D3.1

Metrics and plan for V&V of the concepts and algorithms in the 1st cycle

Project Number:	690705
Classification	Public
Deliverable No.:	D3.1
Work Package(s):	WP3
Document Version:	Vs. 1.0
Issue Date:	31.01.2017
Document Timescale:	Project Start Date: September 1, 2016
Start of the Document:	Month 3
Final version due:	Month 5
Deliverable Overview:	**Main document:** D3.1
Compiled by:	David Käthner
Authors:	David Käthner Paulin Pekezou Mark Eilers Mohamed-Cherif Rahal Sébastien Glaser Stefan Suck
Technical Approval:	Fabio Tango, CRF
Issue Authorisation:	Andreas Lüdtke, OFF

DISTRIBUTION LIST		
Copy type[1]	Company and Location	Recipient
T	AutoMate Consortium	all AutoMate Partners

[1] Copy types: E=Email, C=Controlled copy (paper), D=electronic copy on Disk or other medium, T=Team site (Sharepoint)

RECORD OF REVISION		
Date	Status Description	Author
11.11.2016	Deliverable structure	David Käthner (DLR)
18.01.2016	Section 5 provided	Mark Eilers (OFF)
18.01.2017	Section 6 provided	Stefan Suck (HMT)
18.01.2017	Bullet points rewritten into text	David Käthner (DLR)
19.01.2017	Added Mohamed's part	Mohamed RAHAL (VED)/ David
19.01.2017	added part 2; fixed layout	David
23.01.2017	extended section 5	Mohamed Rahal (VED)
23.01.2017	changed and extended section 6	Stefan Suck (HMT)
25.01.2017	review	Fabio Tango (CRF)
26.01.2017	further notes	Paulin Pekezou (DLR)
30.01.2017	last changes	Mohamed Rahal (VED)

Table of Contents

1 Introduction

The TeamMate car views driver and automation as members on one team that understand and support each other in their collective goal of safe and comfortable driving. To realize this concept, the vehicle must possess certain capabilities. Among those, one is to navigate through traffic on its own. This requires functions to judge risks connected to certain manoeuvres as well as planning and following concrete trajectories on the road. Further, learning from the human driver to negotiate traffic situations safely and comfortably is another integral part of the TeamMate concept. The vehicle therefore needs an appropriate functionality to do so.

It is the task of WP3 to design and implement functionalities which allow the TeamMate car to show the desired behaviour. The work package goal is therefore to design and evaluate functions (i.e. collections of implemented algorithms) which address the above mentioned issues. Specifically this pertains to software that realizes vehicle functions regarding adaptive and safe driving strategies. This will be done for the following aspects: 1) online risk assessment, 2) algorithms for trajectory planning and execution, and 3) algorithms to do online and offline learning of the behaviour of a human driver.

This deliverable lays out the plan on the verification and validation of this software within WP3. Our goal is to show the basic approach to ensure the software quality of this work package’s developments. The approach to verification and validation will be refined and adapted in parallel to the continuous software development, especially after cycle 1 and cycle 2.

As a caveat, the specific algorithms to be developed largely have yet to be determined. We therefore keep an open approach towards verification and validation. Having gained a deeper understanding of the use cases and situations to be addressed, the most useful approach will be determined once the concrete algorithms have been selected.

2 General approach to the verification and validation of techniques and software requirements

For the purposes of *WP3*, verification and validation should be understood both from a modelling and a software engineering perspective.

Following the American Institute of Aeronautics and Astronautics (AIAA), verification and validation of models should be understood as follows (Oberkampf and Barone, 2006):

- *Verification* is defined as the process of determining that a model implementation accurately represents the developer's conceptual description of the model and the solution to the model.
- *Validation* is defined as the process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended use.

In contrast, the view of software engineering can be summarized as follows:

- *Verification* is concerned with whether the system under development is well-engineered, error-free, etc.: *Are we building the system right?*
- *Validation* is concerned with whether the system under development will meet the posed requirements: *Are we building the right system?*

To unite these perspectives, a remain open for requirements which may emerge from algorithmic choices in the future, a *functional approach* appears to be optimal.

