# The Trolley Problem Version of Autonomous Vehicles

**Yair Wiseman [1,*] and Ilan Grinberg [1]**

[1] Computer Science Department, Bar-Ilan University, Ramat-Gan 52900, Israel; wiseman@cs.biu.ac.il
* Correspondence: wiseman@cs.biu.ac.il; Tel.: +972-3-531-7015

**Abstract:** The Trolley problem is very well-known ethics dilemma about actively killing one or sometimes even more persons in order to save more persons. The problem can occur in autonomous vehicles when the vehicle realizes that there is no way to prevent a collision, the computer of the vehicle should analyze which collision is considered to be the least harmful collision. In this paper we suggest a method to evaluate the likely harmfulness of each sort of collision using Spatial Data Structures and Bounding Volumes and accordingly to decide which course of actions would be the less harmful and therefore should be chosen by the autonomous vehicle.

**Keywords:** Autonomous Vehicles; Car Accidents; Embedded Real-Time Systems

---

## 1. Introduction

Ethics is not an exact science. Someone can consider one decision as moral; whereas other one can consider the very same decision as immoral. When it comes to autonomous vehicles, the ethics should be sometimes encoded into the software of the vehicle. The programmer should sometimes decide in questions like whose life is more precious, the occupants in the vehicle or pedestrians go nearby the vehicle?

Such questions have been weighed up over the years and specifically for autonomous vehicles in the recent years [1], but no clear decisions have been made. One of the most popular forms to present the question about the priority of human lives is the "Trolley Problem" [2]. There are some versions for the trolley problem but the common concept is that there is a need to prioritize the life of several persons.

In the "Trolley Problem" the situation is clear i.e. you should kill a certain person in order to save another certain person; however, the reality is not so clear [3,4]. Sometimes the software of the autonomous vehicle cannot be sure whether in a case of an accident, the occupants in the vehicle, nearby pedestrians or anyone else will die. E.g. when there is a rollover crash the occupants in the vehicle have a smaller chance to die relative to a pedestrian that the vehicle falls on him; however, there is still a small chance that an occupant in the vehicles will sustain a severer injures than a nearby pedestrian [5].

There was a claim that the autonomous vehicle makers prefer the live the lives of the occupants in the autonomous vehicle over the lives of the pedestrians in their surroundings, because the occupants are their customers and the vehicle makers want to please them and convince more potential customers that the vehicle is safe [6]; however, no evidence have been found for this hypothetical claim [7].

This paper suggest a way to give the most accurate data to the algorithm and according to this data, the software can decide whatever decision that the software developer believes is the most appropriate decision.

## 2. Methods

Polygons are simple shapes that can simulate the real objects. It is very common to generate models of real objects using simple polygons. This practice is usually called Spatial Data Structures [8].

When it comes to simulation of car accidents, Spatial Data Structures are implemented in order to find the intersections between two objects that are about to collide and realizing which polygons are about to be damaged. Obviously, trying to check all the polygons of each of the objects is pointless, because there are parts of objects that have no chance to collide with another object in some sorts of crashes. E.g. in a front accident, the rear part of the vehicle will not be damaged; therefore there are several methods to reduce the number of polygons intersections when using Spatial Data Structures [9].

Spatial Data Structures are the basis for Space Partitioning [10] and Bounding Volumes [11]. Space Partitioning is a method of space sub-partitioning into convex regions. These convex regions are named "cells". Each of these cells maintains a list of objects that it comprises of. By employing these cells, the intersection algorithm knows how to sift out pairs of polygons that have no chance to intersect.

The other method is Bounding Volume. This method breaks an object into small components; then the algorithm finds a fitted bounding volume for each of the small component. After that, the intersection algorithm checks for intersected components. It should be noted that the sifting out is less demanding in this method, because the algorithm just have to detect the at least partly cover bounding volumes.

Bounding Volumes applications have been intensely studied over the years and many variations of the method have been suggested: Bounding Spheres [70], K-DOPs - Discrete orientation polytopes [71], OBB - Oriented Bounding Boxes [72], AABB - Axis Aligned Bounding Boxes [73] and Hierarchical Spherical Distance Fields [74].

In this paper we have used the AABB approach which is one of the most well-known approaches. In AABB each of the bounding volume in the object model is represented by its minimum and its maximum values. Compared to the "Bounding Sphere" approach, AABB has an advantage and a disadvantage. AABB encompasses the components of the model more tightly which probably yield less intersection checks and also the split of the object into its bounding volumes is faster [75]. The algorithm first checks each of the basic elements that a bounding volume consists of and projects the element on the axes and then finds the minimum and the maximum values for each axis. The fast operation is very essential in real time systems like collisions of autonomous vehicle. The algorithm should be fast and decide promptly what course of actions should be taken.

However, AABB has also a disadvantage. Saving the data for AABB takes more memory space which in the past was very costly, but nowadays, memory space is much less costly and even simple computers have a plenty of memory space, so this disadvantage is not so acute; therefore, we have chosen the AABB approach.

Since our system is a real time system and the computation time is very essential we have decided to implement the AABB approach. We generated the bounding volume tree in a recursive manner. In each step, the algorithm generates bounding volumes for the remaining triangles and splits the triangle set into two sub-graphs. Then, it recursively calls itself to do the same for each of these sub-graphs.

### *2.1. Bounding Volume* hierarchies

Bounding volume hierarchies are actually a tree symbolizes a model of an object [76]. The basic components are the leaves of the tree and each sub-tree rooted by an internal node represents a segment of the model.

Such trees have only one leave for each basic component, so the size of the storage needed for each vehicle model is linear in the number of the basic components. This also impacts on the intersection check time which is therefore quite fast.

