Biosystems Engineering (2003) **86** (2), 135–144
doi:10.1016/S1537-5110(03)00133-8
AE—Automation and Emerging Technologies

Collision-free Motion Planning for a Cucumber Picking Robot

E.J. Van Henten; J. Hemming; B.A.J. Van Tuijl; J.G. Kornet; J. Bontsema

Department of Greenhouse Engineering, Institute of Agricultural and Environmental Engineering (IMAG B.V.), P.O. Box 43, NL-6700 AA Wageningen, The Netherlands; e-mail of corresponding author: eldert.vanhenten@wur.nl

(Received 26 April 2002; accepted in revised form 8 July 2003; published online 29 August 2003)

One of the most challenging aspects of the development, at the Institute of Agricultural and Environmental Engineering (IMAG B.V.), of an automatic harvesting machine for cucumbers was to achieve a fast and accurate eye–hand co-ordination during the picking operation. This paper presents a procedure developed for the cucumber harvesting robot to pursue this objective. The procedure contains two main components. First of all acquisition of sensory information about the working environment of the robot and, secondly, a program to generate collision-free manipulator motions to direct the end-effector to and from the cucumber. This paper elaborates on the latter. Collision-free manipulator motions were generated with a so-called path search algorithm. In this research the A*-search algorithm was used. With some numerical examples the search procedure is illustrated and analysed in view of application to cucumber harvesting. It is concluded that collision-free motions can be calculated for the seven degrees-of-freedom manipulator used in the cucumber picking device. The A*-search algorithm is easy to implement and robust. This algorithm either produces a solution or stops when a solution cannot be found. This favourable property, however, makes the algorithm prohibitively slow. The results showed that the algorithm does not include much intelligence in the search procedure. It is concluded that to meet the required 10 s for a single harvest cycle, further research is needed to find fast algorithms that produce solutions using as much information about the particular structure of the problem as possible and give a clear message if such a solution can not be found.

1. Introduction

In 1996, the Institute of Agricultural and Environmental Engineering (IMAG B.V.) began research on the development of an autonomous cucumber harvesting robot supported by the Dutch Ministry of Agriculture, Food and Fishery (Gieling *et al.*, 1996; Van Kollenburg-Crisan *et al.*, 1997). The task of designing robots for agricultural applications raises issues not encountered in other industries. The robot has to operate in a highly unstructured environment in which no two scenes are the same. Both crop and fruit are prone to mechanical damage and should be handled with care. The robot has to operate under adverse climatic conditions, such as high relative humidity and temperature as well as changing light conditions. Finally, to be cost effective, the robot needs to meet high performance characteristics in terms of speed and success rate of the picking operation. In this project, these challenging issues have been tackled by an interdisciplinary approach in which mechanical engineering, sensor technology (computer vision), systems and control engineering, electronics, software engineering, logistics, and, last but not least, horticultural engineering partake (Van Kollenburg-Crisan *et al.*, 1997; Bontsema *et al.*, 1999; Meuleman *et al.*, 2000).

One of the most challenging aspects of the development of an automatic harvesting machine is to achieve a fast and accurate eye–hand co-ordination, *i.e.* to achieve an effective interplay between sensory information acquisition and robot motion control during the picking operation, just like people do. In horticultural practice, a trained worker needs only 3–6 s to pick and store a single fruit and that performance is hard to beat. Fortunately, in terms of picking speed a robotic harvester does not have to achieve such high performance characteristics. A task analysis revealed that, for economic feasibility, a single harvest operation may take

Notation

c	transition costs between two nodes
f	cost function
g	cost of the motion path from node S to the current node
G	goal node
h	optimistic estimate of the cost to go to the goal from the current node
n	node
n'	successor of node
S	start node

up to 10 s (Bontsema *et al.*, 1999). Still, robot motions should be as fast as possible while preventing collisions of the manipulator, end-effector and harvested fruit with the crop, the greenhouse construction and the robot itself (such as the vehicle and vision system). In a Dutch cucumber production facility, the robot operates in a very tight working environment. Finally, to assure the quality of the harvested fruit, constraints have to be imposed on the travelling speed and accelerations of the manipulator during various portions of the motion path.

To achieve the desired eye–hand co-ordination, one needs (real-time) acquisition of sensory information of the environment as well as algorithms to calculate collision-free motions for the manipulator. In this project, the sensory system is based on computer vision, as reported by Meuleman *et al.* (2000). This paper focuses on the fast generation of collision-free motion trajectories for the manipulator of the picking machine. This issue has not received much attention in agricultural engineering research despite the fact that considerable research effort has been spent on automatic harvesting of vegetable fruits (see *e.g.* Kondo *et al.*, 1996; Hayashi & Sakaue, 1996; Arima & Kondo, 1999).

The paper is outlined as follows. In Section 2, the harvesting robot is described. In Section 3, a task sequence for a single harvest operation is presented. Then Section 4 presents the components of an automatic algorithm for collision-free motion planning. To obtain insight into the operation of the algorithm, in Section 5, the approach is illustrated on a two-degrees-of-freedom (DOF) manipulator. Section 6 contains results of a motion planning experiment with the six-DOF RV-E2 Mitsubishi manipulator used in the harvest robot. Finally, Section 7 contains concluding remarks and suggestions for future research.