Both from the modelling point-of-view and from the engineering perspective, algorithms can be seen as functions. A given function takes as input certain arguments, such as the situation or a criticality metric. It then computes the desired output. In the case of the TeamMate car, this can be a trajectory, a driver intention prediction, or a safety assessment of a certain situation.

In the beginning, these functions will be rather broad and high level, but over the course of the project they will be narrowed and refined. Since functions must serve a specific goal, such as solving a problem, they are use case specific and depend on technical requirements of the demonstrators on which they should run eventually.

When defining the functions, three properties must be addressed:

- *Verification*: The function must always return an output for the given input, or else indicate an error. This can be achieved by unit tests, and tests at the implementation level.
- *Efficiency*: the time until an output is produced. While this can be seen as part of the Verification procedure, it is a safety critical factor when driving in real traffic and thus merits special attention. For driving in real traffic, it is usually not sufficient to state whether a computation will produce the desired output. It must be evaluated as well if this computation executes in the required time, or if it is *real time capable*[2]. This can be addressed with formal approaches, such as complexity

[2] In accordance with the use of *real time* in computer science, we define it as the guarantee for a program to respond to an input within a certain time frame.

calculations. On the implementation level, extreme test cases can be defined, against which the implemented algorithms are evaluated.

- *Validation*: the usefulness of an output provided by the function. Test sets of time streams can be used, such as data sets from real or from simulated traffic situations.

The degree to which these three properties have been addressed can be expressed by *metrics*. Note that the term *metric* refers to the definition from measurement theory: a numerical representation of an empirical matter that fulfils certain properties.

However, it is important to note that the process of verification, validation and the evaluation of efficiency should not exclusively be judged based on metrics. While metrics are an important tool of judging the success or failure of verification and validation, they are only a subset of the possible outcomes.

Our own professional experience shows that a qualitative expert judgement sometimes is the more appropriate outcome of verification and validation procedures. Specifically for purely functional requirements, a "yes" or "no" (whether or not the function fulfils its role) can be the desired evaluation.

To assess if a function is valid, i.e. if it produces the desired outcome or not, a number of approaches will be applied.

First, we can judge whether or not *global objectives* have been reached. Relating to the behaviour of an automated vehicle, such global objectives can be:

- successfully arriving at the desired destination

- collision avoidance
- avoidance of any critical situations
- economic driving
- comfortable driving
- time efficiency, i.e. minimizing travelling time

Another technique will be a comparison of online *computed trajectories* with

- offline generated ideal trajectories
- behaviour of human drivers

The challenge in both cases is to re-produce the dynamic situation in which the online trajectory was shown. Each action and reaction changes the behaviour of the overall dynamic system.

Finally, on a control level, planned trajectories must be compared with executed trajectories. This comparison is also useful to evaluate possible differences between those trajectories planned by a technical system, and those shown by human drivers.

3 Online risk assessment

The purpose of online risk assessment in AutoMate is the calculation of *safety corridors*, where, for the moment, we define safety corridor as a *representation of a constrained multidimensional state space that quantifies the risk of the current and near-future traffic situation for each point contained within the state space according to a metric of risk.* Access to such safety corridors will allow the TeamMate car to assess and plan safe and feasible trajectories, leading to a set of algorithms that allow identifying safe and reasonable arrangements of the driving process. Online risk assessment will be provided to the TeamMate car by a dedicated software module, the *online risk assessment component*. Specification of the online risk assessment component requires the common understanding of risk and corresponding metrics for risk, the definition of potential representations for safety corridors, and functional requirements in form of input, output, and constraints for online risk assessment. In this deliverable, we will provide a first sketch or high-level description of the expected functionalities of the online risk assessment component for the TeamMate car. During the course of AutoMate, this description will then be periodically updated to provide a complete specification of online risk assessment.

