# A PRELIMINARY STUDY ON UPDATING A DETECTED TRAFFIC CONE'S COORDINATES OBTAINED FROM A STEREO CAMERA ON A PSEUDO HIGH DEFINITION MAP DATABASE

Miguel Luis R. Lagahit (1), Yi-Hsing Tseng (1)

1 National Cheng Kung University, No.1, Dasyue Road, East District, Tainan City, Taiwan
Email: miguellagahit@prs.geomatics.ncku.edu.tw; tseng@prs.geomatics.ncku.edu.tw

**KEY WORDS:** Object Detection, Photogrammetry, High-Definition Maps

**ABSTRACT:** Research and development on Autonomous Vehicles (AVs) or self-driving cars have been escalating around the world these past few years. One of those researches is about HD (High-Definition) Maps, which are basically very detailed 3D (three-dimensional) maps produced from mobile mapping techniques that uses a combination of GPS, IMU, LiDAR, cameras, and etc., as its sensors. HD Maps helps the AV better position itself and other objects (traffic cones, traffic signs, lane markings, etc.) on the road by providing all of the necessary geometric and semantic information. This research will propose a preliminary method on how to update said HD Maps. The methodology will mainly consist of: (1) running a pre-trained YOLOv3, an object detection system, on a calibrated stereo camera to detect the traffic cone, (2) applying photogrammetric techniques to determine its 3D position, (3) uploading and updating the observed coordinates to the pseudo HD Map database, which is hosted on a local PostgreSQL platform, (4) executing all of it at the same time on a Python based program, and then (5) visualizing it in a simulated scenario through a browser based application using Cruise Automation's Worldview. Results have shown centimeter level accuracy in terms of obtained distance of the detected traffic cone from the camera setup, and a successful upload and or update of the traffic cone's coordinates to a database. In future works, additional sensors such as LiDAR may be included in the detecting and positioning framework, additional computer vision or image processing techniques such as feature tracking may also be included, and objects of interest can be expanded to include other traffic management objects such as traffic signs and other lane markings.

## INTRODUCTION

### Autonomous Vehicles

Currently a popular concept in engineering is that of Autonomous Vehicles (AV) or self-driving cars. By the year 2016, in this single year, there were already almost 2000 published research work produced all associated with the key words "autonomous" and "vehicle" (Brummelen, 2018). But what exactly makes a vehicle autonomous? To gain a better understanding, we can take a look at the entire autonomous navigation process. The whole process can be summarized and categorized into five main components, namely: (1) perception –where the AV observes or sees and detects objects and features in the surrounding environment, (2) localization and mapping – where the AV positions itself on the road and relatively measures and positions everything it has detected, (3) path planning – where the AV simulates multiple routes and then determines what to take, (4) decision making – where the AV decides on how it will react and interact with everything else on the road whether it be static or moving, and (5) vehicle control – where the AV's inner mechanisms works with its own internal program to synch its mechanical parts with the AV's decisions (Brummelen, 2018). In this research, the focus will be mainly on the components of perception, and localization and mapping, more specifically on its connections with High-Definition (HD) Maps.

### High-Definition Maps

HD Maps are centimeter level precision 3D maps produced from Mobile Mapping Units (MMU) – which uses multiple high-quality sensors such as survey grade GPS, LIDAR, and cameras – that aides the AV better localize and navigate itself on the road. It is mainly composed of 4 layers, namely: (1) the geometric layer – which contains raw and processed sensor data, (2) the semantic layer – which contains information that can be found on objects and markings, (3) the map prior layer – which contains recorded human behavior patterns in an area, and (4) the real-time layer which contains real-time traffic data (Vardhan, 2017). By utilizing a combination of those layers, an AV can make wiser decisions in critical conditions such as maneuvering in highly crowded areas by tracking and predicting human motion, driving on cliff sided roads during bad weather by following virtual vector paths, or even when chasing down an escaping criminal on another speeding car by using real time traffic information to find the fastest route. Of course, those examples were quite extreme but nonetheless it still shows the potential of HD Maps.