2. The harvesting robot

In *Fig. 1* a functional model of the harvesting robot is shown. It consists of an autonomous vehicle used for coarse positioning of the harvesting machine in the aisles of the greenhouse. This vehicle uses the heating pipes as a rail for guidance and support. It serves as a mobile platform for carrying power supplies, a pneumatic pump, various electronic hardware for data-acquisition and control, a wide-angle camera system for detection and localisation of cucumbers in the crop and a seven-DOF manipulator for positioning of the end-effector. The manipulator consists of a linear slide on top of which a six-DOF Mitsubishi RV-E2 manipulator is mounted. The RV-E2 manipulator includes an anthropomorphic arm and a spherical wrist. The manipulator has a steady-state accuracy of $\pm 0{\cdot}2$ mm and meets general requirements with respect to hygiene and the operation under adverse greenhouse climate conditions (high relative humidity and high temperature). The manipulator carries an end-effector that contains two parts: a gripper to grasp the fruit and a cutting device to separate the fruit from the plant. The end-effector carries a laser ranging system or a small camera. They are used to obtain sensory information for fine motion control of the end-effector in the neighbourhood of the cucumber, if needed.

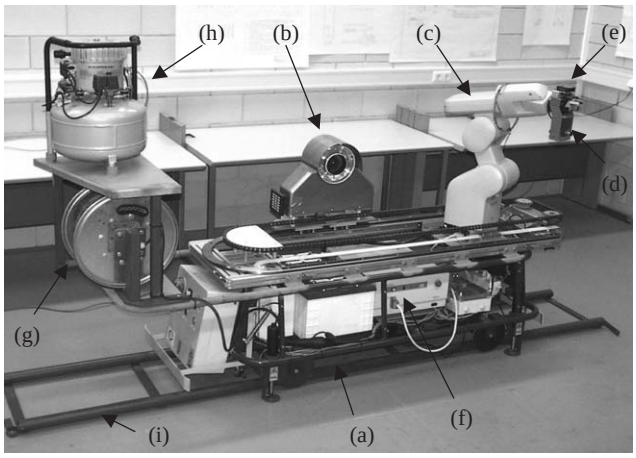

Fig. 1. A functional model of the cucumber harvest robot; (a) vehicle; (b) wide angle camera; (c) seven-degrees-of-freedom manipulator; (d) end-effector; (e) laser ranger and position of camera for local imaging; (f) computer and electronics; (g) reel with 220 V power line; (h) pneumatic pump; (i) heating pipe

3. Task sequence of a single harvest operation

A task sequence of a single harvest operation is presented in *Fig. 2*. Approaching the cucumber during the picking operation is considered to be a two-stage process. First, with the camera system mounted on the vehicle, the cucumber fruit is detected, its ripeness is assessed and its location is determined. In case it is

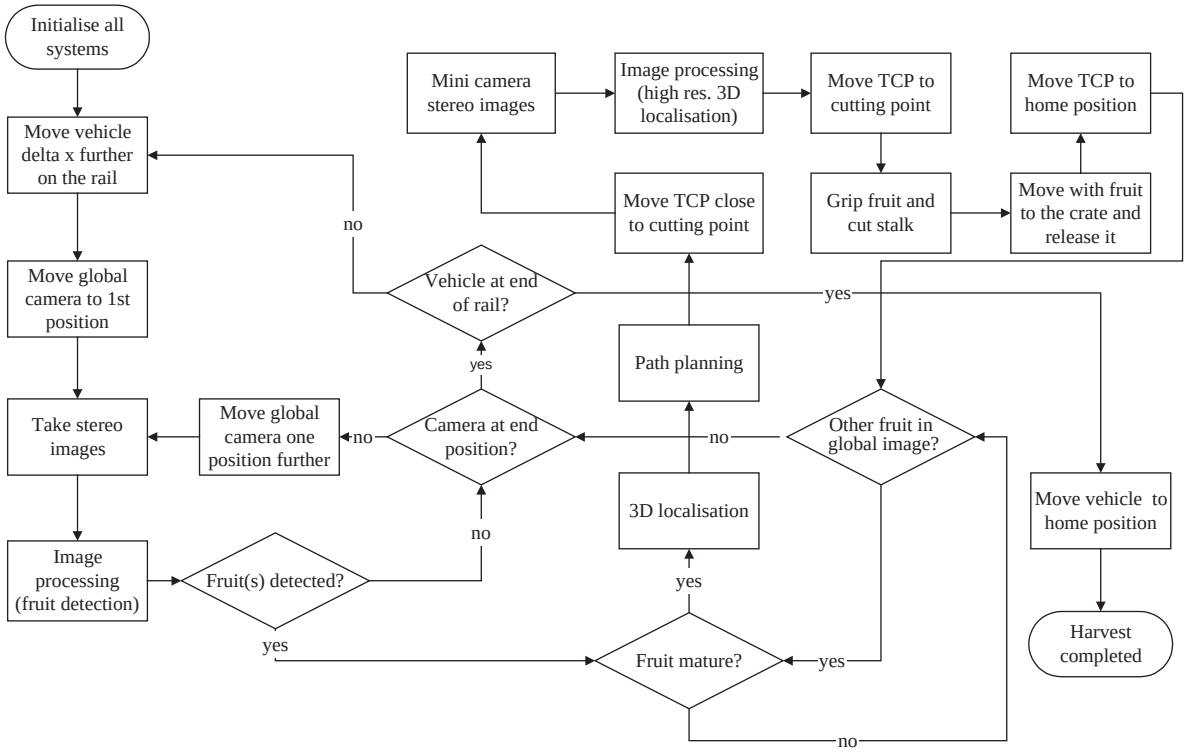

Fig. 2. Task sequence of a single harvest operation: 3D, three dimensional; TCP, tool-centre-point

decided to pick the cucumber, the low-resolution images of the vehicle-mounted camera are used for positioning the end-effector in the neighbourhood of the cucumber. Once the end-effector has arrived in the neighbourhood of the cucumber, then, using the laser ranging system or camera system mounted on top of the end-effector, high-resolution information of the local environment of the cucumber is obtained for the final accurate approach of the cucumber. The end-effector grips and cuts the stalk of the fruit. The gripper fixes the detached fruit and finally the harvested fruit is moved to the storage crate.

