A method to produce minimal real time geometric representations of moving obstacles

David Sanders, Qian Wang, Nils Bausch, Ya Huang, Sergey Khaustov, Ivan Popov

School of Engineering

University of Portsmouth

Portsmouth, PO1 3DJ, UK

david.sanders@port.ac.uk, qian.wang@port.ac.uk, nils.bausch@port.ac.uk, ya.huang@port.ac.uk,

sergey.khaustov@port.ac.uk, ivan.popov@port.ac.uk

***Abstract* — Real time modelling methods are compared for use with a robot manufacturing work-cell and a simple image processing system. The static parts of a robotic manufacturing work-cell are modelled as a number of solid polyhedra. The robot is modelled as a number of connected spheres and cylinders. The static model is renewed when an object enters or leaves the static work-place. Simple polyhedra, spheres and similar 2-D slices in actuator space are compared with other models as representations of objects move in and out of the reach of the robot. Models are compared for their efficiency in accessing data and ability to update as information about moving objects changes. Geometric models of the robot and the robot work-cell are loaded into a path planner to compare the models for efficiency on planning paths around moving objects**

Keywords—model; robot; 2-D Slice; obstacle; path

I. INTRODUCTION (*Heading 1*)

Navigation through dynamic and changing industrial settings is demanding, especially if the movement of objects in the setting is not known or expected and must be updated in real-time. Computers are becoming quicker and more powerful but real-time functions still benefit from effective models and program techniques. Traditional global planning can sometimes be comparatively slow to react in complicated changing situations within complex and flexible manufacturing environments, whereas reactive navigation methods sometimes only look a short time and distance ahead. Some simple but quick real-time transformations are presented here that can provide simple and fast new geometric models of moving objects in an industrial robot work space. That can improve real-time trajectory and path planning.

Complicated industrial environments consist of machinery, objects being worked on and manipulated, and obstacles to avoid [1-5]. Available free space for moving machinery will depend on the modelling methods used in a varying situation and environment. Navigating industrial robots through a changing and dynamic environment is difficult [6], especially when the movement of objects is not known or expected in advance. Traditional planning methods can be slow and reactive navigation tends not to look far enough ahead. Vander-Stappen [7] described efficient methods to calculate an exact solution for motion planning problems if there are only a few moving objects but even he believed that the algorithms used for planning frequently used long and worst-case calculation and computing time [8]. Exact algorithms do not need to be used in real-time though. The processing time will depend on processing speed, problem complexity and how complicated the modelling algorithms are. Processing speed is still increasing but at the same time, the dynamic problems being addressed are becoming more complicated.

If some assumptions can be made about the shape, distribution and size of moving objects then the problem can be simplified. Complexity of free space increases linearly with the number of objects. De-Berg [9] considered how complicated motion planning problems were for bounded-reach robots with only a few objects around them and Tang [10] considered how to geometrically and topologically depict intersections between moving polygon shapes, noting that manufacturing environments where industrial robots are used can often contain polygon shaped objects. Large considered real-time planning based on Non-Linear Vobst [11]. If an environment could be modelled satisfactorily then the velocities that could cause a collision could be modeled.

Complexity of the modelling methods used directly affects planning algorithm complexity. The algorithms need to calculate a sequence of collision-free orientations left for an industrial robot when there are objects moving around it. Complexity results in comparatively lengthy computations and therefore processing times. If the complexity can be reduced then the computation time for motion-planning can also be reduced.

Real time models being used in diverse spaces are studied in this paper. The models are updated by a simple vision system. The work-cell for the robot is modelled by static solid polyhedra stored within computer memory. The static model is revised when objects enter or leave the robot workcell. Simple polyhedral, spheres and two-dimensional slices in an actuator space were compared as potential models for the moving objects. Spheres appear to be the simplest models for moving

objects. Halperin [12] explored ways of manipulating loosely connected spheres. The authors evaluated models based on spheres and identified the things that made them simple and efficient to use. The authors described effective algorithms that calculated places where the spheres touched.

