An Approach Towards Automated Navigation of Vehicles using Overhead Cameras

V. Sri Chakra Kumar, Amit Sinha, Pratheek P. Mallya, Nutanlata Nath
Department of Electrical and Electronics Engineering
BIT Mesra
Ranchi, India
srichakra89@gmail.com, amitfishy@gmail.com, prateek.mallya@hotmail.com, nlnath081054@gmail.com

***Abstract* — At present the need for automated systems is increasing because of the greater demand of services and the improved efficiency of automated systems. In this paper, we present an approach to automatically navigate a vehicle from a starting location 'Point A' to a destination location 'Point B', given a video feed from an overhead camera. Our approach towards automating navigation systems is novel and is effective in cases where a large number of vehicles need to be managed in a small area. In this paper, we present the proof of concept of the proposed system. The applications of this work are directed towards warehouse fleet management, distribution and automated control of industrial vehicles.**

Keywords—Automated Guided Vehicles, graph segmentation, Canny edge detection, A-star, safety factor, search factor, pure-pursuit controller

I. INTRODUCTION

With the advent of Computer Vision, the drive to automate the motion of objects ranging from playful robots to heavy vehicles is widely sought due to the advantages it offers in day-to-day utilities. This paper mentions ceiling mounted camera based vision systems for navigation of vehicles and robots in indoor environments and emphasizes its application in industrial warehouses for automation of vehicle navigation. To navigate a vehicle or robot safely on the floor, information of its surroundings is necessary to decide the direction of motion avoiding hitting of any objects/obstacles. The ceiling mounted cameras provide in their field of view a comprehensive view of the vehicle's surroundings which helps to determine safe direction for its motion. The technology of autonomous navigation in indoor environments has wide applications such as autonomous personal assistants or service robots, autonomous carts in shopping centers, surveillance robots, autonomous robots or vehicles in warehouses, etc.

Some of the earliest work on vision – based navigation [1] used topological navigation to cover large and more generic distances and visual path following to cover smaller and more precise distances. The system designed by Kelly et al. in [2] was one of the earliest systems which was developed that was independent of artificial guidance infrastructure. The developed system uses cameras facing downward, attached to the bottom of the AGV to achieve navigation tasks, by exploiting naturally occurring visual cues.

On board vision systems [1-4] typically involve at least one camera and supporting systems for navigation. Some of the more recent work in on board vision systems is robust to the general case of navigation in various structured and unstructured environments [5, 6]. On board vision systems are economically feasible for scenarios which typically involve coverage of large distances and require very generic navigation capabilities. They are generally more robust than the ceiling mounted camera paradigm.

However, in a scenario where there is high density of vehicles and a small warehouse or arena, the ceiling mounted camera system [7-14] can be much more economically feasible since it will use significantly fewer cameras. Most of the work done in this area in recent times [7-14] mentions the primary advantage being the reduction of number of cameras required in constrained scenarios. Some of the work also explains why it is important to minimize the computation required at each vehicle [13]. The computational burden on each vehicle is unavoidable in on board vision systems. Thus, in a warehouse where a fleet of vehicles are to be managed multiple cameras based navigation system would be ideal.

To illustrate the advantages of a ceiling mounted camera based navigation approach over an on board mounted camera based navigation approach, consider a simplistic scenario such as

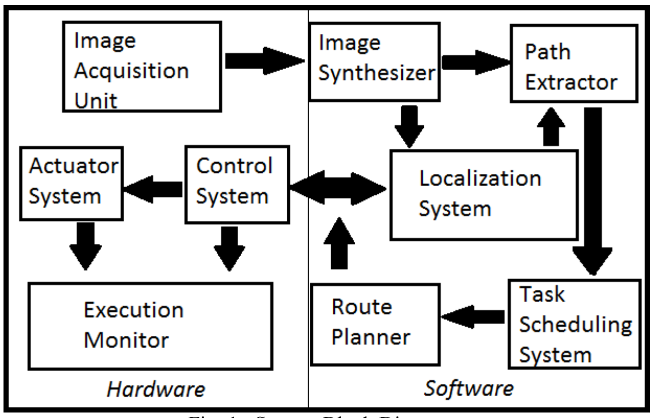

Fig. 1. System Block Diagram

This work is sponsored by Department of Electrical and Electronics Engineering, BIT, Mesra from the UG/PG Project fund granted to the department by the institute BIT, Mesra.

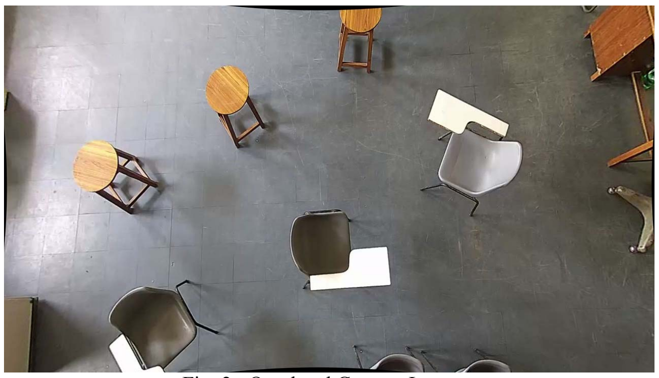
Fig. 2. Overhead Camera Image

that shown in our prototyping arena (Fig. 2). Such an arena would be representative (to some extent) of the industrial warehouse or industrial operation zone. The vehicle we are using to prototype the industrial vehicles and service robots has a small size with respect to the prototyping arena (it has the same size as the '**T**' shaped marker placed on the stool in Fig. 4). It is easy to understand that in such a situation where so much free space is left in the arena, that multiple robots/vehicles can be deployed at the same time. Having more robots/vehicles would lead to more goods transported from point 'A' to point 'B' over a certain duration, thus increasing transporting output in the same time. The major requirements of such a system would be a number of ceiling mounted cameras depending on how large the warehouse is and the height of the warehouse ceiling. A higher ceiling would require lesser cameras for full coverage.