However, since HD Maps are relatively new and is in still in the process of development there are still a lot of things that are needed to be worked on. One of which is updating an HD Map, making it dynamic, how real-time changes happening in the AV environment be reflected on the map (Brummelen, 2018). A crude solution could be to send-out MMUs as frequently as possible to have roads re-mapped in order for the map to adapt to new changes, but that is not only expensive but also very impractical. Imagine sending out an MMU just to find out that something has changed in the road literally a few seconds after it has passed. So, it will be very important that an AV not only use an HD Map but update it as well. This research will be working on that problem by proposing a preliminary updating framework for HD Map databases using a stereo camera.

## METHODOLOGY

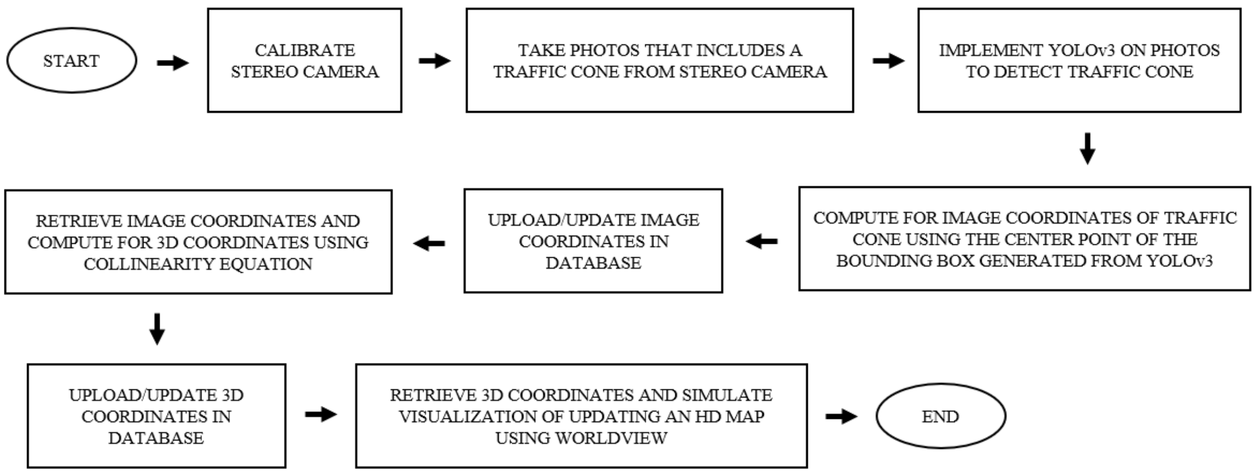

START
CALIBRATE STEREO CAMERA
TAKE PHOTOS THAT INCLUDES A TRAFFIC CONE FROM STEREO CAMERA
IMPLEMENT YOLOv3 ON PHOTOS TO DETECT TRAFFIC CONE
COMPUTE FOR IMAGE COORDINATES OF TRAFFIC CONE USING THE CENTER POINT OF THE BOUNDING BOX GENERATED FROM YOLOv3
UPLOAD/UPDATE IMAGE COORDINATES IN DATABASE
RETRIEVE IMAGE COORDINATES AND COMPUTE FOR 3D COORDINATES USING COLLINEARITY EQUATION
UPLOAD/UPDATE 3D COORDINATES IN DATABASE
RETRIEVE 3D COORDINATES AND SIMULATE VISUALIZATION OF UPDATING AN HD MAP USING WORLDVIEW
END

Figure 1. General Methodology Flowchart

Figure 2 shows a pair of Sony α6000 cameras attached on a piece of wood approximately 27 centimeters apart. This custom setup will act as the stereo camera that will be used for the research. The stereo camera is calibrated using MATLAB’s (version R2019a) built-in application: stereo camera calibrator. The calibration results provided both the camera’s internal orientation parameters, exterior orientation parameters (the relative orientation of the left camera with respect to the right camera), and their corresponding estimated errors. Lens distortion correction was not included in the calibration procedure. This research meant to simulate distance measurement from raw images without any sort of correction or modification.