In order to calculate a path, mapping needs to take place either from a workspace to a joint space (or configuration space) or from the joint space to a workspace. That is required so that the robot, the objects and the environment can be compared within a common space and so that static and moving objects can be avoided. Sanders studied geometric modelling in [1, 3-5] and decided that the following were important for representing a robot work cell with moving objects within it: quick and simple model creation, quick and efficient calculation of intersections between models and ease of use with algorithms that are planning movements. Sharma investigated the fastest time to transfer a vehicle from one place to another while avoiding other vehicles [13]. Other vehicles represented moving obstacles and a conflict was considered to have happened if the distance between vehicles was smaller than a safety distance (and that safety distance depended on their speed). Fiorini described a robot motion planning method within a dynamic environment that took place within common velocity space [14].

Models described here are assessed for their capacity to be revised when new data is available. For example, if an object enters, leaves or is moving within the robot work cell.

The structure of the robot is modelled as a set of connected spheres and cylinders and the range of motion of the robot is quantized. A quick but sub-optimal path is produced using simple models so that the robot can avoid objects as it moves towards a goal. The method inspects a 3-D table of quantized joint space.

Solid polyhedra are used to represent the static parts of the robot work-cell. As new objects enter or depart from the work cell then the available free space is updated. A six-sided parallelepiped, single spheres, multiple spheres and 2-D slices in joint space are employed to model moving objects. They are compared and some results are presented.

II. The Robot and the Static Environment

Industrial environments and robot work cells often consist of solid floors, walls and objects with linear straight edges and flat surfaces. Accurate models of these shapes are more difficult to create, change and use in real-time. If moving objects and robots are modelled using polyhedral shapes then although the accuracy might be higher, computation is more complicated and processing time is lengthened. A static environment only needs to be modelled once because it then does not change. Therefore computational complexity and processing time is not a problem. So, polyhedra can be used to accurately model a static environment.

Providing any model used to represent the industrial robot enclosed the whole of the industrial robot, then? the most important aspect of the model was the time taken to calculate intersections. Many industrial robots have links (often a forearm and upper arm) and joints (often base, shoulder and elbow). The simplest depiction of this type of industrial robot is two lines that are connected by a joint. If fixed distances are then defined forming the lines, then the volume can be considered to enclose the industrial robot casing. This model gives two cylinders with hemispherical ends that are connected by a joint. An advantage of this model is that robot links are efficiently modelled and calculations of intersections between objects and the robot arm are simplified. An end-effector can be modelled by an extra sphere at the end of one of the cylinders with a radius that is large enough to surround the whole of the end effector. Work-pieces can be encompassed within the sphere.

The time taken to calculate intersections was recorded for various models representing moving objects within the work cell of the industrial robot. Three of the models performed well: six-sided parallelepiped in cartesian space, spheres and 2-D slices in joint space. During the testing and comparison, it was presumed that the 2-D cross section of a moving object was known in the X-Y plane and that the length or height (Z) of the moving object could be provided by a sensor system (in this case a simple vision system was used). Moving objects were really two-and-a-half-dimensional. That is to say that they had a height and a 2-D shape. 3-D moving shapes that were considered in this research were cubes and cylinders. 2-D planar slices in joint space, spheres and parallelepiped models were used to model the 3-D moving shapes to a practical precision. In the specific case of the case of 2-D slices, they modelled the moving shapes more rapidly in a discrete three-dimensional space.

A. Modelling using 2-D slices

2-D slices were produced by creating two sets of boundaries: base joint angles ($\theta 1_{min}$ and $\theta 1_{max}$) that bound the object and the minimum distance D_{min} and the maximum distance D_{max} from the origin (the minimum and maximum radii). A moving object can then effectively be modelled as a sequence of 2-D slices. A reference planar slice is determined within the boundary created by a line from the Origin bounded by D_{min} and D_{max} and the Z axis value. These impeded elbow ($\theta 2$) and shoulder ($\theta 3$) joints can then be calculated for this plane and then copied for all $\theta 1$ within the bounding angles, $\theta 1_{min}$ and $\theta 1_{max}$.

