3D Modelling for Underground Mining Vehicles

Martin Magnusson
Örebro University
martin.magnusson@tech.oru.se

Rolf Elsrud
Atlas Copco Rock Drills
rolf.elsrud@se.atlascopco.com

Lars-Erik Skagerlund
Optab
sk@optab.se

Tom Duckett
Örebro University
tom.duckett@tech.oru.se

Abstract *This paper presents the basis of a new system for making detailed 3D models of underground tunnels. The system is to be used for automated control of mining vehicles. We describe some alternative methods for matching several partial scans, and their applicability for making a complete model of a mine environment.*

1 Introduction

When building tunnels — be it for mining, road tunnels or any other purpose — one of the necessary steps is measuring the profile of the tunnel. This measurement is performed both as an active component of the excavation process and for documenting the quality of a completed tunnel.

If an accurate digital 3D model of the actual tunnel is available, it can be used for a number of purposes. Some examples include comparing the scan to a pre-built model and ensuring that new tunnels have the desired shape; measuring the volume of material removed; surveying old tunnels to ensure that they are still safe; and last, but not least, to enable autonomous control of drill rigs and other mining vehicles.

2 Application

The goal of our project is to develop a sensor system for building three-dimensional maps for use by autonomous vehicles — such as mobile robots, drill rigs, and load–haul–dump trucks — in applications such as mining and tunnel building, or for autonomous navigation.

The system will be based on an "optical radar", or lidar, permitting fast range finding in difficult environments. The scanner uses an infra-red laser mounted in a housing that can be rotated around two axes, potentially making full spherical scans, and is described in section 3.

Several problems are associated with this kind of modelling. Naturally, it is only possible to see a very limited part of the mine at a time, and furthermore there is a lot of self-occlusion from the vehicle itself. No matter where on the rig the sensor is placed, the drill booms and part of the rig itself will always be in the line of sight between the sensor and the walls. Therefore it is crucial to have reliable scan registration — that is, matching partial scans taken from different view points to make a complete model. Additionally, the environment can often be wet or dusty. Airborn dust and wet, specular surfaces can lead to quite noisy sensor readings, which makes scan registration more difficult.

When blasting a tunnel, the first step is to drive a drill rig up to the rock face at the current end of the

Photo: Lars Lindgren/Atlas Copco.

Figure 1. An illustration of the sensor and an Atlas Copco drill rig.

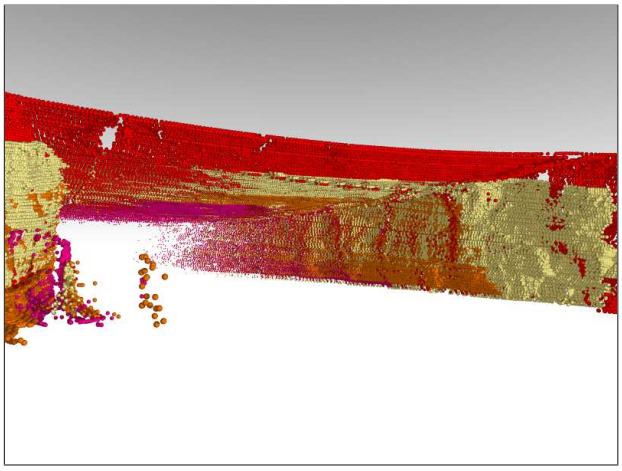

Figure 2. Four registered scans from the Kvarntorp mine, with different colours for clarity.

tunnel and drill holes in a pre-defined pattern. These holes are then filled with explosives, and the rock face is blasted away. After venting out the dust and poisonous gases coming from the blast, the debris is hauled away, the rock in the new part of the tunnel is secured so that there is no risk of the tunnel collapsing, and then the process is repeated with the drill rig moving in again.

For obvious safety reasons, no people can be present in the tunnel before the venting is completed. If the drilling, blasting and removal processes were automated, the vehicles could be driven back shortly after the blast and the excavation would be much more safe and efficient.

The drilling can be performed either manually or in a limited automated mode. Currently, the booms on which the drills are mounted have no sensing capabilities. The rock face is often quite uneven, so to avoid hitting a protruding part of the surface with the drill, the booms must make quite large movements when running in automatic mode. Instead of following the closest possible path from one drill hole to the next, the drill must be moved straight back over a fairly large distance, then sideways to the new bore hole position, and then back in to the rock face again. If a detailed 3D model of the surface were available, this process could be performed much faster.