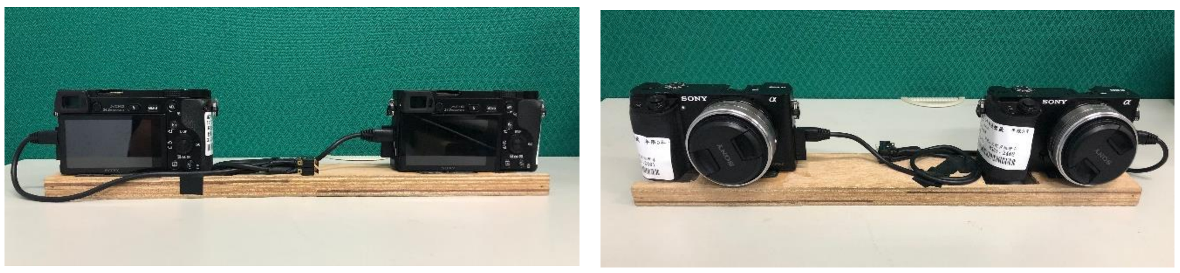


Figure 2. Custom Stereo Camera

A photo of a stationary traffic cone is then taken using the calibrated stereo camera. Its distance with respect to the camera setup is also manually taken using a steel tape for measurement comparison later on. Using a computer with an Intel i7 processor, 16 GB of RAM, and an NVIDIA GeForce GTX 1060 graphics card a custom Python implementation of YOLOv3, a real-time object detection system, is executed on the photos to detect the traffic cone as shown in Figure 3 (Redmon & Farhadi, 2018). It is also shown in the same figure that the detected object is enclosed inside a rectangular bounding box. The center point of this box will be computed and uploaded/updated in a locally constructed PostgreSQL pseudo HD Map database, and will later be retrieved to be used to represent the traffic cone in determining its position.

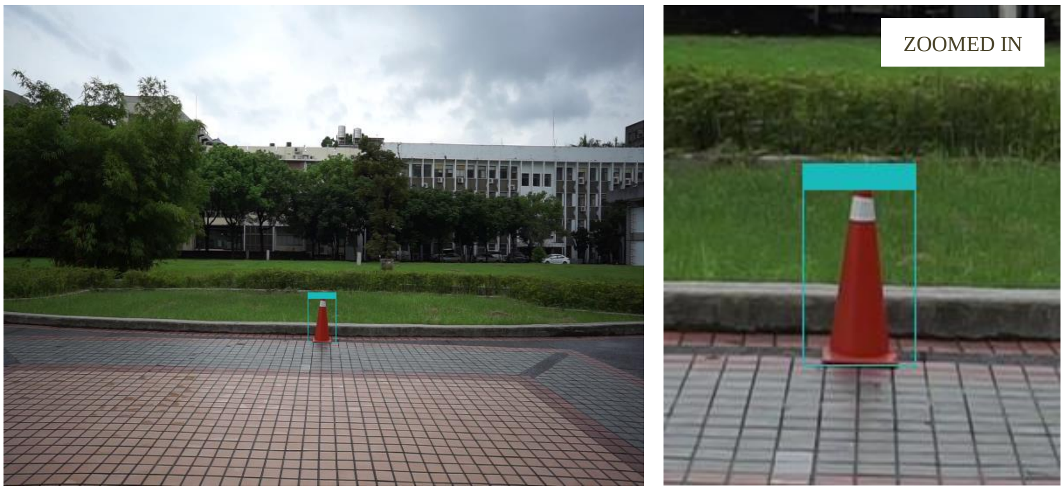


Figure 3. Detected Traffic Cone using YOLOv3

Using the calculated interior and exterior orientation parameters and the stored image coordinates, the relative 3D coordinates of the traffic cone is computed through a Python implementation of a least square iterated collinearity equation shown below in Equation 1 (Förstner & Wrobel, 2016). Delta (Δ) is computed using the previously generated parameters along with an initial assumption of the values of the traffic cone's coordinates. The process is iterated by adding Delta to the assumed coordinates and using the result as the new value of X, Y, and Z. This was repeated until the sum of squares of the values of Delta is greater than $10^{-10}$. The 3D coordinates are then uploaded/updated in the pseudo HD Map database.

$$\Delta = (B^T B)^{-1}(BF)$$

$$F = \begin{bmatrix} x - f\dfrac{N_x}{N_z} \\ y - f\dfrac{N_y}{N_z} \\ \cdots \end{bmatrix} \qquad B = \begin{bmatrix} \dfrac{\partial x}{\partial X} = -f\dfrac{r_{11}\cdot N_z - r_{13}\cdot N_x}{(N_z)^2} & \dfrac{\partial x}{\partial Y} = -f\dfrac{r_{21}\cdot N_z - r_{23}\cdot N_x}{(N_z)^2} & \dfrac{\partial x}{\partial Z} = -f\dfrac{r_{31}\cdot N_z - r_{33}\cdot N_x}{(N_z)^2} \\ \dfrac{\partial y}{\partial X} = -f\dfrac{r_{12}\cdot N_z - r_{13}\cdot N_y}{(N_z)^2} & \dfrac{\partial y}{\partial Y} = -f\dfrac{r_{22}\cdot N_z - r_{23}\cdot N_y}{(N_z)^2} & \dfrac{\partial y}{\partial Z} = -f\dfrac{r_{32}\cdot N_z - r_{33}\cdot N_y}{(N_z)^2} \\ \cdots & \cdots & \cdots \end{bmatrix}$$

