Vision-Based Path Construction and Maintenance for Indoor Guidance of Autonomous Ground Vehicles Based on Collaborative Smart Cameras[*]

Franz-Josef Streit
Nuremberg Institute of Technology Georg Simon Ohm
Nuremberg, Germany
streitfr50552@th-nuernberg.de

Md Jubaer Hossain Pantho
University of Arkansas
Fayetteville, Arkansas, USA
mpantho@uark.edu

Cindy Roullet
University of Arkansas
Fayetteville, Arkansas, USA
ceroulle@uark.edu

Christophe Bobda
University of Arkansas
Fayetteville, Arkansas, USA
cbobda@uark.edu

ABSTRACT

In this paper, we present a guidance and coordination of autonomous ground vehicles in indoor environment. The solution is based on a set of distributed ceiling-mounted smart cameras with overlapping field-of-view for global coordination. A mean shift based algorithm is implemented to extract a map of the environment. This map is used for a distributed routing of autonomous-guided-vehicles from source to destination. Shortest paths will be calculated and updated in real-time. Our approach fits the requirements of decentralized coordination, real-time environmental changes, as it is the case in production facilities, and portability to other fields of application. First, tests in a test-bench showed satisfying results in terms of reliability, validity and performance.

CCS Concepts

•**Computer systems organization** → *External interfaces for robotics;*

Keywords

Distributed Camera Network; Robot Navigation; Path Planning; Vision based Navigation

1. INTRODUCTION

Autonomous ground vehicle (AGV) have the potential to revolutionize operations in areas such as manufacturing, distribution, transportation, and military. They can help move disabled people around in health-care and nursing facilities. In manufacturing and distribution, AGVs can be used for the mundane and often repetitive task of transporting materials in this area. The recent successful introduction of robot systems in distribution centers and manufacturing to transport items around [17, 13] can not mask the limitations of existing solutions to address the general case. They are designed for one fixed environment, and address only a special case of the navigation problem. Generic models, tools, and technologies are missing to actively capture the world with semantically labelled objects [2], radio nodes [7], gestures [16] and events, to generate goals, priorities, and plans with considerable achievements.

One of the main impediments in current indoor AGVs is path building and maintenance. In this regard, most systems rely so far on Simultaneous Localization and Mapping (SLAM) [1], where robots are first trained to recognize routes and then try to build a map of the environment for navigation. In large facilities that are constantly reconfigured, the cost of map building and maintenance becomes too high, thus preventing the feasibility of this approach. SLAM for example is overcome in Amazon's Kiva robots by using an interconnect structure build on the floor. While feasible in new facilities, the cost of retrofitting existing facilities with new robots are also too immense.

Representation of the environment is a prerequisite for coordination and navigation of AGV in an indoor facility. A dynamic 2D map of the facility can be maintained by computer vision algorithms in real-time, provided that a picture of the whole facility is available to the algorithm whenever a change takes place in the environment. Change can be a complete reconfiguration of the facility, a human moving around, or objects lying on the way. Because a single camera cannot provide the entire view of a large facility, distributed cameras represent a straightforward way to collectively cover the complete facility. A distributed set of cameras such as those used in supermarkets can provide a partial view to a

[*]This work was supported in part by the national science foundation (NSF) under Grant 1547934.

ICDSC '16, September 12 - 15, 2016, Paris, France
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4786-0/16/09. . . $15.00
DOI: http://dx.doi.org/10.1145/2967413.2967425

central server where the 2D reconstruction of the entire view can be done followed by the construction and dynamic update of the 2D map. The problem with this approach is the bandwidth required to transfer all the pictures to the central server as well as the amount of computing power the server must provide for large facilities. Furthermore, the whole navigation is compromised with potential fatal damages in case of central server failure.

As mentioned before, our approach will use a set of cameras distributed across the facility to collaboratively build and maintain a 2D navigation map. However, the processing will be done in situ, within the camera, which will communicate over a middleware-layer in a Wi-Fi-network with neighbors for global coordination of the AGV. The main challenges described in this paper are:

1. Finding sound models and tools to capture the dynamic of actions and events in the scene.
2. Using a computing architecture for local high-performance-processing.
3. Defining related on-line embedded optimization along across the boundary of single cameras.

Our results are demonstrated on a testbed, tested in our lab with embedded cameras. We used an embedded camera based on the Raspberry Pi processing platform featuring a four-core Qualcom processor.