For the case of on board systems, it is obvious that at least one camera per vehicle is required for navigation purposes. Using more cameras can increase the quality of localization and provide additional information about the surroundings of the vehicle. However, in the ceiling based approach all the information is available with a single camera (for a particular area of coverage).

If we assume that each camera placed in the warehouse ceiling on average sees more than at least one vehicle at a time, then the effective number of cameras required for this case will be lesser as compared to the case for on board systems. For example, if we accommodate 5 vehicles in the arena, the number of cameras required reduces from 5 to 1. The cameras involved in machine vision applications tend to be very expensive and this method can save a lot of capital. Other advantages include the co-ordination and re-routing by a central server rather than the individual vehicles themselves and this eliminates the possibility of individual vehicles performing conflicting tasks.

This paper presents the ground work necessary for vehicle navigation using multiple ceiling mounted cameras. As the field of view of each downward looking camera is limited, the idea is to employ cameras on ceiling, translated in terms of position to cover entire arena. This idea suits well to the warehouse models similar to the one mentioned in [12]. Fig. 1 shows the subsystems involved in our idea of multiple camera based vehicle navigation system. There have been various developments towards image stitching as a whole, particularly the work done in [15] is capable of taking overlapping images of the same scene and automatically putting these together in one resultant image with minimal manual effort. This method uses features extracted by the SIFT algorithm [16]. However, in the current paper we are considering only a single camera's video feed and the image stitching methodology is left to future work. The key feature of this paper lies in the choice of methods chosen for path extraction, route planning and localization system to produce robust results

This paper is organized as follows. Section II presents some background into the general on board and ceiling mounted camera paradigm as well as the background of concepts we have used. Section III contains details about the methodology used. Sections IV and V describes the software and hardware results along with a description of the equipment used. Sec VI discusses the results and future scope.

II. LITERATURE REVIEW

Presently, there are many existing automated guided vehicles (AGVs) which are based on infrastructural structure. The difficulties of an infrastructure based AGV are elucidated in the work by Kelly et al. [2] and they have used on board cameras and systems, which communicate with a central server. Some systems use omni-directional cameras to increase the area of coverage by on board vision systems [1, 3]. On board vision systems are even used in unstructured environments for agricultural purposes [4]. Some of the more recent approaches are described in [5, 6]. The method described in [5] uses a pair of cameras for stereo vision rather than the traditional single camera based mono vision. A much more realistic scenario was presented in [6] which incorporated visual servoing along with obstacle avoidance in simulated as well as real world vehicles.

There has not been much work concerning systems with ceiling mounted cameras in recent times [7-14]. Most of these papers emphasize the importance this methodology achieves in reducing the number of required cameras and in reducing the computation at each vehicle.

Samo et al. [9] use a ceiling camera to detect the robot and guide it along the path using a predictive controller. Shim et al. [13] use multiple cameras to get a more accurate localization. Hoover et al. [14] use multiple dome cameras to navigate a robot in between specified location. A comprehensive survey on automated navigation of robots has been given by Francisco et al. in [17].

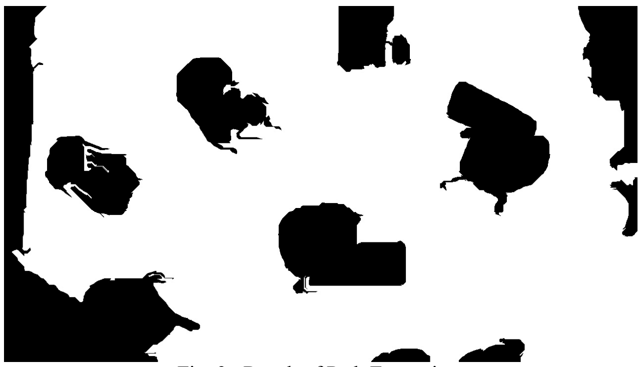
Fig. 3. Result of Path Extraction

In our approach one of the most important steps is differentiating between the overall traversable path and the obstacles (stationary or moving). The method to be used for segmentation of floor portion available for vehicles to traverse is to be generic enough to be used in most of the indoor environments of warehouses. To achieve this, we have tried using the methods based on texture recognition [18], color constancy [19-21] and graph based segmentation algorithms [22]. Most of the color based segmentation algorithms [9, 23] suffer due to variations in illuminations and shadow effects.

Most of the applications for path search use greedy algorithms such as Dijkstra. At each step of progression, the nearest neighbor to the destination is considered to find the shortest path to the destination. A* (read as A star) algorithm is a variation of typical Dijkstra's algorithm. It includes the heuristic based calculations of distance from a neighbor to destination location. This kind of heuristic based approach reduces searching and provides the solution in less time comparatively. Cui and Shi [24] published a survey paper on the A* algorithm, its variations and optimization techniques. The paper [24] reviewed the most relevant topics in the academic domain of pathfinding.

