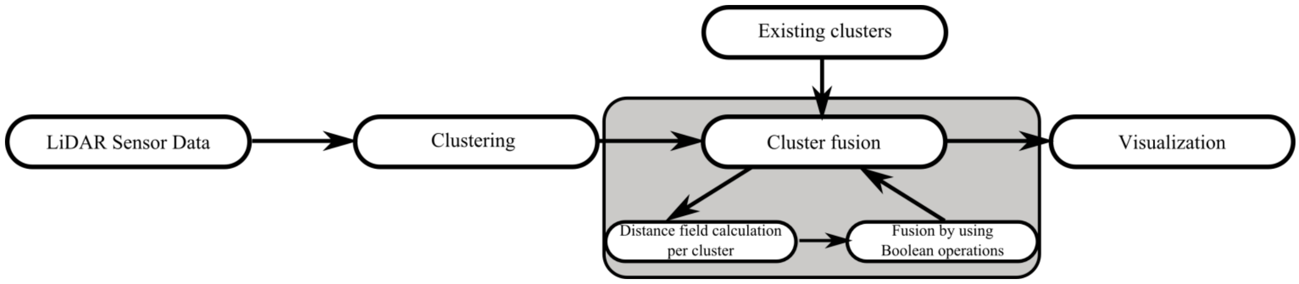
Existing clusters
LiDAR Sensor Data
Clustering
Cluster fusion
Visualization
Distance field calculation per cluster
Fusion by using Boolean operations

Figure 2. System overview. Data from the LiDAR is clustered. The resulting clusters are fused with existing clusters by calculating the distance field and using Boolean operations. The result is visualized to the pilot.

2. RELATED WORK

For clustering arbitrary data several methods exist. One of the most common methods is *k-means* [4], a cluster algorithm with requiring a specified number of clusters. An extension to this algorithm is *x-means* [5], 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. Even more promising approaches are density based algorithms, which became very important in the recent years. The first density-based algorithm *DBSCAN* [6] 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 [7]. Achtert et. al [8] provide an optimization of *OPTICS* based on single linkage clustering called *DeliClu* with the possibility to mark noise without using a density estimator. It outperforms *OPTICS* in terms of robustness, completeness, usability and efficiency.

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) [9]. 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* [10] introduced by O'Callaghan et al. In comparison to *BIRCH*, it needs a lower number of parameters and performs faster. Additionally, an effective *k-means* algorithm for streams was developed by Ackermann et al. called *StreamKM++* that improves the clustering quality of *BIRCH* but is significantly slower [11]. In comparison to *StreamLS* the quality is similar but it scales better with respect to increasing number of cluster centers. Aggerwal et al. present a very effective framework for clustering evolving data streams called *CluStream* [12] 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 [13] is the first density based cluster algorithm for stream data developed by Cao et. al. It makes use of micro-clusters of different types within the data set. Core-micro-clusters summarize clusters, potential core-micro-clusters and outlier micro-clusters are proposed to adapt data to existing clusters or remove clusters. Another density-based approach is *D-Stream* [14] 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. provide with *ClusTree* [15] 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 if new data arrive the model is adapted. Like *CluStream ClusTree* is also divided into an offline and an online component.

Wang et al proposed *OPCluStream* [16] 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-Agglosmerative Clustering* (*ODAC*) [17], 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. introduce *DGClust* [18]. Within this system, continuous cluster structures are adapted for data of an entire network.

In Table 1 an overview of the clustering algorithms of stream data with their underlying cluster method and resulting clustering shape is given.

Table 1. Stream analyzing methods with their underlying cluster algorithm and resulting cluster shapes and outlier detection.

Algorithm	**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
OPCluStram	density based	arbitrary
ODAC	*k-means*	spherical/elliptical

For our data from the LiDAR sensor it is very 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.

3. CLUSTERING, CONVEX HULL AND CLUSTER FUSION

The data we used contain for each data point a classification tag. Tags like "tree", "wire" or "ground" can be visualized within the Advanced Synthetic Vision System with symbols for obstacles or, in case of ground, as a mesh. The unclassified 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.