The remaining of the paper is organized as follows. Section 2 presents related work and shows the differences between them and our work. The overall infrastructure is presented in Section 3, along with the map extracting 3.1 and path finding algorithms 3.2. Subsection 3.3 discusses global routing one level of hierarchy above the local routing. Results of our first implementation are presented in the last Section 4 where the resulting image look like Fig. 5.

2. PREVIOUS WORK

In this section we review the related work pertaining to the different kinds of navigation approaches. If one considers that only vision based camera systems are eligible which are mounted on a ceiling, only few similar system will be found.

For instance, Amazon Kiva's robots require streets and highways mapped on the floor with bar code stickers and a central server to coordinate the robots [17]. Even though the robots can detect obstacles such as humans and tipped pods and stop, the grid is intended only for navigation, which makes them less suitable in environments that are quickly reconfigured by moving stations and objects around. Furthermore, the space requirement for the navigation of hundreds of AGVs in a production facility is very high compared to a system that does not require a dedicated track for navigation. Finally, Amazon's system makes sense only for new acquisition because of the special kind of robots used. Customers that already operate with hundreds of AGVs could not retrofit them for their solution.

Another system, which is already in use is patented by the Seegrid company. Their vision technology does not require environment modification. Their approach uses SLAM for creating a map of the environment. They are able to navigate from one point to the next one using a 3D collision avoidance [13]. Facility reconfiguration with changes in routes requires a new training, which cannot be done in real-time particularly for very large facilities.

The concept published by Reina, et al. [12] has in common with our idea the ceiling-mounted, or flying, cameras to monitor a whole area. Despite this, the setup of the robots is that they are not equipped with cameras and therefore do not navigate autonomously. Path planning and robot navigation is centralized at each camera, which tells the robots where to go. The authors use an "occupancy-map", where the area is divided in cells that can be marked if an object occupies them. Routing is done using Voronoi-Diagrams for path computation. This method is not fully described, for example the size of the cells are not defined, which can vary from square centimeters to square meters. The number of cells to cover makes the method computational intractable, at least in real-time. Besides the computational burden of this approach, it is as stated earlier, difference in infrastructure and operation: No camera is used on the robot, which are not autonomous. Obstacle avoidance is performed by the ceiling-mounted cameras. With dozens of robots in a field-of-view (FoV), no camera can provide the computational power to tackle obstacle detection and avoidance and provide guidance to the robots in real-time.

A distributed network of cameras to control robots is also described in [3]. The approach is similar to the previous one where cameras are still in charge of routing and navigation in their FoV. In an extension of this PhD-thesis, the author extends his method to include some low-level tasks to the robots such as low-level perception and control, which the robot needs anyway to move. It was not clear what the authors meant with low-level perception. However, we infer that the navigation is controlled and guide by the cameras. As in previous methods, the camera tracks all robots, which is not the case with our solution.

3. METHODS

The model proposed in this work is based on the representation of the environment in a mixture of hierarchical network and image understanding, as illustrated in Fig. 1. The whole indoor environment is covered by a set of smart cameras with overlapping FoVs (dashed circle, left), and formally captured by a camera connection graph (upper right) in which cameras represent nodes and overlapping regions between two nodes define edges connecting them. The low-level of hierarchy is defined within each graph node, which represents the FoV of a camera. At this level, navigation paths are represented in a sub-graph consisting of a set of junction-points interconnected with direct edges. The highest level connects the nodes to form the camera-relation graph. In the example of Fig. 1, a robot loaded in a zone covered by camera $C1$ is routed to a zone covered by camera $C6$ using the path $C1 \rightarrow C2 \rightarrow C5 \rightarrow C6$. Each camera is responsible to route the robot in the zone that it covers.