Localization of the vehicle is a vital requirement for visual servoing. Work done in [13] mentions various techniques for localization and presents homography based localization system using multiple surveillance cameras. A work on robot based surveillance system [8] used particle filter method for localization of robot. The paper [11] presents a method for localization of multiple robots with ceiling mounted cameras using pattern matching. It considers error in pose estimation and location of robot due to the perspective of camera mounted on ceiling. However, for industrial warehouses the height of roof top is much greater than the height of vehicle and hence perspective error can be assumed to be less for model simplicity. Visual servoing implementations mentioned in [9, 10] used color markers (circular blobs or isosceles triangle) on the robot for localization.

The paper [25] presented an in-depth look at various path tracking controllers used in autonomous vehicles. It also presented the dynamic equations that describe the vehicle system by approximating it to a bicycle, called the bicycle model. The paper also described in detail several controllers commonly used for autonomous path tracking and gave a comparison of these approaches. One of these, namely the Pure Pursuit Tracker is utilized in our project. In [26] the Ackermann Steering Geometry is described in detail, providing the constraint equations and the derivation of the model equations. It also presented several types of steering mechanisms, the behavior of the system during low and high speed cornering and the advantages of the Ackermann geometry over other types of steering mechanisms.

III. THEORY

The most important steps for implementing an autonomous vehicle are essentially constructing the traversable area as a map as done in [2], localization of the vehicle in terms of position and orientation, determining the exact route the vehicle should traverse and making sure that the vehicle adequately traverses the path from start to end without any major deviations. This section involves the methodology involved in creating a system which is capable of navigation from start to end autonomously.

A. Path Extraction

The first step is to extract the viable portion in which the vehicle can move around (the path) and separate it from the obstacles in the image. In our case the floor is considered as the path and everything else is considered as obstacles. This means, the vehicle itself will be considered to be an obstacle. Hence, at first path extraction is implemented and then the vehicle is identified in the image (section III-B) so that it can be masked to avoid being considered as an obstacle. A typical overhead image of the arena is shown in Fig. 2.

We have obtained robust results for path extraction in spite of texture, variation in illumination and shadow effects (to an appreciable extent) following the graph based segmentation algorithm in [22]. The image is represented as an undirected graph, **W** where the pixels are considered as the vertices, $\mathbf{v}_i$ and the weight between the vertices **i** and **j** is given by (1)

$$W(v_i, v_j) = \left| I(p_i) - I(p_j) \right| \qquad (1)$$

where $\mathbf{I(p_i)}$ is the intensity of the image **I** at pixel **i** and $\mathbf{I(p_j)}$ is the intensity at pixel **j** and the absolute value of this difference is used as the weight between vertex **i** and **j**. In this manner, a graph is constructed which has vertices equal to the number of pixels and each vertex is connected by weights to its adjacent neighbors (adjacent 4 pixels).

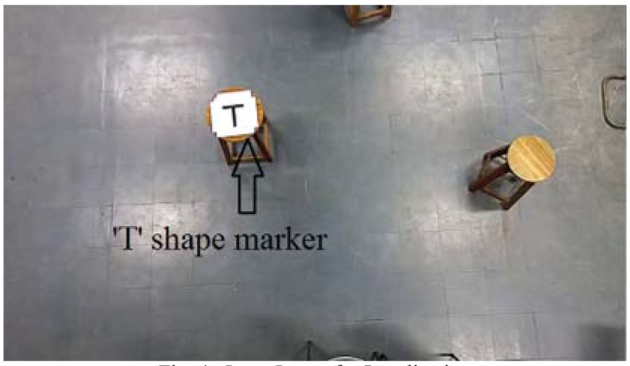

Fig. 4. Input Image for Localization

We can segment the image by separating this graph into various connected sub-graphs (we can even consider unconnected graphs, but for simplicity we are using connected graphs). A segmentation **S** depends on the components **C** which describe the sub-set of pixels chosen for each component (**C**$_1$, **C**$_2$ and so on). The number of elements in set **C** indicates the number of components of the segmentation and each element (e.g. **C**$_1$) contains a sub-set of pixels which it considers as a segment and thus, the result segmentation is described by **C**.

In the beginning, each vertex **v**$_i$ is initialized to its own component **C**$_i$. The weights **W**(**v**$_i$, **v**$_j$) are sorted in ascending order and then each weight is considered in order. If the weight satisfies some criteria (mentioned below), then the two components connected by the weights are merged together into a single component. In this way, we process each weight in the graph and obtain the result component set **C**. This helps us obtain a segmentation which is neither too coarse nor too fine (as described in [22]).

Two components (**C**$_i$, **C**$_j$) are merged if criteria (2) is satisfied. The function *Int(C)* shown in (4) is termed as the internal difference of a component. **C** is the component which is being considered and **E** represents the connecting edges in the graph of **C**. The minimum spanning tree (MST) is found from this graph and then the maximum value of weight is taken from it.

$$W(v_i, v_j) \leq MI(C_i, C_j) \tag{2}$$

$$MI(C_i, C_j) = Min(Int(C_i) + \tau(C_i), Int(C_j) + \tau(C_j)) \tag{3}$$

$$Int(C) = \max_{i,j \in MST(C,E)} W(v_i, v_j) \tag{4}$$

The value of *Int(C)* is considered for both components and then the minimum is considered in (3). A thresholding function **τ** is used to control the trade-off between a coarser or a finer segmentation. It can be set to be inversely proportional to the component size, thereby giving smaller penalties for larger component size and larger penalties for smaller sizes. The criteria (2) can be thought as representing the similarity of a particular weight to the components that it connects.

