A novel surface segmentation approach for robotic manipulator-based maintenance operation planning

Gavin Paul[a,*], Ngai Kwok[b], Dikai Liu[a]

[a]*Centre of Excellence for Autonomous Systems, University of Technology, Sydney, Australia*
[b]*School of Mechanical and Manufacturing Engineering, The University of New South Wales, Australia*

Abstract

This paper presents a novel approach to segmenting a three-dimensional surface map by considering the task requirements and the movements of an industrial robot manipulator. Maintenance operations, such as abrasive blasting, that are performed by a field robot manipulator can be made more efficient by exploiting surface segmentation. The approach in this paper utilises an aggregate of multiple connectivity graphs, with graph edges defined by task constraints, and graph vertices that correspond to small, maintenance-specific target surfaces, known as Scale-Like Discs (SLDs). The task constraints for maintenance operations are based on the characteristics of neighbouring SLDs. The combined connectivity graphs are analysed to find clusters of vertices, thus segmenting the surface map into groups of related SLDs. Experiments conducted in three typical bridge maintenance environments have shown that the approach can reduce garnet usage by 10%-40% and reduce the manipulator joint movements by up to 35%.

Keywords: Surface segmentation, Manipulator trajectory planning, Maintenance operations.

1. Introduction

Rust and paint removal, along with the reapplication of a protective coating are essential operations in the maintenance of large steel infrastructure such as bridges. The assistance of an autonomous robotic system which incorporates an industrial robot manipulator can potentially increase productivity and reduce the workers' exposure to risk. However, an autonomous manipulator-based system that is placed in a partially known or unknown complex environment must determine the 3D surface geometry by exploring the environment before commencing maintenance tasks. Advances in sensing technologies, such as laser

*Corresponding author
Email addresses: Gavin.Paul@uts.edu.au (Gavin Paul), nmkwok@unsw.edu.au (Ngai Kwok), Dikai.Liu@uts.edu.au (Dikai Liu)

range scanners [11], depth cameras [27], and manipulator-based exploration approaches [18][21][13] make it possible to measure the geometry of surfaces with high accuracy even in complex 3D environments. However, the geometry data that is gathered during exploration must first be processed before it can be utilised for manipulator trajectory planning and coverage planning.

It is challenging to represent the geometry data from sensors so the data can be easily and effectively used to plan manipulator movements that interact with surfaces in an environment [7]. The sensor data that is returned from a depth camera or laser range scanner is initially in a primitive form: distance measurements from the sensor to an object, and relative bearing of each distance measurement. When the sensor is attached to the end-effector of a manipulator, which provides accurate information about the position and orientation of the sensor, then the location of a surface point can be determined with high confidence in the 3D coordinate frame of the manipulator. A group of these 3D points, known as a point cloud, can be joined to produce a mesh [6][31] that consists of multiple triangles. Where a robot manipulator must position/orientate a tool within distance bounds of the surface and between angle to the surface normal bounds then pose selection [18][31] can be used to determine the joint angles. Pose selection is an optimisation process used to determine the joint angles of a manipulator, so there is no collision with obstacles in the environment, and so the end-effector is positioned and orientated within a set of task constraints. However, due to sensor noise, the triangle mesh often contains large variations between neighbouring surface normals [15], which makes it difficult to determine a trajectory such that the tool at the end-effector is orientated correctly as it points at a continuous surface [28]. In a given position and orientation, the venturi nozzle that is commonly used in abrasive blasting is observed to clean a circular spot on the surface. A surface representation is required that is smoother than the sensed mesh, and which is more related to the task of trajectory planning for a circular blasting spot. When performing abrasive blasting on disconnected or dissimilar surfaces, the process must be stopped when moving between segments, to guarantee that only the correct areas are covered, and so as to minimise the wastage of the abrasive blasting material (i.e. garnet). Therefore, when a manipulator must interact with surfaces in an environment, such as in abrasive blasting, a map of the environment is required. The map must contain sets of small maintenance-specified targets representing small surface areas that need to be aimed at when performing task-specific manipulator pose selection. Ideally related targets should then be grouped into surface segments so as to improve the efficiency of the system.

