Sensor Central

Automotive Systems

Sara Bahrieh

This thesis is presented as part of Degree of

Bachelor of Science in Electrical Engineering

Blekinge Institute of Technology

August 2013

Blekinge Institute of Technology

School of Engineering

Department of Electrical Engineering

Examiner: Dr. Sven Johansson

Abstract

How to display objects which were detected from different devices in one coordinate system? Nowadays most vehicles are equipped with front and back sensors to help the driver in driving process. Companies who provide this technology need to have an application which enables them for easy data fusion from these sensors and recording the process. Besides sensor's design, programming of them is an important aspect. BASELABS Connect has the solution in a user friendly way.

Creating Sensor Central component for BASELABS Connect is the main goal of this thesis. Sensor Central from BASELABS Connect requires six variables of sensor's position for each sensor to demonstrate the objects from all sensors to one unique coordinate system. In this thesis, it was intended to create such a component which was mounted between all the sensors and the charting component to convert the objects location from different sensor's position to one coordinate system and to be usable from other vehicles too.

Acknowledgments

I would like to express my special thanks to Dr. Eric Richter and Dr. Robin Schubert, my supervisors in BASELABS Company who gave me the opportunity to make this wonderful project and also Dr. Sven Johansson, my supervisor in Blekinge Tekniska Hogskolan who supported me in my thesis.

I would like to thank my colleagues in the BASELABS Company for their valuable time not only in my thesis but also in all other aspects during my stay in Germany.

I would like to appreciate Mr. Anders Hultgren, my program manager at Blekinge Tekniska Hogskola, for his supports and great suggestions.

And finally, I would like to thank my family and my friend, for their great help and support through all the time.

Contents

1. Introduction 8
   1.1 Motivation 8
   1.2 Scope of the thesis work 9
   1.3 Outline of the thesis 9
2. Background and related work 10
3. Foundations/ Design and implementation 13
   3.1 Requirements 13
   3.2 Sensor Central properties (Architecture) 13
      3.2.1 Charting component 13
      3.2.2 Sensors 14
      3.2.3 Sensor Central position 14
   3.3 Work design 15
   3.4 Implementation 16
      3.4.1 Homogeneous matrix 16
      3.4.2 What does exactly Sensor Central do? 18
      3.4.3 How to convert? 19
      3.4.4 How translation works? 19
      3.4.5 How rotation works? 20
4. Network mapping 21
   4.1 Tree 21
   4.2 Graph 21
      4.2.1 Directed and undirected graphs 22
      4.2.2 Weighted and un-weighted Edges 23
      4.2.3 Sparse graphs and dense graphs 24
5. Evaluation and testing 26
6. Conclusion 31
7. Future work 32
8. Reference list 33

Figure 2-1: Single-Sensored Vehicle 10
Figure 2-2: Multi-Sensored Vehicle 11
Figure 3-1: Sensor Central Position 14
Figure 4-1: Different types of Graphs[6] 22
Figure 4-2: Directionality of Graph[6] 22
Figure 4-3: Cost of edges[6] 23
Figure 4-4: Flowchart of the work 25
Figure 5-1: Graph flowchart 26
Figure 7-1: Vehicle to vehicle communications 32

Eq 3-1: Homogeneous Equation 16
Eq 3-2: Translation 17
Eq 3-3: Rotation round Y axis 17
Eq 5-1: Transformation Matrix 28
Eq 5-2: Implementing the transformation 29

1. Introduction

Advanced Driver Assistance Systems (ADAS) are such systems which bring new lifestyle to human's life for high safety driving and less car accidents. These systems are helping for more safety of roads and vehicles. Vast variety of sensors and cameras are used in ADAS to detect objects and alert the driver to avoid accidents. There are three types of sensors:

- Motion sensors
- Environmental sensors
- Detection sensors

Motion sensors are those type of sensors for measuring the acceleration and rotational forces in three dimensional spaces. For environmental parameters such as air temperature, humidity and pressure, developers use another kind of sensors which called environmental sensors. In addition, for detecting the physical position of the objects or devices, the position sensors are used in ADAS systems. The focus of this thesis is on position sensors which can detect the objects. Position sensors can be mounted at different places of a vehicle for specific reasons. Data from all the sensors are gathered and processed by ADAS and a simple and easy to understand result is displayed on a screen inside the vehicle to help the drivers for safe driving and avoid future accidents. Converting all objects positions from gathered data to a unique coordinate system would be a problem for ADAS developers. A solution to that is to implement such a coordinate conversion system which could comfort the developing process for ADAS developers [1].

1.1 Motivation

Developers, who use ADAS systems to improve their products, have different design for sensors in different vehicle models. These designs, including sensors positions on the car, are a significant aspect for improving the performance of an ADAS system. After locating the sensors, the data should be collected in one unique coordinate system to be usable by the driver or ADAS developer. While detecting objects, each sensor and cars have their own coordinate system and deliver the information based on that. Therefore the ADAS developers need to invent an intermediate component which is able to transfer the object's position to a fixed coordinate system after data acquisition. It is time consuming for them to create such codes every time since it is not easy to implement the transformations method. The codes are not reusable for other devices whereof they are very specific for those sensors and devices. Furthermore, the intermediate component which they use it as a converter needs to know the coordinate system of sensors repeatedly during the conversion. To solve this problem, creating a new object which could do the transformation for received data convenient, is presented in this work.

1.2 Scope of the thesis work

The main focus of this thesis is to create such a component for BASELABS customers (ADAS Developers), which is called Sensor Central. It could easily adapt to any system, therefore the ADAS developers can use it in a very broad area of work. The Sensor Central can store all the information about the sensors such as coordinate systems and the relations between each other. It can be quickly implemented during the design. With this component one can easily design the sensors positions without worrying about future extra effort.

1.3 Outline of the thesis

There are the following main chapters which constitute this report:

- Single-Sensored
- Multi-Sensored
- Charting component
- Sensor Central
- Graph and tree
- Vehicle to vehicle communications

2. Background and related work

There are already vehicles available which have one sensor mounted on them to detect the obstacles. As shown in Figure 2-1, there are some advantages and disadvantages to single-sensored vehicles. The advantage to this system is that it is simple to use in a way that there is only data coming from one sensor and is displayed on a monitor for the driver to know about his car's surroundings.

Figure 2-1: Single-Sensored Vehicle

On the other hand, the disadvantage is the area of coverage, in other words one sensor cannot detect a wide range of area and provide the driver with reliable and accurate information.
Another disadvantage of single-sensored vehicles is false positive and false negative. False negative in this content means that there might be reports and data from the sensor which indicate no presence of any obstacle where an obstacle does exist. On the other hand, in false positive there might be moments where the sensor wrongly detects an obstacle where no obstacle exists on the way of the vehicle.