3.1 Clustering with OPTICS

Because unclassified points are scattered over the complete data range within a sensor frame, it has to be clustered. 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 density. To calculate clusters with OPTICS, for each data point the core distance is calculated. The core distance is that maximal distance value to the n neighbor points, where n is the minimal number of points a cluster needs to have. Based on 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 on the reachability plot is given in Figure 3. The synthetic data set in (3a) consists of two clusters denoted by $\otimes$ and $\times$. In (3b) the clusters are depicted in the reachability plot also by $\otimes$ and $\times$.

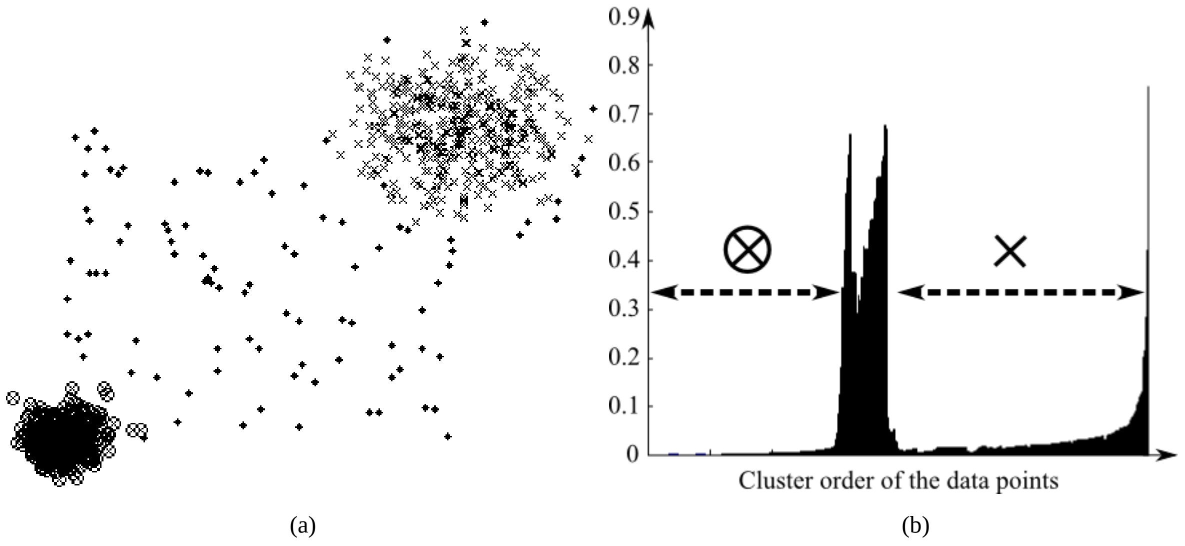

Figure 3. (a) Synthetic data set with different densities, ⊗ and × denote clusters, while simple dots are not clustered noise. (b) shows a reachability plot, the right side is the valley for cluster labeled with ⊗, the left valley is labeled with × for the second cluster.

3.2 Convex Hull calculation

As a preprocessing step for visualization the 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* [19] 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.

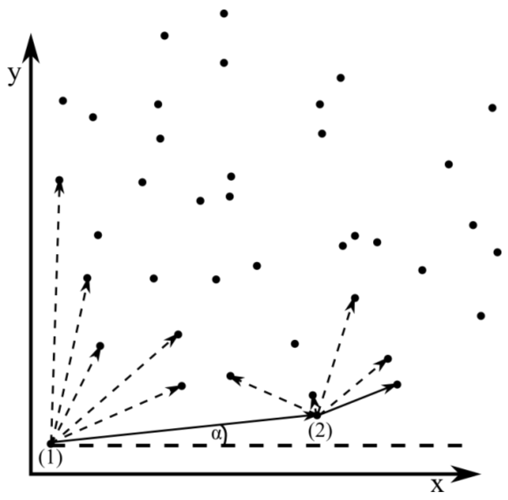

Figure 4. Convex hull calculation with Graham Scan. (1) is the initial point p_1 with the lowest y-coordinate. The connection to point p_2 (2) is based on the smallest angle to the x-axis. From (2) the process is repeated until the beginning point is reached.

