Automation of an excavator based on a 3D CAD model and GPS measurement

*Tomi Makkonen, */***Kalervo Nevala, **Rauno Heikkilä
*University of Oulu, Department of Mechanical Engineering, P.O.Box 4200, FIN-90014 University of Oulu, Finland. E-mail: Tomi.Makkonen@me.oulu.fi
**University of Oulu, Research unit of Construction Technology, P.O.Box 4400, FIN_90014 University of Oulu, Finland. E-mail: Rauno.Heikkila@oulu.fi
***VTT Electronics, P.O.Box 1100, FIN-90570 Oulu, Finland. E-mail: Kalervo.Nevala@vtt.fi

Abstract: This study examines the possibilities of controlling a six degrees of freedom excavator with the final objective of controlling the movements of the excavator by using a CAD model of the road surface. Compared to the traditional excavator with 4 DOF, the excavator was provided with two additional degrees of freedom by applying the 2 DOF Rototilt, an accessory commonly in use. The advantages of automation are faster process of working and less demands on the operator. To study this problem, an Msc.Adams with Matlab/Simulink simulation environment was used.

Keywords: robotic excavator, path generation, dig planning, modeling, simulation, triangular terrain model, CAD

1 INTRODUCTION

A robotic excavator is not a new idea as there are many interesting and advanced systems, for example, Autonomous Truck Loading from Carnegie Mellon University [1], LUCIE from Lancaster University [2] and Komatsu PC05-7 in ACRF [3].
Our studies have focused on 1. heavy use of a modelling environment. 2. an assumption that a CAD model of the target surface is available. 3. a six degrees of freedom excavator compared to the traditional four degrees.
It can be argued that a simulation environment is only used for the lack of money but it does give some advantages such as faster development time, especially, when considering the problem of path generation or checking a set of equations like inverse kinematics.
A road surface CAD model has been used successfully in machine control automation, at least for the road grader [4,5]. The reported speed increase was between 30 - 60%.
Based on the triangular terrain model, the theme in this paper is to derive equations for the bucket position and orientation so that paths L1, L2 and L3 are obtained (see Fig. 5.). We are also interested in defining the kind of information needed about the CAD model and in presenting it.

1.1 Excavator with 6 DOF

The main drawback of a traditional excavator with 4 DOF slightly weakens its possibilities as a robotic excavator, namely the orientation of a bucket's cutting edge is always parallel to y-axel of the coordinate system K_0 (see Fig. 1.) thus limiting the accuracy of bucket control.

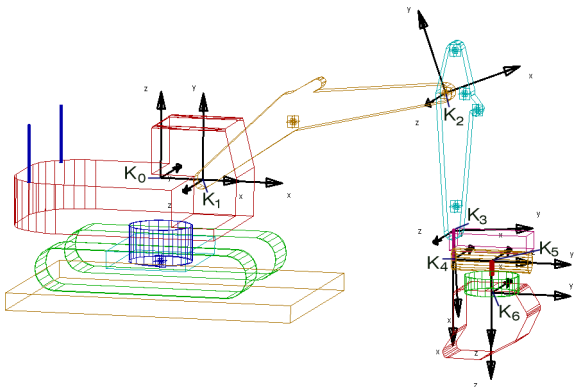

Figure 1. Six coordinate systems attached to an excavator with Rototilt.

So, even if the centre point of a bucket's cutting edge is at the desired location, the corner might travel several tenths of a centimetre misplaced off the target surface. Especially if the yaw angle of an excavator is 45°, displacement is as high as 35 cm for a one-meter-wide bucket.
Rototilt [6] is shown in Figure 2. It is widely used (and well-liked) among excavator operators because it generates two additional degrees of freedom giving more control for the driver to operate. Moreover, the necessity of moving the excavator body is greatly reduced while, for example, digging trenches with a 4 DOF excavator is only possible when the body is located in the same line with the trench. With 6 DOF this can also be done from the side of the line.