Developing a map with a high-level of abstraction (i.e. object recognition) has been shown to be valuable for motion planning of indoor mobile robots [33][30]. Representations at low levels of abstraction that are popular in the literature for improving the map building required for planning tasks include, point clouds fitted with polynomials [2][24], marching cubes [16], marching tetrahedrons [14] and the volumetric method [6] that produces a high resolution triangle mesh. Algorithms also exist which generate local machining tool paths directly from a point cloud [28]. In manipulator pose selection that considers task-specific constraints to position a point projected from the end-effector, then small 3D targets which approximate a surface [32][12] have been shown to aid manipulator trajectory planning [23] in complex environments. In manipulator trajectory planning the aim is to find a collision-free path made up of discrete manipulator poses with a small angular differential between consecutive poses. In steel bridge maintenance operations [19] it has been shown in [29] that manipulator trajectories can be optimised for a small surface area in order to improve efficiency, provided that targets are placed in close proximity to each other with a small amount of overlap.

When a surface map is large and complex then planning a non-stop manipulator trajectory over the entire set of targets in the map is computationally expensive, possibly even intractable, and hence the resulting path is likely to be inefficient. An inefficient blast path wastes blasting material (i.e. garnet) and time, and will redundantly move the loaded manipulator joints through unnecessary blasting configurations - thus increasing wear. It has been shown that by partitioning manipulator coverage problems into multiple planning spaces it is possible to significantly reduce the computational efforts required for planning [4]. An operator could assist the planning process by manually segmenting surfaces. However, this type of interaction would be time consuming and when the manipulator movement considerations are included, the ideal surface segments may be irregular shapes that are not obvious to an operator. Techniques exist in the literature for autonomous surface data segmentation [22][10][1] and refinable surface enclosures for uniform spray painting coverage [3]. Segmentation approaches also exist for use by climbing robots on planar surfaces [9]. These approaches typically group planar surface features into segments based only on geometry considerations. Segmentation techniques need to be extended so as to include the task-specific constraints that are relevant to manipulator pose selection. Therefore, an approach is required that processes the map data gathered so as to create targets that are representative of surfaces in the environment. The approach should also include an algorithm to cluster the targets into segments that are based on the surface geometry, the maintenance task requirements, and the surface-interaction manipulator poses.

This paper presents an approach for segmenting a 3D surface map, which is generated through exploration, by considering the task requirements and the movements of an industrial robot manipulator that performs the task. Initially, the surface map is represented as partially overlapping Scale-Like Disc (SLD) targets so as to facilitate a manipulator-based coverage task. Relevant maintenance-operation task constraints are used to generate SLD connectivity graphs, which are analysed to cluster vertices, thus segmenting the surface map into groups of related SLDs. Surface segmentation based upon task requirements for manipulator motion planning will be shown to improve the efficiency of manipulator-based maintenance operations. Section 2 describes the data collection and the algorithm for target generation which transforms the fused maps into a more usable and compact format via Principal Component Analysis (PCA). Section 3 describes the algorithm to populate graphs based upon a set of task constraints and then groups the targets by performing cluster analysis.

Section 4 presents experimental results which show the segmentation outputs for a number of environments along with the efficiency improvements. Finally, Section 5 presents the conclusions and future work.

2. Task-specific Surface Representation

In order for an industrial robot manipulator to autonomously perform a maintenance task on the surfaces of an initially unknown environment, exploration is needed to generate a map of the environment [18]. To explore 3D environments using a manipulator and a sensor such as a depth camera [27] or a laser range scanning sensor [11], the manipulator is manoeuvred through a sequence of viewpoints that are selected to maximise the quality of the map generated. The result of exploring from multiple viewpoints is a point cloud that represents the surfaces of an environment. It has been shown [29][23][19][31] that for manipulator pose selection and trajectory planning in abrasive blasting operations it is advantageous to have a small target to aim at. The venturi nozzle that is used in maintenance operations must be kept at a specific range and orientation to the surface so as to achieve the desired surface finish characteristics. Fig. 1 shows a nozzle attached to the end-effector of a robot manipulator.