3.3 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 or Rvachev [20]. With R-functions 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 [21][22]. 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 [21].

$$\text{Logical } \textit{not}\text{: } \bar{x} \equiv -x \tag{1}$$

$$\text{Logical } \textit{and}\text{: } x_1 \wedge \mathrm{x}_2 \equiv\ x_1 + x_2 - \sqrt{x_1^2 + x_2^2} \tag{2}$$

$$\text{Logical } \textit{or}\text{: } x_1 \vee \mathrm{x}_2 \equiv\ x_1 + x_2 + \sqrt{x_1^2 + x_2^2} \tag{3}$$

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 [23]. The distance field is positive inside of the convex hull and negative on its outside. Each convex hull can now be interpreted in that way as a function $g\colon \mathbb{R}^2 \to \mathbb{R}$. In this way the two functions (a) and (b) in Figure 5, can be handled as input to a R-function equivalent to the values x_1 and x_2 in the example equations (1) and (2). In Figure 5 Boolean operations for two convex hulls are depicted, including the calculation of the distance field.

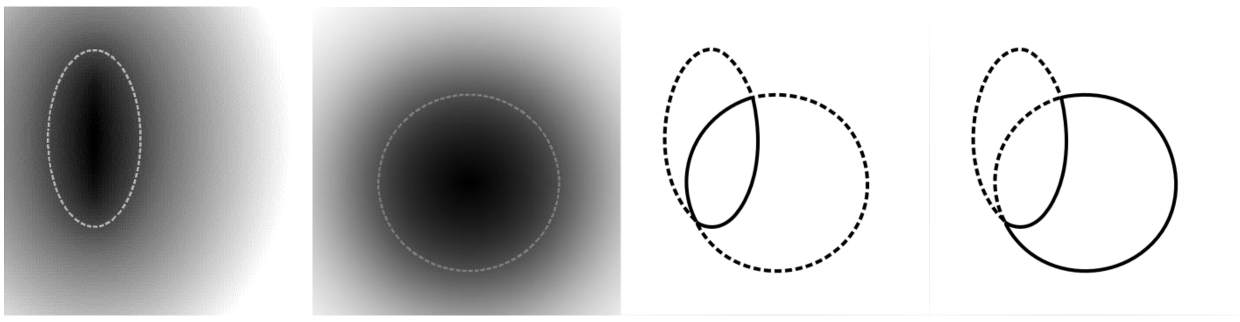

Figure 5. (a) and (b) are distance fields for convex hulls, depicted as dashed line. The distance inside the convex hull is negative and appears black. (c) The union (Equation 2) is given as dashed line, the intersection (Equation 3) is depicted as solid line. (d) The dashed line indicates the result 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.

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 just 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 adjusting 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. 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 6 a conservative union ($\vee$) is shown.

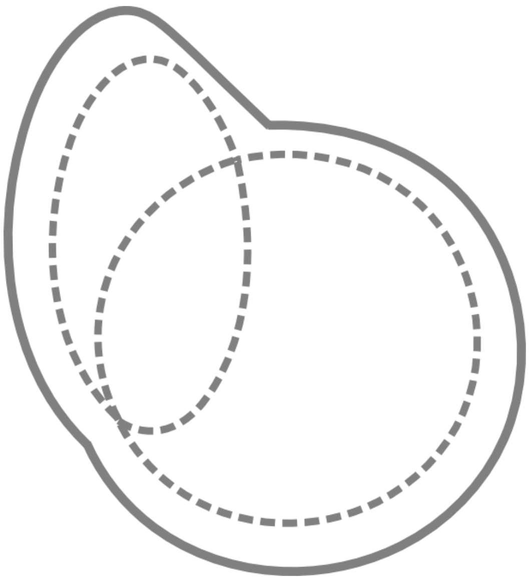

Figure 6. Conservative union (⋁) of the convex hulls from Figure 5. The dashed lines are the initial convex hulls, the solid line is the union of both clusters.