Obstacle avoidance motion planning will be used both for the initial approach of the cucumber as well as the return journey of the harvested cucumber to the crate, to assure that other objects in the work space such as the robot vehicle itself but also stems and, if present, leaves and parts of the greenhouse construction are not hit. Clearly, the harvested cucumber increases the size of the end-effector, which should be considered during the return motion of the manipulator to the storage. The average length of a cucumber is 300 mm.

4. A collision-free motion planning algorithm

Figure 3 illustrates the components of a program that automatically generates collision-free motions for the cucumber picking robot based on the work of Herman (1986). Collision-free motion planning relies on three-dimensional (3D) information about the physical structure of the robot as well as the workspace in which the robot has to operate. So, the first step in collision-free robot motion planning is the 3D *world description acquisition*. This description is based on sensory information such as machine vision as well as *a priori* knowledge about, for instance, the 3D kinematic structure of the harvesting robot, *e.g.* 3D models, contained in a database. With this information, during the *task definition* phase, the overall task of the robot is planned. It is decided which final position and orientation of the end-effector result in the best approach of the cucumber. Also specific position and orientation constraints are defined during this phase. In the *inverse kinematics* phase the goal position and orientation of the end-effector, defined during *task definition*, are translated into the goal configuration of the manipulator. The goal configuration is represented as a combination of a translation of the linear slide and six rotations of the joints of the seven-DOF manipulator. This information is used by the *path planner*. The *path planner* employs a search technique to find a collision-free path from the start configuration of the manipulator to its goal configuration. Once the collision-free path planning has been completed successfully, the *trajectory planner*

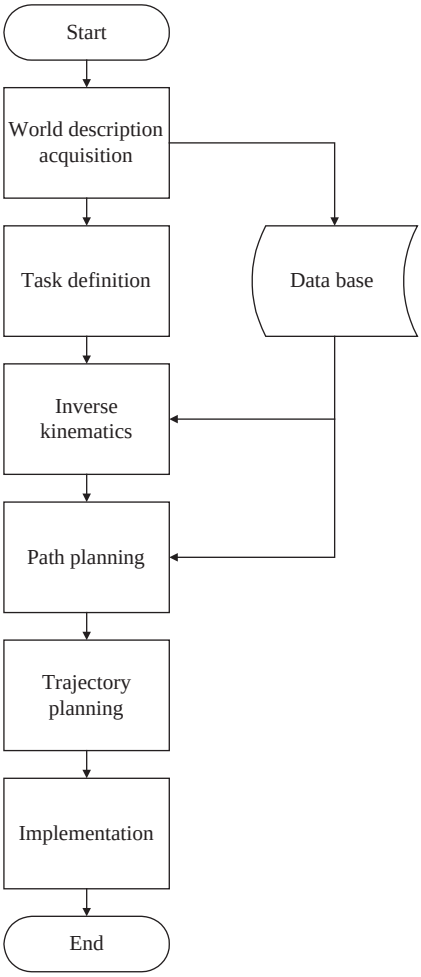

Fig. 3. A program for automatic generation of collision-free motions

converts the collision-free path into a trajectory that can be executed by the manipulator. Typically, the path planning process is concerned only with collision-free configurations in space, but not with velocity, acceleration and smoothness of motion. The *trajectory planner* deals with these factors. The *trajectory planner* produces the motion commands for the servomechanisms of the robot. These commands are executed during the *implementation* phase.

Some components of the motion planning system will be described hereafter in more detail.

4.1. *World description (acquisition)*

The machine vision based world description acquisition for the cucumber picking robot is described in a paper by Meuleman *et al.* (2000). The vision system is able to detect green cucumbers within a green canopy. Moreover, the vision system determines the ripeness of

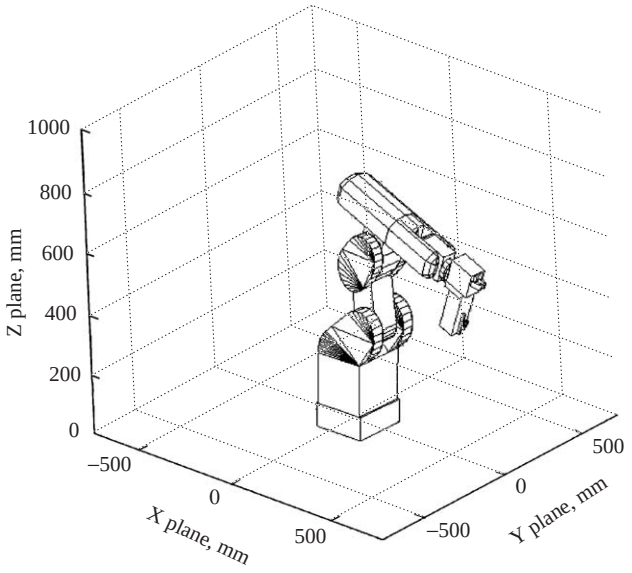

Fig. 4. A three-dimensional model of the six-degrees-of-freedom Mitsubishi RV-E2 manipulator

the cucumber. And finally, using stereovision techniques, the camera vision system produces 3D maps of the workspace within the viewing angle of the camera. In this way the robot is able to deal with the variability in the working environment with which it is confronted.

As stated above, also *a priori* knowledge about, for instance, the physical structure of the robot is needed for collision-free motion planning. As an example, *Fig. 4* shows a 3D model of the six-DOF Mitsubishi RV-E2 manipulator implemented in MATLAB®. The 3D structure of the manipulator is represented by polygons constructed of rectangles and triangles. The model is used for evaluation of motion strategies during simulations and serves as a basis for the detection of collisions of the manipulator with structural components in the workspace of the robot during motion planning.