On a generic and high level, we envision that online risk assessment in AutoMate will be realized in terms of a software module that implements two functions provided to the TeamMate car via dedicated interfaces:

- For input in form of a spatial-temporal estimation of the world state, a so-called episode e and a metric m, the online risk assessment shall provide as output a safety corridor c_m over a multidimensional state space S in respect to the metric:

$$f: (e, m) \rightarrow c_m.$$

- For input in form of a safety corridor c_m based on a metric m and point (or set of points) in the state space $s \in S$ covered by the safety corridor, the online risk assessment shall provide as output a risk r:

$$g: (c_m, s) \rightarrow r.$$

In the context of intelligent driving systems, the notion of *risk* is commonly associated with the idea "that a situation may be dangerous for the driver, i.e. may result in harm or injury" (Lefèvre et al., 2014). In recent years, this definition of risk has been broadened to include situations in which drivers react unexpectedly. As such, Lefèvre et al. (2014) classify risk assessment approaches into two broad families, approaches that consider risk associated with physical collisions between entities (e.g. vehicles) and approaches that relate risk to unexpected behaviour of traffic participants. Accordingly, several metrics for risk assessment have been proposed in the literature, including e.g., time-to-x measures, collision probabilities, or comparisons between expectations and intentions (Lefèvre et al., 2014). Based on the current understanding of safety and corridors, risk assessment in AutoMate belongs to the first of the two families proposed by Lefèvre et al. (2014). Following this notion of safety corridors, the most useful metric for risk in

AutoMate is the *probability of a collision* (Houénou et al., 2014, Lawitzky et al., 2012), or simply *collision probabilities*. Collision probabilities denote the probability that a collision between the driver's vehicle and surrounding objects, like traffic participants or road boundaries, will happen at a certain place and time. They provide an intuitive notion of risk and can be extended by additional information if needed, e.g., by estimating the severity of a collision based on the velocities and/or type of object involved, etc.

As a starting point for the necessary definition of safety corridors based on collision probabilities, we will investigate representations of *free space* proposed in the literature that could be adapted or already represent safety corridors. A simple, but illustrative, example for a safety corridor is that of an occupancy grid (Laugier et al., 2011) that represents the planar environment around the TeamMate car in terms of an evenly spaced field of grid cells that are either free or occupied, with each cell represented by a binary random variable. The resulting occupancy grid can then be used to derive the probability for each cell of being occupied, which directly correlates to a metric of collision probabilities. Other proposals including, e.g., interval maps (Weiherer et al., 2013), parametric free space maps (Schreier et al., 2013), and risk maps (Damerow and Eggert, 2014). The different possible representations differ in their memory and computation requirements and their potential use for path planning.

Regardless of the exact form of representation for safety corridors, their estimation requires input in form of the prediction of likely temporal and spatial evolutions of the traffic situation, so-called episodes, e.g. the future motion, and positions of the traffic participants. For the calculation of safety corridors in AutoMate, this input will be provided by driver, vehicle, and situation models developed in WP2. For now, we envision that such an episode will be provided in terms of a probability density estimate $p(S^{t:t+n}|o^{1:t})$, where S represents a situation and o represent the sensor data provided by the available sensors.

3.1 Plan and Metric for Verification

Verification for online risk assessment should be understood as the assertion that all high-level functionalities of online risk assessment have been implemented and successfully tested according to the requirements. The degree of verification of the online risk assessment will be measured with functional coverage metrics. Functional coverage metrics measure the verification progress with respect to the functional requirements of the intended design by tracking that all identified and specified functional requirements have been implemented and all implemented features have been tested.

Functional requirements will include and cover the input, output, and posed constraints for online risk assessment, more specifically:

- The formal definition of the expected representation of the spatial-temporal estimates of the episodes, used as required input for the online risk assessment.
- The specification of supported risk metrics.
- The definition of the representation of the safety corridor for each supported risk metric.

- The specification of time-, memory-, and computational constraints for the target architecture of the demonstrator.
- Potential additional functional requirements.

Based on the functional requirements, we will then prepare a set of test cases that cover all functional requirements for all considered demonstrators and use cases, defining the input and expected output for the main functionalities of online risk assessment. Online risk assessment will then be verified according to a functional coverage metric *measuring the proportion of functional requirements that have been tested successfully and the total number of functional requirements*. Furthermore, as the name implies, the online risk assessment shall be performed online, i.e. the online risk assessment component will have to provide safety corridors to the other components of the TeamMate car at certain intervals with limited processing time, available memory and computational powers. We will experimentally ensure that these functional requirements concerning time-, memory-, and computational constraints are met by testing the online risk assessment for specific use cases on specific target architectures. If possible, we will furthermore provide formal derivations of time-, memory-, and computational constraints for in respect to the duration and complexity of the episodes provided as input.