We assume that a control component exists for the robots with an interface that can be used by the computing module on the robots for low-level navigation instructions. Upon loading a robot, an operator provides the destination using a keypad on the robot or a remote-controlled input device. A LCD-Display on the robot will be used to display robot specific information such as robot ID, destination, battery level. LCDs are more flexible than traditional fixed codes such as QR-Code and Barcodes. Different destinations and configurations can be displayed in different colors and sizes

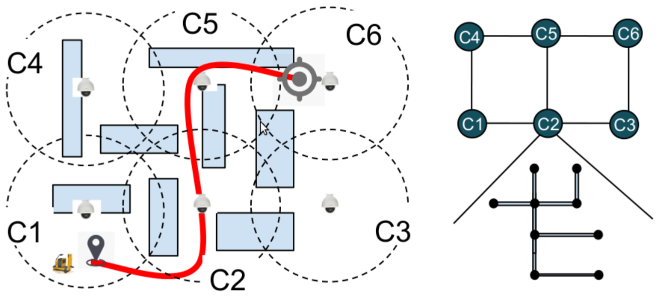

Figure 1: Proposed distributed coordination approach using a set of smart cameras with overlapping FoV and potential path to the destination. A hierarchical graph is used as data structure to capture the camera relationship at first level of hierarchy

without a reprint of the codes. Ceiling-mounted cameras will then read this information to route the robot to destination. Our main innovation in this research as explained in Section 1, is that the global view of the facility is dynamically captured by all cameras in a collaborative way. In the coverage zone of a camera, the pathways are maintained by that camera which coordinates the navigation in that area, in accordance with the routing objectives of the global network. A robot entering a zone covered by a camera receives a navigation plan for that zone, with the goal of minimizing the total routing time to the global destination. When a path becomes obstructed by people or objects the responsible camera evaluates if a re-routing is necessary (this will happen if a segment is so blocked that a robot cannot pass, even with obstacle avoidance), computes a new path if necessary, and provides a new plan to the corresponding robots. Finally, collision avoidance at path intersections in a zone covered by a camera is solved by that camera, since it has a broader view compared to the involved robots with only partial views.

In summary, we address the automatic path building and maintenance problem as well as the routing across cameras FoVs in the following way:

- Ceiling-mounted cameras with overlapping FoV for a gap-less coverage of the whole facility. Those cameras are pan-tilt-zoom, thus they can change their orientation for a better coverage in case of node failure.
- Cameras mounted on robots for immediate obstacle avoidance. This reduces the processing load of ceiling-mounted camera.
- Middleware which provides a seamless and transparent infrastructure to organize share data in the whole network.
- A central server, which will not be used for computation, but to provide configuration data, store global data structures, and if needed record videos.
- A set of locations where a robot can stop for loading and unloading purpose.

To support our research we have developed a prototype camera system that combines the tasks for map extracting path building and global routing. Our test environment is made by big LEGO-bricks, which help us to easy reshuffle our testbed. Small Robots simulate the traffic in our testbed. The robots receive the map upon entering the FoV a camera. For better performance the system runs the image processing tasks in parallel Fig. 2.

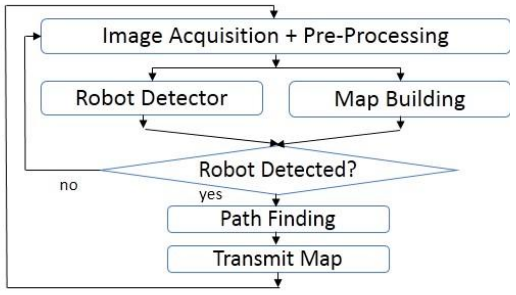

Figure 2: Flow of parallel tasks procedure.

3.1 Map Extraction

Map creation using multiple cameras can offer advantages over more reliable but expensive solutions like laser beams for navigation. For the image processing tasks in this work we considered stable light conditions, which is adoptable in indoor facilities. Guilherme et al. [5] differ between three broad groups of robot indoor navigation: First Map-Based Navigation, second Map-Building-Based Navigation and at last Mapless Navigation. We applied a Map-Building-Based Navigation approach, where we need to create a model of the environment. Therefore we were focused on finding a way to implement a fast map extracting algorithm to provide an "occupancy map" introduced by [9] for the robot. To reach this, we have to be able to dynamically produce and update the 2D map of a facility using a set of smart cameras distributed across the facility. The map must be updated in real-time whenever changes occur in the environment. Because of this, we implemented an efficient way based on mean shift color segmentation. The proposed algorithm consists the following steps:

1) Color Space Conversion: First, after receiving the RGB image from the camera we transform the image to CIELUV color space. CIELUV is an uniform color space that helps to reduce the computational power. As well as makes sure to obtain a meaningful segmentation by its linear addition properties in images with additive mixtures of light. The chrominance components u and v depend on the lightness value L, which can be calculated by its luminance Y. The dependence of all three coordinates of the traditional RGB color values is nonlinear, which would increase the processing time of our algorithm by 300 %.