4.2. *Inverse kinematics*

The inverse manipulator kinematics deal with the computation of the set of joint angles and translations that result in the desired position and orientation of the tool-centre-point (TCP) of the manipulator (Craig, 1989). The TCP is a pre-defined point on the end-effector. For the six-DOF Mitsubishi RV-E2 manipulator Van Dijk (1999) obtained an analytic solution of the inverse manipulator kinematics. For the seven-DOF manipulator, *i.e.* the Mitsubishi RV-E2 manipulator mounted on a linear slide, a straightforward analytic solution of the inverse kinematics does not exist due to the inherent redundancy in the kinematic chain. Recently a mixed analytic-numerical solution of the

inverse kinematics of this redundant manipulator was obtained (Schenk, 2000). Given the position of the ripe cucumber, the algorithm produces a collision-free harvest configuration of the seven-DOF manipulator. Also, it assures that the joints have sufficient freedom for fine-motion control in the neighbourhood of the cucumber.

4.3. *Path planner*

Algorithms for collision-free path planning have been the object of much research. See *e.g.* Latombe (1991) and Hwang and Ahuja (1992) for an overview.

A collision-free path planner essentially consists of two important components: a search algorithm and a collision detection algorithm. The search algorithm explores the search space for a feasible, *i.e.* collision-free, motion from a start point to a goal point. During the search, the feasibility of each step in the search space is checked by the collision detection algorithm. This algorithm checks for collisions of the manipulator with other structural components in the workspace of the robot.

It is important to note that for most path planners the search space is the so-called configuration space of the robot, which crucially differs from the 3D workspace of the robot. In case of the seven-DOF manipulator of the cucumber harvest machine, the configuration space is a seven-dimensional space spanned by the combination of one joint translation and six joint rotations. Then, the search for a collision-free motion from a start position and orientation of the tool-centre-point to the goal position and orientation in the 3D workspace boils down to a search for a collision-free motion of a single point through the seven-dimensional configuration space from the start configuration to the goal configuration. In this way problems with redundancy in the kinematic chain are easily circumvented. There is a one-to-one mapping of a point in the configuration space to a position and orientation of the tool-centre-point in the workspace. However, for most manipulators, the opposite does not hold. Then a single position and orientation of the tool-centre-point in the workspace can be reproduced by several configurations of the manipulator. Due to its unique representation a search in the configuration space is preferred. For collision detection, however, one needs to describe the physical posture of the manipulator in relation with other objects in the 3D workspace. Because every configuration represents a single posture of the manipulator in the 3D workspace, collisions can be easily verified. Then, given the particular kinematic structure of the robot, the workspace obstacles can be mapped into configuration space obstacles as will be shown later.

4.3.1. *The search algorithm*

A path search algorithm should be efficient and find a solution if one exists. The latter property is referred to as completeness (Pearl, 1984). Usually, algorithms that guarantee completeness are not computationally efficient. Computational efficiency, however, is crucial when on-line application is required. To obtain insight into the various aspects of motion planning, in this research, completeness of the algorithm was favoured above computational efficiency. The main reason for that choice is that a complete algorithm will either find the solution or stop using a clearly defined stopping criterion if a solution cannot be found. This is not true for algorithms that do not assure completeness. They either supply a solution or get stuck without further notice. In this research the so-called A*-search algorithm was used (Pearl, 1984; Kondo, 1991; Russell & Norvig, 1995). It is easy to implement and guarantees completeness. Moreover, it minimises a cost criterion that includes a measure for the distance travelled in the search space. The algorithm was implemented in MATLAB®.

To use the A* algorithm for motion planning, the configuration space of the robot is discretised using a fixed grid structure as shown in *Fig. 5*. The user can define both the size and resolution of the grid. Then the A* algorithm searches a path from the start grid point to the goal grid point while minimising a cost function f. This cost function f includes the cost of the path so far g, and an optimistic estimate of the cost from the current position to the goal h. In this research, the Euler norm was used as an optimistic estimate of the cost to go to the goal node. The A* algorithm is both complete and optimal. Optimality assures that the path obtained minimises the cost function used.

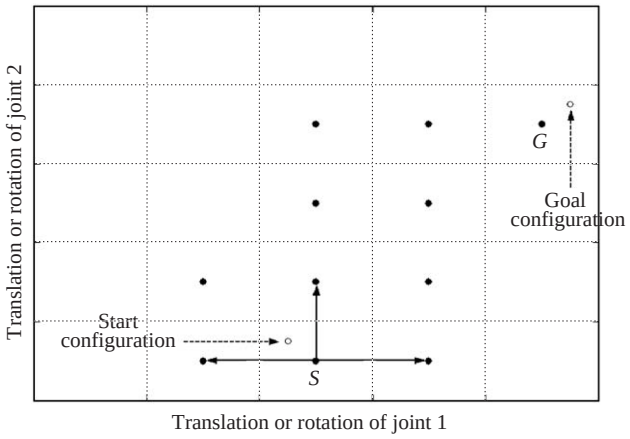

Fig. 5. Orthogonal node expansion in a discretised two-dimensional configuration space: S, start node; G, goal node

The A* algorithm uses two lists with grid nodes, the OPEN list and the CLOSED list. The OPEN list contains grid nodes of which the cost function has not yet been evaluated, whereas function values of the grid nodes on the CLOSED list have been evaluated. It is assumed that the start and goal configuration coincide with grid nodes or that grid nodes in the close neighbourhood of these configurations can be selected.

Then according to Pearl (1984), the A* algorithm manipulates the nodes in the grid as follows.

(1) Put the start node S on OPEN.
(2) If OPEN is empty then exit with failure, else remove from OPEN and place on CLOSED a node n for which f is a minimum.
(3) If n is equal to the goal node G, exit successfully with the solution obtained by tracing back the pointers from n to S.
(4) Otherwise expand n, generating all its successors, and attach to them pointers back to n. For every successor n' of n:
    (a) if n' is not already on OPEN or CLOSED, estimate $h(n')$ (an optimistic estimate of the cost of the best path from n' to goal node G), and calculate $f(n') = g(n') + h(n')$ where $g(n') = g \times (n) + c(n, n')$ with $c(n, n')$ being the transition cost from node n to node n' and $g(S) = 0$;
    (b) if n' is already on OPEN or CLOSED, direct its pointers along the path yielding the lowest $g(n')$; and
    (c) if n' required pointer adjustment and was found on CLOSED, reopen it.
(5) Go to step 2.