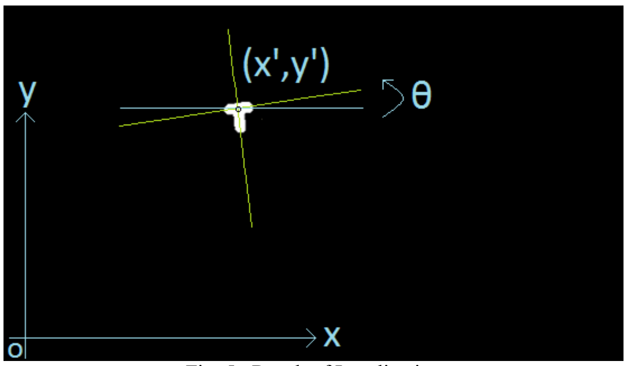

Fig. 5. Result of Localization

Once the segmented regions are obtained, the largest continuous region is considered as the path (white region) and the other regions are considered as obstacles (black region). Here we are assuming that the largest continuous segment will be the path which would be reasonable in a warehouse with fully connected paths. It can be seen in Fig. 3 that the algorithm is robust to some extent towards variation in lighting and light shadows, however it fails to identify the region of the floor with darker shadows. The arena should be adequately lit to avoid darker shadows, so that the algorithm can accurately represent the path (we have intentionally used a photo lit by natural daylight with the room lights off to illustrate the weakness of this algorithm to very dark shadows).

B. Localization System

Fast and accurate localization method is necessary for visual servoing. The primitive threshold method is efficient when the foreground to be segmented and its background have high contrast. Hence a marker of black letter '**T**' in a white patch is placed above the vehicle to identify the location and orientation of the vehicle in the arena image. Fig. 4 shows a marker placed on a stool for testing of localization algorithm being proposed. The marker portion is detected using thresholding operation. The canny edge detection method [27] is used to find the contours of white portion and letter '**T**'. The marker is segmented by looking for sub-contours within contours and by assuming that the ratio of area of the '**T**' marker with the area of its white background is roughly the same all over the arena. Further the segmented '**T**' is morphologically thinned [28] following which it has three end points and a junction. Also, the end to end distance of horizontal line is less than vertical line. Using this information, the position (centroid) and orientation (basic trigonometry) of the marker (vehicle) can be obtained (Fig. 5).

C. Route Planning

Provided the starting location and the destination for a vehicle, it's a trivial task to find a route joining the locations keeping the travelling time as minimum as possible. To find the route, the system must be aware of the region in the arena safe to travel without colliding into any obstacles. The path extraction mentioned in section III-A gives the region (as a binary

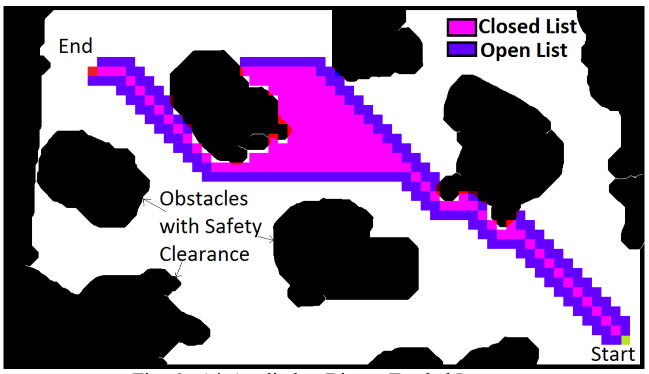

Fig. 6. A* Applied to Binary Eroded Image

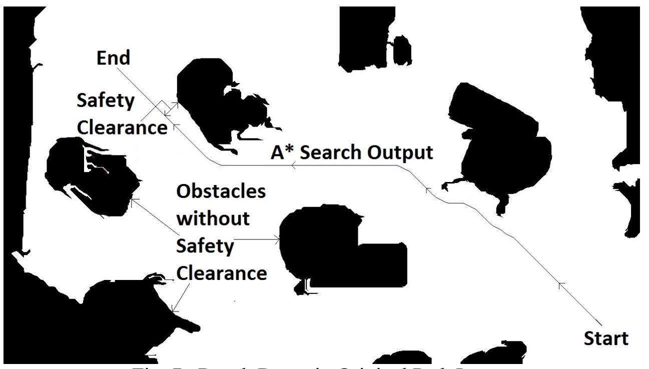

Fig. 7. Result Route in Original Path Image

image) that is safe for the vehicle to traverse in the arena. In this work, we implemented the A* algorithm for determining the shortest route to the destination from starting location.

It is necessary to ensure that the route and hence the vehicle should maintain a safe distance from the obstacles to avoid accidents. The minimum safe distance (in pixels) required is termed as 'safety factor'. The path image is eroded using a circular kernel of radius equal to the 'safety factor'. This allows the obstacles to appear larger than they are (compare Fig. 6 and 7) and hence the A* route has some clearance from the obstacles. This step also modifies the sharp edges of obstacles into curved to an extent, which helps in finding a path with smooth turns.