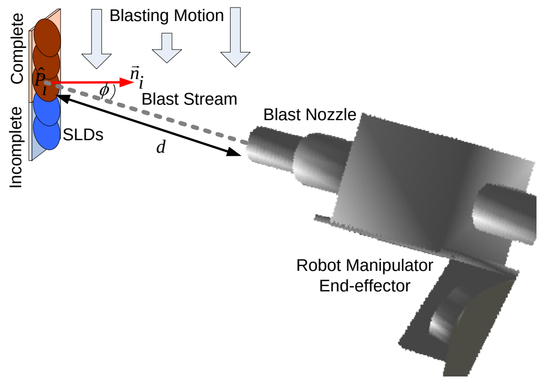


Figure 1: Abrasive blasting operation utilising scale-like disc targets (SLDs).

A manipulator pose selection approach [18] can be used to find joint configurations that correctly position and orientate a tool on the end-effector, while avoiding collisions and manipulator joint limits. Fig. 1 shows the important maintenance task constraints: distance and orientation to a surface. These constraints must be adhered to by the pose selection process. Therefore, it is advantageous to utilise a task specific target called a Scale-Like Disc (SLD) that includes a nozzle-specific diameter, a point, $\hat{\mathsf{P}}_i$, to aim at along with a surface normal, $\vec{n}$. In maintenance operations, such as abrasive blasting, a single blast spot is observed to be circular [5]. In Fig. 1, where three SLDs out of the five SLDs have been blasted, the blasting spot is being moved downwards in a straight line such that the blast path has a rectangular shape with half-circles at either end; a shape that can be approximated by several partially overlapping SLDs. This section formalises the range data collection process and presents an algorithm to generate specifically sized and positioned SLDs.

2.1. Point Cloud Map Generation

When a laser range scanner is mounted on the end-effector of a robot manipulator, and makes horizontal scans along its plane of orientation. Ranges, r_i, to reflecting points on an object's surface are returned together with the corresponding bearings, θ_i. The angle between the ith and $(i+1)$th rays with the bearings, θ_i and θ_{i+1}, is equivalent to the constant angular resolution of the scan. A scan refers to one complete set of range and angle measurements.

Given the manipulator's joint configuration, the instantaneous location and orientation of the sensor can be obtained with reference to a 3D coordinate system, T_0 located at the base of the robot manipulator (Fig. 2). Given a manipulator pose, $\vec{Q} = [q_1, q_2, \dots q_6]^T$ that describes the angular position of each manipulator joint, and using forward kinematics, it is possible to compute the position and orientation of the manipulator's end-effector where the sensor is mounted.

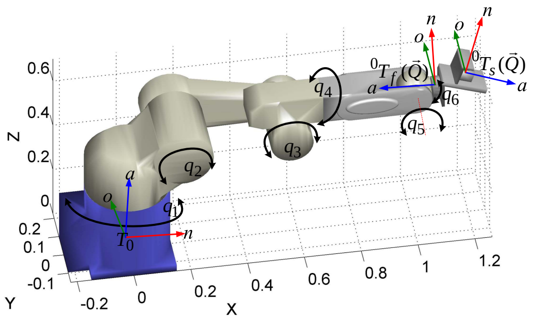


Figure 2: The robot joints and the coordinate frames: robot base, end-effector and sensor.

