2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Virtual 3D Environment for Planning Robotic Paint Routes

Kåre Storgaard Nissum, Troels Hessner Larsen
Ole Madsen, Henning Nielsen
Aalborg University, Denmark

Abstract— **This paper examines the possibilities of utilizing a virtual environment for programming painting robots. Two cameras are utilized to track the position and movement of the robot programmer and the orientation and position of a spray gun. This data is converted into a virtual environment where a 3D stereo rendering is projected onto a screen for a robot programmer to visualize by use of anaglyph glasses. It is possible to program the paint robot in the virtual world after which the program can be simulated using state of the art Graphics Processing Unit (GPU) programming to create a real-time visualization of the coverage of paint. Results concerning the accuracy and visual examples of the system are given.**

I. Introduction

Manual painting of industrial objects is very expensive due to the price of both paint and human labor, and it often yields results of varying quality. As a consequence painting robots are utilized for automatic processing. This has many advantages as the robots both tend to use less paint and produce a more consistent paint quality.

However, the robots must be programmed to be able to paint an object. Today such programming is often performed using an on-line programming technique where the robot is moved to fix points around the object using a remote control.

The problem using this procedure is twofold. Firstly, the process is time consuming. Secondly, while the programming is performed the robot cannot be used for painting. As the paint robot might be part of a production line, this causes the entire production to stop. The cost of programming the robot thus includes both man power and a stopped production line.

Another solution is use of automatic paint planning software. This has been the focus of a number of research and industrial projects [1] [2] [3], and a number of successful implementation can be found in the industry. However, existing paint planning software has limitations when planning complex objects. Use of expert knowledge of a painter is important but in a more optimized way than by using a remote control.

Creating a system which allows the paint robot to keep processing, while programming is performed, requires other means to determine the position of the paint nozzle. An

Kåre Storgaard Nissum is a research assistent at Department of Production, Aalborg University, 9220 Aalborg, Denmark kaare@production.dk
Troels Hessner Larsen is an engineer at Inropa A/S, 9700 Brønderslev, Denmark troels@inropa.com
Ole Madsen is an Associate Professor,PhD at Department of Production, Aalborg University, 9220 Aalborg, Denmark i9om@iprod.auc.dk
Henning Nielsen is an associate professor in image analysis at Department of Computer Vision and Media Technology, Aalborg University, 9220 Aalborg, Denmark hn@cvmt.aau.dk

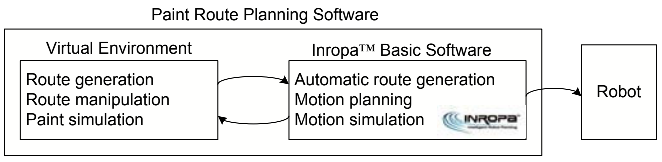


Fig. 1. The relationship between the developed virtual painting environment and Inropa^{TM}Basic.

obvious and intuitive choice is to use a spray gun, and determine the position and orientation of the gun nozzle as a simulated robot tool centre point (TCP).

When planning a painting route by use of a spray gun, the object to be painted is required in order for the programmer to know where to position the TCP. It is preferable to use virtual objects based on CAD models as this renders the real object superfluous during planning. In addition it adds the possibility to apply virtual paint, thereby indicating which part of the object is painted with the currently planned route. Previous research regarding paint simulations includes [4].

Virtual environments exist today such as the Cave at Aalborg University [5]. However, systems such as the Cave are extremely costly due to the required equipment for visualization and tracking. To be able to commercialize a virtual environment for paint route planning, a low cost solution is required.

The solution presented in this article is the creation of a low cost virtual environment which can be utilized to generate a painting route, manipulate an existing painting route and simulate the paint coverage of a painting route. Furthermore, the system is compatible with Inropa^{TM}Basic software which can perform an initial automatically generated route, motion planning and simulate the robot motions. Finally, it is capable of transferring the program directly to a painting robot. The entire system relationship is depicted in Fig. 1.