The construction time however is lengthy. This can be a significant disadvantage when the model is flexible and a reconstruction is frequently needed whenever the object changes its shape; however a vehicle is a rigid object and no changes are made, so the construction can be done only

once and the tree will be good for the entire life of the vehicle, so this disadvantage is irrelevant for the objective of this paper.

When the algorithm checks for collisions, it will begin to check the roots of the model trees and then in a recursive manner it will go down the trees to check whether an intersections between sub-trees or even leaves occurs.

The total time needed to check the severity of a collision between two model trees is given by this formula:

$$T_{total}=N_bC_b+N_pC_p$$

Where,

$T_{total}$ – Toal time for an intersection check between two model trees.

$N_b$ – Number of bounding volume pairs checked for one intersection.

$C_b$ – Time for one intersection check between a particular bounding volume pair.

$N_p$ – Number of primitive polygon pairs checked for one intersection.

$C_p$ –Time for one intersection check between a particular primitive polygon pair.

Methods with a tight-fitting bounding volume like OBB will yield low $N_b$ and $N_p$; however $C_b$ will be in these methods much higher. On the contrary, AABB will yield high $N_b$ and $N_p$; whereas $C_b$ will be lower.

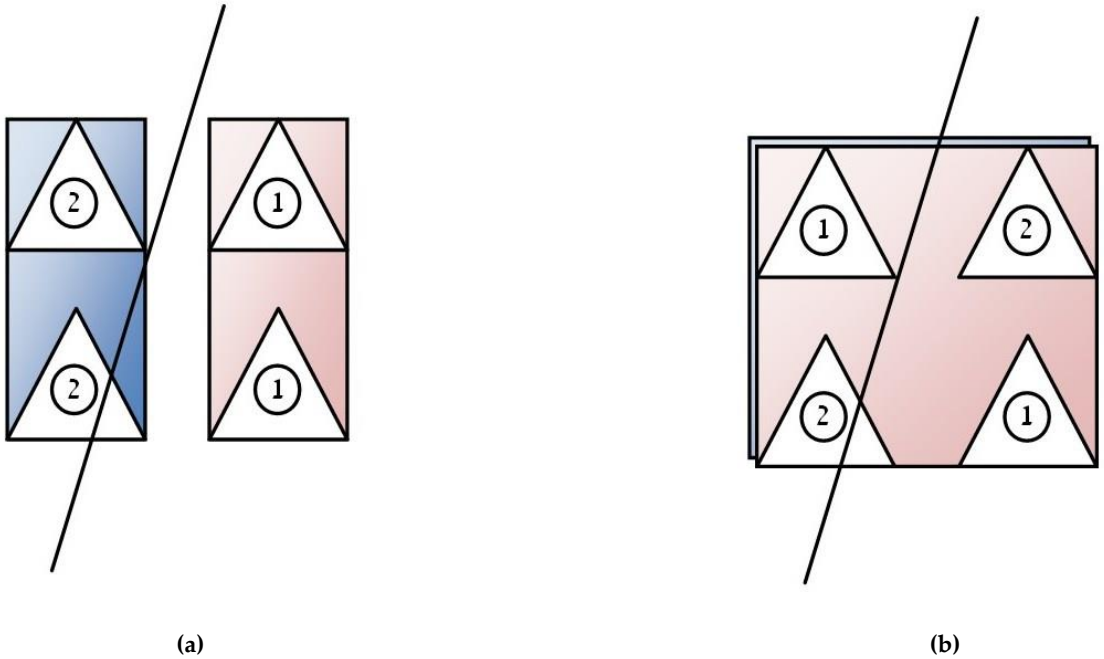



**Figure 1.** Triangle split (**a**) Example of split that causes 2 checks; (**b**) Example of split that causes 4 checks.

*2.2. Implemntation*

As was explained above, the implementation of the method was in a recursive manner. We used triangles as it is a very common in such implementations [77].

Initially, the algorithm finds a bounding volume for the remaining triangles. Then, the algorithm splits the set of triangles into two sub-graphs. Finally, the algorithm calls itself in a recursive manner to handle the two new sub-graphs that have been generated in the previous step.

When a sub-graph has just one triangle, the recursive call will stop and the algorithm will not call itself any longer

The incentive for the triangles split into several sub-graphs is the formation of as small as possible bounding volumes so the model will as accurate as possible.

Figure 1 explains by an example how four triangles can be split in two different ways. The number written in the triangles denotes which sub-graph contains the triangle after the suggested split. This figure clearly shows that a hierarchical intersection checking with a specific segment can generate more triangle intersection checks in the right side figure because the generated bounding volume is bigger. This attribute motivated us to implement a split algorithm that uses better splits as in the left side, so as to minimize the triangle intersection checks.

Actually, the bounding volume algorithms and the triangle split algorithms greatly shape the bounding volume tree generation algorithm and its efficiency. We have employed "Fitting points with Gaussian distribution" [78] to as the basis for the algorithm for generating the bounding volumes.