3.2 Plan and Metric for Validation

Following the American Institute of Aeronautics and Astronautics (AIAA), validation is defined as the process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended use of the model (Oberkampf and Barone, 2006). For online risk assessment, validation can therefore be understood as the assessment of how well the provided risk assessment captures the "real" risk in respect to the metrics of interest.

Under the assumption that the correct functionality of the online risk assessment has been verified, deficits in online risk assessment may arise due to two causes:

- deficits in the spatial-temporal estimates of the world state provided as input to the online risk assessment;
- approximation errors due to the implementation, based on time-, memory-, and computational constraints.

As driver, vehicle, and situation models, as well as their predictive qualities will already be validated within WP2, validation of the online risk assessment will abstract from deficits inherited by the available driver, vehicle, and situation models, focussing on a quantitative assessment of loss induced by the restrictions.

As such, for a given quality of the situation prediction we will therefore *quantify the approximation errors induced by the time-, memory-, and computation constraints on the target demonstrators and target use-cases, compared with an ideal online risk assessment that doesn't has to comply such restrictions* and use them as validation metrics (not to be confused with risk metrics used for online risk assessment) for online risk assessment.

4 Trajectory planning and execution

The trajectory planning and execution is constrained by the output of the corridor computation. Knowing that we have three main families of trajectory planning that we can consider:

- Sampling points: In this case the evolution space is reduced to space- and/or time-related points (preferably part of possible vehicle states set).

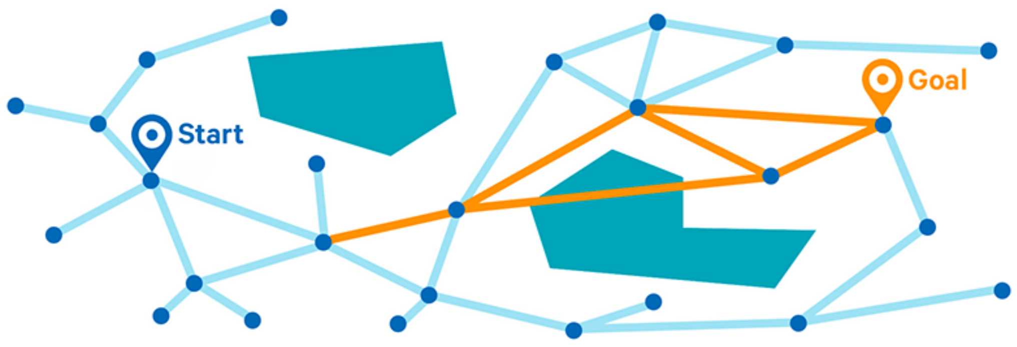

Figure 1 : Set of points and a possible feasible path

- Cell decomposition (occupancy grid): the evolution space is divided into space- and/or time-related cells.

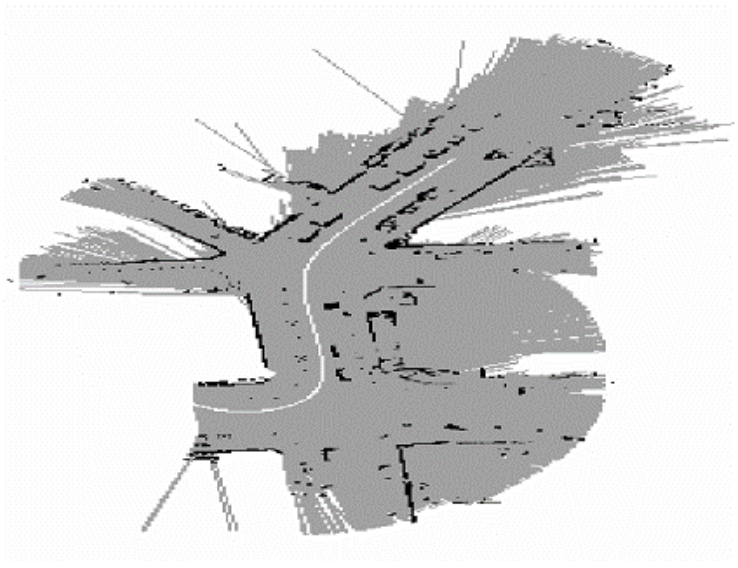

Figure 2: Occupancy grid and a planned trajectory

- Lattice decomposition: The evolution space is divided into pre-defined (spatiotemporal) manoeuvers.