II. Methods

A. System Setup

The setup of the virtual environment is depicted in Fig. 2. The hardware required for the setup is a projector, a screen, a pair of anaglyph glasses, a spray gun, two cameras and four markers. Three markers are mounted on the spray gun and one marker on the anaglyph glasses.

A photo of the anaglyph glasses and the spray gun can be seen in Fig. 3. Three reflector markers are attached to the

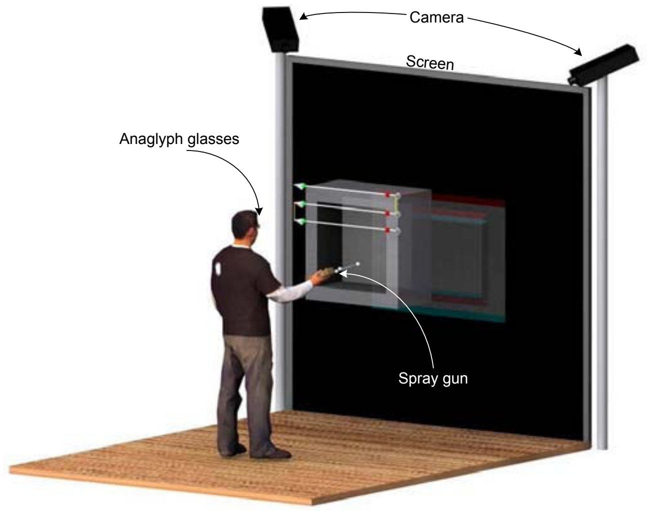


Fig. 2. The setup of the system.

spray gun and an emitting marker is attached to the glasses.

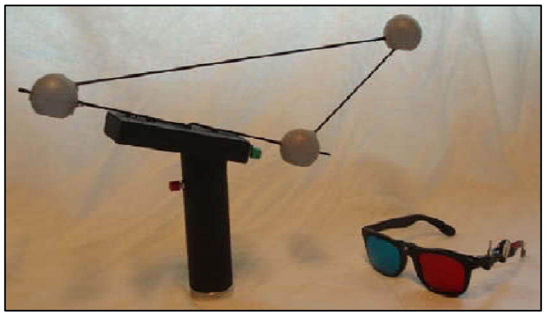


Fig. 3. The anaglyph glasses and the spray gun with markers attached.

The structure of the virtual environment software is illustrated in Fig. 4. The vision tracking part utilizes the calibrated cameras to calculate the spatial coordinates of the markers. The position and orientation of the spray gun and position of the glasses are consecutively calculated and transfered to the stereographics visualization part. In this part stereographic images are created and projected onto the screen. If desired, the real-time GPU based paint simulation part can be activated. The methods used for each part is elaborated upon in the following.

B. Tracking

The objective of the tracking is to determine the location of three markers for determination of the position, movement and orientation of the spray gun, and one marker for the position and movement of the robot programmer. The orientation of the programmer is assumed to be towards the screen.

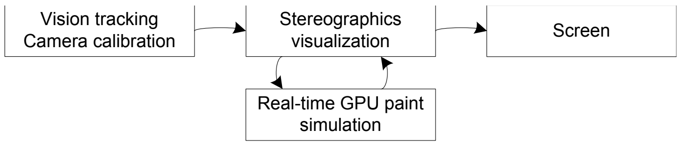


Fig. 4. The structure of the virtual environment.

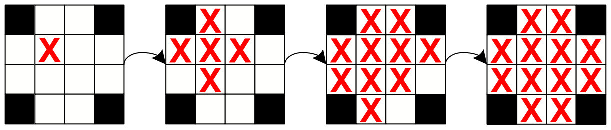


Fig. 5. The process of using recursive region growing for labeling a marker. The first step shows the seed pixel.

Segmentation: The markers appear as high-intensity blobs in the images, and a global thresholding function is consequently utilized to segment the markers. The four markers in each image are separated and labelled using recursive region growing [6] [7].