3.4 Visualization approaches

We introduced different visualization approaches in order to display the unclassified parts of the sensor data in different devices. Our main goal is a representation on the pilot's head-mounted display.

The decision, which symbol is the most adequate for the flight situation – for example hovering, flight in low or high altitudes – has to be evaluated by the pilot and is changeable during flight. Different simple visualization possibilities for head-down and head-mounted displays are given in Figure 6.

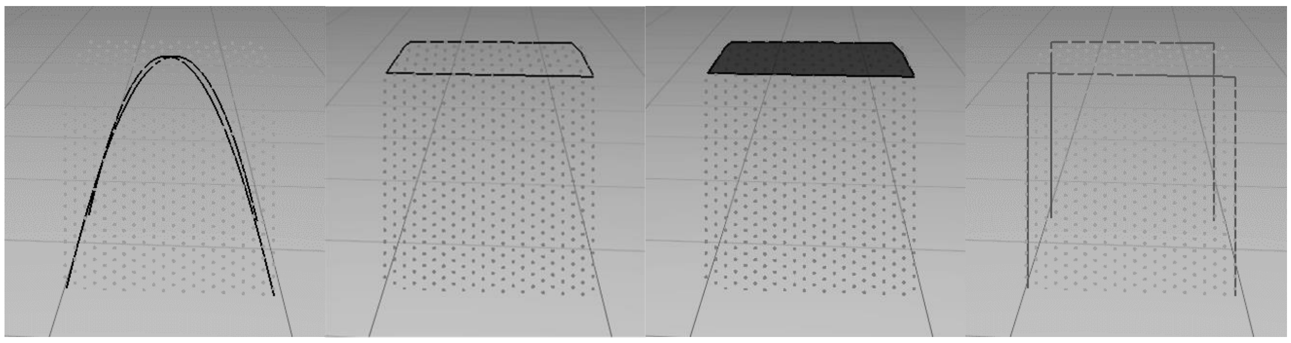

Figure 7. Different visualization approaches for the convex hulls. Grey dots indicate non-classified data points.

4. RESULTS

Our approach with clustering unclassified 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 implemented very efficient and yields the possibility to apply different merging approaches based on Boolean operations. We considered two different cases where we apply different merging techniques. The first case is that a cluster complete 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 simulated LiDAR sensor under perfect conditions that are stable helicopter position and no noise bias in the recorded data. Also the data has the same spatial and temporal resolution as the real sensor. In Figure 8 we applied *OPTICS* and our approach to such data.