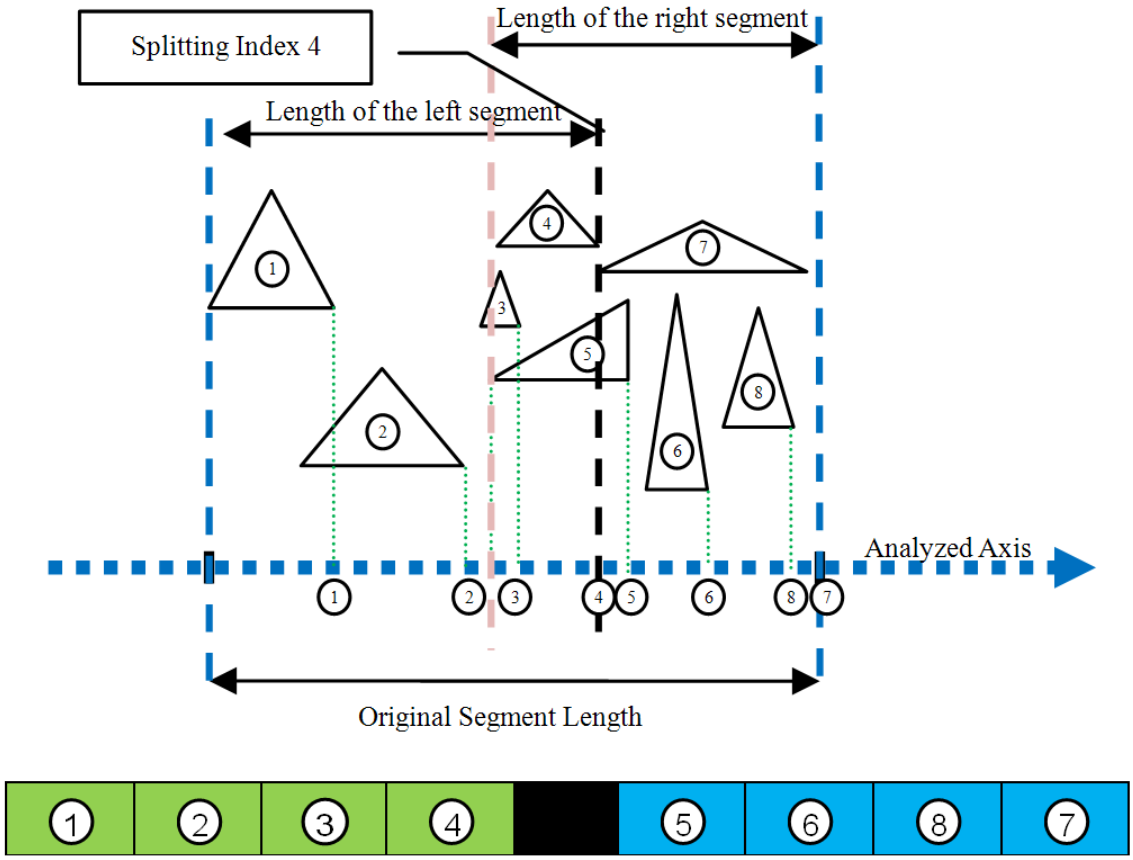



**Figure 2.** Example of triangles' split on the projecting axis.

*2.3. Triangle Split Algorithm*

Each graph of triangles has a corresponding bounding volume that can be split into two sub-graphs. The pseudo code of this split algorithm is:

- For each axis of any volume select a positive direction.
- For each triangle find a vertex with the highest value on the projected axis.
- Sort the triangles vector by their vertex values.

- For each triangle in the triangles vector compute the sum of the projection lengths on the axis divided by the original graph projection lengths of the two sub-graphs and split this triangle in order to get a smaller sub-graphs.

The algorithm strives to do its best in order to generate the least possible intersecting sub-graphs.

Figure 2 and Figure 3 illustrate two different potential index split. Figure 2 shows one potential split at triangle index 4. If indeed the algorithm chooses to split at this index, the split will ineffectually generate a larger intersection between the two new graphs. Figure 3 shows a better option which is a split in triangle index 2 which generates a smaller intersection between the two new graphs. This example shows a case when the algorithm would select a better split over an inferior split. In this particular example it was a split at triangle index 2 rather than a split at triangle index 4.