$$N_x = r_{11}\cdot(X - X_0) + r_{21}\cdot(Y - Y_0) + r_{31}\cdot(Z - Z_0)$$

$$N_y = r_{11}\cdot(X - X_0) + r_{21}\cdot(Y - Y_0) + r_{31}\cdot(Z - Z_0)$$

$$N_z = r_{11}\cdot(X - X_0) + r_{21}\cdot(Y - Y_0) + r_{31}\cdot(Z - Z_0)$$

(Equation 1)

*Where,*

$X, Y, Z =$ *object coordinates of traffic cone*

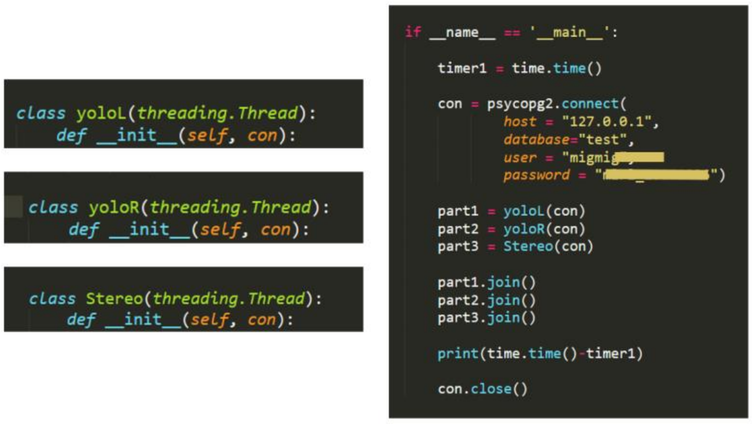
$x, y =$ *image coordinates of traffic cone*

$f =$ *focal length (from camera calibration)*

$X_0, Y_0, Z_0 =$ *exterior orientation of stereo camera (from camera calibration)*

$r_{nm} =$ *value of row n and column m of the rotational matrix (from camera calibration)*

All of the previously mentioned processes of detecting, obtaining and uploading the 3D coordinates of the traffic cone from the photos is executed at the same time to simulate a real-time working environment, much like what an AV would. It is carried out by a Python based program, shown in Figure 4. The program was mainly divided into three working classes: 2 (two) classes on calculating the image coordinates from the implemented YOLOv3 on the image pair obtained from the stereo camera, and 1 (one) class to calculate the 3D coordinates based on the camera parameters obtained from calibration and the image coordinates produced from the other classes. Since the program simultaneously runs the three classes, the class that computes the 3D coordinates was made to continuously run for it to receive the data it needs from the other two classes. All of the classes upload and or update their end products to the pseudo HD Map database.



```python
class yoloL(threading.Thread):
    def __init__(self, con):
```

```python
class yoloR(threading.Thread):
    def __init__(self, con):
```

```python
class Stereo(threading.Thread):
    def __init__(self, con):
```

```python
if __name__ == '__main__':

    timer1 = time.time()

    con = psycopg2.connect(
            host = "127.0.0.1",
            database="test",
            user = "migmi[redacted]
            password = "[redacted]")

    part1 = yoloL(con)
    part2 = yoloR(con)
    part3 = Stereo(con)

    part1.join()
    part2.join()
    part3.join()

    print(time.time()-timer1)

    con.close()
```

Figure 4. Parts of the Python Script for Obtaining 3D Coordinates

The X, Y, Z coordinates that was uploaded/updated by the program in the pseudo HD Map database will act as a simulated update of the traffic cone's detected appearance. The database will be connected to a browser-based application to visualize the simulated updating of an HD Map. The application is implemented through the use of Cruise Automation's Worldview. Worldview is a 3D scene renderer built on React and regl that is used for visualizing, exploring, and analyzing real and simulated AV data (Cruise Automation Team, 2018). The pseudo HD Map database, located at Tainan, Taiwan, will only be containing the raw X, Y and Z coordinates of a portion of an AV test field's LIDAR point cloud (features in red) as well as some lane boundaries vectors (features in green), as shown in Figure 5. Due to some restrictions on the usage of the AV test field's site, the image containing the traffic cone was taken at a different location. This part of the research will just be simulating the visualization process of the whole updating framework, showing the appearance of new/updated data, and as such the detected traffic cone's position inside the test field will only be assumed.