The images processed being large, the block size of A* search route is kept larger than a single pixel to ensure optimal processing speed. This block size of A* search is termed as 'search factor' which is always taken as an odd number so that the center pixel of square block represents the current co-ordinate of the block at any moment in A* search.

If **S** (**x₁**, **y₁**) and **Q** (**x₂**, **y₂**) are the starting and the destination co-ordinates respectively. The heuristic function *f(s)* used to estimate the minimum cost to travel from **S** to **Q** is given by (5), (6) and (7).

$$f(s) = (d_{\max} - d_{\min}) + (\sqrt{2} \times d_{\min}) \quad (5)$$

$$d_{\max} = \max(|x_1 - x_2|, |y_1 - y_2|) \quad (6)$$

$$d_{\min} = \min(|x_1 - x_2|, |y_1 - y_2|) \quad (7)$$

After the path is obtained (Fig. 3), the erosion operation (with disk kernel of radius 9 pixels) followed by the A* search algorithm [24] gives the result shown in Fig. 6. Put in contrast with the normal sized obstacles, we can see that the route obtained has a safety clearance from the obstacles (Fig. 7). This clearance can be increased by increasing the radius of disk kernel used for the erosion operation.

D. *Control System using Ackermann Steering Vehicle*

The control system uses a geometrical route (path) tracker, which is slightly modified from the one featured in [25]. A route tracker is a controller that minimizes the error or the perpendicular distance between the desired route and the vehicle. This is done by minimizing the angular error (δ) between the vehicle heading vector and the path vector [25], as shown in Fig. 8. The orange circle represents the vehicle's current position. The green circle represents the goal point on the desired route. The orange line represents the vehicle heading vector, $\mathbf{V_H}$. The green line represents the desired path vector, $\mathbf{V_P}$. The angle between $\mathbf{V_H}$ and $\mathbf{V_P}$ gives us the angular error (δ).

The vehicle position and orientation are obtained from the method described in III-B, which are then used to calculate and minimize the angular error. Since we have the advantage of knowing the route in advance, we use a 'look-ahead' distance to tune this controller. The 'look-ahead' distance determines the goal point, which is used to calculate the angular error. The steering angle is determined by this error (proportional control) which generates a steering pulse that is applied to the stepper motor (see section V).

Furthermore, the controller is modified to incorporate the vehicle's speed into account, slowing down the vehicle before a turn and speeding up the vehicle after taking the turn. The block diagram of the control system is given in Fig. 9.

The above control system was simulated in MATLAB utilizing a vehicle model derived from [25]. Given a desired

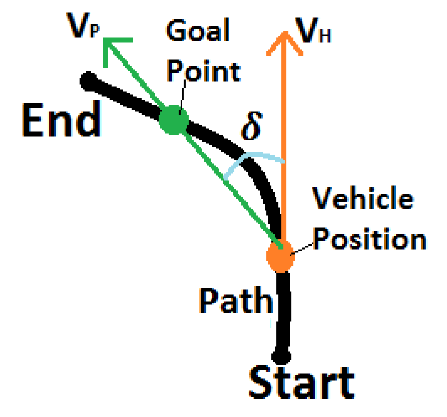

Fig. 8. Geometrical Path Tracker using the Look-Ahead Method

route, the resultant vehicle trajectory was observed for varying look-ahead distances. We also used a 2D animation [29] to observe the steering geometry as the vehicle traversed the route. Fig. 10 & 11 demonstrate the input route and the vehicle trajectory respectively.

The cumulative position error was calculated for obtaining the best look-ahead distance. The best result was obtained for a look-ahead distance of 1 unit (m).

IV. EXPERIMENTAL RESULTS

This section describes the software setup and results obtained. A Hikvision camera, model - DS-2CD2120-I is mounted on ceiling of the room covering an area of 3.2 x 1.8 Sq. meter. The angle of view of camera used is 85° (4mm lens). Maximum resolution of 1920 x 1080, 2MP at a maximum frame rate of 30 fps can be availed from the camera. The camera was connected to the computer via Ethernet switch (Power over Ethernet). The programs written in C++ were run using OpenCV [31] on an Asus K55VM laptop (8 GB RAM, 2.3 GHz) with an Intel i7 3rd Gen Processor. Performance of methods explained above is mentioned in Table-1.

TABLE 1: PERFORMANCE OF ALGORITHMS USED

Image Resolution (width x height)	**Time Taken for Single Iteration in Milliseconds (ms)**				
	Image Acquis-ition	***Path Extract-ion***	***Localiz-ing Marker***	***Route Plann-ing***	***Route Track-ing***
640 x 480	32.511	1141.16	13.83	763.31	< 10
1280 x 720	56.324	3708.62	21.73	1646.15	< 10

V. HARDWARE IMPLEMENTATION

The Ackermann based prototype vehicle (it is essentially a miniaturized car model) was designed by using a stepper motor (1.6A, 200 step, 4.4 kg-cm torque) for steering, and two DC motors (12V,200 rpm each) operated (using L293D motor driver IC) with Pulse Width Modulation for speed control. An Arduino Mega 2560 microcontroller with wireless Bluetooth was used to control motion of robot. A Li-Po rechargeable battery (11.1V, 2200mAh, 8C) is used to power the components on robot.