Figure 2. Rototilt and demonstration of possibilities of extra two degrees of freedom.

1.2 Short introduction to a 6 DOF excavator model and previous work

Geometry of an excavator is drawn by Adams [7], a program designed to simulate mechanical systems. For control and programming the Matlab/Simulink environment is used. Both products are linked so that the model inside Adams can be controlled from Simulink. This speeds up development work as it is clear that a program specifically designed for implementation of mathematics, is superior in this area and vice versa.

A basic model includes:

1. Analytically solved inverse kinematics for a 6 DOF excavator (Fig. 1.).
2. Positioning and orientation using 2xGPS and an inclinometer (Fig. 3.) so that the transformation matrix T_{SCS}^{MCS} between the SCS (Site Coordinate System) and MCS (Machine Coordinate System) can be calculated.
3. A velocity P-controller to the drive joints to a position calculated by inverse kinematics.

Some more information can be found in the list of references [8, 9].

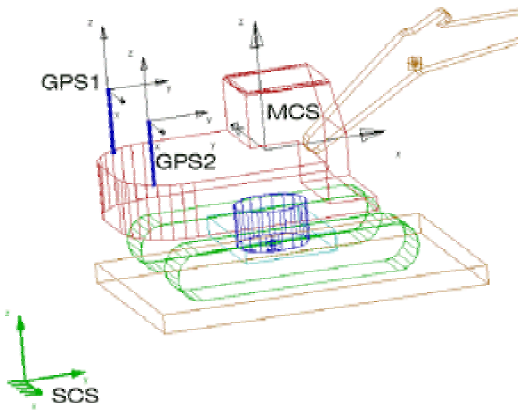

Figure 3. Coordinate systems

2 PATH GENERATION FOR 3D

At the moment the most suitable file format for excavator automation is the triangular terrain model (see Fig. 4. and 8.). Triangles define the plane and are defined by vertices. It is possible to generate a xyz-file, where points defining vertices are written in the text file. This xyz-text-file is the starting point for our study.

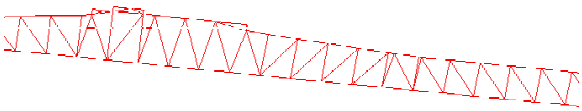

Figure 4. Part of road in a triangular format. Picture taken from MicroStation.

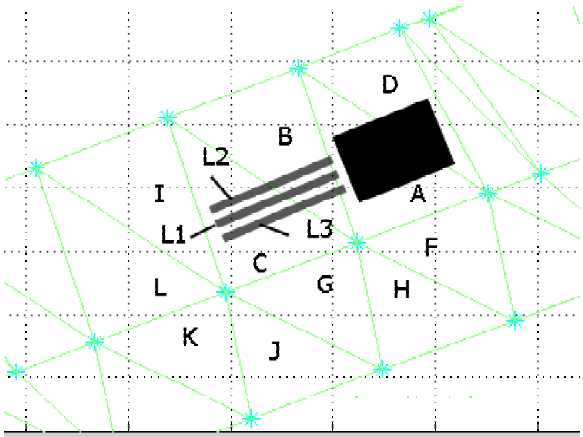

Figure 5. Schematic presentation of an excavator on the triangular surface. L1, L2 and L3 are target paths for the bucket.

2.1 Triangle compilation

As mentioned above, the xyz-format is the base of calculations. The first step is to transform the information to triangles as point information alone is not enough for the purposes of path generation. The Delaunay triangulation [10] was chosen as the method of triangulation because it is included in the Matlab function library.

After triangulation we have a matrix where every row is constructed from three points in a space and a triangle number. The triangle number is called *Tri*.