The problems arise in single-sensored vehicles weigh more than their benefits and due to this fact in this work it is intended to work with multi-sensored vehicles.

Multi-sensored vehicles have higher accuracy and reliability data acquisition respect to their single-sensored counterparts. The main benefit is area of coverage in terms of detection. They are even more optimized in false positive and false negative rates because there are more data sent from the sensors to process and analyze and in this way, more accurate result is derived.

One of the challenges that many vehicle companies still struggle with is the coordinating these sensors together. The more sensors employed on a vehicle, the more challenging the coordination. This problem arises of the fact that each sensor gathers its own data and sends it to the vehicle. This data depends on the spot of the sensor where it is mounted on the car. To get a better imagination of this situation, an example is provided. Consider a vehicle with 3 sensors mounted on the front bumper, one on the right, one the left and the other one right in front of the bumper. Each of these sensors is transmitting data back to the data unit of the vehicle to be processed and displayed to the driver every moment. But the problem is the coordinate of each sensor is based on the spot where it is mounted. Therefore there is always a need to translate each of the coordinates coming from the sensors and bring them all into a unique coordinate system as shown below. This operation is necessary to make sure that each data is meaningful when brought together.

Figure 2-2: Multi-Sensored Vehicle

A way to solve this problem is to define an intermediate component where stands at the intersection of these sensors and translate each coordinate and brings it to the unique and understandable coordinate system. To do this for every system there is a need to build such intermediate component because the sensors are different from system to system and many more variables which vary along the process. This does not seem to be an efficient solution to this challenge.

- Expensive: To build different intermediate component for each system is costly due to labor, time and materials used to build and to test and these are not in favor of the company.

- Time consuming: Intermediate component is not an easy task to be instantiated for every system once there is a new system introduced. Therefore it is time consuming to be built upon request and since each system needs its own intermediate system in this way, therefore many tests should be run to make sure of the quality of service and capability of the new component.

Therefore in this work a system is built to be flexible and reusable to various types of systems. This solution has the following advantages which covers the following capabilities:

- Flexibility: This feature helps this component to be used on every system. Flexibility avoids the re-creation and re-designing a new intermediate component for every system.

- Cost efficient: Reduces many costs due to flexibility of use.

- Time saving: Drastically decreases the time due to flexibility of use.

- User friendly with BASELABS Connect: It is well suited and designed to be used as in one package with BASELABS Connect.

3. Foundations/ Design and implementation

3.1 Requirements

The purpose of this thesis is to accomplish a new method with homogeneous matrices and a data structure graph that can store and convert all the information received from one device. Graphs comprise of nodes and edges which nodes are representative of objects and edges are the links between nodes. The graph library is to build a graph with specific definitions for each edge and is used for converting the data received from all sensors and devices with high readability by different vehicles. Graphs contain nodes. The graph edges have to carry the conversion method with itself, to use when it is needed.

In this thesis the BASELABS Connect used for implementing the algorithm. BASELABS Connect is created for an environment that consists of different component which could able to connect for transferring the data. This data might be data sources like sensors, sinks like processing or both in corporate. Each component carries a particular task namely data acquisition from a sensor, data processing or visualization. There are inputs to and outputs from each component. Inputs and outputs are called in pin and out pin respectively. There is no constraint over the number of pins and also the order of data delivery can be arbitrary. There are some properties to each component such as Name which distinguishes the components and beside that there is a name which the user can feel free to determine and is known as InstanceName, e.g. “Right sensor”.

3.2 Sensor Central properties (Architecture)

Sensor Central is a dependent component which is located between components in a design system for convenient conversion. The component can store all the information from the sensors which are attached to the system. It also contains the position, the central coordinate system and the relations between the sensors locations. In a broad perspective, it can also hold the information which could be usable by other vehicles in a near area as well. The information is only entered once by developers. Sensor Central is adaptive to any system therefore it can be used for any issue related to coordinate system conversion. The Sensor Central ability is not only for storing the data, but also for converting the data based on the saved information. The converted data then transferred to the Charting Component.

3.2.1 Charting component

Charting component is used for drawing a 2 dimensional diagram from sensors data. It is an easy way to display the obstacles which were detected by sensors. In the case of vehicle to vehicle communications, the charting component can also display the information obtained from the stationary system, so that the ADAS developer can have an imagination of the cars surroundings. Charting component contains input pins for receiving the data and performs the calculations to display them. Types of the data differ from each other due to its usage and need to be specified in the coding.

3.2.2 Sensors

Sensors in the vehicles can be placed at any suitable place for more street coverage and efficiency depending on type of vehicle. They also can have different angles for covering the wider area. Sensor Component is the one which plays the role as a sensor and receive the data from the sensor and pass the data to Charting component and is updated every millisecond. The data received from Sensor Component which is connected to Charting Component is displayed in a two dimensional diagram.

Even if there is only one sensor mounted on the vehicle, the detected objects should be displayed with car's coordinate system and therefore that sensor needs to connect to Charting Component for primary calculation.

3.2.3 Sensor Central position

Sensor central position is of important matters. Figure 3-1 shows the place where the sensor central is located. It is connected to the charting component and the sensors on the vehicle where it could modify the data from sensors and presented them in charting component.

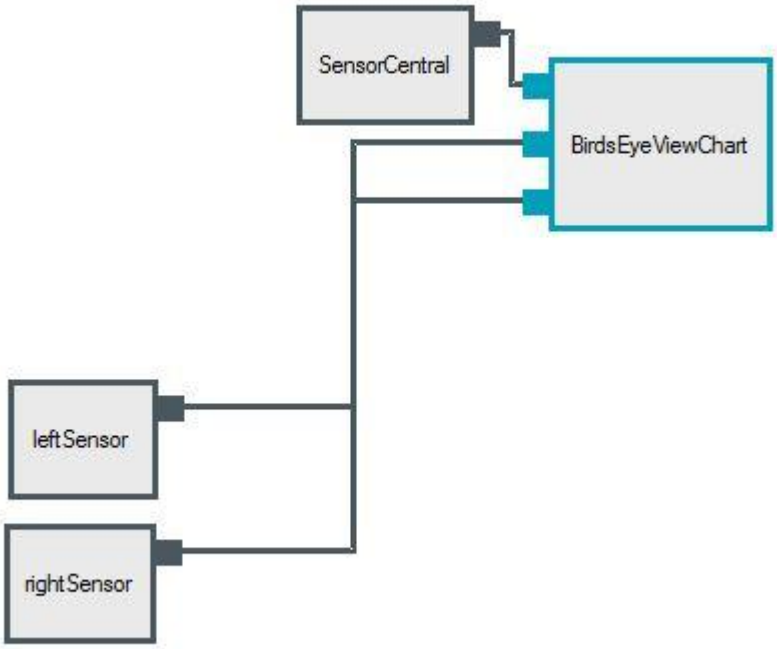

Figure 3-1: Sensor Central Position

3.3 Work design

There are several ways to implement the transformation on the detected points but to find the most convenient one, the homogeneous method for matrices have to be used for multiplication of the translation and rotations [3].
For each translation and rotation, a 4 by 4 matrix is used to represent the data. All matrices must be multiplied to each other and at the end only one matrix is left in Sensor Central for converting the data which is obtained by sensors or any other source point.