2) Mean Shift Segmentation: In the second step, we run a mean shift color segmentation algorithm which is introduced by D. Comaniciu and P. Meer in [4]. The authors undertake high quality edge preserving filtering and image segmentation, obtained by applying the mean shift in the d-dimensional Euclidean space R^d. Comaniciu and Meer defined a vector with the components of a color space. Therefore the, algorithm estimates from a point x, the density gradients over this color space with the kernel density gradients usually called as the Parzen window technique [6, Section 4.3]. Kernel density estimator for an arbitrary set of n points $x_i, i = 1...n$, where the Kernel is $K(x)$ and the bandwidth parameter $h = (h_s, h_r)$ (Spatial-, Color-radius) is defined as [14, p. 76]

$$\hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K(\frac{x - x_i}{h}) \tag{1}$$

The estimate of the density gradient is defined as the gradient of the kernel density estimate:

$$\hat{\nabla} f(x) \equiv \hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} \nabla K(\frac{x - x_i}{h}) \tag{2}$$

If we set Eq. (2) to 0 and define $g(x) = -K'(x)$ we get

$$m(x) = \frac{\sum_{i=1}^{n} g(\frac{x - x_i}{h}) x_i}{\sum_{i=1}^{n} g(\frac{x - x_i}{h})} - x \tag{3}$$

Eq. (3) is called the Mean-shift-vector, which always points toward the direction of maximum increase in density. This means that the local mean is shifted toward the region in which the majority of the points reside, which smooth the image to a certain value. As a result we get clusters with the same density. However, we have selected the Mean Shift because it also takes into account the location of the pixels, which sharpens boundaries of color regions. Depending on the calibration of the camera and the given environment the bandwidth parameters h_s and h_r have to be setup at start-up of the system. After successful segmentation, we exchange every density segment with a different random color. The following picture shows a simple model how the color segmentation works:

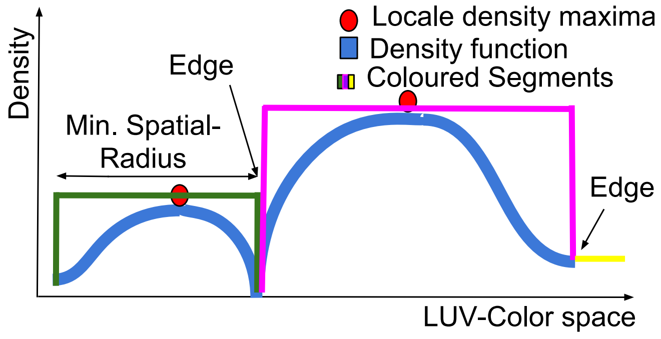

Figure 3: Rough representation of color segmentation with the Mean shift algorithm.

3) Binary Image Extraction: If we assume that the floor constitutes the largest proportion of our segmented image, a histogram calculation can be used to determine the color value of the floor. An "occupancy map" is mostly defined with the numeric value one for obstacles and zero for space or vice versa where the robot can operate. Therefore, we have to extract a binary image by setting all color values of the histogram maximum to zero and everything else to one. As a result we get a lightweight map for further path planning. The algorithm proposed in this paper is by reasons of simplicity and performance basically fulfilling all kinds of requirements of decentralized coordination, real-time environmental changes with smart embedded cameras.

3.2 Path Building

A Map-Building-Based Navigation requires not only a 2D "occupancy map" but also an reliable map how to navigate between the obstacles to reach in the shortest way to a specific destination. Path-finding is a well discussed real-world problem. We implemented following four common path searching algorithms:

- A* algorithm [8]
- Dijkstra's algorithm [15]
- Breadth-first search algorithm [19]
- Wavefront algorithm [11]

The extended 4-connected pixel Von Neumann Neighborhood with a Manhattan distance of $r = 2$ where used in our algorithms. Von Neumann used his neighborhood theory in the field of cellular automata [10] and can can be described on a given cell (x_0, y_0) of a distance r by:

$$N^v{}_{(x_0,y_0)} = \{(x, y : |x - x_0| + |y - y_0| \leq r)\} \tag{4}$$

Eq. (4) defined on a square grid, consequences in a diamond-shape. The algorithms were evaluated in terms of costs and performance. For our test-bench we used the image 4 with with two different resolutions. The resulting path is shown in Fig. 5. Every algorithm had the same start and goal point. We evaluated the algorithms on an ARMv7 Cortex-A53, 1.2GHz, rev 4 (v7l) processor on gcc version 4.8.4 -O3. The complexity / efficiency is expressed using the Bachmann-Landau notation, where $|V|$ is the number of vertices and $|E|$ is the number of edges in the graph.