A position and orientation combination of an end-effector can be expressed in the 4×4 rotation and translation homogeneous transformation matrix, $^0T_f(\vec{Q})$, by performing transformations based upon the manipulator model and the manipulator joint angles q_i for $i \in \{1, \dots 6\}$ as,

$$^0T_f(\vec{Q}) = \prod_{i=1}^{6} {}^{i-1}T_i(q_i). \tag{1}$$

The transformation matrix between the end-effector and the sensor is denoted as ^fT_s. Together these two matrices describe the viewpoint in the manipulator base coordinate frame, $^0T_s(\vec{Q}) = {}^0T_f(\vec{Q})^fT_s$. The sensor position is $[x, y, z]^T$, the centre laser ray is in the direction of the vector, $[a_x, a_y, a_z]^T$, the ray's axis of rotation is $[n_x, n_y, n_z]^T$, and the $\pm 90°$ laser ray direction is $[o_x, o_y, o_z]^T$. Then the transformation matrix is,

$$^0T_s(\vec{Q}) = \begin{bmatrix} n_x & o_x & a_x & x \\ n_y & o_y & a_y & y \\ n_z & o_z & a_z & z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

The location of a data point, $\mathbf{p}_i = [\mathsf{x}_i, \mathsf{y}_i, \mathsf{z}_i]^T$, on an object returned from the ith ray with range r_i and angle to the centre ray θ_i is calculated as,

$$[\mathsf{x}_i, \mathsf{y}_i, \mathsf{z}_i, 1]^T = {}^0T_s(\vec{Q})\,[0, -r_i \sin\theta_i, r_i \cos\theta_i, 1]^T\,. \tag{3}$$

The laser scanner is driven in a tilting movement using the fifth joint of the manipulator, q_5, such that both the end-effector transformation matrix, $^0T_f(\vec{Q})$, varies along with the viewpoint, $^0T_s(\vec{Q})$. The tilting motion enables the laser sensor to deliver a set of range data to the 3D surfaces in the environment that surrounds the manipulator. The accurately calibrated manipulator provides time-stamped configuration data that is combined with the continuously-acquired range data. By combining the range data with the positional information of the end-effector, a set of n points, is generated as a point cloud,

$$\mathsf{P} = \{\mathbf{p}_i\},\ i = \{1, \cdots n\}. \tag{4}$$

At each viewpoint, the adjacency of the range data is maintained so that a triangle mesh can be created. The ordering of vertices in the face definition of each triangle inherently contains the normal direction. The normal direction is defined so that it is directed more towards the sensor location than its negative alternative. To fuse data from multiple viewpoints into a single triangle mesh, an adaptive distance field map representation (i.e. a volumetric technique) has been chosen. The data fusion process is based on the technique proposed by Curless [6] and improved by Webb [31] for real-time and online implementation, and for being able to handle thin plates and sharp features. The implementation includes a spatial index over multiple signed distance fields, implemented as an octree of small 3D grids. This provides a sparse representation to minimise memory usage and an index for efficient updates. The output of the fusion process is a mesh map where the vertices (i.e. the point cloud, P) can be rapidly queried. Fig. 3 shows a point cloud generated by sensing an overhanging structure from multiple viewpoints.

The point cloud that is generated must be transformed into a set of SLD targets, which are smoothed circular targets that can be used to plan the task-specific motions of a manipulator. Therefore, after obtaining the point cloud, P, the goal is to segment or classify the points into n_{ss} distinct point groups, PG_j, for $j \in \{1, \ldots n_{ss}\}$ such that,

$$\mathsf{P} = \bigcup_{j=1}^{n_{ss}} \mathsf{PG}_j, \tag{5}$$

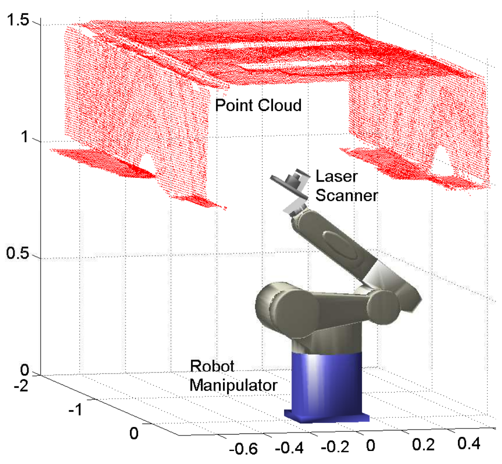


Figure 3: A manipulator explores using a range sensor and generates a fused map consisting of a point cloud, P containing $n_p = 80977$ points.

where $\bigcup$ stands for set union combining the n_{ss} point groups. Each point group is a candidate to be represented by a SLD.