Today's tools for tunnel profile scanning are either very slow or very expensive, and profiling currently needs to be performed separately from any other activity in the tunnel. The rock drill industry has been searching for tools for a fast, active, and cheap solution to this problem for a long time. One of the demands is that the measuring equipment should be incorporated into the machine — which in the case of Atlas Copco's products is the drill rig.

Fast and reliable three dimensional scanning and model building is also highly applicable in many other different industrial, medical and robotic applications.

Traffic simulation The Swedish National Road Administration are using a tunnel simulator which gives rescue personnel, police and others the possibility of practising rescue missions and traffic control for different types of accidents in tunnels, without disturbing traffic. Automatic 3D modelling could make it easier to build true models of the sites used for simulation.

Archaeology 3D scanning could also be used to measure archaeological artifacts and historical sites. A digital model will preserve a deteriorating site pretty much indefinitely, and will also allow objects to be studied remotely or processed automatically. A good example is the Stanford Digital Forma Urbis Romae project, where fragments of a shattered enormous stone map of ancient Rome are analyzed with scan matching methods, trying to reconstruct the puzzle [5].

Architecture and construction A 3D model of a building can be used to verify that it was constructed according to plans, and it can also be used by architects when planning a renovation or extension.

Mobile robots Robots or autonomous mining equipment could use 3D maps for route planning and obstacle avoidance. Automated modelling is especially useful in hard to reach areas, not least for extraplanetary vehicles such as the Mars rovers. Scan matching can also be used for localisation — recognising a location within a previously built model of the world from a scan of the current surroundings.

3 Scanner

The scanner we use is built by Optab Optronikinnovation AB. It is based on an infra-red laser and measures the distance to the closest surface by exploiting the phase shift of the reflected laser beam.

A laser range finder is the only reasonable sensor type for building 3D maps with the required level of detail. Other sensor types — such as radar, sonar, stereo vision, and projected-light triangulation — would not be able to provide the same resolution when scanning tunnels several metres wide with smooth textures.

Our scanner is built to be a cost-efficient, flexible and rugged sensor that can cope with the challenging environment of an active mine. We currently use a prototype, but figure 1 is an illustration of the finished product. The laser is mounted in a housing that is rotated on two axes so that the beam is swept in a spherical fashion, measuring points along the longitude lines of a spherical coordinate system. Both the longitudinal and latitudinal resolutions can be adjusted continuously to trade off accuracy for speed. When the vehicle is stationary, the rotation speed can be turned down to get a resolution of 5 cm or less on surfaces 25 m away; and when using the scanner for localisation on a moving vehicle, it can be adjusted for a higher update frequency. The range is approximately 3 – 45 m.

4 Registration

Three-dimensional pair-wise registration is the problem of matching two overlapping data sets to build a consistent model. If the exact position and rotation (the position-rotation tuple is generally referred to as the

pose) of the scanner is known for each scan, with respect to some global coordinate system, this is a straightforward task. However, unless some sort of localisation infrastructure, such as radio frequency tags, that information is generally difficult to obtain with any accuracy.

Measuring the distance between scans from odometry — calculating the distance travelled from the number of wheel turns — is highly unreliable. Even small measurement errors for each turn will accumulate very quickly over time. There may also be other, less systematic errors, stemming from wheel slip, differing tyre pressure or times when the vehicle is airborne.

However, registration algorithms find a good pose if given an initial estimate that is close enough to the true solution. How close it has to be depends on the shape of the data.

4.1 ICP

The *iterative closest point* (ICP) algorithm is widely used today for registration of 3D point clouds and polygonal meshes. ICP iteratively updates and refines the relative pose by minimising the sum of squared distances between corresponding points in the two data sets, and was introduced by Besl and McKay in 1992 [1].

Since its conception, a large number of variants have been developed, and a good survey of different variations of ICP is presented in [6]. It can be thought of as a modular algorithm, with different ways to perform its different parts.