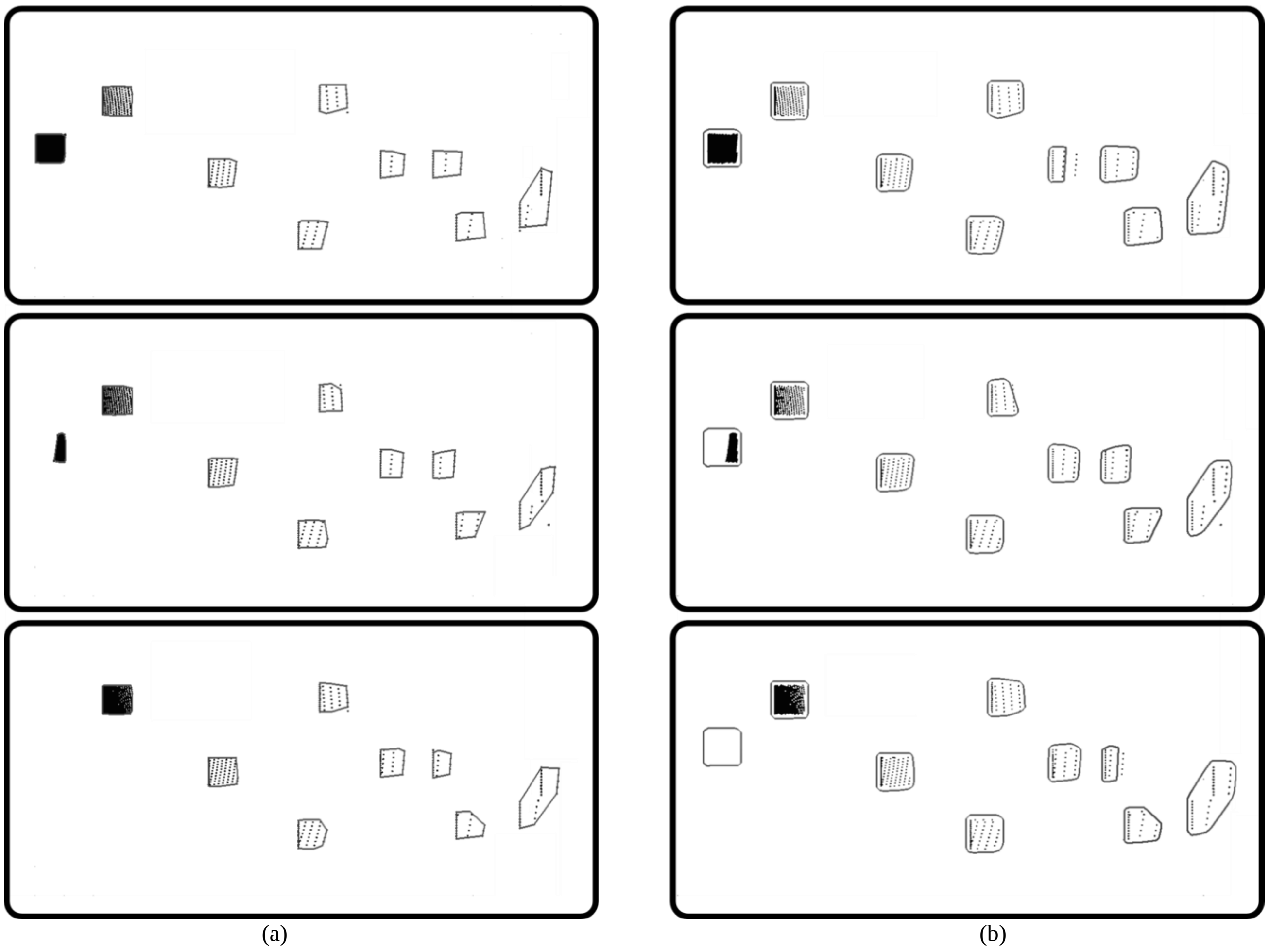

Figure 8. Bird-eye perspective on not classified data in three consecutive sensor frames. (a) Clustering with OPTICS. (b) Clustering with our approach that results in smooth contours and a stable clustering with small changes over time.

All these methods are scalable in their computation time by reducing the amount of data: Data in bigger distance to the helicopter are optionally not necessary to be clustered. Also the resolution of the grid used to apply the Boolean operations is definable by the user and can be adapted through different platforms. Additionally the visualization methods are independent from the cluster approach. To generate a smooth visualization, we morph between two sensor frames. Also only clusters next to the helicopter are visualized full opaque. Clusters become with increasing distance more transparent. The result can be shown in different ways to the pilot and can be applied in different ways on the head-down or the head-up display. The whole processing from an initial clustering with *OPTICS* to the merging of clusters can be calculated within the 300ms time frame from on sensor image to the next. The whole system was applied in our flight simulator test environment. In Figure 7 potential visualizations for the pilot are given as an overlay within the flight simulator.

