TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

ACTA TECHNICA NAPOCENSIS

Series: Applied Mathematics, Mechanics, and Engineering
Vol. 67, Issue Special I, February, 2024

PROCESSING THE POINT CLOUD STREAM FROM HELIOS2 TIME-OF-FLIGHT CAMERA

Adrián VODILKA, Marek KOČIŠKO, Martin POLLÁK

Abstract: *This article discusses the use of the Helios2 ToF camera in manufacturing and industrial applications. The aim of this research is to design a script for the processing of point cloud stream from the Helios2 Time-of-Flight camera. It is a device suitable for digitizing objects and is provided with its own ArenaSDK with ArenaView GUI. These are suitable for simple tasks. To be used in an industrial environment, a script needs to be proposed to implement ArenaAPI functionality for specific applications. In this research, a script was proposed to acquire point cloud stream from Helios2 camera. This data was then converted into Open3D point cloud data and visualized. The point cloud was then processed using Voxel downsampling, Statistical outlier removal and Estimate normals. The resulting point cloud is then suitable for use within manufacturing and industry.*
Key words: *Time of Flight, depth sensing, Helios2, ArenaSDK, point cloud, Open3D.*

1. INTRODUCTION

At present, a variety of digitisation techniques and technologies can be used within the context of reverse engineering across the industry. It is possible to use industrial 3D scanning devices to digitise complex forms on the surface of objects with high accuracy and precision. The disadvantage of 3D scanners is the often time-consuming scanning process. To perform real-time digitization of the surface of objects, 3D cameras are designed. These cameras provide a relatively accurate and precise option for digitizing the surface of objects in real or almost real time. 3D cameras are not primarily designed for the accurate digitization of surfaces for use in the design of 3D models. 3D cameras are suitable for use in computer and robotic vision. The use of 3D cameras is particularly advantageous in the manufacturing process. A number of 3D cameras based on digitization techniques are available, including LiDAR, Stereovision, Time-of-Flight, Structured Light, Photogrammetry, Infrared Depth Sensing and others.

Time-of-Flight (ToF) technology is one of these available technologies. ToF camera technology utilizes the principle of emmiting light pulses from a laser or LED. These light signals travel to the object and reflect back to a sensor on the camera. The camera can calculate the distance to each point on the object by measuring the time it takes for the light to travel to and from the object. The result of digitizing with a ToF 3D camera is a depth image or point cloud, and similar to conventional cameras, ToF cameras can stream data at a frame rate of e.g. 30 frames per second [1].

Among the ToF 3D cameras is the Helios2. This device is provided with its own ArenaSDK along with the ArenaView GUI application. It is used to perform simple tasks, which include displaying streams or point cloud images, changing settings, etc. To use the Helios2 ToF camera within industrial and manufacturing applications, it is essential to use ArenaAPI and custom scripting for specific applications. The aim of this research is to design a script for the processing of point cloud stream from the Helios2 Time-of-Flight camera.

This paper is related to previous research within the Department of Computer Aided Manufacturing Technologies, Faculty of Manufacturing Technologies, Technical University of Kosice. In 2022 an article was

published discussing the 3D Time-of-Flight digitization solution Helios2 along with the ArenaSDK software package and their use in digitizing objects with different surface properties [2]. Subsequently, in 2023 a paper was published, whose aim of the research was to verify the use and functionality of the camera for the needs of digitization of objects, specifically the process of polygonization of point clouds obtained by the Helios2 ToF camera [3].

2. HELIOS2 CAMERA

The Helios2 Time-of-Flight camera is depth sensing 3D optical imaging technology introduced in 2020 by LUCID Vision Labs Inc. in Canada. Utilizing Time-of-Flight technology, this 3D camera allows the surface of physical objects to be digitized in real-time by calculating the phase shift of the time-of-flight of the light pulse illuminating the object and the corresponding reflection. Helios2 utilizes Sony DepthSense IMX556PLR CMOS sensor. Its previous version, the 2019 Helios, was the first commercially available industrial grade ToF technology. Unlike Helios and other available ToF technologies, Helios2 has improved accuracy and precision of the acquired data. The Helios2 3D camera is suitable for a variety of industrial environments due to its IP67 certification, small size of 60 mm x 60 mm x 77.5 mm and weight of 398 g, operating range -20° to 50°C without the need to use of cooling techniques. Helios2 has a resolution of 640 x 480 px (0.3 MP) at a refresh rate of 30 fps. It's possible to power the device using PoE + IEEE 802.3at or using GPIO with a voltage of 18-24V. The average power consuption is 12W with a maximum of 30W. Digitization with Helios2 is possible in 6 operating distance modes from 0.3 m to 8.33 m. Scene representation of object digitization using Helios2 ToF camera is seen on Fig. 1. [4].