In order to do the labelling, the image is searched for a white seed pixel which is then labeled. This is depicted as the first step in Fig. 5. The four-connected neighbourhood of this pixel is then investigated, and if any neighbouring pixels are white they are labelled too, as illustrated in the second step of Fig. 5. The neighbourhood of these pixels are then investigated etc. The process terminates when all pixels in a blob have been labelled. The search then continues to find a new seed pixel which has not been labelled yet until the four marker blobs are all located and labelled.

The labelled marker pixels are undistorted using the calibrated camera parameters of each camera [8].

Because circular markers are utilized the goal is to locate the centre of each marker. In order to account for flickering of edge pixels in the grabbed images, a weighted centre of mass calculation is performed on each labelled marker yielding a high-precision sub-pixel accuracy determination of the circle centre [9]. The formula for the weighted centre of mass is:

$$(x, y) = \frac{\sum_{i=1}^{n} (I_i - T) \cdot (x_i, y_i)}{\sum_{i=1}^{n} (I_i - T)} \tag{1}$$

where I is the intensity in the original image, T is the global threshold and n is the number of pixels used for the calculation.

Triangulation: In order for the triangulation of the markers to be accurate, corresponding images from each camera must be grabbed at the same time. This is not the case here since unsynchronized video cameras are used. A first order linear interpolation is consequently utilized to account for the small displacements and to stabilize the tracking. The process is depicted in Fig. 6.

The extrinsic camera calibration parameters are utilized to project a ray in space through the located marker centre and the centre of the camera. This produces four rays for each camera resulting in a total of 16 intersections in space. An intersection is defined as the midpoint of the shortest line between a ray from each camera.

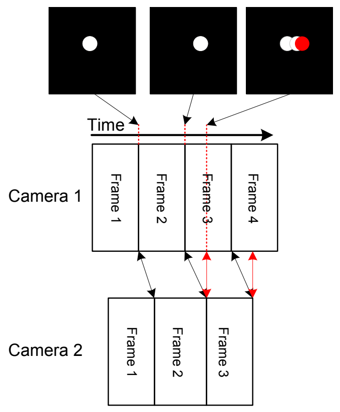


Fig. 6. The frames from the cameras are not synchronized and a first order linear interpolation is utilized to account for the displacement.

The initial rejection rule is to reject an intersection if the distance between the two rays is above a certain predefined threshold. This will produce a candidate set of positions in space. In order to locate the correct markers from the candidate set, a priori knowledge about the physical placement of the markers in relation to each other is utilized. The candidate set is paired in all combinations of three producing three lines. The sum of squared difference between the length of these lines and a priori knowledge of the physical placement of the markers are calculated yielding the best match for the markers constituting the spray gun. Only one point from each image will remain which implicitly yields the position of the robot programmer's glasses.

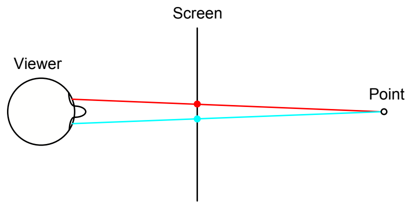
C. Visualization

The visualization part receives the calculated marker positions from the tracking part. Based on these positions the part is to visualize the virtual scene for the robot programmer.

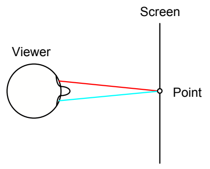
Stereoscopic Renderings The visualization of the CAD-model and paint route is based on stereoscobic renderings. In stereoscopic renderings a number of depth cues are utilized to yield images which seemingly come out of the screen. Apart from an implementation of monoscopic depth cues, which are well known from 3D games, stereoscopic depth cues are utilized. The stereoscopic depth cues are based on the fact that humans have two eyes. Instead of projecting only one image on the screen, two superimposed images are projected onto the screen [10]. Different display techniques for presenting the superimposed images exist, e.g. using polarized glasses, shutter glasses or anaglyph glasses. To be able to meet the low cost demands, the superimposed images are projected using the anaglyph technique in the proposed solution. When using the anaglyph technology the two images are colour separated, such that they consist of a red and a cyan image. When viewed through anaglyph glasses each eye can only see one of the two images. Fig. 7 illustrates an example of an anaglyph image.