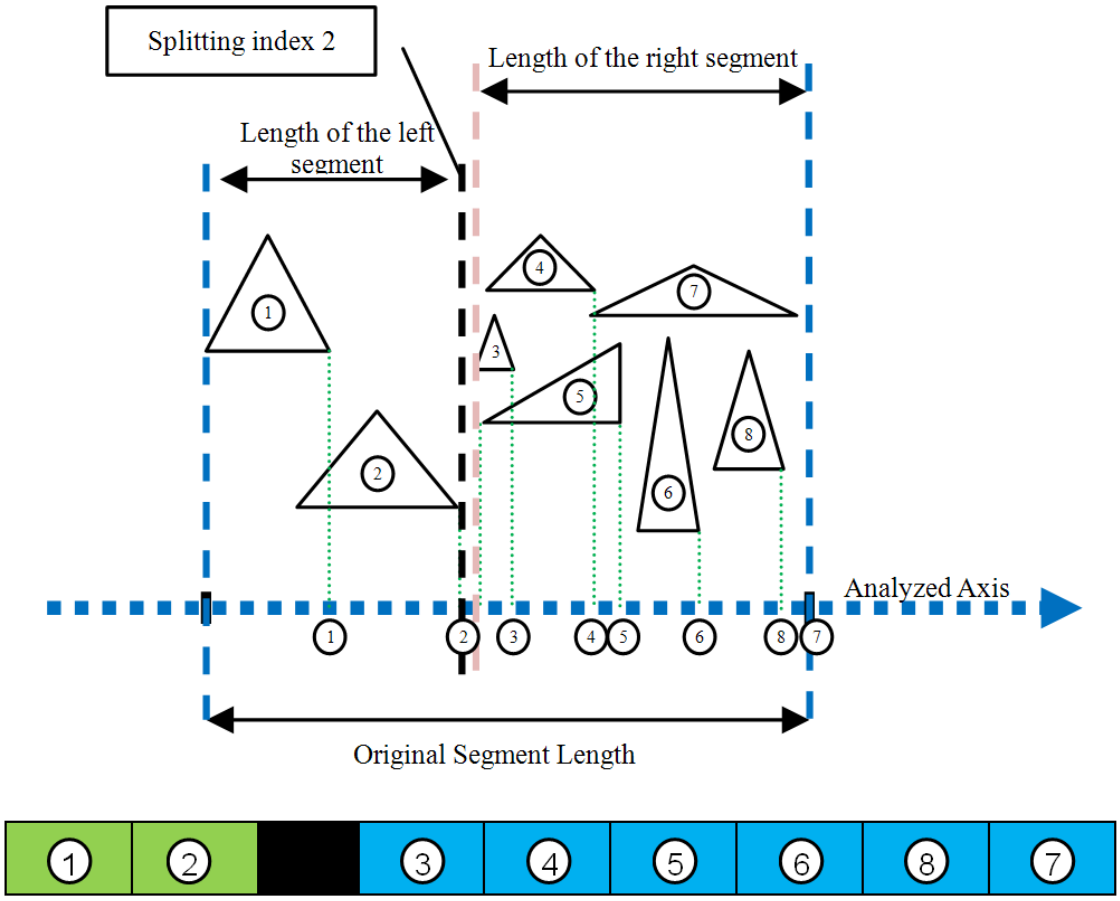



**Figure 3.** Yet another example of triangles' split on the projecting axis.

**4. Results**

We evaluated several schemes for parallel implementation of the algorithm:

- Best Match – Select a core with the maximum number of correlated model sections.
- Random Match – Select a random core.
- Lowest Match - Select a core with the minimum number of correlated model sections.
- Best Match-Load – This is a combination of the Best Match scheme and a Load parameter that calculated for each core. The algorithm prefers less loaded cores over more loaded cores. The load on any core is computed by dividing the buffered checks waiting for this core by the maximal size of this core's buffer. The algorithm takes into account both the load in the core and the correlation between the model sections that the core checks and the new arrived model section, so as to get the best decision.

**Figure 4.** Vehicle with a substantial damage.

The suggested method has been implemented on a four cores processor. We endeavored to evaluate the efficiency of the suggested triangle scheme. The particular processor was Intel® Pentium® Processor N3540 which is a very widespread quad-core processor with 2.16GZ; however, another process would have yield similar results.

We checked two kinds of damages. In Figure 4 and Figure 5 we can see a vehicle with a substantial damage. Figure 4 shows the vehicle before it has been triangulated and Figure 5 shows the same vehicle after before it has been triangulated. Similarly, Figure 6 and Figure 7 show a vehicle before it has been triangulated and after before it has been triangulated. However, Figure 6 and Figure 7 show a vehicle with only scratches.