When matching high-resolution scans, it is usually necessary to choose a subset of points to compare, in order to reduce the running time. This can be done using a uniform or random subsampling. If topology data in the form of mesh faces or surface normals at the measured points are available, it is also possible to subsample in a more selective manner, such as choosing points where the normal gradient is high. The preferred strategy for choosing points varies with the general shape of the data. Surfaces that are generally flat and featureless, such as long corridors, are notoriously difficult, but choosing samples such that the distribution of normals is as large as possible [6] forces the algorithm to pick more samples from the features that do exist (incisions, crevices and the like), and increases the chance of finding a correct match.

Once a set of points has been chosen, the algorithm proceeds to finding corresponding points on the other data set. This search is where most of the execution time is spent. The naive way to do this is to pick for each point its closest neighbour. While this will not in general be the correct corresponding point, especially if the data sets are far from each other, successive iterations will still usually converge to a good solution.

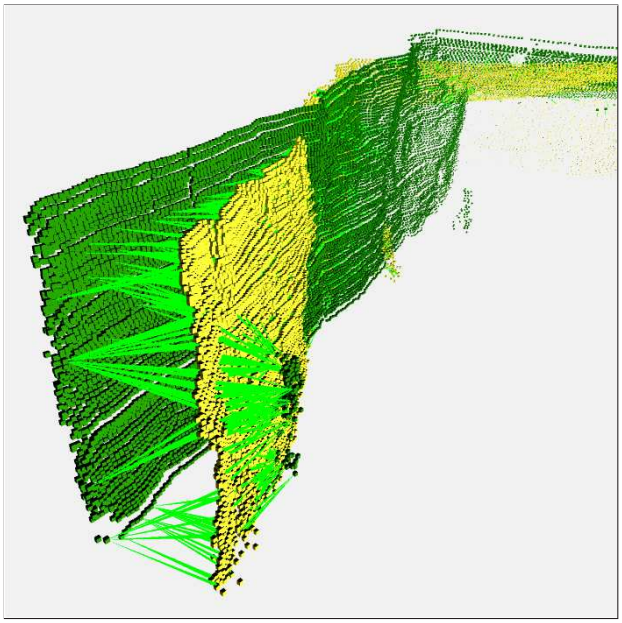

Figure 3. Matching two misaligned scans from a mine tunnel using ICP. The yellow (light) scan is matched to the dark green scan. The point-to-point correspondences are shown with bright green (medium gray) arrows.

This correspondence finding is illustrated in figure 3. If topological data are available, more sophisticated point finding methods can be applied. One alternative is "normal shooting" [3] — finding the intersection of a ray originating at the source point in the direction of the source point's normal and the destination surface. It can also be wise to only select point pairs where the normals are sufficiently well aligned, to avoid matching a point from a wall to one from the ceiling, for example.

To speed up the nearest-neighbour search, the points in the target data set are commonly stored in a kd-tree. If there are n points in the target data, and one wants to find the nearest neighbours of k points, the time needed for the search grows as $O(n^{2/3} + k)$ for 3D data, plus the $O(n \log n)$ time needed for building the tree structure.

Which method to choose for this step also depends on the geometry of the input. Picking the closest point is a robust method for difficult surfaces, but there are other strategies which are faster for easier surfaces, with many small-scale and large-scale features. There may also be a trade-off between strategies which give convergence after few iterations and ones which take little time per iteration. Rusinkiewicz [6] suggests finding a point in the target scan by projecting a ray from the scanner viewpoint through the current point in the source scan. However, this method only works well when the camera is perpendicular to the surface, which is usu-

ally not the case in a tunnel. It is better to either use the nearest point, or to perform normal shooting.

To minimise the influence of badly selected point pairs, different weights can be assigned to different pairs. This can be done in a number of ways. One alternative is to set the weight proportional to the point-to-point distance and have lower weights for points further apart. While this may seem like a sensible idea, in some cases pairs with longer distances are more important than others. If surface normals are available, larger weights can be assigned to point pairs with aligned normals ($w = n_1 \cdot n_2$). Another version is weighting based on scanner characteristics so that points where the uncertainty of the scanner is higher get lower weights. The Optab scanner records the signal strength for each measured point, in addition to the distance. In this case, smaller weights might be assigned where points have a weaker reflected signal. Some combination of the above could also be implemented. For applications in which the surface texture is also recorded, higher weights can be assigned to pairs in which the points have similar colours [4], although this is rarely an option in an underground environment. Which weighting scheme to use is highly data dependent, but we have found that simple constant weighting for all pairs works well when the scans are initially far apart, and linear weighting based on the distance is better for well-aligned scans where different parts are occluded.