In order to determine how the two superimposed images are related see Fig. 8. The figure illustrates a point in the two images projected onto the screen. The point is perceived as being either behind Fig. 8(a), on Fig. 8(b) or in front of the screen Fig. 8(c). The perceived position depends on the relation between the position of the point in the red and cyan image respectively.

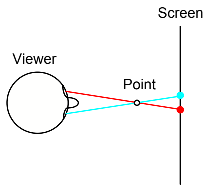
Fig. 7. Anaglyph image consisting of two colour separated superimposed images.



(a) Behind screen



(b) On screen

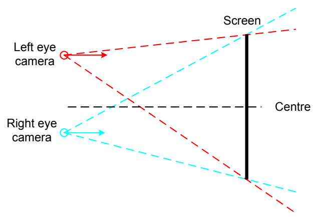


(c) In front of screen

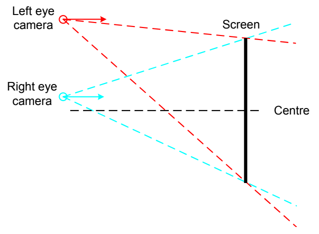
Fig. 8. Depending on the relation of the point on the two images, the point is perceived as being either behind, on or in front of the screen. Based on [10].

To create the superimposed images two view-ports with corresponding cameras are created. Each camera corresponds to an eye of the robot programmer. The frustums of the cameras are set up using the mathematically correct off-axis projection [11]. When using this approach the frustums of the two cameras are skew. The skewness of the frustums depends on the position of the robot programmer in relation to the screen. This is illustrated in 2D in Fig. 9, where the robot programmer moves to the side. To determine the skewness and position of the cameras the tracked position of the robot programmer is utilized in the calculations. This enables the robot programmer to move around a virtual CAD-based object and corresponding paint route which appear stationary in space.

Route Planning In order to plan a paint route for a visualized object the spray gun is utilized. When the trigger



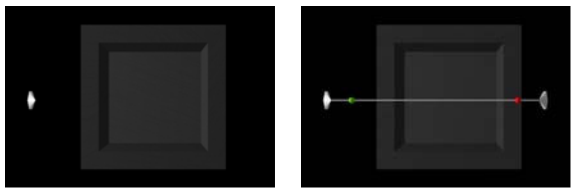
(a) At center



(b) Moves left

Fig. 9. As the robot programmer moves to the left the two frustums of the two cameras become increasingly skew.

of the gun is pressed a paint stroke is created along with the first fix point of the stroke, as illustrated in Fig 10(a). The fix point is created at the position of the gun according to the orientation of the gun. The gun can then be moved to where the stroke is desired to end. When the trigger is released a second fix point of the stroke is inserted at the new position of the gun. In addition a *gun on* (green sphere) and a *gun off* event (red sphere) are automatically inserted. This is illustrated in Fig. 10(b). The position and orientation of the fix points can be altered after they are created.



(a) First fix point (b) Second fix point

Fig. 10. When the trigger is pressed a fix point is inserted. When the trigger is released a second fix point is inserted. In addition a *gun on* and a *gun off* event is automatically inserted.

D. Simulation

When a route has been planned it is possible to simulate the paint route to get an indication of the paint coverage. The simulation is only an indication of the coverage as not all parameters are included in the painting model, e.g. electrostatic objects or air current produced at corners of the object. The system simulates the route by updating the robot nozzle position based on a linear interpolation of the position and on the orientation of the fix points. Creating a real-time CPU based paint simulator is very resource demanding and is thus not a feasible solution. Instead it is possible to create texture based painting, utilizing the GPU, by performing state of the art GPU programming to enhance performance significantly. In texture based painting a texture is mapped onto the CAD-model and the paint is applied to this texture in each rendered frame. In order to utilize the GPU a viewpoint and corresponding camera is created at the robot TCP position. This is illustrated in Fig. 11.