As explained earlier, the way which is used to obtain the final transformation matrix is important in calculation and implementing in Sensor Central. There is priority level in implementing translation and rotation which must be followed. Translation always has a higher priority over rotation meaning that the received data is first multiplied by translation and after that it is multiplied by rotation around any axes.

In order to implement these multiplications and also creating a component in BASELABS Connect, a high performance programming language is needed. In this thesis, all implementations, calculations and processes are done using C# programming language. In order to make sure of the quality of service Microsoft Visual Studio 2010 was chosen. This choice also gives the advantage of using a high prices and great debugger tool. Since Microsoft Visual Studio is considered as the main working environment of C# and is updated frequently, it gives the developer to work on a stable platform.

There are various types of libraries that could have been employed in this work, but to make sure of popularity of the library and its reliability, QuickGraph library was selected from Codeplex which is a Microsoft public license. It is an open source library which provides generic graphs data structures and algorithms for .NET. The QuickGraph library was selected due to some requirements in this work. The necessity of a data structure tree which allows tagging the nodes and edges for transferring the received data from sensors between the vehicles is one of the significant requirements. Moreover, the algorithms for searching via the tree for finding the proper object, such as depth first search, shortest path and breadth first search are included in the library. The shortest path search was employed during this work. In this work the shortest path is used for converting and transferring the data. This library also covers the nodes and edges with any required data [2].

3.4 Implementation

3.4.1 Homogeneous matrix

This work is based on homogeneous transformation. Homogeneous transformation enables us to composite several translation and rotation. The homogeneous matrix equations are the equations in which there is an A coefficient matrix, x the variable vector and b the known constant. Therefore the equation Ax = b is a homogeneous equation if and only if b is zero. The following matrix equation is an example of a homogeneous equation [3]:

$$\begin{bmatrix} 2 & 3 & 0 \\ 0 & 5 & -1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Eq 3-1: Homogeneous Equation

The homogeneous matrix equations have some important properties as follows:

- The equations always have at least one solution where all the variables are zero, the equation is still valid. This is the case where it is called trivial solution.
- Base on a theorem: "A square matrix ***M*** is invertible if and only if the homogeneous matrix equation ***Mx=0*** does not have any non-trivial solutions.", if there is a non-trivial solution exists for ***Mx=0***, then ***M*** is not invertible. Besides, the return way of this theorem is also working, means that if the ***M*** matrix has an inverse matrix ***M'***, then the only solution for ***Mx=0*** is the trivial solution where ***x=0*** [4].
- In the case of fewer equations than parameters, there is always a non-trivial solution in homogeneous systems.
- Another theorem says that: "If ***x=p*** is a solution to the matrix equation ***Mx=b***, then any other solution to the equation is of the form ***p + h***, where h is a solution to the homogeneous equation ***Mx=0***". Meaning that for solving a matrix equation, the single particular solution only needed when a homogeneous case was solved [4].

This work is done by using the homogeneous matrix equation for the properties which is useful in matrix multiplication during the conversion.

Transformation order is important in calculation due to some reasons. One significant reason is that rotation and scaling depend on the origin of the coordinate system, means that the result of an object's rotation is different when the origin moves. The origin will be shifted during translation. Therefore, when the rotation is done first, and afterward the translation comes, the result would be based on two origins. One with the first origin and second one with the new translation and this leads to the wrong result. However, when the translation is exerted first, the

origin will be changed and rotation uses the new origin for rotating. This also stands for scaling. Since conversion is constituted of translation and rotation, the priority of multiplying must be considered and implemented in the right order [4].

For instance, imagine that the sensor detects one obstacle with the coordinate of (4, 4, 4) base on its own coordinate system. A translation with coordinate (3, 3, 3) and one rotation around Y axis with angle π is applied to the system. The coordinate of detected obstacle in the new system is (-1, 1, -1), if the translation applied first. However, the answer would be (-7, 1, -7) if the rotation is applied before translation and in this case the answer is different and of course it is wrong.

$$\boldsymbol{P_1} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

The following calculations, implement the translation and rotation in different order to demonstrate that the calculation would be wrong if the rotations comes first. In this calculation T represent the translate matrix and R represent the rotation matrix, so when translation is exerted first the calculation would be as follows:

$$\boldsymbol{T}.\boldsymbol{P_1} = \boldsymbol{P'}$$ *Eq 3-2: Translation*

$$\begin{bmatrix} 1 & 0 & 0 & -3 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 4 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \times 4 + 0 \times 4 + 0 \times 4 + (-3) \times 1 \\ 0 \times 4 + 1 \times 4 + 0 \times 4 + (-3) \times 1 \\ 0 \times 4 + 0 \times 4 + 1 \times 4 + (-3) \times 1 \\ 0 \times 4 + 0 \times 4 + 0 \times 4 + 1 \times 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\boldsymbol{Ry}.\boldsymbol{P'} = \boldsymbol{P_2}$$ *Eq 3-3: Rotation round Y axis*

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \times 1 + 0 \times 1 + 0 \times 1 + 0 \times 1 \\ 0 \times 1 + 1 \times 1 + 0 \times 1 + 0 \times 1 \\ 0 \times 1 + 0 \times 1 + (-1) \times 1 + 0 \times 1 \\ 0 \times 1 + 0 \times 1 + 0 \times 1 + 1 \times 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}$$

However, the answer is completely different when rotation is employed before translation and the calculation below show how it effects the answer:

$$\boldsymbol{Ry}.\boldsymbol{P_1} = \boldsymbol{P'}$$

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 4 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \times 4 + 0 \times 4 + 0 \times 4 + 0 \times 1 \\ 0 \times 4 + 1 \times 4 + 0 \times 4 + 0 \times 1 \\ 0 \times 4 + 0 \times 4 + (-1) \times 4 + 0 \times 1 \\ 0 \times 4 + 0 \times 4 + 0 \times 4 + 1 \times 1 \end{bmatrix} = \begin{bmatrix} -4 \\ 4 \\ -4 \\ 1 \end{bmatrix}$$

$$\boldsymbol{T}.\boldsymbol{P'} = \boldsymbol{P_2}$$

$$\begin{bmatrix} 1 & 0 & 0 & -3 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -4 \\ 4 \\ -4 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \times (-4) + 0 \times 4 + 0 \times (-4) + (-3) \times 1 \\ 0 \times (-4) + 1 \times 4 + 0 \times (-4) + (-3) \times 1 \\ 0 \times (-4) + 0 \times 4 + 1 \times (-4) + (-3) \times 1 \\ 0 \times (-4) + 0 \times 4 + 0 \times (-4) + 1 \times 1 \end{bmatrix} = \begin{bmatrix} -7 \\ 1 \\ -7 \\ 1 \end{bmatrix}$$

The calculation above, showed that it is important to consider the order of the multiplication during the conversion. Therefore, this property is implemented in the code so that any data which is received from the sensors is multiplied by translation matrix first and afterward by rotation.