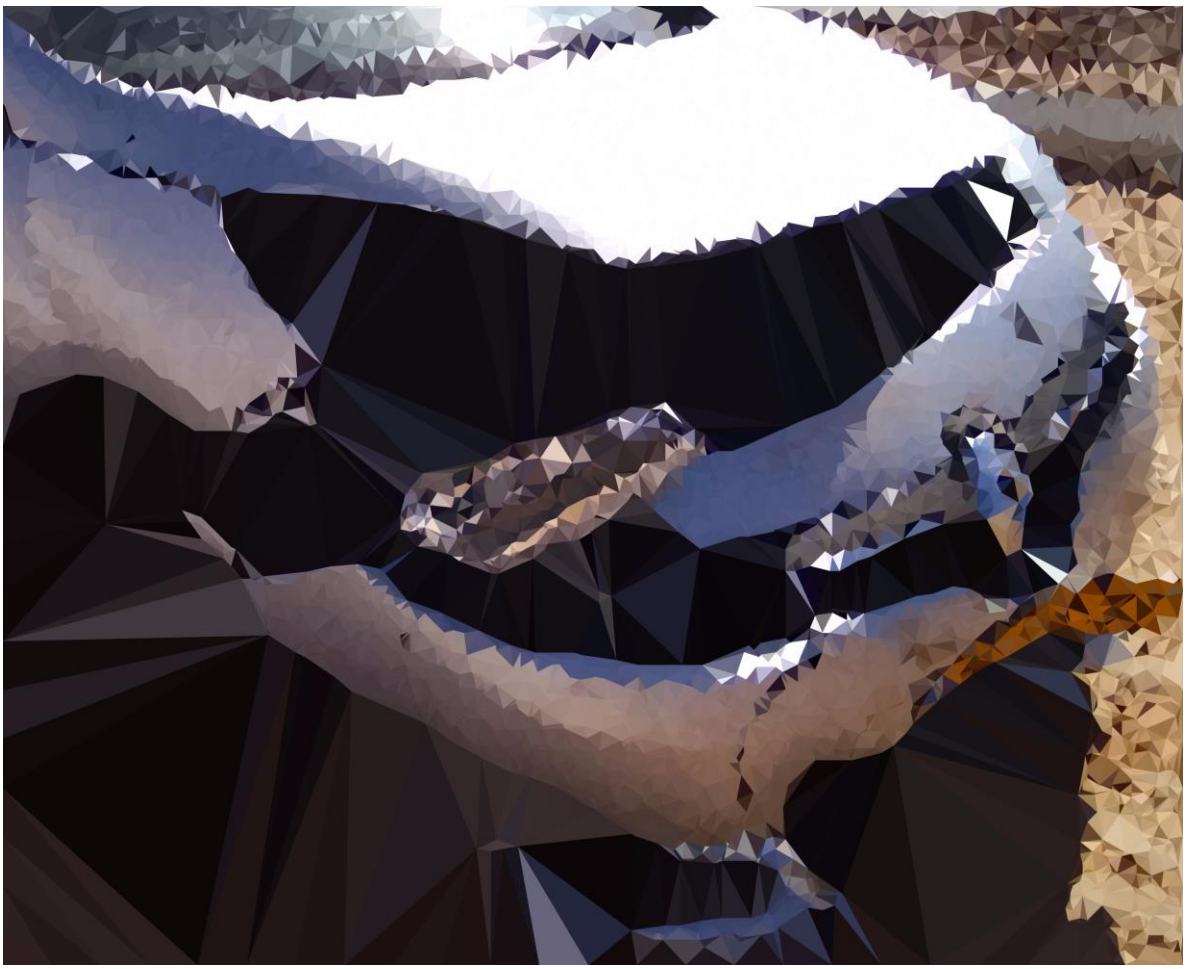


**Figure 5.** Triangulated version of the vehicle from Figure 4.

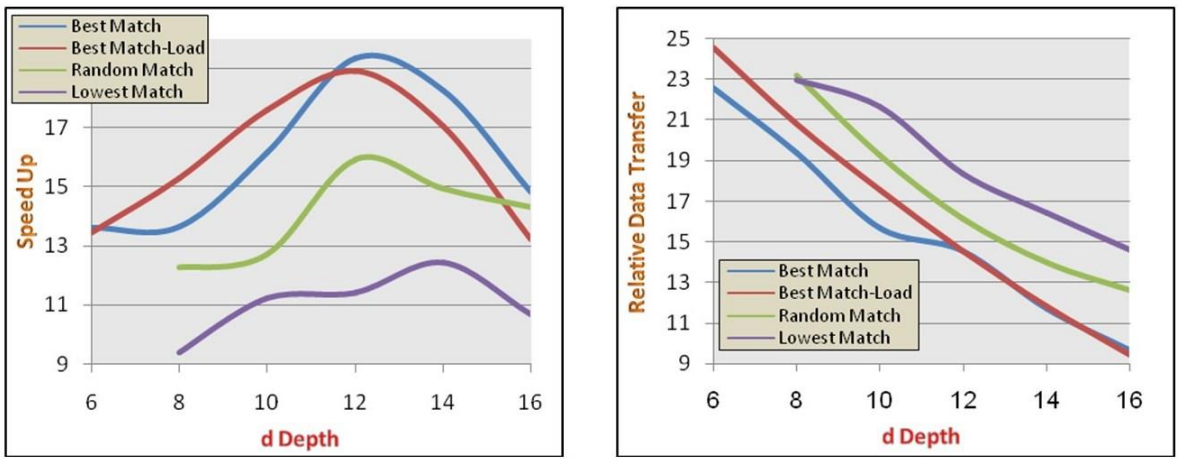
**Figure 6.** Vehicle with small scratches.

**Figure 7.** Triangulated version of the vehicle from Figure 6.

Selecting the distribution depth of the bounding volumes is also very important and should be carefully selected. Small depths result in large overlapping bounding volumes, which will generate a too many node collision pairs. However, large depths result in a long analyze of the first step by the main process while the other cores are idle. This is an unnecessary bottleneck that the algorithm should prevent.

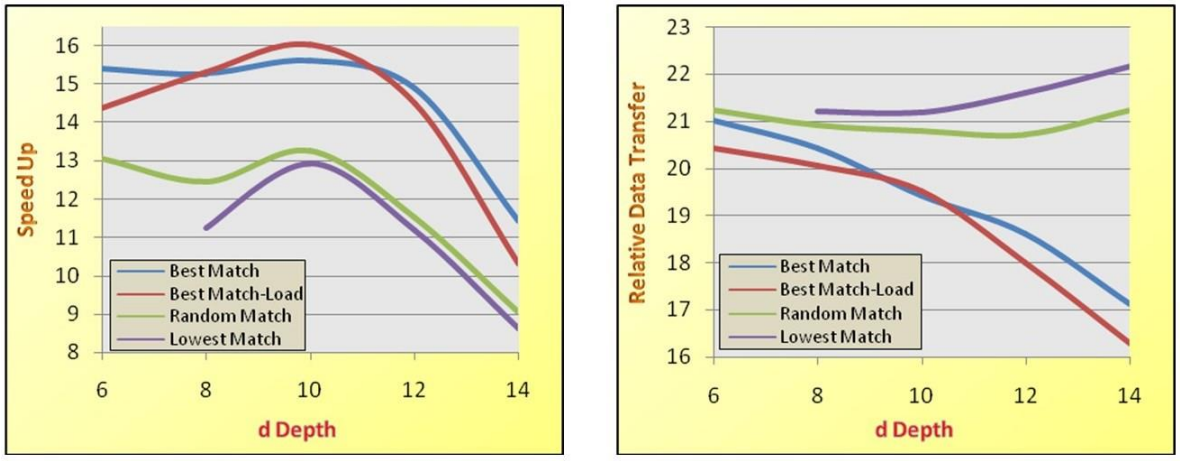
We have checked the effect of several distribution depths on the algorithm performance. We have used the images of the vehicle in Figure 5 and the vehicle in Figure 7. The findings can be seen in Figure 8 and Figure 9. It can be noticeably observed that the Best Match algorithm yields better performance, both in relative data transfer and speedup.

The best possible distribution depth can be different in different cases. An example for this difference can be seen in Figure 8 and Figure 9. The vehicle in Figure 8 has an optimal distribution depth of 12; whereas the vehicle in Figure 9 has an optimal distribution depth of 10. Other vehicles can have other optimal distribution depths.




**(a)** **(b)**

**Figure 8.** Distribution depth effect in the vehicle of Figure 5 on (a) speedup (b) relative data transfer




**(a)** **(b)**

**Figure 9.** Distribution depth effect in the vehicle of Figure 7 on (a) speedup (b) relative data transfer

## 5. Discussion

The results clearly show that different distribution depths should be chosen for different vehicle models.

Moreover, we can also figure out from the results that if the vehicle model is larger, the Best Match algorithm will have a greater benefit over the other algorithms.

The performance of the algorithms is considerably affected by the buffer size allocated for each core. By using a buffer, a number of jobs can be gathered and sent to another core. So as a result the main process is less busy and therefore it will able to prepare more tasks intended for some other cores.