That significantly reduced the number of tests and searches for blocked points. The planning algorithm is largely reduced by copying values rather than complicated mathematical calculations.

B. Transformations into joint space

A single point in a Cartesian space does not transform to be a single point in an actuator space. If a single point lies inside the operational volume of an industrial robot then it will transform into one or more complicated 3-D shapes. These complicated 3-D shapes may be represented by approximating the profiles of the shapes using mathematical curves to describe the geometric shapes, or they may be represented by units of space. In this research, moving objects were represented as regions of small units within a joint space. That technique would not be restricted to any particular robot and is suitable for a robot with any number of degrees of freedom.

A KUKA KR125 robot was used and the research considered the 3 major axes. A 3-D table was created and within the table, each dimension corresponded to one of the degrees of freedom for the industrial robot ($\theta 1$/ $\theta 2$ / $\theta 3$). The configuration of the wrist was not studied but the wrist and end effector could easily be included within an extra sphere.

Each table element was set to "free"' and orientations (in actuator space) where the industrial robot interconnected with the moving objects were determined. An element denoted a robot configuration in terms of, ($\theta 1_{center}$, $\theta 2_{center}$, $\theta 3_{center}$), with a value added to represent some movement away from the central values. All the units together embodied the whole work space for the industrial robot. The number of units in the table, TotalNodes, was:

$$\{(\theta 1_{max}-\theta 1_{min})/2 \times \delta\theta 1\} * \{(\theta 2_{max}-\theta 2_{min})/2 \times \delta\theta 2\} * \{(\theta 3_{max}-\theta 3_{min})/2*\delta\theta 3\} \quad (1)$$

where, $\theta 1_{max}$ and $\theta 1_{min}$ = upper and lower boundaries of $\theta 1$.
$\theta 2_{max}$ and $\theta 2_{min}$ = upper and lower boundaries of $\theta 2$
$\theta 3_{max}$ ad $\theta 3_{min}$ = upper and lower boundaries of $\theta 3$.

If the body of the industrial robot intersected a moving object at any configuration in a unit, then that unit was set to "obstructed". If the industrial robot did not intersect a moving object at all configurations within a unit then the unit continued to be set to "free". The global path planning problem then reduced to searching for a series of neighboring units between a BEGINNING and an END configuration that were "free". Free space tends to be bigger than obstructed space so that a quick and efficient method was to test each transformed moving object and for nodes which could contain the robot. This method was adopted. The algorithm was:

Create the Static Model (only done once):

Initialise data structures to create a 3-D table to represent joint space.

Calculate trigonometric solutions.

Set all units in the table to `"free" status.

Associate flags with each node.

After calculating the static model there were a large number of free elements remaining that represented a set of configurations where the industrial robot wouldn't bump into the static environment (the work cell).
Data about moving objects was then either simulated or provided by the vision system.

Repeatedly…

Read the data about moving objects.

Create the two and a half dimensional model.

For an element where the industrial robot could intersect with a moving object

Recursively test elements next to the original element to decide whether they can also be reached by the industrial robot.

III. 2-D SLICES

To begin with, the limits in X were increased by the upper-arm radius:

$$StartofRow\_clear = StartofRow - UpperlimRad \quad (2)$$

$$EndofRow\_clear = EndofRow + UpperlimRad \quad (3)$$

The center point modulus and the ends of an edge StartofCol were determined.

$$(TopLeftcorner, Ang) = InvTan (StartofCol / EndofRow\_clear) \quad (4)$$

$$Corner(TopLeft, Modulus) = \sqrt{(StartCol^2 + EndRow^2)} \quad (5)$$

The subsequent model parameters were calculated: largest base angle, ($\theta 1_{max}$), smallest base angle, ($\theta 1_{min}$), outside radius from origin, (D_{max}), and inside radius from origin, (D_{min}),