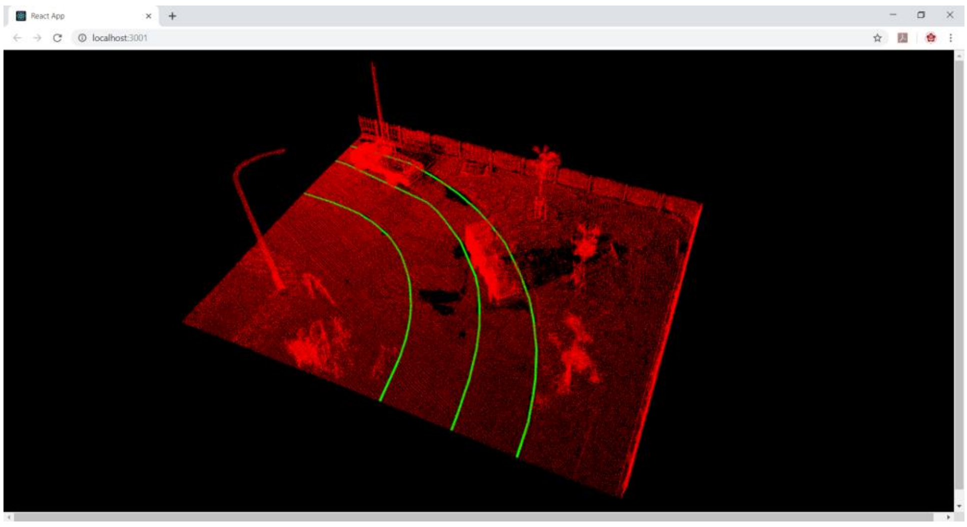


Figure 5. Visualizing the Pseudo HD Map in Worldview

## RESULTS AND DISCUSSION

The results of MATLAB's stereo camera calibration all have sub-millimeter estimated errors for both the internal and exterior parameters, as shown in Table 1 and 2. Its computed focal lengths for both the camera pair are almost only 0.1 mm short of the 16 mm value indicated by Sony for the camera model a6000. The 27.4 cm computed distance between both the cameras (the X translation of the relative exterior orientation) is also fairly close to the manually measured distance of 27 cm.

Table 1. Internal Orientation Parameter in Millimeters (mm)

| LEFT | | | RIGHT | | |
|---|---|---|---|---|---|
| | VALUE | ESTIMATED ERROR | | VALUE | ESTIMATED ERROR |
| FOCAL LENGTH | 15.9004 | 0.0151 | FOCAL LENGTH | 15.8703 | 0.0152 |
| PRINCIPAL POINT (X) | 11.5591 | 0.0187 | PRINCIPAL POINT (X) | 11.4181 | 0.0200 |
| PRINCIPAL POINT (Y) | 7.6025 | 0.0161 | PRINCIPAL POINT (Y) | 7.3865 | 0.0165 |

Table 2. Relative Exterior Orientation Parameters (Left with Respect to Right) in Millimeters (mm)

| | VALUE | ESTIMATED ERROR | | VALUE | ESTIMATED ERROR |
|---|---|---|---|---|---|
| X | 274.4578 | 0.1296 | $\omega$ | -0.2224 | 0.0648 |
| Y | -3.0607 | 0.1064 | $\phi$ | 1.7567 | 0.0842 |
| Z | -1.9746 | 0.5825 | $\kappa$ | 0.5294 | 0.0077 |

The proposed methodology's calculated distance from the camera set-up to the traffic cone has an 8.1 cm difference from the steel tape measured value and the calculated height of the traffic cone's center has a difference of 1.4 cm. The whole processing time of detecting the traffic cone from the images, computing its 3D coordinates and uploading them in the database, took approximately 20 ms to complete.