3.4.2 What does exactly Sensor Central do?

There are different tasks that are available in Sensor Central component namely translation and rotation around 3 axes. The data received at Sensor Central requires user interaction to determine about the operations needed to be performed on the data. There might be cases where only translation is needed and no rotation calculation and also there might be cases where rotation around x or y or z or even all of them are needed. Even the user could give the data for translation and rotation and in this case user takes advantage of Sensor Central at full capacity.

Upon receiving the data, Sensor Central creates a matrix for each of the entries given by the user. The needs to have these matrices are to apply mathematical operations on them.
Depending on the aforementioned cases, if there are translation data available then those translation matrices are multiplied to each other and the result will be multiplied by all the other matrices which are rotational matrices.

If there is no translation matrices the order is not important and all the matrices can be multiplied in any order. This order is of important factor due to fact that when translation happens the origin of coordinate changes and this creates a new origin for the coordinate. If the rotation matrices are multiplied first then after that translation happens, in this way the calculation of translation and

rotation are based on different origin.

3.4.3 How to convert?

The result matrix after all the multiplications is the most important and final matrix for converting. This final matrix is multiplied to any received data in Charting Component and the destination point which was expected from the desired coordinate system is calculated. This means in every edge there is an assigned matrix and in case of vehicle to vehicle communication, data going and coming via each edge is multiplied by the matrix of that edge and the result is converted.

3.4.4 How translation works?

There are 2 types of translations. One case is that the coordinate itself is translated respect to the axis. In this case the axis is fixed and remains still, but the coordinate is translated based on the axis. This is a good solution where there is possibility to move the coordinate. But since in this case the coordinates are objects and obstacles on the road, there is no chance to be able to move them. Due to this fact, other type of translation has been used which is the second type where the coordinate remains fixed and only the axis are translated based on the coordinate. The translation matrix is a 4×4 matrix.

When the axis is fixed the translation matrix is as follows:

$$\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & cx \\ 0 & 1 & 0 & cy \\ 0 & 0 & 1 & cz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For instance, the new coordinate A (2, 2, 2) with the translation (3, 3, 3) would be:

$$\begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \\ 5 \\ 1 \end{bmatrix}$$

However, the new coordinate A (2, 2, 2) after translation based on axis, has different position compared to the first type. This matrix shows the calculation of translation in case where the coordinate is fixed and the axis are moved.

$$\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & -cx \\ 0 & 1 & 0 & -cy \\ 0 & 0 & 1 & -cz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

And here is the calculation of the same coordinate A:

$$\begin{bmatrix}1 & 0 & 0 & -3\\0 & 1 & 0 & -3\\0 & 0 & 1 & -3\\0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}2\\2\\2\\1\end{bmatrix} = \begin{bmatrix}-1\\-1\\-1\\1\end{bmatrix}$$

3.4.5 How rotation works?

As there were 2 cases in translations, the same cases happen in rotations. Therefore one case is when the coordinate rotates around its axis and another case is when the axis is rotated based on coordinate. Still due to aforementioned reason, only axis is rotated in this work. The rotation matrix is a 4×4 matrix.

The matrices below are the ones which the coordinate itself moves and the axis are fixed [5]:

$$\boldsymbol{Rx}(\boldsymbol{\alpha}) = \begin{bmatrix}1 & 0 & 0 & 0\\0 & \cos(\alpha) & -\sin(\alpha) & 0\\0 & \sin(\alpha) & \cos(\alpha) & 0\\0 & 0 & 0 & 1\end{bmatrix}$$

$$\boldsymbol{Ry}(\boldsymbol{\gamma}) = \begin{bmatrix}\cos(\gamma) & 0 & \sin(\gamma) & 0\\0 & 1 & 0 & 0\\-\sin(\gamma) & 0 & \cos(\gamma) & 0\\0 & 0 & 0 & 1\end{bmatrix}$$

$$\boldsymbol{Rz}(\boldsymbol{\theta}) = \begin{bmatrix}\cos(\theta) & -\sin(\theta) & 0 & 0\\\sin(\theta) & \cos(\theta) & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & 1\end{bmatrix}$$

Where the matrices which are used in this work are:

$$\boldsymbol{Rx}(\boldsymbol{\alpha}) = \begin{bmatrix}1 & 0 & 0 & 0\\0 & \cos(\alpha) & \sin(\alpha) & 0\\0 & -\sin(\alpha) & \cos(\alpha) & 0\\0 & 0 & 0 & 1\end{bmatrix}$$

$$\boldsymbol{Ry}(\boldsymbol{\gamma}) = \begin{bmatrix}\cos(\gamma) & 0 & -\sin(\gamma) & 0\\0 & 1 & 0 & 0\\\sin(\gamma) & 0 & \cos(\gamma) & 0\\0 & 0 & 0 & 1\end{bmatrix}$$

$$\boldsymbol{Rz}(\boldsymbol{\theta}) = \begin{bmatrix}\cos(\theta) & \sin(\theta) & 0 & 0\\-\sin(\theta) & \cos(\theta) & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & 1\end{bmatrix}$$

4. Network mapping

As this work is part of the big project that enables vehicles to know about each other's coordinates in order to avoid accidents, traffic and many other applications, it requires an easy to understand way of plotting to show the relation between each vehicle and how they are connected to each other. To manage this task a nice map is needed. There were 2 options to pick from. One is called Graph and the other one is Tree [6]. Since there are major differences between these two, a precise consideration had to be taken into account.

4.1 Tree

Trees follow the parent and children pattern. In trees there are certain properties as follows;

1. All the nodes of the tree are connected to each other, in other words, there is no node that cannot be reached through some simple path.
2. There are no loops or cycles in the tree meaning that there is no such path that starts from one node; say v_1 and passing through nodes v_2,v_3,...v_n and comes back again at node v_1.
3. In trees the numbers of edges are always one less than the number of nodes.