Table 1: Comparison of path-finding algorithms.

Algorithm	Complexity	Reso.	Costs	Time (ms)
A* algorithm	$\mathcal{O}(\|E\|)$	640x480	611	608 (± 18.2)
	$\mathcal{O}(\|E\|)$	320x240	305	58 (± 1.7)
Wavefront	$\mathcal{O}(\|E\| + \|V\|)$	640x480	611	1778 (± 53.3)
	$\mathcal{O}(\|E\| + \|V\|)$	320x240	305	199 (± 6.0)
BFS	$\mathcal{O}(\|E\| + \|V\|)$	640x480	611	1943 (± 58.3)
	$\mathcal{O}(\|E\| + \|V\|)$	320x240	305	212 (± 6.4)
Dijkstra's	$\mathcal{O}(\|E\| + \|V\|log\|V\|)$	640x480	611	4875 (± 146.3)
	$\mathcal{O}(\|E\| + \|V\|log\|V\|)$	320x240	305	489 (± 14.7)

All algorithms found their goal with the same length / costs. Per contra the image size influence the execution time disproportionately high. The A*-algorithm is as expected the fastest. As it can be seen in table 1, A* has the best time complexity. A* is admissible and take into account fewer nodes than any other search algorithm with similar heuristic. Therefore, we selected this algorithm and concurrently set the image size as small as possible for our further research.

3.3 Global Routing

Describing a dynamic network amounts to capturing the distributed cameras on the ceiling, the navigation path within a camera region, and the moving robots. Similar to [18] our camera network is viewed as a 2-dimensional mesh topology wherein each node is identified by a 2-Tuple coordinate (c_x, c_y). Because of possible changes in the network structure at run-time, the functions, set and variables used to capture our problem must explicitly be time dependent.

At any time-stamp t, a ceiling-mounted camera is modeled relatively to its 4-connected Von Neumann Neighborhood explained in Eq. (4). Using atom $C^t(n, s, e, w)$, where $\{n, s, e, w\} \in \mathbb{N}^+$ are variables indicating here the North-, South-, East-, and Westside neighbors. For example, atom $C^0(4, 0, 2, 0)$ expresses the fact that at system start-up ($t = 0$), the current camera is only connected to node 4 at north and to node 2 at east.

The navigation path within a camera's FoV is captured as a directed graph with nodes as junction points and edges representing single track segments. We call this graph the navigation graph, one example is illustrated in Fig. 1 (lower right) to represent paths in the FoV of camera $C2$. The navigation graph is captured using atoms P for nodes and E for edges. While P_t^t represents a junction point k on this graph, $E^t(k, l, c)$ is used for modeling the connection between two junction points k and l with a cost c at time t. Cost of an edge, connecting two junction points is equal to the edge traversal time which will increase at run-time proportionally to its use to avoid congestion and enhance a balance utilization of the paths within a camera region.

A robot j is captured using atom $T_t^t(k, t_a, p, d_x, d_y)$, with k being the junction point at which the robot is located at time t, t_a the arrival time at junction point k, p the priority attached to the delivery task, and (d_x, d_y) the coordinate of the destination. The recognition process based on a common background (BG) subtraction task. According to [20] we decide that pixel vector $\vec{x}$ belongs to the background if $p(\vec{x} \mid \mathcal{X}, BG) > c_{thr}$, where we set the threshold c_{thr} to an empirical value of **24**. In addition to this, we used their Gaussian mixture model with a training data set $\mathcal{X}_T$ of **400** frames to get a background model defined by the first largest clusters B:

$$p(\vec{x} \mid \mathcal{X}, BG) \sum_{m=1}^{B} \hat{\pi}_m \mathcal{N}(\vec{x}; \hat{\vec{\mu}}_m, \sigma_m^2 I) \tag{5}$$

$\vec{x}$ is denoted as the value of a pixel in a color space. $\hat{\vec{\mu}}_m$ and σ_m^2 describe the Gaussian components estimation of mean and variance. The co-variance matrices are assumed to be diagonal and the identity matrix I has proper dimensions. The mixing weights denoted by $\hat{\pi}_m$.