3. ARENAAPI

ArenaSDK is a software kit that provides a programming API from the technology manufacturer. ArenaSDK includes libraries and example programs in C++, C, .NET and Python programming languages. ArenaSDK includes the ArenaView GUI. Advanced features of Helios2 can be used and controlled using ArenaView, while creating depth images or displaying a 3D point cloud in real time. The ArenaView can be operated in 2D and 3D views. Within 3D view it is possible to change the view orientation in real time, so that the actual digitized data can be displayed or a snapshot captured. Helios2 digitizing data is exported by ArenaView in 3D PLY and RAW format.

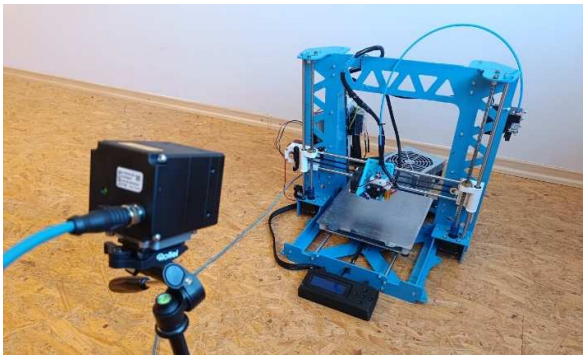

Fig. 1. Scene representation of object digitization using Helios2 ToF camera.

ArenaAPI (Arena Application Programming Interface) is the actual application programming interface that developers use to interface with and control the cameras. ArenaAPI provides direct access to the camera and allows users to control a variety of parameters, from frame rate and exposure to more advanced features specific to certain camera models. Buffer management is essential for efficient image processing and streaming. The ArenaAPI interface provides tools to manage and control buffers, ensuring a smooth data flow and minimizing the risk of framerate loss. ArenaAPI offers the use of multiple programming languages including popular Python, but also C, C# and C++, allowing developers to work in the environment that suits them best.

ArenaView as a GUI application available to users of Lucid Vision Labs technology allows users to conveniently view, capture and adjust camera settings without the need for ArenaAPI and any programming. It allows the Helios2 ToF 3D camera to be used as a Plug&Play without previous experience with the technology. ArenaView provides the ability to stream and visualize data within the GUI in real-time, adjust camera parameters, change display modes, and

save data in Depth Images format or save point cloud in PLY format. In the Fig. 2. it is possible to observe the point cloud display in ArenaView GUI.

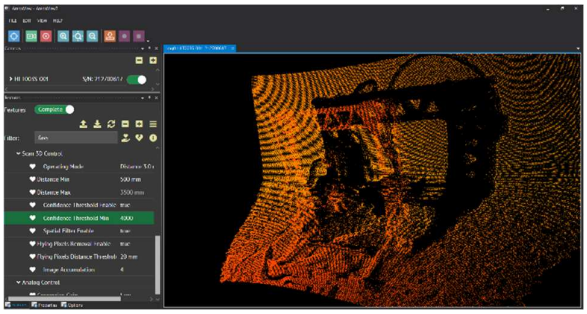

Fig. 2. View of Helios2 ToF camera digitization process in ArenaView GUI.

ArenaAPI is intended for developers to integrate Helios2 ToF camera functions into their specific applications. ArenaAPI is suitable to be used in advanced applications and requires programming and software development knowledge. This is convenient for data acquisition and processing, with the possibility of subsequent use of external software systems and libraries for point cloud processing or for applications in computer and machine vision. For the use of the Helios2 ToF camera in industrial applications, the use of ArenaAPI is crucial. Despite the provision of ArenaSDK from the manufacturer with additional documentation, this documentation is quite generic and extensive, so the interface for acquisition and use needs to be designed in a custom approach for a given application. To summarise, ArenaView is suitable for the launch and selection of suitable process parameters for the acquisition of 3D coordinates of a digitised surface, while ArenaAPI is suitable for implementation within the industrial sector for its applications. To design custom software for use with ArenaAPI and the Helios2 camera, it is convenient to use the Python programming language, which is used extensively in industrial and scientific applications and makes suitable support for ArenaAPI.