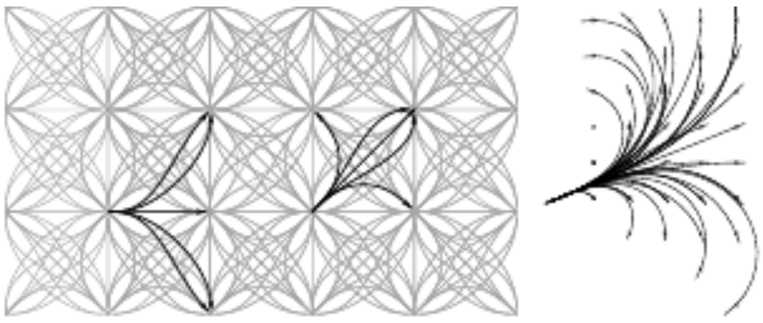

Figure 3: An example of a lattice decomposition

Each elementary information (cells, points, branches of the lattices) in the free space and in the uncertain space are stored with their collision probabilities.

In each one of these three cases, the computed corridor will help to restrict the search space. Once this part is realized we can apply the Dijkstra-like algorithm (Roy, A*, D*, D*Lite, ARA*, etc.), for more details see (Soltani et al. 2002, and Raja, P., & Pugazhenthi, S., 2012). These kinds of algorithms aim to have the minimum weighted path between two nodes in a graph and find out a "skeleton" of the desired path. The result of such algorithms gives as an output as a broken path to follow. Such kind of paths are not feasible by a car due to some constraints that we have to take into account:

- the kinematic constrains of the vehicle, which are the nonholonomic dynamic and a smooth path to travel through,
- the trajectory should be differentiable
- the curvature must be continuous (second order geometric continuity).
- the width and heights of the ego-vehicle and the surrounding obstacles

The resulting paths must be smoothed and fitted. To do that, several methods can be used. We can, for example, use a constrained smoothing cubic spline or Beziers curves and splines.

To execute the path computed above, we need some useful information :

- An accurate localisation at least in lateral.
- A set of points to follow or at least the boundaries of the corridor in which the vehicle can drive safely.
- The starting and the end point
- The steering wheel angle and the speed
- The desired "speeds" to go from the actual point to the desired point

A control is realized in order to translate the computed path into control variables.

In order to validate the generated paths we must take into account the following metrics:

- The algorithm memory complexity: the algorithm must take a reasonable memory space.
- Time execution: the time needed to generate a path having a corridor.
- Smoothness of the planned path: along the generated path, the lateral acceleration and its derivative must be bounded at a given value.
- Minimum and maximum curvature: minimum and maximum magnitude of the curvature along the path.
- Length of the path in meters: the generated path must correspond to a time window of x seconds (x can be 2 seconds for example.).

5 Online & offline learning algorithms to learn from the driver

The AutoMate system should adapt the automation strategies to the driver's preferences and guarantee a human expert-like driving behaviour. This means there should be algorithms to learn offline from recorded and annotated driving and situation data, as well as algorithms which allow the automation system to learn online from situation data and driver inputs during the driving process.

For offline learning the collected data is annotated by an expert to generate a ground truth. Model is trained, usually on a part of the ground truth, to fit the data as good as possible. Via cross validation trained model is tested against the part of the ground truth which was not used for training. Thus, the validation of the offline learning algorithms is the validation of the learned model.

Via online learning the parameters of the driver model are adapted to the actual driver behaviour. By doing so the AutoMate car will learn to perform manoeuvres similar to the driver and make the driver's preferred decisions and perform the preferred manoeuvre, e.g., *lane change left* or *car following* in specific situations.

The driving behaviour of the driver is continuously observed. The data regarding the driver's state, behaviour etc. is received from the driver and situation models (WP2). This collected data will not be annotated by an expert. The used learning algorithms shall only learn behaviour that leads to safe driving.

The learning algorithm shall also be used to explore unknown regions within the safety-corridor computed by T3.3.

5.1 Plan and Metric for Verification

The goal of the verification of the offline and online learning algorithms is to assure that the implemented software satisfies all the expected requirements. This means that is ensured that all relevant functionalities of the offline and online learning algorithms are implemented and tested. Some of the functionalities and requirements have already been defined during T1.3.