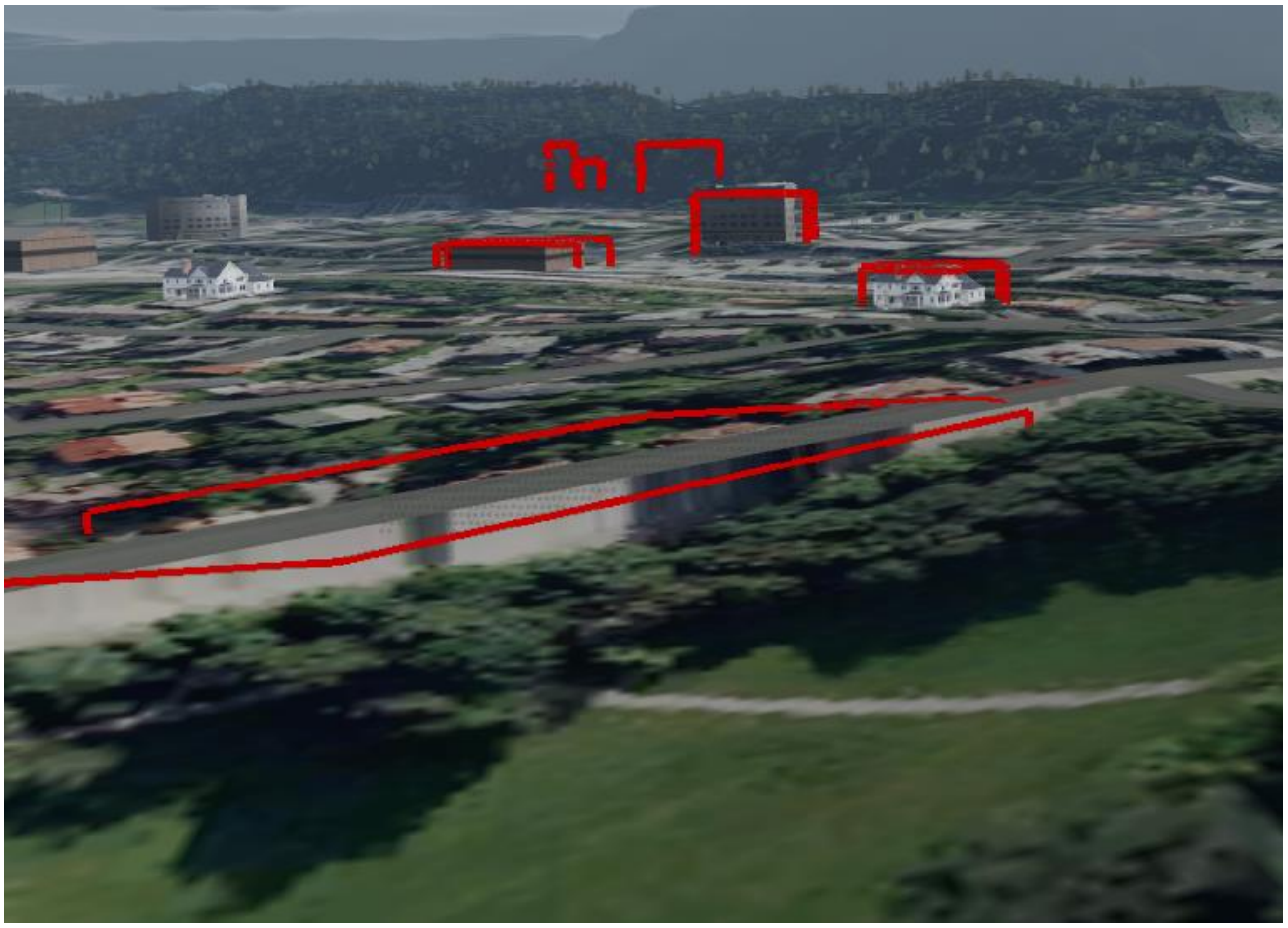

Figure 9. Screenshot from simulator environment. Cluster visualization from the head-mounted display is projected to the simulator environment.

5. CONCLUSION AND FUTURE WORK

An initial implementation of *DBSCAN* turned out to be too slow and instable through time. Therefore we increased the first solution by using *OPTICS* resulting in a faster solution but also with the lack of being instable for visualization. With our approach we generated a stable visualization for a data stream from a LiDAR sensor. Additionally, we demonstrated that our technique is real-time capable on mission computers within the time frame of 300ms. The proposed visualization is based on the outcome of recent workshops with focus on other symbols with helicopter pilots in a flight simulator environment. Our proposed cluster representations are a smoother approach than drawing every unclassified pixel in order to give a stable visualization of the environment to the pilot over time. The representation of the unclassified data has now to be evaluated. To avoid unnecessary cluster visualizations a potential application case could be to visualize only unclassified pixels on the landing site.