Some pairs will also need to be rejected entirely, in order to eliminate outlier points. This can be done using some heuristic to reject pairs where the points are far apart. Points lying on mesh boundaries should always be rejected. Otherwise, points from non-overlapping sections of the data may cause a systematic "drag" bias (see figure 4).

Since it is difficult to determine the boundary points for point cloud data, it can sometimes be good to use a decreasing distance threshold for rejection instead. This way, pairs that are separated by a large distance are used in early iterations to bring the data sets closer, but in later iterations pairs where the points are far from each other are likely to be non-overlapping points matched to boundary points, and are rejected. For many cases, this leads to a better final result; although for some cases, where the scans are far apart initially, linearly decreasing the threshold will make it too short before the meshes are close, and in such cases, it can be better to use a constant threshold. See figure 5 for an example of the results. The reason that the version with a decreasing threshold always takes the same amount of time to converge is that the algorithm must allow the threshold to decrease to zero, as the change in the rejection threshold would have no effect otherwise. In these cases, the threshold was set to decrease to zero in 50 iterations.

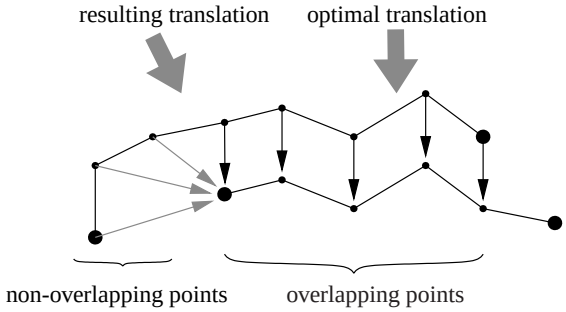

Figure 4. When the two data sets do not overlap completely, as is normally the case, allowing point pairs on the boundaries can introduce a systematic error to the alignment process. If the pairings including the non-overlapping points in the figure were rejected, the resulting transformation would be straight down, resulting in a perfect match in this case.

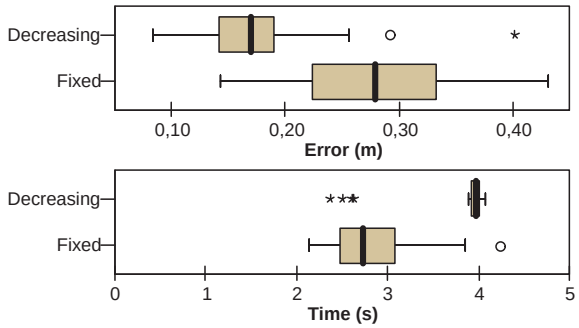

Figure 5. Comparison of running ICP with a fixed distance rejection threshold and a linearly decreasing threshold for two of the scans illustrated in figure 2.

The box plots used show the distributions of the total results. The median value is indicated by the thick black bar. The ends of the box show the first and third quartiles. The "whiskers" are then drawn towards the minimum and maximum values, but the maximum length is 1.5 times the interquartile range. Any values beyond the ends of the whiskers are considered outliers, and drawn as individual points. This method is the one commonly accepted among statisticians for drawing box plots.

Finally, the measured distances between the point pairs are minimised. There is a closed form solution for determining the transformation that minimises the point-to-point error, which is described in [1].

Then the process is repeated again, with a new selection of points, until the algorithm has converged.

4.2 NDT

The *normal distributions transform* (NDT) method for registration of 2D data was recently presented in [2].