Table 3. Comparison of Observed and Computed Measurements in Meters (m)

| | Steel Tape | Collinearity Equation | Difference |
|---|---|---|---|
| Ground Distance from Camera Setup to Traffic Cone | 8.530 | 8.449 | 0.081 |
| Height of Traffic Cone's Center | 0.350 | 0.336 | 0.014 |

Figure 6 shows the visualization of the pseudo HD Map before the traffic cone was detected (left) and after the traffic cone's simulated appearance was updated (right). The blue feature on the right image of the figure represents the traffic cone.

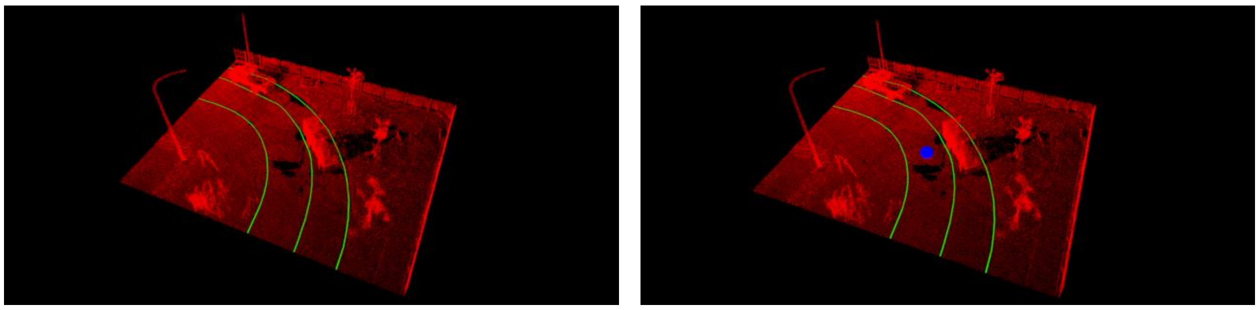


Figure 6. Visualization of Pseudo HD Map's Updating Simulation

Provided that both the camera setup and the feature of interest are stationary and is done on fair weather with moderate sunlight, the proposed methodology has shown that it can provide, at the least, below 10 cm positioning accuracy in the X, Y and Z axis which is relatively close to the 5 cm accuracy requirement of an HD map (Vardhan, 2017). The visualization of the traffic cone's appearance on the browser-based application also reflects the proposed methodology's successful stable connection between the pseudo HD Map database and the Python program that computes the traffic cones coordinates. Given that this all happened below a quarter of a second signifies that the proposed methodology can be regarded as a working functional preliminary HD Map database updating framework.

## CONCLUSIONS, RECOMMENDATIONS AND FUTURE WORKS

The research has successfully shown that the proposed preliminary updating framework for HD Map databases using a stereo camera has been successful. It provided below 10 cm positioning accuracy of the detected traffic cone and established a working stable connection with a local database in a span of 20 ms. It has also been shown that YOLOv3 can be a reliable tool for fast image object detection and that Python, PostgreSQL and Worldview are all viable connecting platforms in terms of AV, more specifically, HD Map applications.

It is highly recommended that the research be re-done by taking images of the stationary traffic cone on different sides and angles and test it on varying increasing distances of the camera pair with respect to each other to check if there will be significant changes in the results.

In future works LIDAR will be included in the setup to act both as an independent sensor as well as a checking mechanism for the stereo camera's results. Real AV data will also be used to factor in localization and mapping in the updating framework.

## REFERENCES

Brummelen, J.V. et al., 2018. Autonomous vehicle perception: The technology of today and tomorrow. Elsevier: Transportation Research Part C 89, pp. 384-406.

Cruise Automation Team, 2018. WORLDVIEW, Retrieved August 13, 2019, from https://webviz.io/worldview/#/docs/guides/quick-start.

Förstner, W. & Wrobel, B., 2016. Photogrammetric Computer Vision: Statistics, Geometry, Orientation and Reconstruction. Springer, pp. 547-615.

Redmon, J. & Farhadi, A., 2018. YOLOv3: An Incremental Improvement. arXiv.

Vardhan, H., 2017. HD Maps: New age maps powering autonomous vehicles, Retrieved August 13, 2019, from https://www.geospatialworld.net/article/hd-maps-autonomous-vehicles/.