4. APPLICATION OF HELIOS2 IN MANUFACTURING

The Helios2 camera can be implemented in a number of applications within industry and manufacturing. An interesting possibility is the digitization of the surface of physical objects. Several deficiencies occur within this application, including low accuracy and precision of the device, multipath effect of reflected light from multiple walls at an angle, deficiencies occurring from imaging in perspective view. These deficiencies can be eliminated within future research using Iterative Closest Point (ICP) alignment of multiple point clouds or by using the implementation of Simultaneous Localisation And Mapping (SLAM) technology. The acquired data can be used within another possible implementation option in pick and place sequences to determine the position and orientation of objects on the conveyor belt or as part of collision avoidance. It can also be implemented in computer vision for object detection, dimensional and volume verification of products or for securing safety environments. The advantage of using Helios2 ToF technology in manufacturing is the capability to use real-time acquisition of data.

To use the Helios2 camera in production applications, it is essential to develop a custom program within the software and utilizing the accessible ArenaAPI. Initially, it is advisable to use ArenaView software within ArenaSDK, launch the Helios2 camera, select the appropriate digitization process parameters for the specific conditions of the digitization scene and object surface. It is advisable to empirically verify these process parameters and select the most appropriate options. For the software program development, the programming language Python was chosen. The first objective was to establish a communication interface with the Helios2 device for the acquisition of 3D surface coordinates. The first part of the program was to initialize and link the Helios2 camera to the computer. Nodemaps were then selected as a structured representation of all the features, settings, and functionalities of a camera. Features and settings of the process have been selected within these nodemaps, namely PixelFormat to Coord3D_ABCY16, Scan3dOperatingMode to Distance3000mm SingleFreq, ExposureTimeSelector to

Exp1000Us, Conversion Gain to Low, Scan3dImageAccumulation to 4, Scan3dSpatialFilterEnable to True and Scan3dConfidenceThresholdEnable to True. These settings were selected empirically. Subsequently, a class was created for obtaining 3D x, y, z coordinates in mm. The stream was then launched and the buffer was started loading to store the 3D coordinates in RAM. To verify functionality, a selection of the 3D coordinate section was selected to be displayed in the command line with the print function.

5. OPEN3D

After verification of establishment of interface with Helios2, it was proposed to subsequently process this stream of 3D coordinates into a point cloud for later implementation. Open3D v0.17.0 libraries were used for working with 3D coordinates, their conversion, visualization, processing and storage. Open3D is an open source library that provides tools for faster development of software working with 3D data in the languages C++ and Python. Core features of Open3D include installation via conda and pip, 3D data structures, 3D data processing algorithms, scene reconstruction, surface alignment, PBR rendering, 3D visualization and Python binding [5].

As part of the software development, the Open3D libraries for working with point clouds and numpy for working with calculations were imported. The stream of 3D coordinates from the buffer was converted using the Open3D class o3d.geometry.PointCloud() and o3d.utility.Vector3dVector(points). Open3D libraries work natively in meters, while the Helios2 camera obtains 3D coordinates in millimeters. For this reason, it was necessary to simply divide these coordinates by 1000 before converting the 3D coordinates to Open3D point cloud data. After converting the 3D coordinates to Open3D point cloud format, it was then possible to perform a visualization of the point cloud using o3d.visualization.draw_geometries([point_cloud]) as seen on Fig. 3., opening a window displaying the point cloud, allowing to rotate, pan, and zoom to view the point cloud from different angles. Apart from visualization, the point cloud can be saved in Point Cloud Library (PCL) format using o3d.io.write_point_cloud().

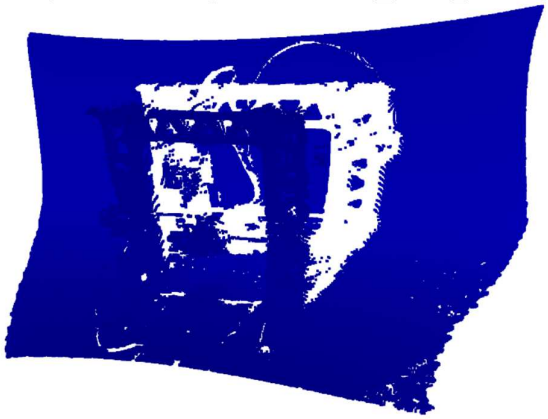

Fig. 3. Visualization of the point cloud from the Helios2 camera.