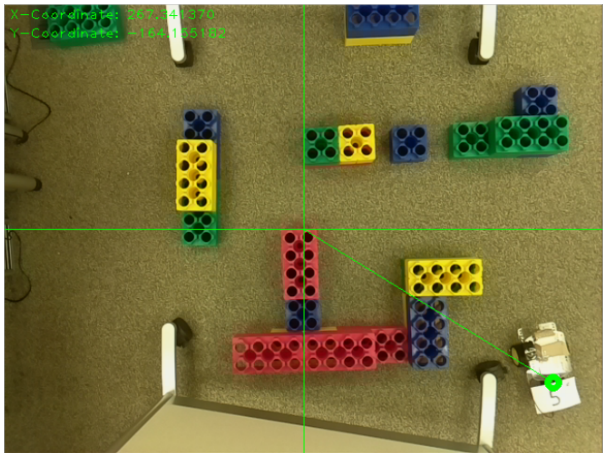

Figure 4: Test bench with robot who enters the FoV.

As soon as we know the position of the robot we can start with the inter-camera routing. The goal of inter-camera routing is to determine the next intermediate stop/hop of a moving robot j based on its final destination. Due to its simplicity, a XY-routing algorithm is leveraged for implementing point-to-point camera exchange in a 2-Dimension mesh topology network. In this algorithm, a packet is first routed on the X-axis until horizontal alignment and then vertically until it reaches its final destination. Whenever a new robot is captured at the entry point of a camera in the distributed system, its final destination (d_x, d_y) will be compared to a camera address (c_x, c_y). If both addresses match, then the robot will be considered as delivered. Otherwise, the horizontal coordinates (c_x and d_x) will first be compared and the robot will be routed to the east if $c_x < d_x$ or to the west if $c_x > d_x$ until horizontal alignment ($c_x = d_x$).

Prerequisite for a reliable routing is that, every camera is calibrated and knows its exact position relative to the other cameras. Camera failures are not considered yet. A middleware is implemented for Wi-Fi communication, independent of the underlying hardware. This connection makes use of TCP/IP-sockets to send the navigation data to the robot as well as their surrounding cameras. The robot translates the commands into movement orders, accessing the motors.

4. RESULTS

Performance analyses of our system is given in Table 2. The view of our camera is divided into six overlapping dynamic scenes as illustrated in Fig. 1, simulating a distributed camera network. In our test-case the robot gets detected via his unique ID by camera $C3$ and the tracked coordinates, gets the initial starting point for the path planning:

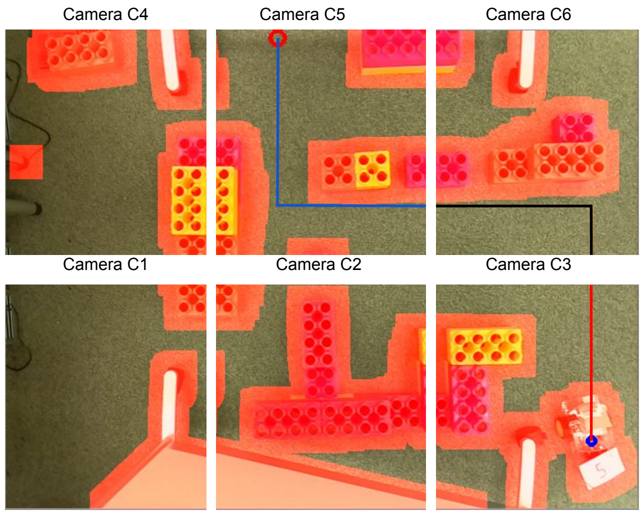

Figure 5: Distributed camera simulation with path from the tracked robot position (blue cycle) to the destination point (red cycle).

A map of the environment is already calculated when the Robot enters the FoV. Based on the given map and the tracked coordinates the camera computes necessary steps to send the robot to the next camera. Following by forwarding the path resulting from the robot's entry point and goal point toward the chosen camera nodes, to the robot. This is possible because the camera is aware of its position relative to the other cameras and uses regular dimension-ordering (XY-Routing). On the robot is a TCP/IP-Server listening to receive the navigation and orientation informations. Once the robot received the information, it translates the map into motor commands and continues along its way. Using this path planning the robot is able to navigate autonomously to his destination. At last knowing the next destination of a robot and the entry and exit points within the camera region, the robot drives to the exit, which applies to the minimized time-to-destination-cost path $C3 \rightarrow C6 \rightarrow C5$.