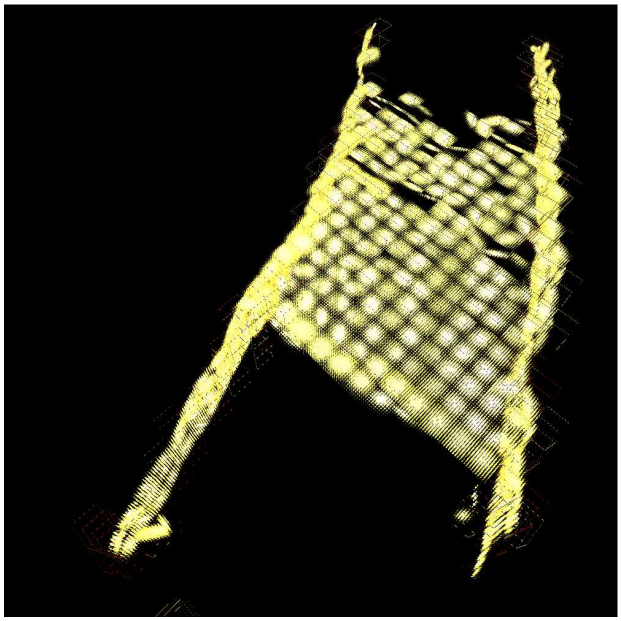

Figure 6. The probability functions used by NDT for one of the tunnel sections shown in figure 2, seen from above. Brighter, denser parts represent higher probabilities. The cells here have a side length of 1 m.

The key element in this algorithm is a new representation for the target point cloud. Instead of matching the source point cloud to the points in the target directly, the probability of finding a point at a certain position is modelled by a linear combination of normal distributions. This gives a piecewise continuous representation of the target data, which makes it possible use standard and mathematically sound numerical optimisation methods for registration.

The first step of the algorithm is to subdivide the space occupied by the target data into regularly sized cells (squares or cubes). Then, for each cell b_i, the average position $\mathbf{q}_i$ of the points in the cell and the covariance matrix $\mathbf{C}_i$ (also known as the dispersion matrix) are calculated.

$$\mathbf{q}_i = \frac{1}{n}\sum_j \mathbf{x}_j \tag{1}$$

$$\mathbf{C}_i = \frac{1}{n-1}\sum_j (\mathbf{x}_j - \mathbf{q}_i)(\mathbf{x}_j - \mathbf{q}_i)^T \tag{2}$$

$x_{j=1,\ldots,n}$ are the points contained in the cell.

The probability that there is a point at position $\mathbf{x}$ in cell b_i can then be modelled by the normal distribution $N(\mathbf{q}, \mathbf{C})$, like this:

$$p(\mathbf{x}) = c\exp\left(-\frac{(\mathbf{x}-\mathbf{q}_i)^T\mathbf{C}_i^{-1}(\mathbf{x}-\mathbf{q}_i)}{2}\right) \tag{3}$$

where c is a constant that can be set to one. Here $\mathbf{q}_i$ and $\mathbf{C}_i$ are the average and covariance for the cell that contains point $\mathbf{x}$. The normal distribution in 3D has the shape of an ellipsoid centred around the average. If the covariance of the points is small, the distribution will be close to spherical, and if the points are highly correlated, it will be closer to a line or a disc.

Now a score function is needed to give a measure of the fitness of a particular pose. Since optimisation problems are generally formulated as minimisation problems, the score function is defined so that good parameters yield a large negative number. The rotation and translation parameters can be encoded in a vector $\mathbf{p}$. Given a set of points $\mathbf{x}_i$, and the transformation function $T(\mathbf{p}, \mathbf{x})$ to transform a point in space, the score function $s(\mathbf{p})$ is defined as the negated sum of probabilities that the transformed points of the source data set are actually lying on the target surface.

$$s(\mathbf{p}) = -\sum_i p(T(\mathbf{p}, \mathbf{x}_i)) \tag{4}$$

The score function can than be optimised using Newton's method, for example. Further details on how to find the first and second order derivatives of this function needed by Newton's method can be found in [2].

Given the vector of transformation parameters $\mathbf{p}$, Newton's algorithm can be used to iteratively solve the equation $\mathbf{H}\Delta\mathbf{p} = -\mathbf{g}$, where $\mathbf{H}$ and $\mathbf{g}$ are the Hessian and gradient of s. The increment $\Delta\mathbf{p}$ is then added to the current estimate of the parameters, so that $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$.