If a moving object corresponded to a known template then a third dimension (a height Z) was obtained for the moving object from a template, otherwise height was set to infinity. That segment was then extrapolated on the Y axes and calculation only took place in a Z, Y plane. The model of the moving object was expanded by the upper-arm radius in the Z and Y plane. $\theta 1$ was changed to a new lower limit and an inverse kinematic solution was calculated for all points within the moving objectfor the Z and Y axis:

```
FOR Y = (MinUpperRad) TO (MaxRadius + UpperRadius)
  FOR Z = 255 TO (Rad(Z) + UpperRadius)
     CALL InvKin
  NEXT Z
NEXT Y
```

Coordinates in Z and Y were translated into robot joint angles with an inverse kinematic solution subroutine called InvKin. Distance from the cartesian point (L3) to the origin and the angle to a point (Pointθ) were calculated:

$$Point\theta = Inv\_Tan\ Z / Y \quad (6)$$

$$sqofL3 = Y^2 + Z^2 \quad (7)$$

$$L3 = \sqrt{sqofL3} \quad (8)$$

L3 was checked against the upper-arm to test whether an impact was conceivable. If within reach of upper-arm and if $\theta 2$ was within limits, then $\theta 3$ was set to "blocked" between its limits and $\theta 2$ was set to Curveθ and. If L3 was less than Forearm plus Upper-arm then the Forearm could collide with that point. $\theta 3$ and $\theta 2$ were computed using the cosine rule. If $\theta 3$ and $\theta 2$ were within limits then a flag was set to "obstructed".

2-D slice models have been used with sensor systems [15-16] mobile robots [17-21] and powered wheelchairs [22-23].

IV. SPHERES

The table of data mentioned earlier was initialized. The limits of the table corresponded to the limits for the angles of the robot joints inside the work cell. Objects outside the work cell were disregarded. So, the table was used to conduct checks for intersections at a limited number of orientation and positions. That meant that a smaller number of trigonometric solutions was needed and they could be calculated at the beginning when calculation time did not matter. Before the objects were determined, all the elements in the table were set to `EMPTY'. Another 4 flags were associated with each table element: `ON LIST', `UPPERARM TESTED', `FOREARM TESTED', and `NEW OBJECT'. The code representing each element was kept as a byte of data in an array and the five flags each used one bit. Object data was extracted from a file or sent from the vision system and the initial job for the program was to read this data.

A sub-task then calculated the upperarm and forearm blocked space in the table. A configuration was calculated at which the part of the arm being considered was nearest to the center of the object. If a forearm was being studied, then a configuration where the Foretip was at the sphere center was computed. For the upperarm, a configuration was calculated for which the center line of the upperarm pointed at the center of the sphere. If the object was within reach of the link being examined, then this configuration was the first unit for the transformed object. A base angle was computed from the X,Y coordinates for the sphere. First, the modulus (L3) and the angle (Sphereθ) from the robot to the center of the sphere was computed and a check was performed to find whether the sphere was out of the range of the robot.

$$\text{Waist}\theta = \theta 1 = \text{InverseTan}(Y/X) \quad (9)$$

$$\text{ModulusXY} = \sqrt{(X^2+Y^2)} \quad (10)$$

$$\text{Sphere}\theta = \text{InverseTan}(Z/\text{ModulusXY}) \quad (11)$$

$$L3 = \sqrt{(X^2 + Y^2 + Z^2)} \quad (12)$$

Shoulder and elbow angles (θ2 and θ3) were calculated using τhe cosine rule.

Upper-Arm length = L1 = 225mm and ForeArm = L2 - 165mm

$$\theta 3 = \text{InverseCos}\,[(L1^2 + L2^2 - L3^2) / (2*L1*L2)] \quad (13)$$

$$\theta 2 = \text{InversCos}\,[(L1^2+L3^2-L2^2) /(2*L1*L3)] + \text{Sphere}\theta \quad (14)$$

If the center of the sphere was close to the robot then θ3 would exceed its lower limit (θ3 < 90°). In this case, θ3 was set to 90° and θ2 was calculated using InversTan:

If θ3 < 90° THEN
θ3 = 90°
θ2 = InversTan (L2 / L1) + Sphereθ
END

This provided a first configuration that was near the center. When the lower boundary of θ2 was surpassed, (θ2 < -30°), then the angle was set to -30°. A calculation was then made to find the distance between the center of the sphere and the upperarm (Mod) using FindtheModulus. The cosine rule could be used from that to find a new θ3:

If -30° < θ2 THEN
θ2 = -30°
θ3 = InversCos[(L1²+L2²-Mod²) / (2*L1*Mod)]
END

The table element representing the first configuration was set to obstructed. Neighboring elements were then tested. If they were also obstructed then their neighbors were tested. The position problem was solved using forward kinematic calculations and the minimum distance between the arm of the robot and the object (as long as that calculation had not been performed before). This continued recursively until all of the object had been transformed. Elements were set to "obstructed" if they had any two opposite neighboring elements that were also obstructed. Any elements at the edge of the now solid model were added to a record. Neighboring elements that were recorded were tested and that was repeated until the surface of the transformed sphere was fully outlined.

Obstructed elements were kept so that they could be expanded later on. When an element was expanded then it was retrieved from the record store and new obstructed points were added. When every element had been considered then the transformation of the object was completed. The time taken to calculate objects was recorded and some examples are given in Section VI of this paper.

V. A SIMPLE POLYHEDRAL SHAPE

Polyhedra have often been used to model objects and a simple polyhedral model was used here. Moving objects were modelled as simple six-sided-parallelepiped. The edges of the model in X and Y were obtained by calculating the limits of the columns and rows set by the vision system. As before, if it was possible then the height of an object could be retrieved from a template. Edge positions were expanded by the radius of the part of the robot being tested (for example the forearm or upperarm). The example of expanding the forearm in X is shown here:

$$\text{ExpandXLow} = \text{EdgePos(LowX)} - \text{ForRad} \quad (15)$$

$$\text{ExpandXHigh} = \text{EdgePos(HighX)} + \text{ForRad} \quad (16)$$

The expanded polyhedral edge limits were then tested against the cartesian coordinates of the arm.

VI. RESULTS

The time taken to transform objects was recorded.

Time taken for the various models to transform a large cube into an actuator space are displayed within Table 1.

TABLE I. TRANSFORMATION TIMES FOR A CUBE

A. Static environment

The system needs the time taken to convert moving objects to be short. It can take a long time to model a static environment but this only needs to be done once at the start.

In this work that conversion took up to a minute of computer time depending on the complexity of the static environment.

B. 2-D Similar Slices

Model	Time (ms)	Blocked elements.
Single Sphere	18.7	2425
2 x Spheres	94.2	2335
Simple Polyhedra	126.2	1982
2-D Similar Slices	15.2	2488

An advantage of this model was that once collision coordinates for θ3 and θ2 were computed for a specific θ1 then the collisions could be repeated for all specific θ1 that might collide. This downgraded the principal processing job to a copy function rather than having to calculate a reverse or a forward kinematic solution. This model was the quickest to transform into a discrete 3-D actuator space.

If a moving object enlarged beyond a certain size or if at relocated nearer the origin, then the moving object could intersect both the Forearm and Upperarm. So, the actuator space occupied by the moving object will abruptly increase and so calculation time will increase.

C. Sphere Model

Initially, the objects were modelled as individual spheres with the smallest radius that enclosed the object.

If there was time then the object was re-modelled by 2 smaller spheres and then four smaller spheres etc. Elements set to obstructed that were related to the first sphere would usually also collided with the models using multiple spheres. Forward kinematic solutions did not need to be recalculated for these elements. Despite that, total processing time increased with the number of spheres. That was because the overhead involved in calculating intersections for each sphere was more than any time that was saved because the spheres were smaller. So, calculations with a single sphere were quicker than multiple spheres despite the fact that the single sphere had a larger volume and was less accurate.

A problem with using more than one sphere was that the center of several spheres could be set to "blocked" (with some surrounding elements) after the first sphere was expanded. As these elements were obstructed, later spheres might not retest and some nodes might not be recorded.

D. Simple Polyhedra Model

Although this was the most accurate, it took the longest to calculate.