4.2 Graph

Graphs are abstract connections of points. Each point is also referred to as Vertex or Node. Graphs are employed to describe the state transitions. These states could be anything. Some are waiting for a decision; some are waiting for user interaction with the program and many other states.
The graphs could be directed and undirected meaning that the flow direction could be of matter which is directed and could be of no matter which is undirected. Edges of the graph could be weighted or unweighted. This actually means that each edge might have a number or value associated with it. Even when direction matters, each direction might have different value of as the weight. In this work a graph is a map that simply shows vehicles as nodes and the links between them as its edges. Graphs are like maps to show different aspects of relation between two objects such as the number of ways to get from one node of the graph to another, to find the shortest path possible from one node to another, and many more possibilities. One thing to note about graphs is that there are no rules dictating the connection among the nodes. When dealing with graphs, there is no worries about concept of a root node, nor is there a concept of parents and children. In simple words, a graph is just a collection of interconnected nodes.

Another advantage with graph is that graphs can have separate sets of nodes from other sets of nodes. As shown in Figure 4-1 in graph (a) contains two totally separate sets of nodes but they are still considered as one graph. Another important factor that distinguishes graphs from trees is that graphs have cycles can have cycles as indicated in graph (b). As shown there is a path starting from v1 to v2 and then to v4 and finally going back again to v1, this is said to be one

cycle. Graph (c) has exactly the properties of a tree since it does not contain any cycles; the numbers of edges are one less than number of nodes. Therefore, it is a tree. Then it can be said that trees are some type of graph.
In our daily life there are many applications that use graphs such as search engines which model the web and link the web pages as the nodes and link them together as the edges of the graph.

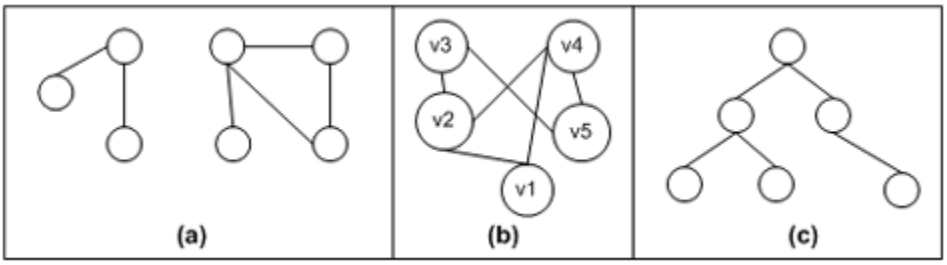

Figure 4-1: Different types of Graphs[6]

4.2.1 Directed and undirected graphs

In graphs, edges are used to make connections between nodes. When speaking of graphs, it is presumed to be bidirectional. This means that if there is an edge between node v and u, then one can go from v to u and vice versa as shown in Figure 4-2. A graph with bidirectional edges is said to be undirected graph since there is no implicit direction in its edges.

On the other hand there are cases where there is a need to have implicit direction in the edges. This indicates that if there exists an edge between node v and u, that edge is only allowed in either v to u or u to v and not both directions. These types of graphs are said to be directed graphs which have unidirectional edges.

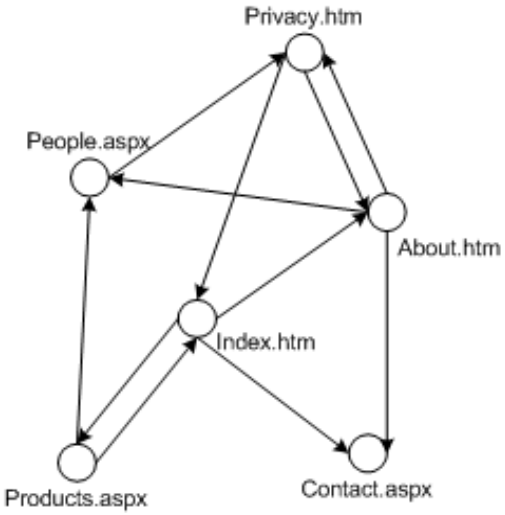

Figure 4-2: Directionality of Graph[6]

4.2.2 Weighted and un-weighted Edges

Sometimes it is essential for edges to have cost associated with them. For example in creating a map between cities one needs to consider the distance between cities and take into account the shortest path between cities. This task using graph is possible through assigning weight to the edges which basically indicates how far the distance is from one node to another which is linked using this edge.

Figure 4-3 shows graph with weighted edges. From this graph one can calculate the distance between one city to another by summing costs of the edges on the way. The shortest path would be of course the edge with the least cost. For example in the Figure 4-3 there are multipath from San Diego to Santa Barbara. If one goes from Riverside and then to Barstow and back to Santa Barbara, he has traveled for 210 miles which is the cost of the whole way. But if he decides to choose the shortest path then he should travel to Los Angeles, and then to Santa Barbara which is only a 130 miles trip.

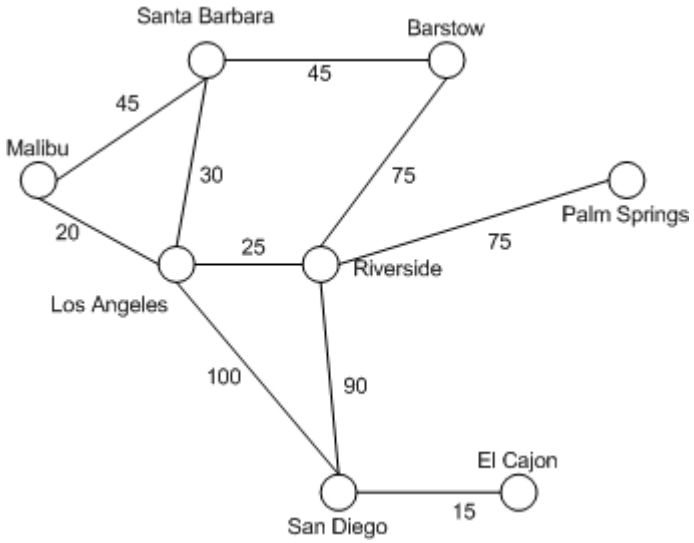

Figure 4-3: Cost of edges[6]

Note that directionality and weightedness of edges are the properties which make up to four different combinations for graph type as below:

- Directed, weighted edges
- Directed, unweighted edges
- Undirected, weighted edges
- Undirected, unweighted edges

4.2.3 Sparse graphs and dense graphs

Although the number of edges varies from graph to graph, but typically a graph has more edges than nodes. This number depends on whether the graph is directed or undirected. A directed graph could have an edge for each node to another node. Therefore n nodes could have $n-1$ edges which is $n \times (n-1)$edges that is nearly n^2 (self-edges which are edges that start from one node and returns back to the same node is not considered in this content) [6].