Grid expansion in step 4 can take various forms. In this research a so-called orthogonal expansion was used. This approach is illustrated in *Fig. 5*.

Figure 5 also illustrates that the start and goal nodes do not have to coincide with the actual start and goal configuration of the manipulator. In such cases the nearest neighbouring nodes are selected.

In this algorithm, the stopping criterion is very clearly defined. The algorithm stops if at step 3, the node removed from the OPEN list equals the goal node. Alternatively, the algorithm will stop at step 2 if all the grid nodes are evaluated and the OPEN list has become empty. In that case a solution is not found.

There are two ways of dealing with collision detection during the path search. First of all, the colliding configurations can be identified before the path search by scanning the whole discretised configuration space. This will be computationally expensive in case of a high-dimensional configuration space discretised with a high resolution grid. It will be more efficient to evaluate the feasibility of the grid nodes during the search process. That is to say, that during the node expansion step, step 4, a collision detection algorithm checks whether the robot configuration associated with that node yields a collision with the environment or not. Since the A* algorithm usually evaluates only a small part of the configuration space, this will yield a considerable improvement in the efficiency. A collision can be penalised by adding a large penalty to the cost function mentioned in step 4a. Alternatively, a grid point resulting in a collision can be omitted directly from the OPEN list during the grid expansion phase. In this research the latter approach was used.

4.3.2. *The collision detection algorithm*

For collision detection an algorithm was implemented in MATLAB® based on the ideas reported by Boyse (1979). This algorithm evaluates the intersection of surfaces of the robot model with the surfaces of other structural components in the workspace. Calculating the intersection of two surfaces essentially boils down to determining the intersection of all the edges of one surface with the other surface which can be achieved using standard tools from geometry. All in all, collision detection is a computationally intensive task. Therefore, collision detection in a real-time application such as the cucumber robot, requires a trade-off between the desired accuracy of the collision detection and the available calculation time. The accurate CAD-model of *Fig. 4* contains 600 triangular and rectangular surfaces. A factor 15 reduction in calculation time was achieved by replacing the accurate manipulator model by a less accurate model built from so-called oriented bounding boxes (OBB). This 3D OBB-model of the manipulator consisting of only 36 moving surfaces is shown in *Fig. 6*. Clearly, with the OBB-model some accuracy has been offered for the sake of calculation speed. For the current investigations it was considered justifiable.

5. Example 1: collision-free motion planning for a two-degrees-of-freedom manipulator

To illustrate the approach, results of collision-free motion planning for a manipulator with two rotational joints (two DOF) are presented. *Figure 7*(*a*) shows a top-view of an artificial greenhouse environment in which the squares represent cucumber stems and the objective is to move the tool-centre-point of the manipulator from the path (straight lay out) to a cucumber hanging behind a cucumber stem, without hitting any cucumber stems. This is considered to be one of the most difficult motions during cucumber picking.

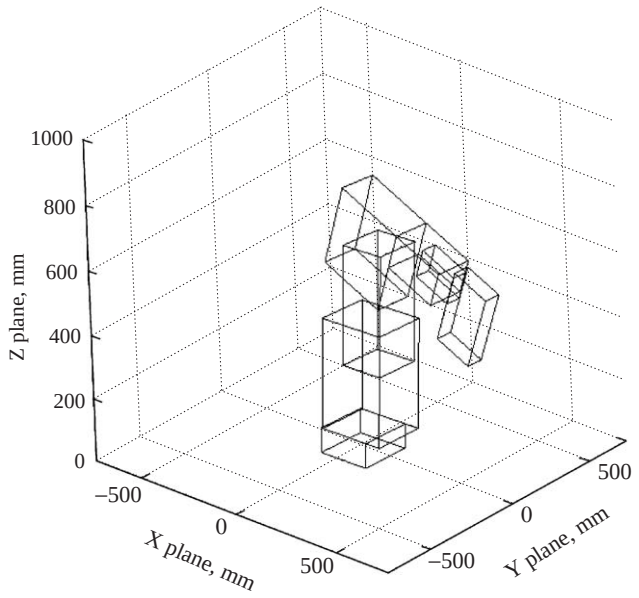

Fig. 6. An oriented-bounding-box model of the six-degrees-of-freedom RV-E2 manipulator

5.1. *Results*

To illustrate the operation of the motion planning algorithm, *Fig. 7*(*b*) shows the associated two-dimensional configuration space. A discretisation step of five degrees was used. The solid black squares, called configuration obstacles, represent configurations yielding a collision between robot and a cucumber stem. The start configuration is indicated by the letter *S*, the goal configuration is indicated by the letter *G*. They represent the start posture and the picking posture shown in *Fig. 7*(*a*). The objective of the path search is to find a connection between the start node *S* and the goal node *G*. Observe, first of all, that the map of the configuration space reveals the true complexity of the motion planning problem that may look trivial in the work space. Secondly, observe that a straight path from the start node to the goal node results in a collision and therefore is not feasible. *Figure 7*(*c*) shows the grid nodes, denoted by *, evaluated by the A* algorithm during a forward search from the start node to the goal node. The optimal path in the configuration space is shown in *Fig. 7*(*d*) and snapshots of the associated collision-free motion of the manipulator in the workspace are shown in *Fig. 7*(*e*). Observe that the collision-free motion in the configuration space results in a collision-free motion in the workspace; the manipulator does not interfere with the workspace obstacles: the cucumber stems. Finally, *Fig. 7*(*f*) shows the grid nodes evaluated by the A* algorithm when a backward search is performed from the goal node to the start node.