The stepper motor was placed in the center of the vehicle, with its shaft positioned vertically, and connected to the center of the front axle through an L-shaped arrangement using rods to transfer the steering torque.

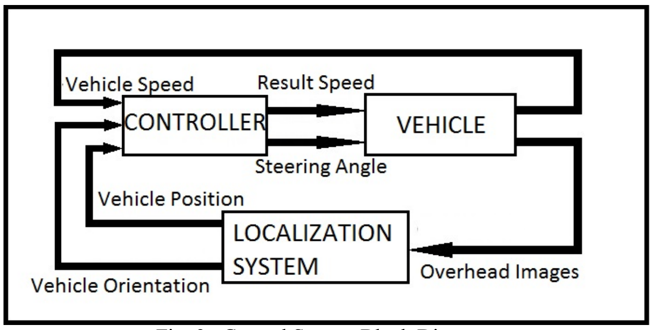

Fig. 9. Control System Block Diagram

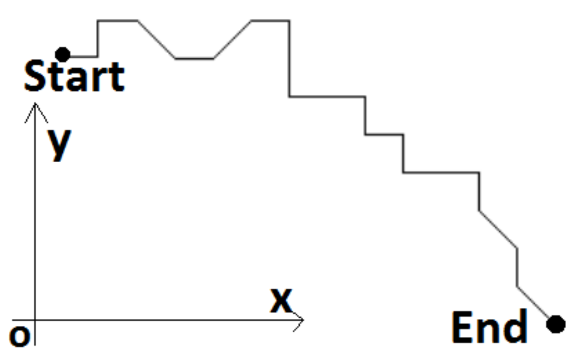

Fig. 10. Sample Input Route

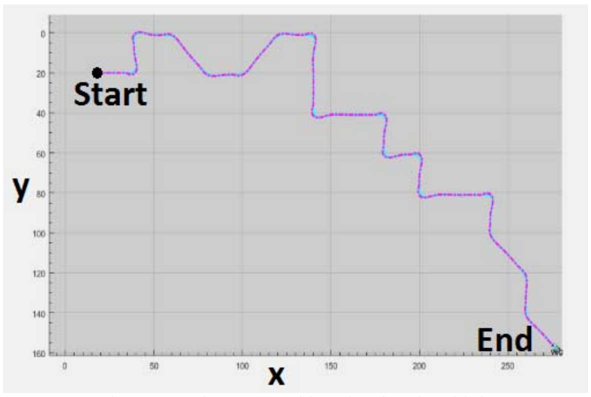

Fig. 11. Path Traversed by Simulated Vehicle

The four wheels used were of 65mm diameter and 12mm width. The two front wheels were connected in an Ackermann steering geometry for the purpose of steering with negligible sideways slip. The two rear wheels were connected to the DC motors for the purpose of driving the vehicle. The chassis and structure of the vehicle is made using parts from a construction set as shown in Fig. 13. The necessary hardware connections are shown in Fig. 12. The vehicle designed is capable of being controlled using Bluetooth from a laptop.

VI. CONCLUSIONS AND FUTURE SCOPE

The individual steps in the overall framework have been implemented and tested. Most of the work done in [7-14] on obstacle avoidance involves detecting the obstacles rather than detecting the actual path. In this manner, the obstacles are detected and the actual map is reconstructed in every iteration. The method we have adopted instead involves detecting the feasible path and the obstacles and then initializing the map there itself. Any dynamic obstacles that come into the picture are easily recognized and immediately added to the map. In this way, our map is initialized and maintained very easily and failure to detect obstacles is very unlikely. The limitation is that it is also more computationally expensive. However, the trade-off is worth it because there are plenty of lighting,

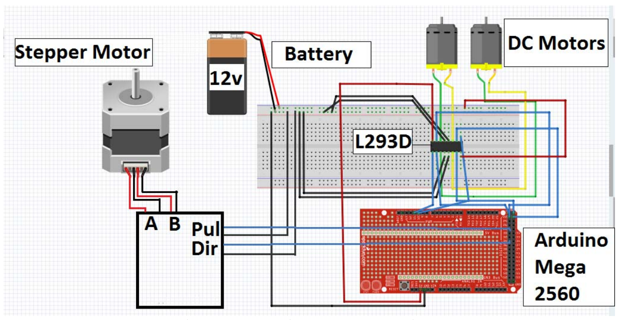

Fig. 12. Hardware Connections

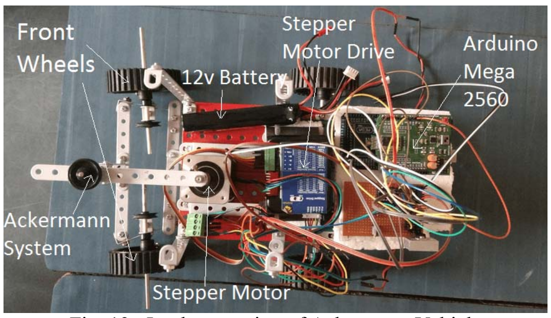

Fig. 13. Implementation of Ackermann Vehicle

texture, noise, etc. which other algorithms are not able to handle (color constancy approaches cannot identify texture). Once the stationary obstacles are detected, this step need not be repeated unless the stationary obstacles change position. In this manner any moving object on the path can be classified as either vehicle/robot (by recognizing the template) or a non-stationary obstacle.