Due to completeness of graph library chosen in this work, the graph representation was used. These differences are, searching for the shortest path, in the tree it was not possible to store the data in the nodes, but it was possible to do so though graph library. Another factor was that there was no specific algorithm to tag transformation to the edges of tree. Furthermore, in the future work, when the task is expanded, the tree cannot support all the requirements for converting the nodes and data.

To get a better perception of the overall steps of the work the following flowchart is provided.

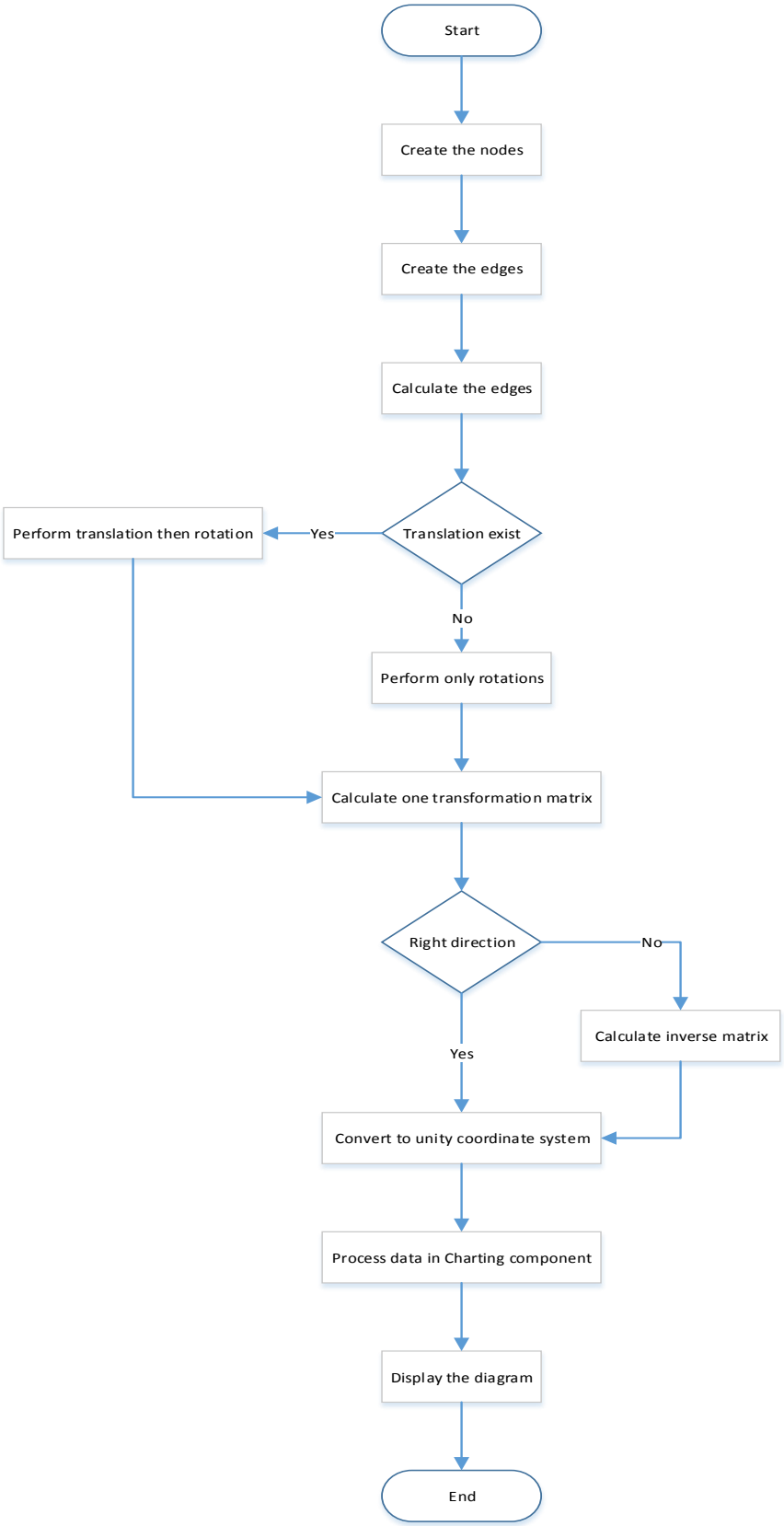

Figure 4-4: Flowchart of the work

5. Evaluation and testing

There were various testing in every implemented parts of the work. These include testing different part of the code, mathematical testing and algorithm testing. Besides testing, analyzing the work took up part of the testing procedure.

To evaluate and test the overall performance of the work, two vehicles which each has two sensors mounted on them are to be considered as shown in Figure 5-1.