5.2. *Discussion*

The results illustrate that path search in the configuration space boils down to finding a point trajectory from the start configuration to the goal configuration. *Figure 7*(*c*) and (*f*) clearly demonstrates the advantages of collision checking during the path search in stead of *a priori* collision detection, since the A* algorithm only partially evaluates the grid points in the configuration space. Additionally, the results suggest, that if an obstacle is located between the start configuration and the goal configuration, a considerable amount of grid nodes have to be evaluated before a solution is found. In such cases, the A* algorithm is not very efficient in finding a way around the configuration space obstacle. In case obstacles closely circumvent a goal node, a backward search may yield a solution with less computation time as illustrated by *Fig. 7*(*f*). In this example the backward search yielded a solution after 117 iterations instead of 146 during a forward search; a reduction of 20%. If the goal node was located to the far end of the alley between the two obstacle ridges, even a higher reduction in the number of iterations was obtained using a backward search (results not shown). The best search direction clearly depends on the particular structure of the problem at hand. Both *Fig. 7*(*d*) and (*e*) demonstrates that to achieve an optimal path in the sense of the cost function, the algorithm tends to cut corners resulting in small distances between robot and obstacle. This characteristic has to be kept in mind during practical motion planning experiments when the sensor-based world description data are prone to inaccuracies. Then collisions may occur that were not accounted for during the motion planning. Finally, *Fig. 7*(*d*) shows that due to the grid structure and the orthogonal expansion of the grid nodes during the path search, the motion path contains a few sharp corners. This will result in strong unwanted accelerations and decelerations of the links when implemented in practice. Such undesirable behaviour calls for smoothing of the motions by the trajectory planner as suggested in Section 4.

6. Example 2: collision-free motion planning for a six-degrees-of-freedom manipulator

This paragraph demonstrates the motion planning program on the six-DOF Mitsubishi RV-E2 manipulator. *Figure 8*(*a*) shows a three-dimensional view of the six-DOF manipulator in an artificial greenhouse environment. Again, the objective is to move the tool-centre-point of the manipulator from a position in the path to a cucumber hanging behind a cucumber stem without hitting the cucumber stems represented by the black posts.