If we choose an algorithm that consumes a plenty of memory like Lowest Match or Random Match, the processor will swiftly run out of memory and therefore will be incapable to accomplish the task. The processor will have to bring into play virtual memory mechanism that can turn out even to Thrashing [79].

Such a situation can be seen in the last Figures where there are trimmed lines. It happens in algorithms that inefficiently allocate memory and when a low distribution depth is select. The lines are trimmed because the algorithm has been gone into a thrashing situation and therefore no data could be obtained.

## 6. Conclusions

There is no confident answer to the "Trolley Problem" and different people gave different answers to this problem. Our paper also did not aim at realizing the "correct answer" for the "Trolley Problem" and it was out of scope of this paper.

The paper has suggested a real-time estimation system for vehicle crash probable damages. The aim of this system is to facilitate autonomous vehicle's embedded computer to decide which crash is the least harmful when it comprehends that a crash is unavoidable by analyzing all the damaged for each of the possible crashes.

Actually, this paper has suggested how to adapt the well-known Primitive Intersection scheme into an estimation tool of potential vehicle crash damages so as to decide which course of actions should be taken by the vehicle in an inescapable crash.

## References

1. N. J. Goodall, 'Can you program ethics into a self-driving car?" IEEE Spectrum, Vol. 53, No. 6, pp. 28–31, 2016.

2. J. J. Thomson, "Killing, letting die, and the trolley problem", Ethical Theory - An Anthology, Second Edition, John Wiley & Sons, Inc. Publication, pp. 204-217, 1976.
3. N. J. Goodall, "From Trolleys to Risk: Models for Ethical Autonomous Driving", American Journal of Public Health (AJPH), Vol. 107, No. 4, p. 496, 2017.
4. N. J. Goodall NJ, "Away from trolley problems and toward risk management", Applied Artificial Intelligence Vol. 30, No. 8, pp. 810–821, 2016.
5. A. M. Eigen, "Rollover Crash Mechanisms and Injury Outcomes for Restrained Occupants", NHTSA Technical Report, U.S. Department of Transportation, Washington, DC, 2005.
6. E. Aharonson, "Actuality in 'Halacha' – Autonomous Cars", available online at: https://www.youtube.com/watch?v=gmgUJDVDreg , 2017.
7. Y. Wiseman and Y. Giat, "Multi-modal passenger security in Israel Multimodal Security in Passenger and Freight Transportation: Frameworks and Policy Applications, Edward Elgar Publishing Limited, Chapter 16, pp. 246-260, 2016.
8. K. Keul, P. Lemke and S. Müller, "Accelerating spatial data structures in ray tracing through precomputed line space visibility", In Proceedings of the 24th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, pp. 17-26, 2016.
9. I. Grinberg and Y. Wiseman, "Scalable Parallel Simulator for Vehicular Collision Detection", International Journal of Vehicle Systems Modelling and Testing, Vol. 8(2), pp. 119-144, 2013.
10. R. Vuillemot, and J. Boy, "Structuring Visualization Mock-ups at the Graphical Level by Dividing the Display Space", To appear in IEEE Transactions on Visualization and Computer Graphics, 2017. Available at: http://romain.vuillemot.net/publis/infovis17-visualization-mockups.pdf
11. J. S. Park, C. Park and D. Manocha, "Efficient probabilistic collision detection for non-convex shapes" In proceedings of IEEE International Conference on Robotics and Automation (ICRA), pp. 1944-1951, Singapore, 2017.
12. R. B. Yehezkael, Y. Wiseman, H. G. Mendelbaum and I. L. Gordin, "Experiments in Separating Computational Algorithm from Program Distribution and Communication", LNCS of Springer Verlag Vol. 1947, pp. 268-278, 2001.
13. Y. Wiseman, "A Pipeline Chip for Quasi Arithmetic Coding", IEICE Journal - Trans. Fundamentals, Tokyo, Japan, Vol. E84-A(4), pp. 1034-1041, 2001.
14. E. Fredj and Y. Wiseman, "An O(n) Algorithm for Edge Detection in Photos Compressed by JPEG Format", Proc. IASTED International Conference on Signal and Image Processing SIP-2001, Honolulu, Hawaii, pp. 304-308, 2001.
15. Y. Wiseman and E. Fredj, "Contour Extraction of Compressed JPEG Images", ACM - Journal of Graphic Tools, Vol. 6(3), pp. 37-43, 2001.
16. Y. Wiseman, K. Schwan and P. Widener, "Efficient End to End Data Exchange Using Configurable Compression", Proc. The 24th IEEE Conference on Distributed Computing Systems (ICDCS 2004), Tokyo, Japan, pp. 228-235, 2004.
17. Y. Wiseman, "ARC Based SuperPaging", Operating Systems Review, Vol. 39(2), pp. 74-78, 2005.
18. Y. Wiseman, "Advanced Non-Distributed Operating Systems Course", ACM - Computer Science Education, Vol. 37(2), pp. 65-69, 2005.
19. M. Reuven and Y. Wiseman, "Reducing the Thrashing Effect Using Bin Packing", Proc. IASTED Modeling, Simulation, and Optimization Conference, MSO-2005, Oranjestad, Aruba, pp. 5-10, 2005.
20. Y. Wiseman, "The Relative Efficiency of LZW and LZSS", Data Science Journal, Vol. 6, pp. 1-6, 2007.
21. Y. Wiseman and I. Gefner, "Conjugation Based Compression for Hebrew Texts", ACM Transactions on Asian Language Information Processing, Vol .6(1), article no. 4, 2007.
22. Y. Wiseman, "Burrows-Wheeler Based JPEG", Data Science Journal, Vol. 6, pp. 19-27, 2007.
23. I. Grinberg and Y. Wiseman, "Scalable Parallel Collision Detection Simulation", Proc. Signal and Image Processing (SIP-2007), Honolulu, Hawaii, pp. 380-385, 2007.
24. Y. Wiseman, "ASOSI: Asymmetric Operating System Infrastructure ", Proc. 21st Conference on Parallel and Distributed Computing and Communication Systems, (PDCCS 2008), New Orleans, Louisiana, pp. 193-198, 2008.
25. Y. Wiseman, J. Isaacson and E. Lubovsky, "Eliminating the Threat of Kernel Stack Overflows", Proc. IEEE Conference on Information Reuse and Integration (IEEE IRI-2008), Las Vegas, Nevada, pp. 116-121, 2008.