Table 2: Timing analysis of the system on architecture 3.2 with an image resolution of 320 x 240.

Task	**Time (ms)**	**In parallel**
Grab image + preprocessing	107 (± 3.2)	NO
Robot detector	40 (± 1.2)	YES
Map building + Path finding	1386 (± 41.6)	YES
Transmit Map	0.3 (± 0.01)	NO
Total	**1493 (± 2.1)**	**-**

5. CONCLUSIONS AND FUTURE WORK

We have presented a new method for vision based path construction and maintenance for indoor navigation of AGVs, based on collaborative smart cameras. The test-bench is purposeful to prototype a distributed camera system in a small area where the processing is done in situ, within the camera. Nevertheless, it cannot simulate the standards in huge facilities with hundreds of cameras and robots. For further concepts we define a real-time requirement of 3 ms. In other words, each camera has only 3 ms to assess the environment and guide the robots. In this time, it must perform image processing operations, path computation, obstacle detection and provide guidance to each AGV. For that reason the future work will concentrate to address the complexity of image processing tasks, in a HW/SW implementation, followed by a FPGA-based target platform, where low-level inherent parallel image processing tasks are mapped onto hardware while high-level reasoning is kept in software.

6. ACKNOWLEDGMENTS

The project is conducted as a joint effort between the University of Arkansas in Fayetteville and R-Dex in Atlanta. We thank Jörg Arndt, Nuremberg Institute of Technology Georg Simon Ohm for his expertise in this research.

7. REFERENCES

[1] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.

[2] R. Chatila and J.-P. Laumond. Position referencing and consistent world modeling for mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 138–145. IEEE, 1985.

[3] Y. Cheng. *Wireless mosaic eyes based robot path planning and control. Autonomous robot navigation using environment intelligence with distributed vision sensors.* PhD thesis, University of Bradford, 2010.

[4] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, 2002.

[5] G. N. DeSouza and A. C. Kak. Vision for mobile robot navigation: A survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(2):237–267, 2002.

[6] R. O. Duda, P. E. Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

[7] M. Fiala. Artag, a fiducial marker system using digital techniques. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 590–596. IEEE, 2005.

[8] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[9] H. P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 116–121. IEEE, 1985.

[10] J. v. Neumann and A. W. Burks. Theory of self-reproducing automata. 1966.

[11] A. Pal, R. Tiwari, and A. Shukla. A focused wave front algorithm for mobile robot path planning. In *Hybrid Artificial Intelligent Systems*, pages 190–197. Springer, 2011.

[12] A. Reina, L. Gambardella, M. Dorigo, and G. Di Caro. zeppelin: Ceiling camera networks for the distributed path planning of ground robots. Technical report, IRIDIA Technical Report Series, 2012.

[13] SeegridCorporation. *How Seegrid Vision Guided Vehicles (VGVs) Navigate at Giant Eagle Retail Support Center*, 2011 (accessed April 27, 2016).

[14] B. W. Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

[15] S. Skiena. Dijkstra's algorithm. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley*, pages 225–227, 1990.

[16] S. Waldherr, R. Romero, and S. Thrun. A gesture based interface for human-robot interaction. *Autonomous Robots*, 9(2):151–173, 2000.

[17] M. Wulfraat. *Is Kiva Systems a Good Fit for Your Distribution Center? An Unbiased Distribution Consultant Evaluation.*, 2012 (accessed April 27, 2016).

[18] W. Zhang, L. Hou, J. Wang, S. Geng, and W. Wu. Comparison research between xy and odd-even routing algorithm of a 2-dimension 3x3 mesh topology network-on-chip. In *Intelligent Systems, 2009. GCIS'09. WRI Global Congress on*, volume 3, pages 329–333. IEEE, 2009.

[19] R. Zhou and E. A. Hansen. Breadth-first heuristic search. *Artificial Intelligence*, 170(4):385–408, 2006.

[20] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th Int. Conf. on*, volume 2, pages 28–31. IEEE, 2004.