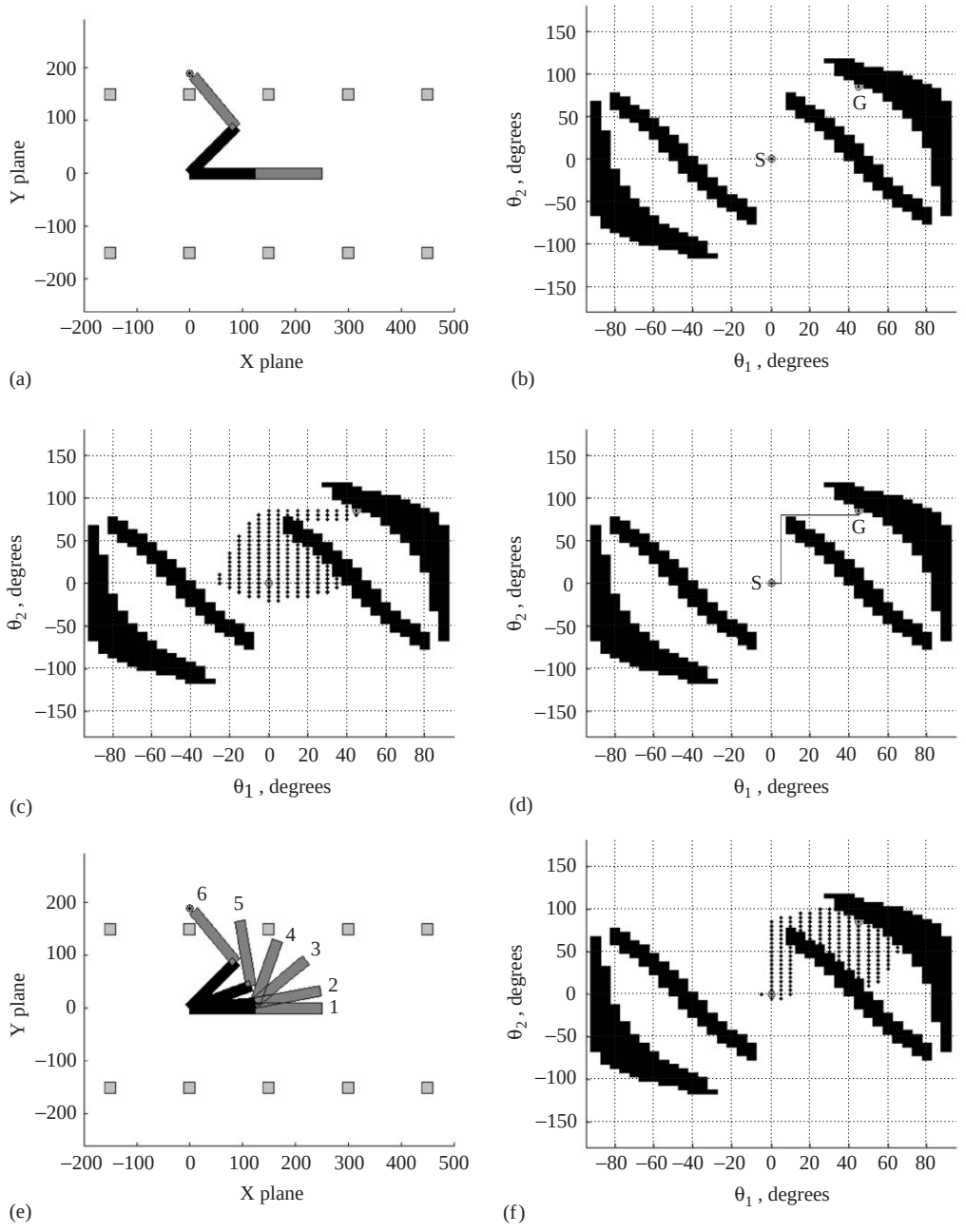

Fig. 7. Collision-free motion planning for a cucumber picking operation of a two-degrees-of-freedom manipulator in an artificial greenhouse environment: (a) topview of the workspace with the manipulator in start posture (straight) and goal posture to pick a cucumber hanging behind a cucumber stem represented by the grey square; (b) the associated configuration space with the black areas representing configurations resulting in a collision and S and G representing the start and goal configuration, respectively; (c) the configuration space sampled by the A algorithm during a forward search from the start to the goal node; (d) the collision-free trajectory through the configuration space; (e) six snapshots of the collision-free motion of the manipulator to the cucumber; (f) the configuration space sampled by the A* algorithm during a backward search from the goal to the start node; θ_1 and θ_2 are the rotations of the first and the second joint, respectively*

6.1. *Results*

Since this example involves a six-DOF manipulator, the search is performed in a six-dimensional configuration space. It is impossible to visualise the collision-free point motion through the configuration space as was done with the previous example. Therefore only snapshots of the collision-free motion through the workspace are presented in *Fig. 8*(*a*)–(*f*). The motion involves all six rotational joints. Essentially, the motion

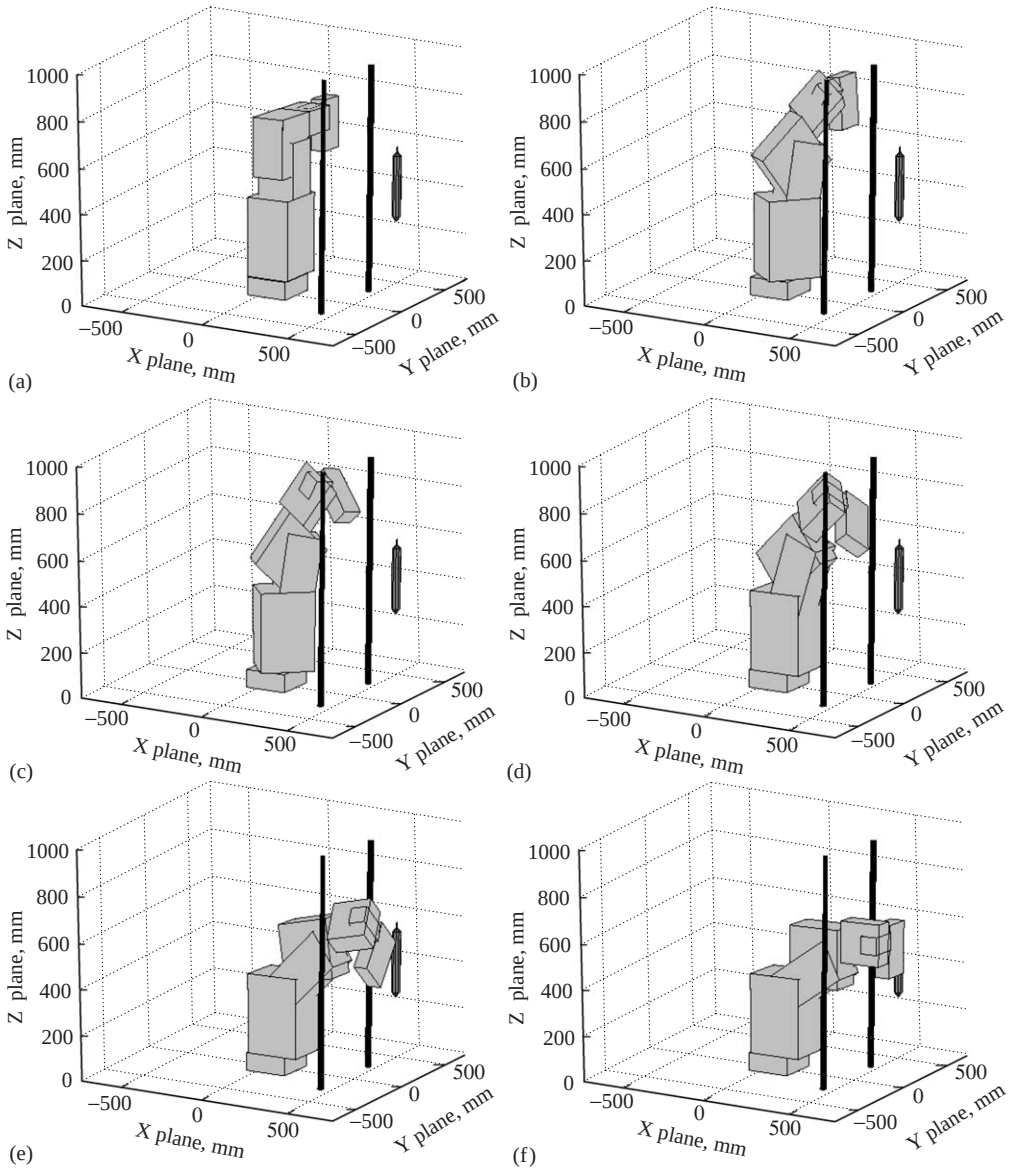

Fig. 8. (a)–(f): Six snapshots of a collision-free motion of the six-degrees-of-freedom RV-E2 manipulator to a cucumber hanging behind a cucumber stem represented by the black vertical posts

to the cucumber consists of two components. First of all, the manipulator tilts backwards while rotating around the main vertical axis and then tilts forwards again to bring the tool-centre-point between the cucumber stems. Secondly and simultaneously, the last three joints rotate so as to be able to position the tool-centre-point at the cucumber behind the cucumber stem. By doing so, the cucumber stem is circumvented.