26. M. Itshak and Y. Wiseman, "AMSQM: Adaptive Multiple SuperPage Queue Management", Proc. IEEE Conference on Information Reuse and Integration (IEEE IRI-2008), Las Vegas, Nevada, pp. 52-57, 2008.
27. P. Weisberg and Y. Wiseman, "Using 4KB Page Size for Virtual Memory is Obsolete", Proc. IEEE Conference on Information Reuse and Integration (IEEE IRI-2009), Las Vegas, Nevada, pp. 262-265, 2009.
28. I. Grinberg and Y. Wiseman, "Scalable Parallel Simulator for Vehicular Collision Detection", Proc. IEEE Conference on Vehicular Electronics and Safety (IEEE ICVES-2010) Qingdao, ShanDong, China, pp. 116-121, 2010.
29. Y. Wiseman, "Take a Picture of Your Tire!", Proc. IEEE Conference on Vehicular Electronics and Safety (IEEE ICVES-2010) Qingdao, ShanDong, China, pp. 151-156, 2010.
30. R. Ben Yehuda and Y. Wiseman, "The Offline Scheduler for Embedded Transportation Systems", Proc. IEEE Conference on Industrial Electronics (IEEE ICIT-2011), Auburn, Alabama, pp. 449-454, 2011.
31. D. Livshits and Y. Wiseman, "Cache Based Dynamic Memory Management For GPS", Proc. IEEE Conference on Industrial Electronics (IEEE ICIT-2011), Auburn, Alabama, pp. 441-446, 2011.
32. Y. Wiseman, "The Effectiveness of JPEG Images Produced By a Standard Digital Camera to Detect Damaged Tyres", World Review of Intermodal Transportation Research, Vol. 4(1), pp. 23-36, 2013.
33. Y. Wiseman and P. Weisberg, "Economical Memory Management for Avionics Systems", IEEE/AIAA 31st Digital Avionics Systems Conference (DASC), 2013.
34. Y. Wiseman and A. Barkai, "Diminishing Flight Data Recorder Size", IEEE/AIAA 31st Digital Avionics Systems Conference (DASC), 2013.
35. R. Ben Yehuda and Y. Wiseman, "The Offline Scheduler for Embedded Vehicular Systems", International Journal of Vehicle Information and Communication Systems, Vol. 3(1), pp. 44-57, 2013.
36. D. Livshits and Y. Wiseman, "The Next Generation GPS Memory Management", International Journal of Vehicle Information and Communication Systems, Vol. 3(1), pp. 58-70, 2013.
37. Y. Wiseman, "Fuselage Damage Locator System", Advanced Science and Technology Letters, Vol. 37, pp. 1-4, 2013.
38. Y. Wiseman and A. Barkai, "Smaller Flight Data Recorders", Journal of Aviation Technology and Engineering, Vol. 2(2), pp. 45-55, 2013.
39. Y. Wiseman, "Camera That Takes Pictures of Aircraft and Ground Vehicle Tires Can Save Lives", Journal of Electronic Imaging, Vol. 22(4), 041104, 2013.
40. P. Weisberg and Y. Wiseman, "Efficient Memory Control for Avionics and Embedded Systems", International Journal of Embedded Systems, Vol. 5(4), pp. 225-238, 2013.
41. Y. Wiseman, "Steganography Based Seaport Security Communication System", Advanced Science and Technology Letters, Vol. 46, pp. 302-306, 2014.
42. Y. Wiseman, "Device for Detection of Fuselage Defective Parts", Information Journal, Tokyo, Japan, Vol. 17(9(A)), pp. 4189-4194, 2014.
43. Y. Wiseman, "Noise Abatement at Ben-Gurion International Airport", Advanced Science and Technology Letters, Vol. 67, pp. 84-87, 2014.
44. Y. Wiseman, "The still image lossy compression standard – JPEG", Encyclopedia of Information and Science Technology, Third Edition, Vol. 1, Chapter 28, pp. 295-305, 2014.
45. Y. Wiseman, "Protecting Seaport Communication System by Steganography Based Procedures", International Journal of Security and Its Applications, Sandy Bay, Tasmania, Australia, Vol. 8(4), pp. 25-36, 2014.
46. Y. Wiseman, "Noise Abatement Solutions for Ben-Gurion International Airport", International Journal of U- & E-Service, Science & Technology, Vol. 7(6), pp. 265-272, 2014.
47. P. Weisberg & Y. Wiseman, "Virtual Memory Systems Should Use Larger Pages rather than the Traditional 4KB Pages", International Journal of Hybrid Information Technology, Vol. 8(8), pp. 57-68, 2015.
48. Y. Wiseman, "Improved JPEG based GPS picture compression", Advanced Science and Technology Letters, Vol. 85, pp. 59-63, 2015.
49. Y. Wiseman, "Diminution of JPEG Error Effects', The Seventh International Conference on Future Generation Information Technology, Vol. 117, pp. 6-9, 2015.