2.2. Scale-Like Disc Generation

The generation of Scale-Like Discs (SLDs) is considered to be both a data reduction and a dimensionality reduction problem. It is observed that if a set of 3D points could form a small planar surface, one of the dimensions will vanish, and hence a reduction of the dimensionality is achieved. If the remaining dimensions are utilised, the 3D points are reduced to a single surface normal, a 3D centre point, and a radius from the centre point. The representation which is created is therefore equivalent to a disc. To this end, an approach based on Principle Components Analysis (PCA) is adopted [25][8]. One of the benefits of using PCA is that a normal vector, which describes the orientation of the surface, can be determined. This is since the surface normal is the eigenvector that corresponds to the minimum eigenvalue of the covariance matrix. A further advantage of PCA is that if the centre point of each disc is strategically positioned, then a surface in the environment can be represented by a set of discs which are partially overlapping in a scale-like pattern (i.e. SLDs). SLDs can be used to represent both flat and curved surfaces, and most importantly the SLDs facilitate the processes of manipulator pose selection and manipulator trajectory planning [17].

A flowchart of the SLD generation algorithm is presented in Fig. 4. The SLD generation algorithm simultaneously creates the SLDs through the PCA data reduction technique, and arranges the SLDs in the required scale-like pattern.

The first step of the algorithm is to divide the environment into cubic voxels (i.e. equally sized, cube-shaped volumetric pixels) so that each point in the point cloud is associated with a voxel [26][19]. This technique is used to both,

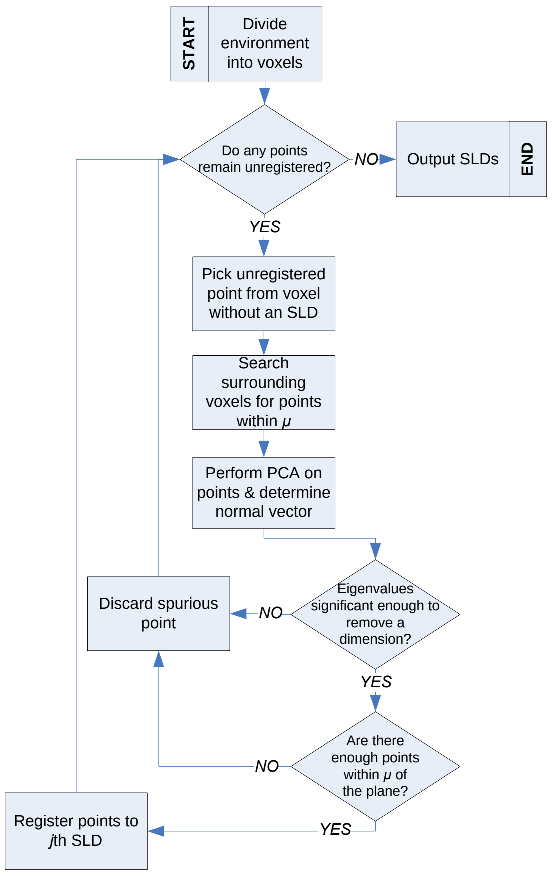

START
Divide environment into voxels
Do any points remain unregistered?
NO
Output SLDs
END
YES
Pick unregistered point from voxel without an SLD
Search surrounding voxels for points within μ
Perform PCA on points & determine normal vector
Eigenvalues significant enough to remove a dimension?
NO
Discard spurious point
YES
Are there enough points within μ of the plane?
NO
YES
Register points to jth SLD

Figure 4: The SLD generation algorithm: reduces the dimensionality of the point cloud and positions the SLDs in a partially overlapping pattern.