The A* algorithm [24] is one of the most elementary algorithms for this application. It gives us the route from start to destination in the image plane. We use the concept of 'safety factor' here to make sure that the vehicle follows a path to reach its destination while maintaining a safe distance from all the obstacles. This can be observed in Fig. 6 and 7. This factor can further be increased to give larger clearances from obstacles. While it is not easy to compare this with other methods [7-14] because there is no well-defined metric, our method gives us control over how much clearance we want to keep. However, it is typically more computationally expensive over other algorithms. To overcome this issue, we have used the idea of 'search factor'. By reducing the high-resolution image to a smaller image with larger block size, we can reduce the time taken in this step by trading-off the fine resolution path obtained from the original image, to a coarser path with lesser details. By setting these parameters to certain fixed values in a particular scenario, a particular operational constraint can be achieved.

Path extraction and route planning take a significant amount of time; however, these steps do not need to be carried out repeatedly. The path extraction only needs to be carried out if the stationary obstacles change position significantly. Otherwise, the path does not actually change in anyway. Since the localization procedure can be used to determine position of any vehicles, and it is known that the vehicles must be on the path, any portion of the image which is considered as a vehicle will also be considered as the path. Any moving obstacles on the path can be recognized since they will not be identified in the localization step. In this manner, the map of the arena is maintained and updated with the path extraction algorithm only if there are structural changes in the arena (a new warehouse shelf is created/removed).

The route planning step only needs to be carried out once from start point to destination point, unless an anomaly arises (failure of vehicles, thus blocking the path) which requires re-calculation of the route. In such a failure scenario, the vehicle would pause and recalculate a new path quickly. It should be noted however, that we are only considering the case of a single vehicle and traffic control is a very involved problem.

The localization of the template and therefore the vehicle is done by identifying the high contrast region within the image and then processing it to obtain the position and orientation in the image plane. It is shown in [11] that due to perspective error, a small correction should be applied to the vehicle position estimate which is dependent on the height of the vehicle and the position of the vehicle with respect to the camera optical axis. This error increases if the height of the vehicle increases and the ceiling height decreases. It also increases when the vehicle moves further from the camera optical axis. Not accounting for perspective error leads to an incorrect estimate of the vehicle position. However, in our case, for simplicity and because the vehicle height is small compared to the ceiling height, we are assuming that there is no perspective error. In general, the perspective error should be accounted for. On placing the template at different locations, we observed detection of the template in most cases over the arena. In the case of multiple vehicles, each vehicle can be tracked by monitoring the previous frame and selecting the vehicle with the least displacement from the current detected templates.

Since the route is obtained in the image plane and the localization is also obtained in the image plane, the effects of error due to distortion in the camera are reduced. The route tracking has been simulated successfully in MATLAB at 10 Hz. The path traced out by the simulated vehicle (Fig. 11) is almost identical to the randomly generated path (Fig. 10), although on close observation the straight lines of Fig. 11 are a bit wavy in appearance. This is because our model is essentially a proportional controller which is susceptible to overshoot errors. However, the errors are small and can be ignored. Similar performance was observed with many other randomly generated paths.

All the other functionality is executed in C++ with OpenCV [31]. For the system to run in real-time, Image Acquisition, Localizing Template and Route Tracking need to run continuously. From the given results, it is seen that 10Hz can be easily achieved in a real system.

The advantages of our system compared to the systems used in on board vision systems [1-6] can be seen if we consider a scenario with a high density of vehicles to be used in a small warehouse. The proposed systems of [1-6] use at least one camera for the purpose of navigation, while our system can use a single camera to handle multiple vehicles. The system proposed is also controlled by a centralized server while on board vision systems are controlled by separate individual systems, thereby increasing their physical and computational load.

A relatively cheap method of automating navigation of vehicles and service robots in warehouses and other closed spaces is proposed. Each step of the proposed methodology has been carried out, showing promising results. The integration of the overall system is left to future work, along with using multiple overhead cameras to increase area coverage.

A GitHub implementation on algorithms used in this paper is available at - https://github.com/amitfishy/Autonomous-Control-of-Vehicles-using-Overhead-Computer-vision

VII. ACKNOWLEDGEMENT

We are very grateful to the Professors of the department of Electrical and Electronics Engineering at BIT Mesra, Dr. R. C. Jha, for extending the required facilities and Dr. D. K. Mohanta, for giving valuable suggestions at all times. We are also thankful to all the faculty and other staff of the department for their help during this project.

REFERENCES

[1] Gaspar, José, Niall Winters, and José Santos-Victor. "**Vision-based navigation and environmental representations with an omnidirectional camera.**" IEEE Transactions on robotics and automation 16.6 (2000): 890-898.

[2] Kelly, B. Nagy, D. Stager, and R. Unnikrishnan. "**An infrastructure free automated guided vehicle based on computer vision**". IEEE Robotics and Automation Magazine, 14(3):24–34, 2007.

[3] Marius Drulea, Istvan Szakats, Andrei Vatavu, Sergiu Nedevschi, "**Omnidirectional stereo vision using fisheye lenses**", Intelligent Computer Communication and Processing, IEEE, pp 251 – 258, September 2014.