After joining the triangles together unit normal vector w is calculated by using a cross product for every triangle. Let's denote vertices as $N_1(x_1,y_1,z_1)$, $N_2(x_2,y_2,z_2)$ and $N_3(x_3,y_3,z_3)$ we get:

$$a = \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix} \quad b = \begin{pmatrix} x_3 - x_1 \\ y_3 - y_1 \\ z_3 - z_1 \end{pmatrix} \tag{1}$$

And using a cross product and division:

$$w = \frac{a \times b}{|a \times b|} \tag{2}$$

There are two possible directions for unit normal vector *w* and only one is chosen so that its direction in an SCS is up:

$$\begin{array}{c} \text{IF } w_z \geq 0, \text{ THEN } w = w, \\ \text{ELSE } w = -w \end{array} \tag{3}$$

When considering a large model of a highway with many triangles it is practical to bind shorting parameters for every triangle. A simple solution is to calculate the maximum and minimum distance between the triangle and SCS:

$$\begin{aligned} \text{distmin} &= \min\begin{pmatrix} \sqrt{x_1^2 + y_1^2 + z_1^2} \\ \sqrt{x_2^2 + y_2^2 + z_2^2} \\ \sqrt{x_3^2 + y_3^2 + z_3^2} \end{pmatrix} \\ \text{distmax} &= \min\begin{pmatrix} \sqrt{x_1^2 + y_1^2 + z_1^2} \\ \sqrt{x_2^2 + y_2^2 + z_2^2} \\ \sqrt{x_3^2 + y_3^2 + z_3^2} \end{pmatrix} \end{aligned} \tag{4}$$

When the position of a point is known in an SCS, only the triangles that fulfil the condition - a point can be in a triangle only if the distance of the point is between [distmin, distmax] - can be considered.
The resulting triangle compilation matrix size is *Tri* x 14 and the form is then as follows:

$$\mathbf{tricom}_{Tri,i} = \begin{bmatrix} N_1 & N_3 & N_3 & w & \text{distmin} & \text{distmax} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{5}$$

2.2 Basic math for triangles, lines and points

Surprisingly simple mathematics is required when working with the L1-3 path generation problem. Two basic problems can be found:
At what point p does a line intersect a plane refined by triangle vertices? This can be solved [11]:

$$\begin{aligned} p &= \frac{-(D + w \cdot N)}{w \cdot N} \\ D &= -(w \cdot N) \end{aligned} \tag{6}$$

Any of the three points can be chosen as *N*. If $w \cdot N = 0$, the line is parallel to the plane.
If a point is on the plane defined by the triangle, does it locate inside the area defined by vertices? Figure 6. presents a situation where point p (x,y,z) is located inside the triangle, defined by the points N_1, N_2 and N_3. In a simulation environment, it is practical to give two local origins which are located at points N_1 and N_2.
By studying Figure 6. it becomes clear that point *p* can be defined as the components of vectors *a*, *b* and *d* (7). In addition, the parameters of those components have to be positive. Only then is the point located inside the triangle.

$$\begin{aligned} t_1 a + t_2 b &= g_1 \\ t_3(-a) + t_4 b &= g_2 \end{aligned} \tag{7}$$

As the point is on the plane, we can discard the z-coordinates. Solving equations (7) yields:

$$\begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{pmatrix} = \begin{pmatrix} \dfrac{-b_x g_{1y} + g_{1x} b_y}{-a_y b_x + b_y a_x} \\ \dfrac{-a_y g_{1x} + g_{1y} a_x}{-a_y b_x + b_y a_x} \\ \dfrac{-d_y g_{2x} + g_{2y} d_x}{a_x d_y - d_x a_y} \\ \dfrac{a_x g_{2y} - g_{2x} a_y}{a_x d_y - d_x a_y} \end{pmatrix} \tag{8}$$

$$g_1 = \begin{pmatrix} x - x1 \\ y - y1 \\ z - z1 \end{pmatrix} \quad g_2 = \begin{pmatrix} x - x2 \\ y - y2 \\ z - z2 \end{pmatrix} \quad d = \begin{pmatrix} x3 - x2 \\ y3 - y2 \\ z3 - z2 \end{pmatrix}$$

$$a = \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix} \quad b = \begin{pmatrix} x_3 - x_1 \\ y_3 - y_1 \\ z_3 - z_1 \end{pmatrix}$$

And if all $t_i \geq 0$, the point is located inside the area defined by the vertices.