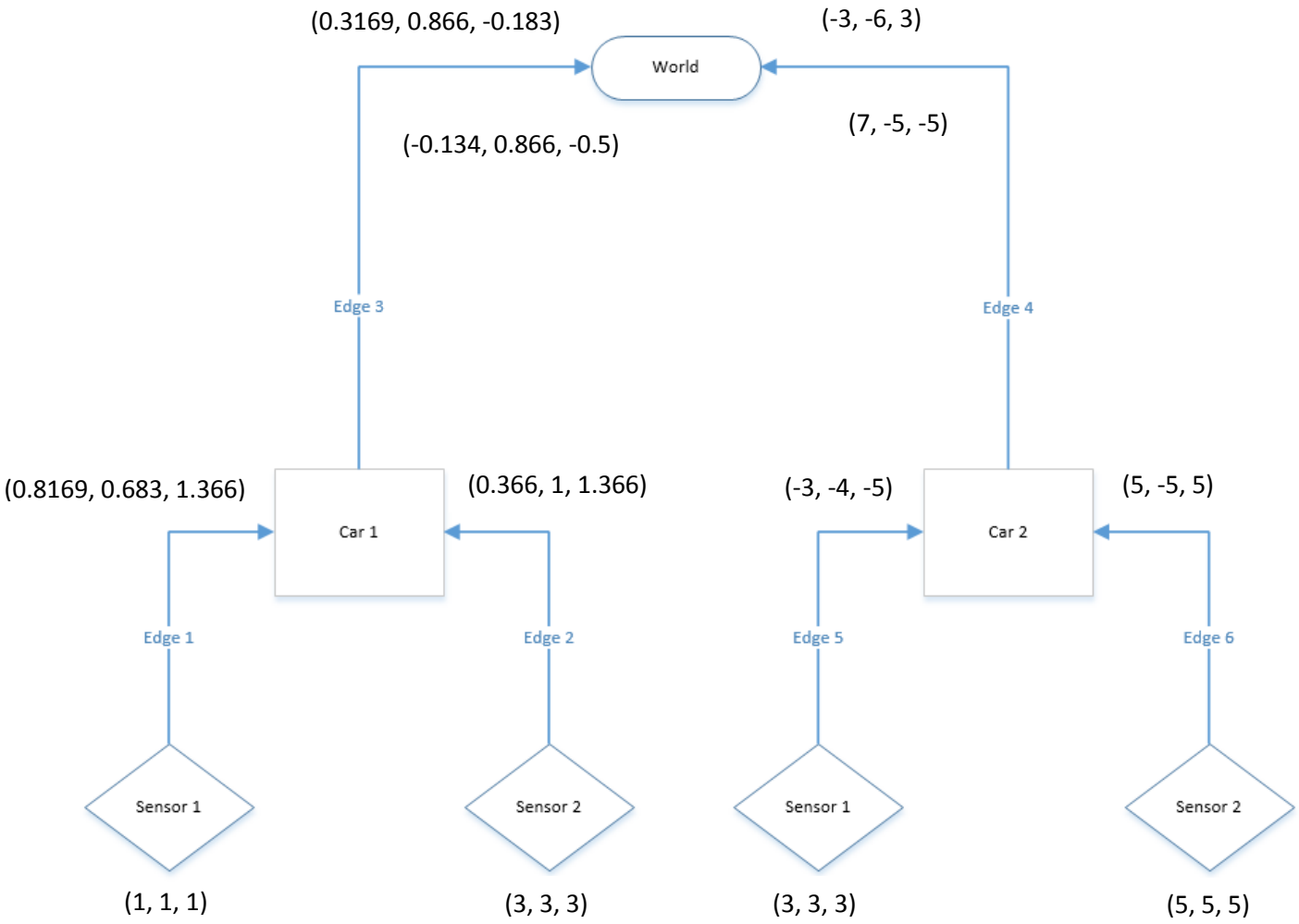

Figure 5-1: Graph flowchart

As shown above depending on the design of the car which determines the sensors position, the edges have different transformations calculation. The figure shows that sensor 1 has detected an obstacle based on its own coordinate system at location P_1 (1, 1, 1) and is sending this coordinate for instance to car 2. The following lines also indicate the test for each edges which is consider in the thesis to be tested for calculation. Here, it is assumed that each sensor coordinate system, has a translation or rotation respect to the main coordinate system of the car. Therefore, each edge should carry the information based on the coordinate system's transformation.

The first car has two sensors, one with two rotation around the axes and the other one has a translation and a rotation respect to the coordinate system of the car. These information are saved to edge 1 and edge 2 of the graph in Sensor Central respectively. The car itself is located in a situation where it has a translation and rotation respect to world coordinate system which could be either UTM or UPS. These coordinate systems are explained more in the Future work part and it is not implemented in this work, but to demonstrate that this graph can be expanded in broad area, this part is done as a pattern. The information for car location based on the world coordinate system are stored in edge 3.

$$Edge\ 1: \boldsymbol{R_y(\pi/6)} = \begin{bmatrix} 0.866 & 0 & -0.5 & 0 \\ 0 & 1 & 0 & 0 \\ 0.5 & 0 & 0.866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \ \&\ \boldsymbol{R_z(\pi/6)} = \begin{bmatrix} 0.866 & 0.5 & 0 & 0 \\ -0.5 & 0.866 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Edge\ 2: \boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \ \&\ \boldsymbol{R_y(\pi/2)} = \begin{bmatrix} 0.866 & 0 & -0.5 & 0 \\ 0 & 1 & 0 & 0 \\ 0.5 & 0 & 0.866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Edge\ 3: \boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & -0.5 \\ 0 & 1 & 0 & -0.5 \\ 0 & 0 & 1 & -0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \ \&\ \boldsymbol{R_x(\pi/2)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The second car also is in the same situation as first car, means that it has two sensors where each of them has different coordinate system respect to the main coordinate system and the information are supplied to the edges which is shown in the Figure 5-1.

$$Edge\ 4: \boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \ \&\ \boldsymbol{R_y(\pi/2)} = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Edge\ 5: \boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & -6 \\ 0 & 1 & 0 & -7 \\ 0 & 0 & 1 & -8 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Edge\ 6: \boldsymbol{R_x(\pi/2)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \& \boldsymbol{R_y(\pi/2)} = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \&$$

$$\boldsymbol{R_z\left(\frac{\pi}{2}\right)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The following calculations show how the transformation is applied on the graph. As mentioned earlier, supposed that the sensor one in car one is detected one obstacle in front of itself with the coordinate $\boldsymbol{P_1}$ based on its own coordinate system. To alert car two about this object and avoid future accident, this information is sent to car two. However, the coordinate $\boldsymbol{P_1}$does not make any sense for car two and the reason is their difference in their position and origin of their coordinate system. Therefore, the calculation should have employed in coordinate $\boldsymbol{P_1}$to have a correct information at destination. The matrix $\boldsymbol{edge1}$ is calculated as a transformation matrix for edge one and it is attached to the edge so any data which is received from that sensor is multiplied by the transformation matrix of edge one and so on.

$$\boldsymbol{P_1} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\boldsymbol{edge1} = \boldsymbol{R_y} \times \boldsymbol{R_z} \qquad \textit{Eq 5-1: Transformation Matrix}$$

$$\boldsymbol{edge1} = \begin{bmatrix} 0.866 & 0 & -0.5 & 0 \\ 0 & 1 & 0 & 0 \\ 0.5 & 0 & 0.866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.866 & 0.5 & 0 & 0 \\ -0.5 & 0.866 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.75 & 0.5 & -0.433 & 0 \\ -0.433 & 0.866 & 0.25 & 0 \\ 0.5 & 0 & 0.866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The inverse calculation is also calculated to demonstrate the ability of the graph to use bidirectional flow. So the data can converted from anywhere on the graph.

$$\boldsymbol{edge1Inverse} = \begin{bmatrix} 0.75 & -0.433 & 0.5 & 0 \\ 0.5 & 0.866 & 0 & 0 \\ -0.433 & 0.25 & 0.866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The edge 1 now store two matrices, one for converting from sensor to car and one the inverse flow. These matrices are used for all the data coming from the sensor and is not changeable. In case, the sensor position change, the data have to be correct with the user. The calculation below, show a received data from sensor one which is converted by the information attached to the edge.