In the current state, the point cloud is based on 3D coordinates streamed by the Helios2 camera. This data can be adjusted early on using the nodemaps adjustment. But some of this data is hardcoded by the camera, e.g. the camera resolution. The camera resolution directly defines the quality and density of the acquired point cloud. Although it would be possible to adjust the camera resolution by adjusting the camera settings, such adjustments may reduce the accuracy and precision of the acquired point cloud. A more appropriate method is to modify the resulting point cloud for specific needs using post-processing functions. Various available libraries can be used for this purpose. Open3D allows the use of different classes and libraries. [6].

Voxel downsampling is a post-processing method used to recalculate the density of points within a point cloud. It partitions the point cloud within 3D space into equally-sized cubic "voxels" and averages the points within each voxel to produce a single representative point. This is useful for reducing the point cloud density to perform faster computation if the point cloud density is too high and unnecessary within the application. Voxel downsizing can be performed using pcd.voxel_down_sample(voxel_size=0.05). By adjusting the voxel_size value, the density of the point cloud can be adjusted. The resulting point cloud after voxel downsampling can be seen at Fig. 4.

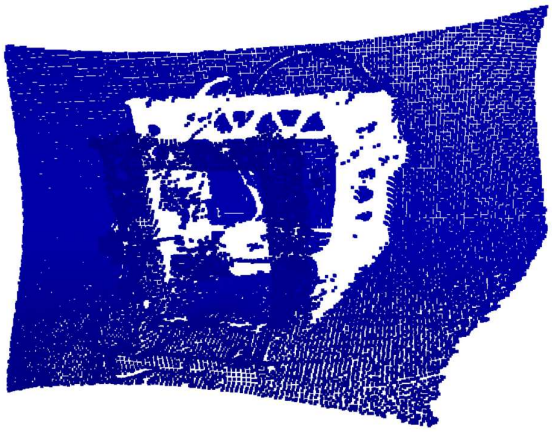

Fig. 4. Visualization of the point cloud after voxel downsampling.

Another post-processing method is remove_statistical_outlier. It deletes points that are more distant from their nearest points in comparison to the averaged point cloud. It requires two input parameters. The nb_neighbors parameter determines the number of nearest points to be considered when averaging the distance for a given point. Std_ratio parameter enables to set the threshold value according to the standard deviation of the mean distances in the point cloud. When a point is more distant than the average distance of the points, it is deleted, as seen in Fig. 5. Normals within point cloud are important for many 3D processing tasks. Normals are vectors that are perpendicular to the tangent plane at each point in a point cloud or surface. They are important for surface reconstruction, segmentation, and visualization. Open3D allows to use the available function to calculate the normals using estimate_normals.

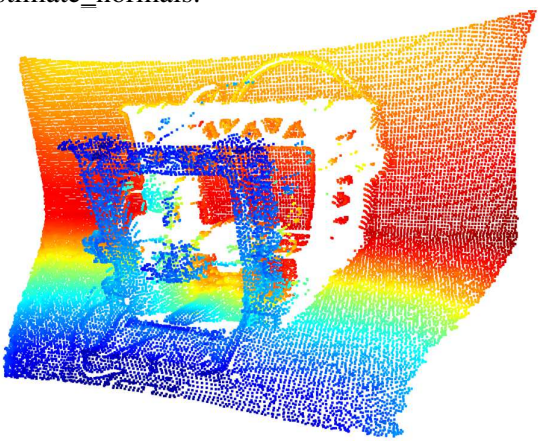

Fig. 5. Visualization of the point cloud after the removal of statistical outlayer.

Function estimate_normals computes the normal at each point in the point cloud. For this specific case using Helios2 to capture point clouds, it is possible and beneficial to orient normals toward Helios2 camera. it is appropriate to orient the vectors to the Helios2 camera, since the zero of the camera coordinate system is known. The Helios2 camera captures the point cloud from the view of the camera coordinate system zero, and thus it is possible to orient the vectors in the correct direction to the camera without the need to resolve the possible uncertainty in the orientation of the vectors that may occur. The calculated normals can be seen on Fig. 6.

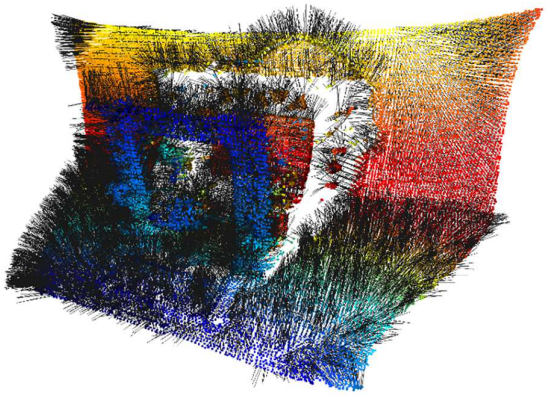

Fig. 6. Visualization of the point cloud with calculated normal displayed.