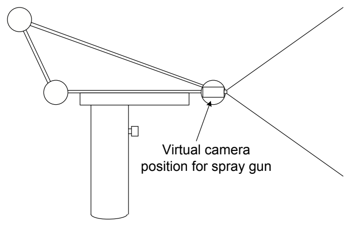


Fig. 11. An additional camera is inserted at the front marker of the spray gun. This camera is used to paint the object.

It is necessary to use several render-passes for the spray gun view-port to apply the paint. In the first pass the position of the object vertices is stored into the texture coordinates in the vertex program. Similarly the texture coordinates are stored into the vertex coordinates. The result of this pass is illustrated in Fig. 12(a). As illustrated this corresponds to an UV map rendering where the colour corresponds to the fragment position. In the second render-pass a depth-map is created. In a depth-map the distance to the rendered objects is outputted instead of the colour of the object. The result of this pass is illustrated in Fig. 12(b). The final pass is a render-quad-pass in which the results of pass one and two are used. A lookup into the UV map is made to determine the position of the fragment. Given the position is within painting range, the depth of the fragment position is considered. The position of the fragment is used to perform a lookup into the depth map. This depth value is compared to the depth value of the fragment. The depth of the UV map, which is stored in the blue colour channel, is illustrated in gray scale in Fig. 12(c). Given that the two depth values are the same, paint is applied to the fragment. If they are not the same the fragment is occluded by other fragments and paint is thus not applied. The result of applying the paint in the texture is illustrated in Fig. 12(d). The result of this final pass is stored in a texture.

Instead of the original object texture this new texture containing the paint information is mapped onto the object which is illustrated, from the spray gun viewpoint, in Fig. 12(e). Finally, when viewed through the left eye view point, which is to the right and further away than the spray gun, the object is as illustrated in Fig. 12(f).

III. Results

Fig. 13 shows the precision of the marker localization. Four different marker positions at distances between

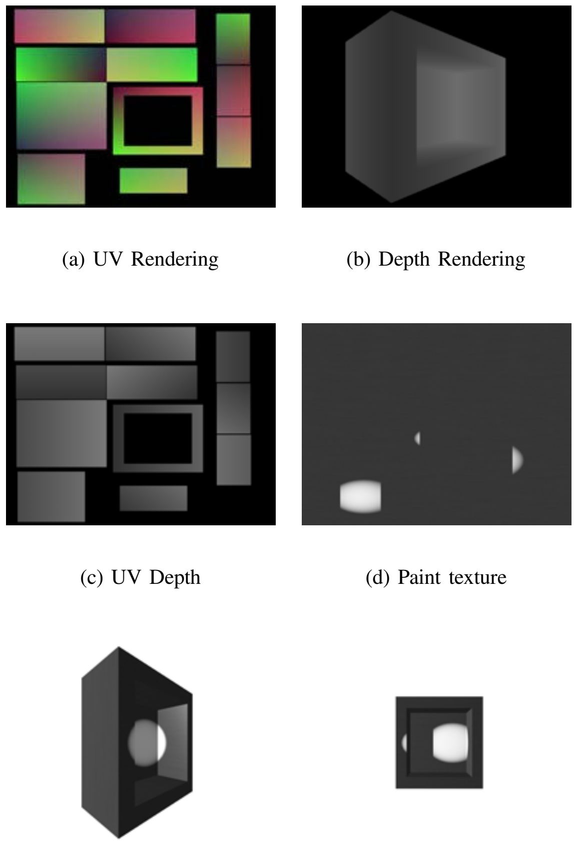

(a) UV Rendering (b) Depth Rendering (c) UV Depth (d) Paint texture (e) Spray gun view (f) Eye view

Fig. 12. The result of the different render passes during the painting.

2500mm and 3500mm from the two cameras are shown. The figure depicts all positions of the markers which resulted in maximum deviations of 6mm at a 95% level of confidence. Considering the context of the system, where the robot has a distance of approximately 250mm to the object, a precision of ±6mm is considered to be sufficient.