$$\boldsymbol{P'_1} = \boldsymbol{edge1} \,.\, \boldsymbol{P_1}$$ *Eq 5-2: Implementing the transformation*

$$\boldsymbol{P'_1} = \begin{bmatrix} 0.75 & 0.5 & -0.433 & 0 \\ -0.433 & 0.866 & 0.25 & 0 \\ 0.5 & 0 & 0.866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.8169 \\ 0.683 \\ 1.366 \\ 1 \end{bmatrix}$$ Converted point

$$\boldsymbol{P_1} = \begin{bmatrix} 0.75 & -0.433 & 0.5 & 0 \\ 0.5 & 0.866 & 0 & 0 \\ -0.433 & 0.25 & 0.866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.8169 \\ 0.683 \\ 1.366 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$ Inverse way

The coordinate $\boldsymbol{P_2}$ is also assumed to be a detected object from sensor 2 and the calculations below show the final coordinate for that.

$$\boldsymbol{P_2} = \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix}$$

$$\boldsymbol{edge2} = \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.866 & 0 & -0.5 & 0 \\ 0 & 1 & 0 & 0 \\ 0.5 & 0 & 0.866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.866 & 0 & -0.5 & -0.73 \\ 0 & 1 & 0 & -2 \\ 0.5 & 0 & 0.866 & -2.73 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Calculation of inverse matrix:

$$\boldsymbol{edge2Inverse} = \begin{bmatrix} 0.866 & 0 & 0.5 & 2 \\ 0 & 1 & 0 & 2 \\ -0.5 & 0 & 0.866 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Implementation of the edge matrix to an assumed detected object from the car in coordinate $\boldsymbol{P_2}$. The following line also examined for inverse calculation. It means that by implementing the inverse matrix, the origin coordinate has to be obtained.

$$\boldsymbol{P'_2} = \begin{bmatrix} 0.866 & 0 & -0.5 & -0.73 \\ 0 & 1 & 0 & -2 \\ 0.5 & 0 & 0.866 & -2.73 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 3 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.366 \\ 1 \\ 1.366 \\ 1 \end{bmatrix}$$ Converted point

$$\boldsymbol{P_2} = \begin{bmatrix} 0.866 & 0 & 0.5 & 2 \\ 0 & 1 & 0 & 2 \\ -0.5 & 0 & 0.866 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.366 \\ 1 \\ 1.366 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \\ 3 \\ 1 \end{bmatrix}$$ Inverse way

6. Conclusion

In terms of quality, this work holds a great quality due to various testing in each step of progress and is able to perform the transformation between any nodes in the graph. It can be used in a broad area where the ADAS developer can locate the sensors anyplace on the car based on their design without worrying about data acquisition. All the data after passing through the Sensor Central is ready to use either to alert the driver or displaying on a graph. This work will be continued to develop due to its stability and scalability. Since the work was done based on mathematics calculation, the result is proven by solutions and analysis.

7. Future work

In broad perspective vehicle to vehicle communication could make a drastic improvement in transportation system. It basically aims to increase the driving comfort and safety for drivers. Various wireless technologies are currently developed to reach this goal. With sharing the information received from vehicles via a stationary system, a tremendous reduction in traffic jam, accidents and other unpredicted incidents is achieved. All the information obtained from vehicles; should be readable for each car individually in its own position. ADAS developers must use the coordinate conversion between their coordinate system and also the global coordinate system which leads to realize the obstacles and car's location. One global system such as UTM (Universal Transverse Mercator) coordinate system is employed to recognize the locations of the vehicles and obstacles, on the Earth independently. It is also for vehicles to give their locations on the surface of the earth in vertical position. The importance of the conversion between cars' location and detected obstacles arises here, since each vehicle needs to know the exact location of the obstacles base on their own position. Sensor central now works on a single vehicle and as the part of future work, it will be a coordinate convertor when sending data from vehicle to vehicle. Part of the future work would be to figure out a way to deal with the number of vehicles to support in the graph.

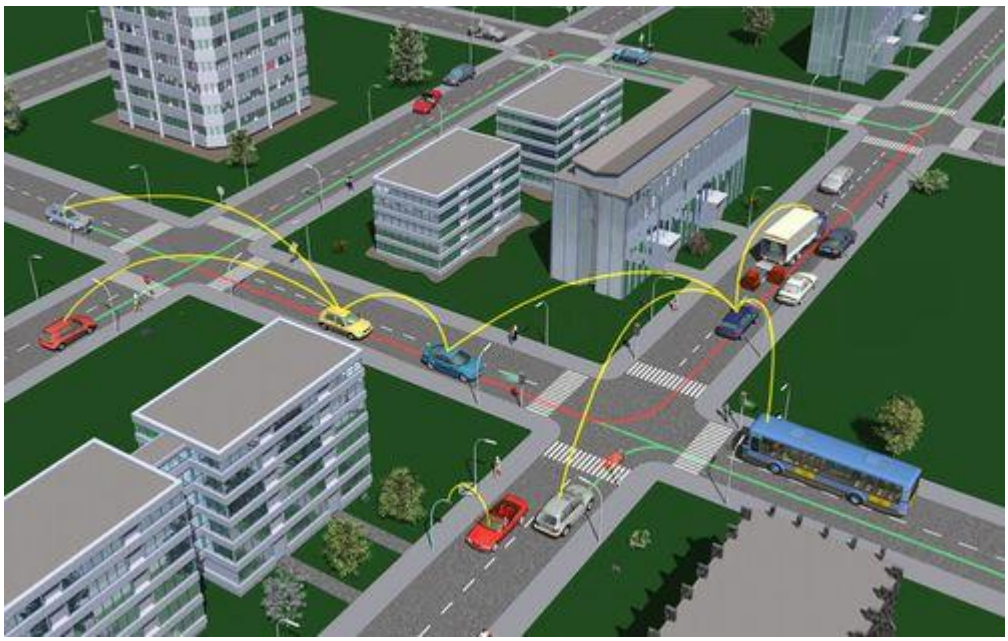

Figure 7-1: Vehicle to vehicle communications

8. Reference list

[1] http://developer.android.com/guide/topics/sensors/sensors_overview.html [Accessed on 12 July 2013]
[2] https://quickgraph.codeplex.com/documentation [Accessed on 20 July 2013]
[3] http://www.sophia.org/homogeneous-linear-systems-tutorial [Accessed on 18 July 2013]
[4] http://msdn.microsoft.com/en-us/library/windows/desktop/ms533864%28v=vs.85%29.aspx [Accessed on 12 July 2013]
[5] https://en.wikipedia.org/wiki/Transformation_matrix [Accessed on 25 July 2013]
[6] http://msdn.microsoft.com/en-us/library/ms379574(v=vs.80).aspx [Accessed on 12 July 2013]