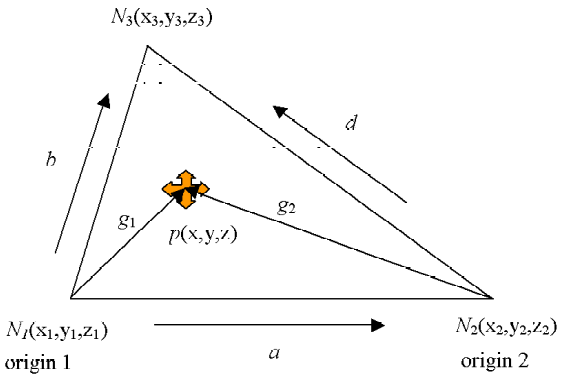

Figure 6. Point *p* inside a triangle

2.3 Identification of triangles on the path in 3D

The next step is to find triangles which are on the path of vectors L1, L2 and L3 (see Fig. 5.). We can remove the z-coordinate information to simplify calculations as the projection to the xy-plane includes all necessary geometric information.

Let's denote the end point of L1 as L1end [x,y,z] and we chose the direction of L1 in the following way:

$$x_{proj} = T_{SCS}^{MCS}\,[1..2,1] \tag{9}$$

This is not a totally accurate way, but if the excavator is close to the target plane and the reach is approximated as a sphere, it gives a close enough approximation for an initial study.
For the L1end we then get:

$$\text{L1end}[x,y] = O_{\text{MCS}}[1..2] + \text{distmaxL1}\cdot x_{\text{proj}} \tag{10}$$

, where $O_{\text{MCS}}[1..2]$ is the MCS origin in an SCS, and distminL1 is a system-specific parameter giving the maximum reach of an excavator, similarly distminL1 is defined as the minimum reach. For the complete vector L1 we can then write a discrete function:

$$\text{L1part}[x,y]_n = \text{L1end}[x,y] - n/\text{kmax} \cdot (\text{distmaxL1-pitminL1})\cdot x_{\text{proj}} \tag{11}$$

, where n = [1,2,3.. kmax] and kmax is the number of samples and gives a resolution for triangle change. Especially when n = kmax the starting point of L1 is acquired.
The procedure is the same for L2 and L3, only L2end[x,y] and L3end[x,y] points are calculated a bit differently by using the width parameter klev. It should be noticed that klev is given in 2D, and thus in 3D the distance might vary.

$$\begin{aligned} \text{L2end}[x,y] &= \text{L1end}[x,y] - \text{klev}\cdot ty \\ \text{L3end}[x,y] &= \text{L1end}[x,y] + \text{klev}\cdot ty \\ ty &= \frac{T_{SCS}^{MCS}\,[1..2,2]}{\left|T_{SCS}^{MCS}\,[1..2,2]\right|} \end{aligned} \tag{12}$$

The next step is to find triangles by using equations (8) and then reduce all unnecessary data points. An example of the resulting group is given in Figure 7. One can see how the points are chosen at the end and the start of the path, and between them only when the triangle changes.
After specifying the path matrix in 2D it is transformed to 3D by using the normal vector *w*, which was created at the triangle compilation stage. One form of equation for the plane is:

$$\begin{pmatrix} w_x \\ w_y \\ w_z \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} - \begin{pmatrix} w_x \\ w_y \\ w_z \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} = 0 \tag{13}$$

Which is solved for z:

$$z = \frac{-1}{w_z}\left(w_x x + w_y y - \begin{pmatrix} w_x \\ w_y \\ w_z \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \right) \tag{14}$$

A data set for the path point matrix in 3D is illustrated in Figure 8. The same set is in Figure 7

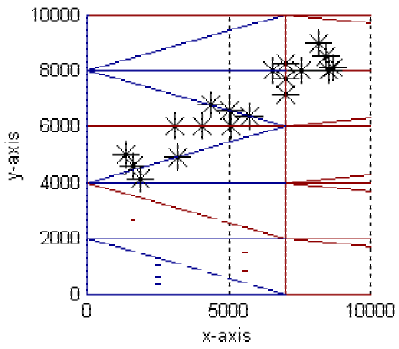

Figure 7. Points defining L1, L2 and L3 on 2D triangle surface.