The framerate of the tracking system is $25 Hz$ which is the maximum when using PAL cameras. This framerate provides a smooth visualization for the robot programmer.

Fig. 14 illustrates two objects visualized using the system. In Fig. 14(a) and (c) the user is standing to the left of the object. In Fig. 14(b) and (d) the user is standing to the right of the object.

Fig. 15 shows a simulation in progress (without anaglyph view). As illustrated, the created paint route consists of six fix points. In Fig. 15(a) the simulation has been started. In Fig. 15(b) the simulation is at a yet later stage and the additionally applied paint is visible. The update frequency of a standard projector is 60Hz and the simulation has a framerate above this using an ATI X-300 graphics card and a laptop.

The planned painting route can be transfered to Inropa^{TM}Basic Software as depicted in Fig. 16(a) where the motion planning is done. The final robot program is

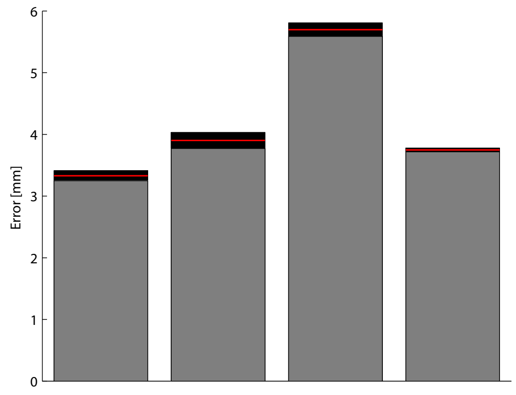


Fig. 13. The result of the marker localization test. The bars represent the error of the localization at four different locations. The distance to the camera is between 2500mm and 3500mm. The black areas indicate the 95% confidence intervals.

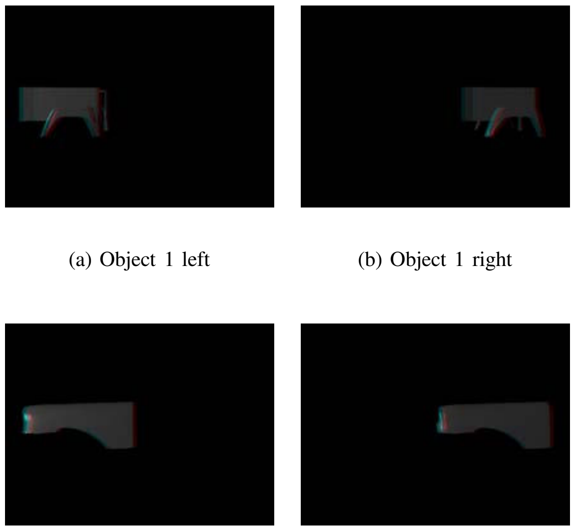

(a) Object 1 left (b) Object 1 right (c) Object 2 left (d) Object 2 right

Fig. 14. Visualization of two objects viewed from two different positions.

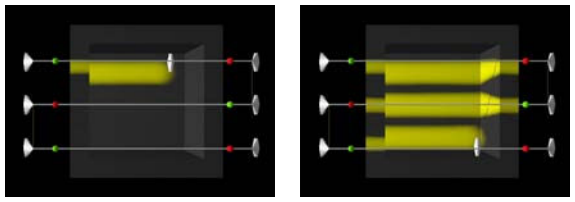
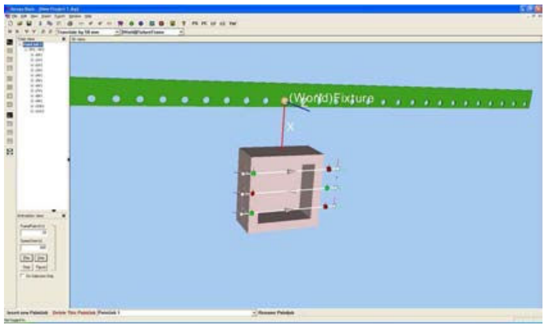
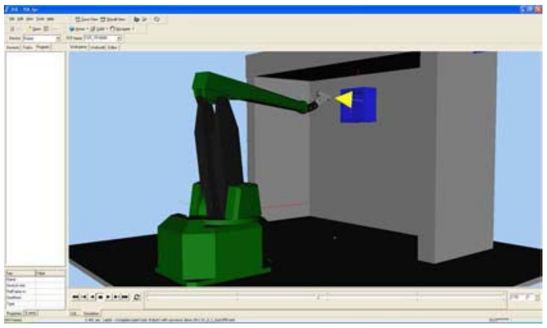