For brevity, let $\mathbf{q} \equiv T(\mathbf{p}, \mathbf{x}_i) - \mathbf{q}_i$ and drop the index i for the covariance matrices. The entries for the gradient of the score function can then be written as

$$g_i = \frac{\delta s}{\delta p_i} = \mathbf{q}^T\mathbf{C}^{-1}\frac{\delta \mathbf{q}}{\delta p_i}\exp\left(\frac{-\mathbf{q}^T\mathbf{C}^{-1}\mathbf{q}}{2}\right) \tag{5}$$

The entries of the Hessian are

$$H_{ij} = \frac{\delta s}{\delta p_i \delta p_j} = \exp\Big(\frac{-\mathbf{q}^T\mathbf{C}^{-1}\mathbf{q}}{2}\Big)\Bigg(\Big(\mathbf{q}^T\mathbf{C}^{-1}\frac{\delta \mathbf{q}}{\delta p_i}\Big)\Big(-\mathbf{q}^T\mathbf{C}^{-1}\frac{\delta \mathbf{q}}{\delta p_j}\Big) + \mathbf{q}^T\mathbf{C}^{-1}\frac{\delta^2 \mathbf{q}}{\delta p_i \delta p_j} + \frac{\delta \mathbf{q}}{\delta p_j}^T\mathbf{C}^{-1}\frac{\delta \mathbf{q}}{\delta p_i}\Bigg) \tag{6}$$

The first-order and second-order partial derivatives $\frac{\delta \mathbf{q}}{\delta p_i}$ and $\frac{\delta^2 \mathbf{q}}{\delta p_i \delta p_j}$ in the previous equations depend on the transformation function.

For the 2D case presented in [2], these derivatives are very inexpensive to compute. Since only small angles need to be considered, the corresponding terms are similarly cheap in the 3D case as well. Rotations are represented here with three angles ϕ_x, ϕ_y, and ϕ_z, rotating each point around the principal coordinate axes.

$$T(\mathbf{x}) = \mathbf{R}_x\mathbf{R}_y\mathbf{R}_z\mathbf{x} + \mathbf{t} \tag{7}$$

$$= \begin{bmatrix} \cos\phi_y\cos\phi_z & -\cos\phi_y\sin\phi_z & \sin\phi_y \\ \sin\phi_x\sin\phi_y\cos\phi_z + \cos\phi_x\sin\phi_z & -\sin\phi_x\sin\phi_y\sin\phi_z + \cos\phi_x\cos\phi_z & -\sin\phi_x\cos\phi_y \\ -\cos\phi_x\sin\phi_y\cos\phi_z + \sin\phi_x\sin\phi_z & \cos\phi_x\sin\phi_y\sin\phi_z + \sin\phi_x\cos\phi_z & \cos\phi_x\cos\phi_z \end{bmatrix} \mathbf{x} + \mathbf{t} \tag{8}$$

$$\approx \begin{bmatrix} 1 & -\phi_z & \phi_y \\ \phi_z & 1 & -\phi_x \\ -\phi_y & \phi_x & 1 \end{bmatrix} \mathbf{x} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{9}$$

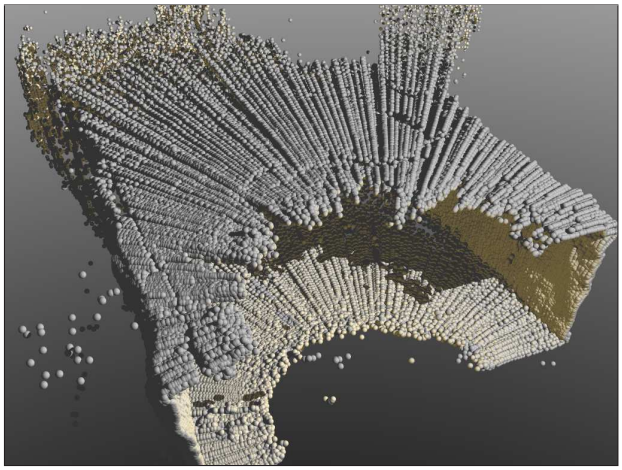

Figure 7. The source scan from the `junction` data set.