[4] Ming Li, Kenji Imou, Katsuhiro Wakabayashi, Shinya Yokoyama, "**Review of research on agricultural vehicle autonomous guidance**", International Journal ABE, 2009; 2(3): 1.

[5] Bukin, A. G., et al. "**A computer vision system for navigation of ground vehicles: Hardware and software.**" Gyroscopy and Navigation 7.1 (2016): 66-71.

[6] de Lima, Danilo Alves, and Alessandro Corrêa Victorino. "**A hybrid controller for vision-based navigation of autonomous vehicles in urban environments.**" IEEE Transactions on Intelligent Transportation Systems 17.8 (2016): 2310-2323.

[7] Anh Kim Tran, et al. "**Implementation of an obstacle avoidance mobile robot using a ceiling-mounted camera system**." Advances In Dynamics, Instrumentation And Control. 2004. 367-376.

[8] Punarjay Chakravarty and Ray Jarvis, "**External Cameras & A Mobile Robot: A Collaborative Surveillance System**", Australasian Conference on Robotics and Automation (ACRA), December 2-4, 2009, Sydney, Australia.

[9] Simončič, Samo, and Primož Podržaj. "**Vision-based control of a line-tracing mobile robot.**" Computer Applications in Engineering Education 22.3(2014):474-480.

[10] Liang, Xinwu, et al. "**Adaptive image-based trajectory tracking control of wheeled mobile robots with an uncalibrated fixed camera**." IEEE Transactions on Control Systems Technology 23.6 (2015): 2266-2282.

[11] R. Visvanathan et al., "**Mobile robot localization system using multiple ceiling mounted cameras**," 2015 IEEE SENSORS, Busan, 2015, pp. 1-4.doi:10.1109/ICSENS.2015.7370454

[12] Culler, David, and James Long. "**A Prototype Smart Materials Warehouse Application Implemented Using Custom Mobile Robots and Open Source Vision Technology Developed Using EmguCV**." Procedia Manufacturing 5(2016):1092-1106.

[13] Shim, Jae Hong, and Young Im Cho. "**A Mobile Robot Localization via Indoor Fixed Remote Surveillance Cameras.**" Sensors 16.2 (2016): 195.

[14] Hoover, Adam, and Bent David Olsen. "**Path planning for mobile robots using a video camera network.**" Advanced Intelligent Mechatronics,1999. Proceedings. 1999 IEEE/ASME International Conference on.IEEE,1999.

[15] Matthew Brown, David G. Lowe, "**Automatic Panoramic Image Stitching using Invariant Features**", International Journal of Computer Vision, Vol. 74, Issue 1, pp 59 – 73, 2007.

[16] D.G. Lowe, "**Object recognition from local scale-invariant features**", Proceedings of the Seventh IEEE International Conference on Computer Vision, Vol. 2, pp 1150-1157, 1999.

[17] Bonin-Font, Francisco, Alberto Ortiz, and Gabriel Oliver. "**Visual navigation for mobile robots: A survey**" Journal of intelligent and robotic systems 53.3 (2008): 263.

[18] T. Ojala, M Pietikainen, T. Maenpaa, "**Multiresolution gray-scale and rotation invariant texture classification with local binary patterns**", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, Issue 7, pp 971-987, 2002.

[19] Graham D. Finlayson, Steven D. Hordley, "**Color constancy at a pixel**", Journal of the Optical Society of America A, Vol. 18, pp. 253-264, 2001.

[20] Graham D. Finlayson, Steven D. Hordley, Mark S. Drew, "**Removing Shadows from Images**", ECCV, LNCS, Volume 2353, pp 823-836, 2002.

[21] J.M. Alvarez, A. Lopez, R. Baldrich, "**Illuminant-invariant model-based road segmentation**", IEEE Intelligent Vehicles Symposium, pp 1175 – 1180, 2008.

[22] P.F.Felzenszwalb, D.P.Huttenlocher, "**Efficient Graph-Based Image Segmentation**", International Journal of Computer Vision, vol. 59, pp 167-181, 2004.

[23] H.D. Cheng, X.H. Jiang, Y. Sun, Jingli Wang, "**Color image segmentation: advances and prospects**", In Pattern Recognition, Volume 34, Issue 12, 2001, Pages 2259-2281, ISSN 0031-3203

[24] X. Cui and H. Shi, "**A*-based pathfinding in modern computer games,**" International Journal of Computer Science and Network Security, vol. 11, pp. 125–130, 2011.

[25] Jarrod M. Snider, "**Automatic Steering Methods for Autonomous Automobile Path Tracking**", Robotics Institute, Carnegie Mellon University, CMU-RI-TR-09-08, 2009.

[26] Thomas D. Gillespie, "**Fundamentals of Vehicle Dynamics**", Society of Automotive Engineers, 1992.

[27] Canny J., "**A Computational Approach to Edge Detection**", IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.

[28] Lam, L., Seong-Whan Lee, and Ching Y. Suen, "**Thinning Methodologies-A Comprehensive Survey**", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 14, No. 9, September 1992, page 879, bottom of first column through top of second column.

[29] Marc Compere, "**Simple 2D kinematic vehicle animation**", MATLAB Central, 2016.

[30] R. Gonzalez and R. Woods, "**Digital Image Processing**", Addison Wesley, 1992, pp 414 - 428.

[31] Kaehler, Adrian, and Gary Bradski. **Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library.** " O'Reilly Media, Inc.", 201