(a) Simulation stage 1 (b) Simulation stage 2

Fig. 15. Two different stages during the runtime paint simulation.

simulated in TUL from AMROSE, Ltd. as depicted in Fig. 16(b).



(a) Planned painting Route shown in Inropa^{TM}Basic Software



(b) Motion simulation in TUL software from AMROSE, Ltd.

Fig. 16. The simulation from Fig. 15 has been transfered to Inropa^{TM}Basic Software and the motion planning is simulated in TUL from AMROSE, Ltd.

IV. Conclusion

A virtual environment capable of visualizing paint routes has been created. The location and movements of the robot programmer and the location and orientation of the spray gun are tracked in space thereby allowing for the robot programmer to interact with the virtual environment. It is possible to create a paint route and consecutively edit it. Furthermore, state of the art GPU programming is utilized to simulate the paint coverage of the created paint route in real-time.

When a paint route has been created it can be transfered to Inropa^{TM}Basic Software which handles the motion planning. Afterwards the paint route created in the virtual environment can be replicated by a robot.

The automatic paint route generation tool of Inropa^{TM}Basic Software can be used to create an initial painting route for an object.

It is possible to transfer this route to the virtual environment and edit it. Thereafter the motion planning is performed by Inropa^{TM}Basic Software and a robot can replicate it.

A. Future Work

A menu in the virtual environment is currently being created allowing for the robot programmer to change all settings of the paint route. This menu and the interaction of the robot programmer has to be tested by robot programmers to find the most intuitive way for the robot programmers to interact.

The described system utilized two cameras for the tracking. This may cause occlusion at certain points, hence more cameras to perform the tracking is desired.

Acknowledgment

This article is partly financed by the ECO2PAINTER project.

References

[1] N. Asakawa and Y. Takeuchi, “Teachingless spray-painting of sculptures by an industrial robot,” *Proc. of IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico*, 1997.

[2] G. Biegelbauer, A. Pichler, M. Vincze, C. L. Nielsen, H. J. Andersen, and K. Haeusler, “The inverse approach of flexpaint,” *IEEE Robotics & Automation Magazine*, pp. 24–34, September 2005.

[3] M. Vincze, A. Pichler, G. Biegelbauer, K. Häusler, H. Andersen, O. Madsen, and M. Kristiansen, “Automatic robotic spray painting of low volume high variant parts,” *Proc. of 33th International Symposium on Robotics*, 2002.

[4] M. S. Arikan and T. Balkan, “Process modeling, simulation, and paint thickness measurement for robotic spray painting,” *Journal of Robotic Systems*, vol. 17, no. 9, pp. 479–494, 2000.

[5] V. M. Lab, “Cave,” 2006, http://www.vrmedialab.dk/pr/facilities/cave.html.

[6] A. Wallace and S. Price, “Region growing: A recursive approach,” CV Online, 2002.

[7] D. Marshall, “Region growing,” CV Online, 1997.

[8] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2003, iSBN: 0521 54051 8.

[9] G. Chiorboli and G. P. Vecchi, “Comments on "design of fiducials for accurate registration using machine vision",” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 12, pp. 1330–1332, 1993.

[10] S. Corporation, “Stereographics developers’ handbook,” 1997, http://www.stereographics.com/.

[11] P. Bourke, “Calculating stereo pairs,” 1999, http://astronomy.swin.edu.au/ pbourke/stereographics/stereorender/.