Like in section 3.2 we intend to apply functional coverage metrics to track that all functional requirements and features have been implemented and tested.

Especially for the online learning, efficiency is a crucial factor. Since, depending on the learning strategy, the learning algorithms might run in parallel, with other components of the TeamMate car. This means limited processing time, memory and computational powers.

We will test the online learning algorithms for specific use cases on specific target hardware. Thus we experimentally test if requirements concerning time-, memory-, and computational constraints are met. If possible, we will furthermore provide formal derivations of time-, memory-, and computational constraints.

5.2 Plan and Metric for Validation

For the offline and online learning algorithms to learn from the driver the validation can be understood as the assessment of how well the learned driver model reflects the “real” driver in respect to the relevant metrics e.g. the driver’s behaviour or manoeuvres in certain situations.

During experiments we will create test sets of time-streams, i.e. sequences of traffic situations, and compare the model performance before and after learning steps. For a useful leaning the output of the model after learning should fit better to the new data than the output of the previous model and it should also still perform well on older data, which was known before the learning step.

6 Conclusions, outlook

This deliverable shows the work package's approach to verification and validation of software. Specifically this involves those implemented functions or algorithms that realize vehicle functions regarding adaptive and safe driving strategies.

We will assume a functional perspective, in which desired vehicle behaviour in a situation is understood as the output of a computation. The inputs of the computation will be entities such as vehicle properties, the situation, or even the behaviour of the vehicle's human passengers.

Such a functional approach produces output that can be well communicated, is sufficiently high level to not restrict interesting ideas, but still can be developed further towards formalization.

The next steps in WP3 will centre around requirements elicitation and interface specification.

We will further investigate the realization of the use cases together with the demonstrator owners. This will lead us towards the definition of requirements, both functional and non-functional. Taking the online risk assessment as an exemplar, the necessary step to follow is a definition of the safety corridor's exact nature. This will also entail the discussion of appropriate risk metrics.

Together, the use case based discussion will yield a better understanding of the precise functions, algorithms or implementations which are required to make the TeamMate car successful.

References

Damerow, F. and Eggert, J. (2014). Predictive risk maps. *Proceeding of the IEEE 17th International Conference on Intelligent Transportation Systems (ITSC 2014)*, 703-710.

Houénou, A., Bonnifait, P., and Cherfaoui, V. (2014). Risk assessment for collision avoidance systems. *Proceedings of the IEEE 17th International Conference on Intelligent Transportation Systems (ITSC 2014)*, 386-391.

Laugier, Ch., Paromtchik, I. E., Perrollaz, M., Yoder, J-D., Tay, Ch., Yong, M., Nègre, A., and Makhnacha, K. (2011). Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety. *IEEE Intelligent Transportation Systems Magazine*, 3(4), 4-19

Lawitzky, A., Wollherr, D., and Buss, M. (2012). Manoeuvre-based risk assessment for high-speed automotive scenarios. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1-6.

Lefèvre, S., Vasquez, D., and Laugier, Ch. (2014). A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH Journal*, 1:1, 1-14.

Oberkampf, W. L. and Barone, M. F. (1985). Measures of agreement between computation and experiment: Validation metrics. *Journal of computational physics*, 217, 5-36.

Schreier, M., Willert, V., and Adamy, J. (2013). From Grid Maps to Parametric Free Space Maps – A Highly Compact, Generic Environment for ADAS. *Proceedings of the 2013 IEEE Intelligent Vehicles Symposium (IV)*, 938-944.

Weiherer, T., Bouzouraa, S., and Hofman, U. (2013). An interval based representation of occupancy information for driver assistance systems. *Proceedings of the 16th International IEEE Annual Conference on Intelligent* Transportation Systems (ITSC 2013), 21-27.

Soltani, A. R., Tawfik, H., Goulermas, J. Y., & Fernando, T. (2002). Path planning in construction sites: performance evaluation of the Dijkstra, A∗, and GA search algorithms. Advanced Engineering Informatics, 16(4), 291-303.

Raja, P., & Pugazhenthi, S. (2012). Optimal path planning of mobile robots: A review. International Journal of Physical Sciences, 7(9), 1314-1320.