6.2. *Discussion*

The results illustrate that for a six-DOF manipulator a collision-free motion can be found. It is expected that this result can be extended to the seven-DOF manipulator used in the cucumber picking device. However, the example revealed a weakness of the A* algorithm. For the six-DOF manipulator under consideration, the search was performed in the six-dimensional configuration space. Then, due to the large amount of grid points that have to be evaluated, the search becomes prohibitively slow. This is partly due to the implementation in MATLAB®. This software package is not very efficient when large numbers of iterations have to be performed.

Again the results show the sharp corners in the motion trajectories. When high speed motions are required, these motion trajectories have to be smoothed to prevent to heavy loading on the manipulator links.

7. Conclusion

In this paper a methodology was presented to achieve a proper eye–hand co-ordination for the cucumber harvesting robot developed at the Institute of Agricultural and Environmental Engineering (IMAG B.V.). The paper outlined a program that is able to generate collision-free motions for the manipulator. With some numerical examples the approach was illustrated and analysed.

Main conclusion of this research is that collision-free motions can be calculated for the six-degrees-of-freedom (DOF) RV-E2 manipulator used in the harvesting machine. It is expected that these results can be extended to the seven-DOF manipulator, *i.e.* the RV-E2 manipulator mounted on the linear slide. The A*-search algorithm was found to be easy to implement and robust. In this way it offered a lot of insight into the particular problems of robot motion planning. Additionally a big advantage of this algorithm is that it either produces a solution or stops when a solution cannot be found. This completeness property, however, makes the algorithm prohibitively slow. It was found that calculation of motion trajectories for manipulators with many degrees of freedom is computationally very involved with the algorithm described in this paper. To comply with the desired cycle time of 10 s for a single harvest action, further research is required to reduce the computation time needed for the motion planning. Research can be directed along two lines. First of all, computation time can be reduced by using special computing hardware, *e.g.* parallel processors. Alternatively, and simultaneously, reduction of computation can be achieved by using faster and efficiently implemented algorithms. Also, the results showed that the algorithm does not include much intelligence. Though it tries to generate a goal directed motion, if configuration obstacles are encountered it just samples the grid points in the search space until the solution is found without using much information about the particular structure of the problem. So, further research is required to derive fast algorithms that efficiently exploit information about the particular structure of the problem and do not get stuck without further notice.

Acknowledgements

This work was supported by the Dutch Ministry of Agriculture, Food and Fishery. The constructive comments of the anonymous referees are gratefully acknowledged.

References

Arima S; Kondo N (1999). Cucumber harvesting robot and plant training system. Journal of Robotics and Mechatronics, **11**(3), 208–212

Bontsema J; Van Kollenburg-Crisan L M; Van Henten E J (1999). Automatic harvesting of vegetable fruits, Proceedings of the BRAIN International Symposium 2000—Progressive Technologies in Agriculture and Environment towards 21 Century, November 24, 1999, IAM-BRAIN, Omiya, Japan, pp 44–51

Boyse J W (1979). Interference detection among solids and surfaces. Communications of the ACM, **22**(1), 3–9

Craig J J (1989). Introduction to Robotics. Addison-Wesley, Reading, MA, USA

Gieling Th H; Van Henten E J; Van Os E A; Sakaue O; Hendrix A T M (1996). Conditions, demands and technology for automatic harvesting of fruit vegetables. Acta Horticulturae, **440,** 360–365

Hayashi S; Sakaue O (1996). Tomato harvesting by robotic system. ASAE Annual International Meeting, Phoenix, Arizona, USA, ASAE Paper No. 96-3067

Herman M (1986). Fast, three-dimensional, collision-free motion planning. Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, pp 1056–1063

Hwang Y K; Ahuja N (1992). Gross motion planning—a survey. ACM Computing Surveys, **24**(3), 219–291

Kondo K (1991). Motion planning with six degrees of freedom by multistrategic bidirectional heuristic free-space enumeration. IEEE Transactions on Robotics and Automation, **7**(3), 267–277

Kondo N; Monta M; Fujiura T (1996). Fruit harvesting robots in Japan. Advances in Space Research, **18**(1/2), 181–184

Latombe J C (1991). Robot Motion Planning. Kluwer Academic Publishers, Boston, USA

Meuleman J; Van Heulen S F; Kornet J G; Peters D G (2000). Image analysis for robot harvesting of cucumbers. AgEng 2000, Agricultural Engineering International Conference, Warwick, UK, EurAgEng Paper No. 00-AE-003

Pearl J (1984). Heuristics. Intelligent Search Strategies for Computer Problem Solving. Addision-Wesley, Reading, MA, USA

Russell S; Norvig P (1995). Artificial Intelligence—A modern Approach. Prentice-Hall, Englewood Cliffs, NJ, USA

Schenk E J (2000). Modelvorming, voorwaartse kinematica, inverse kinematica met botsingsdetectie en padplanning van een 7 DOF manipulator systeem voor het automatisch oogsten van komkommers. [Modelling, forward kinematics, inverse kinematics with collision detection and motion planning of a 7 DOF manipulator system used for cucumber harvesting]. IMAG, Wageningen, The Netherlands, IMAG Report V2000-77

Van Dijk G (1999). Modelvorming en padplanning van een 6 DOF manipulator voor het oogsten van komkommers. [Modelling and motionplanning of a 6 DOF manipulator used for cucumber harvesting.] IMAG, Wageningen, The Netherlands, IMAG Report V99-04

Van Kollenburg-Crisan L M; Wennekes P; Werkhoven C (1997). Development of a mechatronic system for automatic harvesting of cucumbers. In: Proceedings of BIO-ROBOTICS 97, The International Workshop on Robotics and Automated Machinery for Bio-productions, Valencia, Spain, pp 143–148