50. Y. Wiseman, "Enhancement of JPEG Compression for GPS Images", International Journal of Multimedia and Ubiquitous Engineering, Vol. 10(7), pp. 255-264, 2015.
51. P. Weisberg and Y. Wiseman, "Virtual Memory Systems Should use Larger Pages", Advanced Science and Technology Letters, Vol. 106, pp. 1-4, 2015.
52. Y. Wiseman and Y. Giat, "Red Sea and Mediterranean Sea Land Bridge via Eilat", World Review of Intermodal Transportation Research, Vol. 5(4), pp. 353-368, 2015.
53. Y. Wiseman, "Alleviation of JPEG Inaccuracy Appearance", International Journal of Multimedia and Ubiquitous Engineering, Vol. 11(3), pp. 133-142, 2016.
54. Y. Wiseman, "Can Flight Data Recorder Memory Be Stored on the Cloud?", Journal of Aviation Technology and Engineering, Vol. 6(1), 16-24, 2016.
55. Y. Wiseman and I. Grinberg, "Autonomous Vehicles Should Not Collide Carelessly", Advanced Science and Technology Letters, Vol. 133, pp. 223-228, 2016.
56. Y. Wiseman, "Traffic Light with Inductive Detector Loops and Diverse Time Periods", Contemporary Research Trend of IT Convergence Technology, Vol. 4, pp. 166-170, 2016.
57. Y. Wiseman, "Unlimited and Protected Memory for Flight Data Recorders", Aircraft Engineering and Aerospace Technology, Vol. 88(6), pp. 866-872, 2016.
58. Y. Wiseman and I. Grinberg, "When an Inescapable Accident of Autonomous Vehicles is Looming", International Journal of Control and Automation, Vol. 9(6), pp. 297-308, 2016.
59. Y. Wiseman, "Conceptual Design of Intelligent Traffic Light Controller", International Journal of Control and Automation, Vol. 9(7), pp. 251-262, 2016.
60. Yair Wiseman, "Compression Scheme for RFID Equipment", Proc. IEEE International Conference on Electro Information Technology (EIT 2016), Grand Forks, North Dakota, USA, pp. 387-392, 2016.
61. Y. Wiseman and I. Grinberg, "Circumspectly Crash of Autonomous Vehicles", Proc. IEEE International Conference on Electro Information Technology (EIT 2016), Grand Forks, North Dakota, USA, pp. 382-386, 2016.
62. Y. Wiseman, "Efficient RFID Devices", Proc. The 42nd Annual Conference of IEEE Industrial Electronics Society (IECON 2016), Firenze (Florence), Italy, pp. 4762-4766, 2016.
63. Y. Wiseman, "Computerized Traffic Congestion Detection System", International Journal of Transportation and Logistics Management, Vol. 1(1), pp. 1-8, 2017.
64. Y. Wiseman, "Remote Parking for Autonomous Vehicles", International Journal of Hybrid Information Technology, Vol. 10(1), pp. 313-324, 2017.
65. Y. Wiseman, "Tool for Online Observing of Traffic Congestions", International Journal of Control and Automation, Vol. 10(6), pp. 27-34, 2017.
66. Y. Wiseman, "Safety Mechanism for SkyTran Tracks", International Journal of Control and Automation, Vol. 10(7), pp. 51-60, 2017.
67. Y. Wiseman, "Self-Driving Car - A Computer will Park for You", International Journal of Engineering & Technology for Automobile Security, Vol. 1(1), pp. 9-16, 2017.
68. Y. Wiseman, "Real-Time Monitoring of Traffic Congestions", IEEE International Conference on Electro Information Technology (EIT 2017), Lincoln, Nebraska, USA, pp. 501-505, 2017.
69. Y. Wiseman, "Automatic Alert System for Worn Out Pipes in Autonomous Vehicles", International Journal of Advanced Science and Technology, Vol. 107, 2017.
70. T. Larsson, "Exact bounding spheres by iterative octant scan", In Proceedings of SIGRAD 2015, Stockholm, Sweden, pp. 9-12, 2015.
71. Y. Wang, Z. X. Luo, J. C. Liu, X. Fan, H. Li, and Y. Wu, "Real-time estimation of hand gestures based on manifold learning from monocular videos", Multimedia tools and applications, Vol. 71, No. 2, pp. 555-574, 2014.
72. Q. Fu, X. Chen, X. Su, J. Li and H. Fu, "Structure-adaptive Shape Editing for Man-made Objects", In Computer Graphics Forum, Vol. 35, No. 2, pp. 27-36, 2016.
73. P. Cai, C. Indhumathi, Y. Cai, J. Zheng, Y. Gong, T. S. Lim and P. Wong, "Collision detection using axis aligned bounding boxes", In Simulations, Serious Games and Their Applications, pp. 1-14. Springer Singapore, 2014.
74. R. Weller, "New geometric data structures for collision detection and haptics", Springer Science & Business Media, 2013.

75. U. Schwesinger, R. Siegwart, and P. Furgale, "Fast collision detection through bounding volume hierarchies in workspace-time space for sampling-based motion planners", In proceedings of IEEE International Conference on Robotics and Automation (ICRA), pp. 63-68, 2015.
76. D. Meister and J. Bittner, "Parallel Locally-Ordered Clustering for Bounding Volume Hierarchy Construction", To appear in IEEE Transactions on Visualization and Computer Graphics, IEEE Xplore Digital Library, 2017.
77. K. J. Mei, and R. S. Lee, "Collision detection for virtual machine tools and virtual robot arms using the Shared Triangles Extended Octrees method", International Journal of Computer Integrated Manufacturing, Vol. 29, No. 4, pp. 355-373, 2016.
78. L. Maarten, and J. M. Phillips, "Shape fitting on point sets with probability distributions" In Algorithms-ESA 2009, pp. 313-324. Springer Berlin Heidelberg, 2009.
79. M. Reuven and Y. Wiseman, "Medium-Term Scheduler as a Solution for the Thrashing Effect", The Computer Journal, Oxford University Press, Swindon, UK, Vol. 49, No. 3, pp. 297-309, 2006.