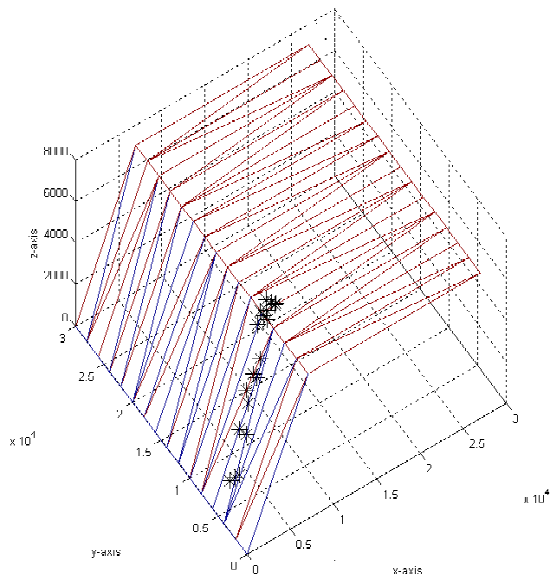

Figure 8. Points defining L1, L2 and L3 on 3D triangle surface. Data set is the same as in Figure 5.

2.4 Path and orientation of a bucket

In a previous study of a 6 DOF excavator, inverse kinematics was created to bind the K_0 and K_6 coordinate systems (see Fig. 1.) and the bucket was left out of the study. The reason for this was mainly that the bucket shape and size could vary a lot, and that it is still unclear what the best position and orientation for K_7 is. So, we need to create a transformation matrix from the bucket tip to K_6. Matrix (15) and Figure 9.

$$A_7^6 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & LV \\ 0 & 0 & 1 & -LK \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (15)$$

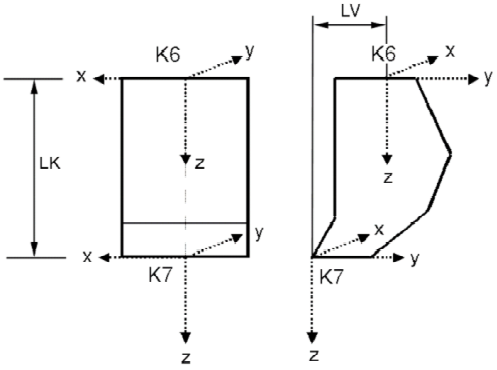

Figure 9 Coordinate systems K_6 and K_7

To simplify the creation of path vectors, a control matrix is defined by using points the created in section 3.3. Format is:

$$\mathbf{pathcont} = \begin{bmatrix} Tri & x & y & z \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \quad (16)$$

As one can see, the first column keeps the triangle number, so we can use **tricom** matrix (5) to read specific information from triangles. The last three columns define the points illustrated in Fig. 8. The path is then defined:

$$pathvec_i = \mathbf{pathcon}[i,2..4] + s \cdot \left(\frac{\mathbf{pathcont}[i+1,2..4] - \mathbf{pathcont}[i,2..4]}{\left| \mathbf{pathcont}[i+1,2..4] - \mathbf{pathcont}[i,2..4] \right|} \right) \quad (17)$$

Parameters (above) can be expressed as a function of time so that $s = \text{velocity} \cdot \text{time}$. In this way the constant speed for bucket movements is achieved. For index i, a triangle change, we get a simple condition:

$$s < \left| \mathbf{pathcont}[i+1,2..4] - \mathbf{pathcont}[i,2..4] \right| \quad (18)$$