REFERENCES

[1] Wehr, A. and Lohr, U., "Airborne laser scanning - an introduction and overview," ISPRS Journal of Photogrammetry and Remote Sensing 54, 68–82 (1999).

[2] Schafhitzel, T., Hoyer, M., Völschow, P., “Increasing situational awareness in DVE with advanced synthetic vision,” Proc. SPIE 8737, (2013)

[3] Aviation, U.S.F., [Installation of Terrain Awareness and Warning System Approved for Part 23 Airplanes], General Books, (2011).

[4] MacQueen, J. B., “Some Methods for Classification and Analysis of MultiVariate Observations,” Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability, 281-297 (1967)

[5] Pelleg, D., Moore, A., “X-means: Extending K-means with Efficient Estimation of the Number of Clusters,” Proc. of the Seventeenth International Conference on Machine Learning, 727-734 (2000)

[6] Ester, M., Kriegel, H., P., Sander, J., Xu, X., “A density-based algorithm for discovering clusters in large spatial databases with noise”, Proc. of the Second International Conference on Knowledge Discovery and Data Mining, 226-231 (1996)

[7] Ankerst, M., Breunig, M., Kriegel, H.P., Sander, J., “OPTICS: Ordering Points to Identify the Clustering Structure,” Proc. of the 1999 ACM SIGMOD International Conference on Management of Data, 49-60 (1999)

[8] Achtert, E., Böhm, C., Kröger, P., „DeLiClu: Boosting Robustness, Completeness, Usability, and Efficiency of Hierarchical Clustering by a Closest Pair Ranking,” (2006)

[9] Zhang, T., Ramakrishnan, R., Livny, M., “BIRCH: An Efficient Data Clustering Method for Very Large Databases,” (1996)

[10] O'Callaghan, L., Mishra, N., Meyerson, A., Guha, S., Motwani, R.: “Streamingdata algorithms for high-quality clustering,” Proc. 18th International Conference on Data Engineering, 685-694 (2002)

[11] Ackermann, M., R., Märtens, M., Raupach, C., Swierkot, K., Lammersen, C., Sohler, C., “Streamkm++: A clustering algorithm for data streams,” ACM Journal of Experimental Algorithmics 17(1), (2012)

[12] Aggarwal, C., C., Han, J., Wang, J., Yu, P., S. “A Framework for Clustering Evolving Data Streams,“

[13] Cao, F. , Ester, M., Qian, W., Zhou, A., “Density-based clustering over an evolving data stream with noise,” Proc. SIAM Conference on Data Mining, 328-339 (2006)

[14] Chen, Y., Tu, L., “Density-based clustering for real-time stream data,” Proc. of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, 133-142 (2007)

[15] Kranen, P., Assent, I., Baldauf, C., Seidl, T.: “The clustree: indexing micro-clusters for anytime stream mining,” Knowledge and Information Systems 29(2), 249-272 (2011)

[16] Wang, H., Yu, Y., Wang, Q., Wan, Y., “A Density-based Clustering Structure Mining Algorithm for Data Streams,” Proc. of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications, 69-76 (2012)

[17] Rodrigues, P., Gama, J., Pedroso, J., P., “ODAC: Hierarchical clustering of time series data streams,” Proc. of the Sixth SIAM International Conference on Data Mining, 499-503 (2006)

[18] Gama, J., Pereira, P., R., Lopes, L., “Clustering distributed sensor data streams using local processing and reduced communication,” Intelligent Data Analysis 15(1), 3-28 (2011)

[19] Graham, R., L., “An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set,” Information Processing Letters 1, 132-133 (1972)

[20] Rvachev, V., L., “On the analytical description of some geometric objects,” Reports of Ukrainian Academy of Sciences, 765–767 (1963)

[21] Shapiro, V., “Semi-analytic geometry with R-Functions,” Acta Numerica, Cambridge University Press, 239-303 (2007)

[22] Shapiro, V., “Theory of R-functions and applications: A primer,” Technical Report, Cornell University (1991)

[23] Felzenszwalb, P., F., Huttenlocher, D., P., „Distance transforms of sampled functions,“ Technical Report, Cornell University (2004)