The 3D transformation function is shown in equation (8). For small ϕ, $\sin\phi \approx \phi$ and $\cos\phi \approx 1 - \frac{\phi^2}{2}$, and $\phi^2 \approx 0$. If these approximations are exploited, many of the terms reduce to zero. The simplified transformation function is shown in equation (9), and the first-order derivatives with respect to the transformation parameters in $\mathbf{p}$ can be found in equation (10). Here, $\frac{\delta\mathbf{q}}{\delta p_i}$ is the i-th column of $\mathbf{J}$. The second-order partial derivatives all reduce to zero. Using these approximations is motivated by noting that if the scans need to be rotated much, there is little chance that the registration will succeed anyway.

$$\mathbf{J} = \begin{bmatrix} 1 & 0 & 0 & 0 & x_3 & -x_2 \\ 0 & 1 & 0 & -x_3 & 0 & x_1 \\ 0 & 0 & 1 & x_2 & -x_1 & 0 \end{bmatrix} \tag{10}$$

5 Results

ICP and NDT were compared on a data set with two scans from the underground test mine at Kvarntorp in Sweden. The scans are made with different resolutions from the same position at the end of one tunnel, including a small passage to a neighbouring tunnel (see figure 7). The source scan has 139 642 points and the target scan has 72 417 points. These scans have obvious large-scale features, such as the end face of the tunnel and the passage. The ground truth pose is simply zero translation and rotation, since the scans are taken from the same position and orientation.

The times reported for ICP include the creation time for a kd-tree used to speed up the nearest-neighbour search. This only needs to be done once for each data set as long as it is not transformed. For a mapping application, where scans are matched in succession to the last one, these numbers are a good indication of the true time needed, but for an object recognition task where several models are registered to one fixed environment, it would be more fair to count only the registration time. The time needed to create the kd-tree for the target scan in this case is approximately 4.4 s.

For these tests we used ICP with nearest-neighbour correspondences and constant weights. A fixed rejection threshold of 1 m was used. NDT uses axis/angle rotations and a cell size of 1 m. Because no surface normals were available for these scans, uniformly random sampling was used. 3000 points were taken from the source scan, and the complete target scan was used. The initial estimate for each run was the ground truth plus an added translation vector $\mathbf{e}_t$, with different orientations for each run, but a fixed magnitude. The experiments were run on a Dell Latitide D800 with a 1600 MHz CPU and 512 MB of RAM.

As can be seen from figure 8, NDT works very well for data with large features. If a coarser sampling is used, or the target scan is also subsampled, it is possible to get running times comparable to those of NDT from ICP, but doing so also further reduces the accuracy proportionally.

6 Conclusion

We have found that for our application, scan registration with the novel NDT algorithm is generally more efficient and accurate, as long as there are sufficient

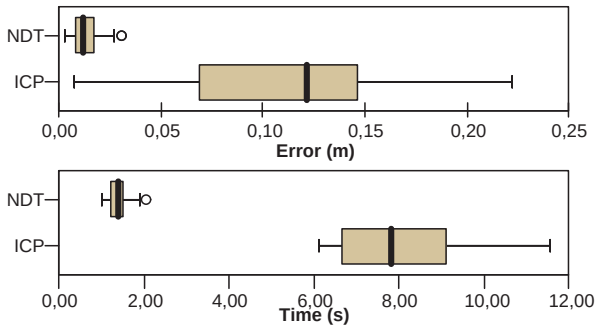

Figure 8. Registration errors and running times for 100 registrations with initial error $\|\mathbf{e}_t\| = 1$. Errors below 0.15 m can be considered good matches.

large-scale features in the data set. Data lacking features, however, are difficult for all registration algorithms.

It should also be noted that neither of these algorithms are completely "hands-free". It is not possible to find a set of settings that works for all cases. Sometimes, depending on the shape of the data and the amount of error in the inital pose, the output will not be a correct match, and then the registration will have to be done again, using a different pose estimate, a larger number of samples, a different weighting scheme, a different cell size, or simply a different random seed.

It would be desirable to investigate how to further improve the robustness of scan registration without sacrificing speed, and we intend to work on this in the near future.

References

1. Paul J. Besl and Neil D.McKay. A method for registration of 3–d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239 – 256, February 1992.
2. Peter Biber and Wolfgang Straßer. The normal distributions transform: A new approach to laser scan matching. In *IEEE/RJS International Conference on Intelligent Robots and Systems*, 2003.
3. Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, April 1992.
4. Andrew Johnson and Sing Bing Kang. Registration and integration of textured 3-d data. Technical Report CRL96/4, Cambridge Research Lab, 1996.
5. Geoff Koch. Technology speeds up efforts to piece together ancient marble map of rome. *Stanford Report*, April 19 2004.
6. Szymon Marek Rusinkiewicz. Efficient variants of the ICP algorithm. In *The Third International Conference on 3D Digital Imaging and Modeling*, 2001.