Orientation of a bucket is easily obtained from the calculation above. Let us name the transformation matrix, which defines orientation and position of a bucket in an SCS as T_{SCS}^7. By examining Figure 1. it is clear that the z-axis of the K_7 coordinate system has to be in the opposite direction of the triangle normal vector *w.* Thus we get:

$$T_{SCS}^7[1..3,3] = -w \quad (19)$$

Logically (see Fig. 1) we can see that the direction of the y-axel should point away from the excavator. More exactly its direction should be opposite to *pathvec*:

$$T_{SCS}^7[1..3,2] = -\frac{pathvec_i}{\left| pathvec_i \right|} \quad (20)$$

We know about the x-axel that it is on the plane defined by the triangle (as is y-axel) and its direction is obtained by using a cross product:

$$T_{SCS}^7[1..3,1] = \frac{T_{SCS}^7[1..3,2] \times T_{SCS}^7[1..3,3]}{\left| T_{SCS}^7[1..3,2] \times T_{SCS}^7[1..3,3] \right|} \quad (21)$$

The transformation matrix T_{SCS}^7 defining the desired bucket position and orientation in an SCS is established.

3 CONCLUSION

It was show hot it is possible to use CAD reference for bucket positioning. Triangular terrain model was first compiled to xyz-file format and from there it was derived equations to formulate target transformation matrix T_{SCS}^7 for purposes of excavation bucket control.
Combining this result with a transformation matrix T_{SCS}^{MCS}, which in our model was created by using 2xGPS and a inclinometer, it is possible to form set points for bucket in 3D and in case of 6 DOF excavator also orientation is defined.
Also for further use a form of matrices defining target points **pathcont** and information gathered from triangles **tricom** is found to be efficient.

4 REFERENCES

[1] Stentz A., Bares J., Singh S. & Rowe P., A Robotic Excavator for Autonomous Truck Loading. Proceedings of the 1998 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems. Victoria, B.C. Canada. pp. 1885-1893, 1998.

[2] Ha Q., Santos M., Nguyen Q., Rye D. & Durrant-Whyte H., Robotic excavation in construction automation. IEEE Robotics and Automation Magazine, Vol. 9, No. 1, pp. 20-28, 2002.

[3] Bradley, D.A., Seward, D.W., Developing real-time autonomous excavation – the LUCIE story, Proceedings of the IEEE Conference on Decision and Control, Vol. 3, pp. 3028-3033,1995.

[4] Heikkilä, R., Jaakkola, M., The Efficiency of a 3-D Blade Control System in the Construction of Structure Layers by Road Grader - Automated Design-Build of Road Construction in Finland, ISARC'2002, 9th International Symposium on Automation and Robotics in Construction, Washington, DC, the United States of America, pp. 475-480, 2002.

[5] Heikkilä, R. , Jaakkola, M., Intelligent Road Construction Site - Development of Automation into total working Process of Finnish Road Construction., ISARC'2003, 20th International Symposium on Automation and Robotics in Construction, Eindhoven, the Netherlands, pp. 265-269, 2003.

[6] Indexator, company home pages, WWW-pages, Url: http://www.indexator.se, 17.6.2004.

[7] MSC.Software Corporation, company home pages, WWW-pages,
Url: http://www.mscsoftware.com, 17.6.2004.

[8] Makkonen, T., Nevala, K., Simulating an excavator equipped with a rotating bucket – the 6 DOF system, OST – 03 Symposium on machine design, ISBN 951-42-7215-3, pp. 46-52, 2003.

[9] Makkonen, T., Desing and Simulation of the bucket positioning system excavator using Adams and Simulink programs, Diploma thesis, University of Oulu, 2003. (in Finnish, confidential until 6.10.2006)

[10] George, P., Borouchaki, H., Delaunay triangulation and meshing: application to finite elemets, Paris, Hermes, p. 413, ISBN 2-86601-692-0, 1998.

[11] Katara, M., Computer graphics, PDF format, www.cs.tut.fi/~tgraf/2003/luennot/T0-39.pdf, 17.6.2004. (in Finnish)