VII. DISCUSSION

Higher accuracy models tended to need more processing time and took longer to reach a solution. Lower accuracy models needed robot links or moving objects to be artificially increased in size to remove the risk of unnoticed collisions. Lowering accuracy led to rejecting some potentially useable solutions.

The speed of calculation was important for moving objects and their models. The easiest possible intersection calculation involved spherical models. Computation was slashed to only calculating distances from a line (representing a robot link) to a point (the center of a sphere) and then subtracting the sphere radius to produce a distance from the robot to the sphere surface.

The use of more than one sphere was studied. As the robot working environment became more complicated then more spheres could model it more accurately. Increasing the number of spheres increased the accuracy of a model. Cubic numbers of spheres were considered, that is 1 / 8 / 27 / 64 etc. If they were of equal size then they formed regular lattice patterns. An infinite number of spheres could model a cube exactly but it was found that modelling objects using spheres of equal size was not the most efficient way of doing it. For example, if a cube is modelled using sixty-four equally sized spheres, eight of those spheres are completely enclosed. Those eight could be replaced by one larger sphere and that would not reduce accuracy or increase the volume of the model.

To compare modelling using single and multiple spheres, as an extra example, a model of a cylinder using one and two spheres was compared. The volume of two spheres of radii 35 mm was compared to that of one sphere of 70 mm:

Volume of the two spheres: $2 \times 4/3 \times \pi \times 353 = 359{,}188$ mm3

Volume of the single sphere: $4/3 \times \pi \times 703 = 1{,}436{,}755$ mm3

The area of the two spheres would be much smaller except that the model of the robot must then be considered to find the union volume:

$$\text{Model} \cup \text{Robot} \tag{17}$$

The Upperarm model radius= 85mm: Union radius for a single sphere was 70+85 = 155. Union radius for two spheres is 35+85 = 120.

Union volume of a single sphere is $4/3 \times \pi \times 1503 = 14{,}137{,}167$ mm3. Union volume of two spheres is $2 \times 4/3 \times \pi \times 1153 = 12{,}741{,}211$ mm3.

A similar number of obstructions was recorded for both models. If elements in a second sphere were not tested for obstruction during calculations for a previous sphere, then that would change the number calculated. That partially explains a lack of reduction in calculation times when using two spheres to model objects.

The simple six-sided parallelepiped had a volume that was less than that of the 2-D slice model. The volume for the Parallelepiped was:

Volume = $(60+160)^2 \times (140+80) = 10{,}648{,}000$ mm3.

This could reduce the number of obstructed elements, but shape and therefore the associated computations would be more complicated. So, the calculation time would increase.

VIII. CONCLUSIONS

Angles for θ3 and θ2 were only calculated for a single slice so that processing time was reduced because that slice of

"obstructed" elements was just then copied for all the base angles that might intersect the object. Copying is quicker than calculating and so this method appears to be faster than methods requiring real-time calculations.

The number of "obstructed" elements found was similar for all the models. Intersection volumes were similar and that implied a corresponding accuracy.

Modelling using similar 2-D slices produced the quickest intersection calculations. The 2-D slices were simpler than polyhedra, as the model only required two bounding angles for the base joint (an inner and outer radius) and a height.

The model using similar 2-D slices were the most effective of the models considered. 2-D slices in joint space represented moving objects at least as well as the other models but they were quicker.

REFERENCES

[1] D.A Sanders. "Recognizing shipbuilding parts using artificial neural networks and Fourier descriptors". Proc' Institution of Mechanical Engineers Part B-Journal of Eng Man 223 (3) pp: 337-342 (2009).

[2] Z. Rasol and D.A. Sanders. An automatic system for simple spot welding tasks. Total Vehicle Technology Conf, pp: 263-272 (2001).

[3] D. A. Sanders and P. Harris, "Image modelling for real time manufacturing applications using 2-D slices in joint space and simple polyhedra," Journal of Design and Manufacturing 3, pp: 21 - 27 (1993).

[4] D.A. Sanders. "Real time geometric modelling using models in an actuator space and cartessian space". Journal of Robotic Systems. 12 (1). pp: 19-28 (1995).