6. CONCLUSION

The aim of this research was to design a script for the processing of point cloud stream from the Helios2 Time-of-Flight camera. First, the hardware had to be set up and the set had to be prepared. Once the Helios2 camera was started, the process parameters for the digitizing experiment were selected within ArenaView. Subsequently, a simple Python script was proposed to acquire a stream of 3D coordinate points from the Helios2 camera. The acquired data were converted to Open3D data and validated by visualization. After validation, further sections of the point cloud processing script were proposed. The first step was Voxel downsampling, in which the point cloud density was recalculated to 10 mm as needed. The second step was the removal of statistical outlayer. The last step was the calculation of the

normals and their subsequent visualization. The resulting point cloud can be used in manufacturing and industrial applications. In the proposed future work, it is possible to build on this research and use the obtained point cloud in robotic vision for the Pick&Place sequence.

7. ACKNOWLEDGEMENTS

Paper is the result of the Project implementation: Development of excellent research capacities in the field of additive technologies for the Industry of the 21st century, ITMS: 313011BWN5, supported by the Operational Program Integrated Infrastructure funded by the ERDF. The authors would like to thank the Ministry of education of Slovak Republic for the support of this research through the grants VEGA no. 1/0121/23, KEGA no. 038TUKE-4/2021, KEGA no. 002TUKE-4/2023 and project APVV-19-0550.

8. REFERENCES

[1] Y. He, S. Chen, IEEE Access, 7, 12495-12510 (2019)
[2] A. Vodilka, M. Pollák, M. Kočiško, IJMPE, 4, 79-85 (2022)
[3] A. Vodilka, M. Pollák, M. Kočiško, EAISICC, *7th EAI Int. Con. on Man. of Man. Sys.*, 31-43 (2023)
[4] Helios2 Time of Flight (ToF) IP67 3D Camera, *https://thinklucid.com/product/helios2-time-of-flight-imx556/*
[5] Q. Zhou, J. Park, V. Koltun, ARVIX, 1-6 (2018)
[6] C. Li, Y. Ren and B. Liu, PCGen: Point Cloud Generator for LiDAR Simulation, ICRA, 11676-11682 (2023)

PRELUCRAREA FLUXULUI DE NOR DE PUNCTE DE LA CAMERA HELIOS2 TIME-OF-FLIGHT

Rezumat: Acest articol discută despre utilizarea camerei Helios2 ToF în aplicații industriale și de producție. Scopul acestei cercetări este de a proiecta un script pentru procesarea fluxului de nori de puncte de la camera Helios2 Time-of-Flight. Camera 3D Helios2 Time-of-Flight este un dispozitiv potrivit pentru digitalizarea obiectelor și este prevăzut cu propriul ArenaSDK cu ArenaView GUI. Acestea sunt potrivite pentru sarcini simple. Pentru a fi utilizate într-un mediu industrial, trebuie să se propună un script care să implementeze funcționalitatea ArenaAPI pentru aplicații specifice. În această cercetare, a fost propus un script pentru a achiziționa fluxul de nori de puncte de la camera Helios2. Aceste date au fost apoi convertite în date de nor de puncte Open3D și vizualizate. Norul de puncte a fost apoi procesat folosind Voxel downsampling, eliminarea statistică a valorilor aberante și normalele estimate. Norul de puncte rezultat este adecvat pentru utilizare în producție și în industrie.

Adrián VODILKA, Eng. MSc. PhD., Assistant Professor, Technical University of Kosice, Faculty of Manufacturing Technologies, Bayerova 1, 080 01 Presov, Slovakia, Department of Computer Aided Manufacturing Technologies, adrian.vodilka@tuke.sk, +421 55 602 6360.
Marek KOČIŠKO, Prof. Eng. MSc. PhD., Professor, Technical University of Kosice, Faculty of Manufacturing Technologies, Bayerova 1, 080 01 Presov, Slovakia, Department of Computer Aided Manufacturing Technologies, marek.kocisko@tuke.sk, +421 55 602 6354.
Martin POLLÁK, Eng. MSc. PhD., Assistant Professor, Technical University of Kosice, Faculty of Manufacturing Technologies, Bayerova 1, 080 01 Presov, Slovakia, Department of Computer Aided Manufacturing Technologies, martin.pollak@tuke.sk, +421 55 602 6460.