[5] D.A. Sanders, G Lambert and L. Pevy. "Pre-locating corners in images in order to improve the extraction of Fourier descriptors and subsequent recognition of shipbuilding parts. Proc' Institution of Mechanical Engineers Part B-Journal of Eng Man 223 (9). pp: 1217-1223 (2009).

[6] S. Carpin, " Randomized motion planning: A tutorial ", Int Jrnl of Robotics & Automation 21 (3): pp 184-196 (2006).

[7] R-P. Berretty, M.H. Overmars and A.F. van der Stappen, "Dynamic motion planning in low obstacle density environments", Computational Geometry 11(3–4), pp 157–173 (1998).

[8] D. A. Sanders, A. Moore and B. L. Luk. "A Joint Space Technique for Real Time Robot Path Planning". Robots in Unstructured Environments, IEEE 91TH376-4, pp: 1683 - 1689. (ISBN 0-7803-0078-5. (1991).

[9] M. de Berga, M.J. Katzb, , M.H. Overmarsa, A.F van der Stappena, and J. Vleugels. "Models and motion planning", Computational Geometry 23 (1), pp 53–68 (2002).

[10] K. Tang, "A geometric method for determining intersection relations between a movable convex object and a set of planar polygons", IEEE Transactions on Robotics 20 (4) , pp 636 – 650 (2004).

[11] F.Large, C. Laugier and Z. Shiller, "Navigation Among Moving Obstacles Using the NLVO: Principles and Applications to Intelligent Vehicles", Autonomous Robots 19 (2), pp: 159 – 171 (2005).

[12] D. Halperin and M.H Overmars, "Spheres, molecules, and hidden surface removal ", Proc' 10th annual symp' on Computational Geometry, pp: 113-122 (1994).

[13] V. Sharma, M. Savchenko, E Frazzoli, and P.G. Voulgaris, "Transfer time complexity of conflict-free vehicle routing with no communications", International Journal of Robotics Research 26, pp 255-271 (2007).

[14] P. Fiorini, "Robot motion planning among moving obstacles", PhD disertation, University of California, (1995).

[15] D.A. Sanders. "Environmental sensors and networks of sensors". Sensor Review Volume: 28 Issue: 4, pp: 273-274. (2008).

[16] D.A. Sanders, G. Lambert, J. Graham-Jones, J; et al. "A robotic welding system using image processing techniques and a CAD model to provide information to a multi-intelligent decision module". Assembly Automation 30 (4), pp: 323-332 (2010).

[17] D.A Sanders. "Comparing ability to complete simple tele-operated rescue or maintenance mobile-robot tasks with and without a sensor system. Sensor Review 30 (1), pp: 40-50 (2010).

[18] D.A Sanders. "Comparing speed to complete progressively more difficult mobile robot paths between human tele-operators and humans with sensor-systems to assist". Assembly Automation 29 (3), pp: 230-248 (2009).

[19] [19] D.A Sanders, J. Graham-Jones, and A. Gegov. "Improving ability of tele-operators to complete progressively more difficult mobile robot paths using simple expert systems and ultrasonic sensors". Industrial Robot 37 (5). Pp: 431-440 (2010).

[20] D.A Sanders, I.J. Stott, D.C. Robinson et al. "Analysis of successes and failures with a tele-operated mobile robot in various modes of operation". Robotica 30, pp: 973-988 (2012).

[21] [21] D.A Sanders, G.E. Tewkesbury, I.J.Stott et al. "Simple expert systems to improve an ultrasonic sensor-system for a tele-operated mobile-robot". Sensor Review 31 (3), pp: 246-260 (2011).

[22] D.A Sanders, M. Langner and G.E. Tewkesbury. "Improving wheelchair-driving using a sensor system to control wheelchair-veer and variable-switches as an alternative to digital-switches or joysticks". Industrial Robot 37 (2), pp: 157-167. (2010).

[23] D.A Sanders, I.J. Stott and J Graham-Jones. "Expert system to interpret hand tremor and provide joystick position signals for powered wheelchairs with ultrasonic sensor systems". Industrial Robot 38 (6), pp: 585-598. (2011).