determine which voxel each 3D point is associated with, and to improve the efficiency of the algorithm. An occupied voxel is defined as a voxel that contains one or more 3D points. The number of points in an occupied voxel is a function of the size of the voxels and the density of the point cloud. In order to determine which voxel contains which 3D point, each point is divided by the length of the sides of a voxel. The three real number resultants are then mapped to the largest previous rounding integer (i.e. $\lfloor \cdot \rfloor$). Thus, if the length of the sides of a voxel is $10mm$, then the ith point, $\mathbf{p}_i = [130, -25, 35]^T$, is associated with the voxel, $[13, -3, 3]^T$. Each point is thereby associated to a voxel which is labelled with a 3×1 vector of integers. The required pattern for the SLDs is maintained by ensuring that the radius of the SLDs, μ, is greater than the empirically determined $\sqrt{3}$ times larger than the voxel size, just as the longest diagonal of a cube is $\sqrt{3}$ times larger than each side. Therefore, the 3D points contained by two adjacent voxels, can be used to generate SLDs which at least touch at the edges, or that will more likely partially overlap. To ensure the required SLDs do not completely overlap, each occupied voxel is limited to only contain one SLD.

The next step of the algorithm is to focus on an occupied voxel that does not contain a SLD. An unregistered point is then randomly selected from this voxel and denoted as a potential SLD centre point $\hat{\mathsf{P}}_j$. A search is conducted on the enclosing voxel and the surrounding $3^3 - 1 = 26$ voxels, to find a point group, PG_j, that are within the specific distance, μ, from $\hat{\mathsf{P}}_j$. This is equivalent to searching the spherical volume, $(4/3\pi\mu^3)$ around $\hat{\mathsf{P}}_j$.

PCA must then be performed on each point group, PG_j, so a normal of the point cloud, $\vec{n}_j$, can be extracted. The measurements obtained from the laser sensor give a 3D description of the location of a point on an object's surface. The goal is to reduce the dimensionality to 2D such that the surface is still adequately represented.

A subset point cloud, PG_j containing ${}^j n_p$ points returned by the laser sensor and transformed into global coordinates is,

$$\mathsf{PG}_j = \{\mathbf{p}_i\},\ i = \{1, \cdots {}^j n_p\}. \tag{6}$$

The mean of the point group, $\overline{\mathbf{pg}}_j$, is first calculated as,

$$\overline{\mathbf{pg}}_j = \frac{1}{{}^j n_p} \sum_{i=1}^{{}^j n_p} \mathbf{p}_i. \tag{7}$$

where $\overline{\mathbf{pg}}_j$ is a 3D vector, the difference between the point group and the mean is,

$$\mathbf{\Delta} = \mathsf{PG}_j - \overline{\mathbf{pg}}_j \times \mathbf{1}, \tag{8}$$

where $\mathbf{\Delta}$ is a $3 \times {}^j n_p$ matrix and $\mathbf{1} = [1, \cdots 1]$ is a vector containing ${}^j n_p$ values of 1.

A covariance matrix, $\mathbf{C}$, representing the spread of the points is calculated from,

$$\mathbf{C} = \frac{1}{{}^j n_p} \mathbf{\Delta}\mathbf{\Delta}^T,\ \mathbf{C} \in \mathbb{R}^{3\times 3} \tag{9}$$

In order to determine the principal components of the matrix, the covariance matrix, $\mathbf{C}$, is used to determine the eigenvectors, $\mathbf{V}$, which diagonalises $\mathbf{C}$ as in,

$$\mathbf{V}^{-1}\mathbf{C}\mathbf{V} = \mathbf{D} \tag{10}$$

where the diagonal matrix, $\mathbf{D} = \{D_{i,i}\}$ and $D_{i,i}$ is equivalent to the ith eigenvalues, λ_i, which in turn correspond to the eigenvectors, $\mathbf{v}_i$ in the matrix, $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]^T$.

The magnitude of the eigenvalues has a significant role in the representation of a data set. In conventional applications of PCA, the dimensions of data corresponding to larger eigenvalues are selected to represent the data set. Thus, the eigenvalues are further sorted in descending order. A dimensionality reduction is possible in this case if the ratio between the maximum and minimum eigenvalue, and the ratio between the median and minimum eigenvalue is such that the eigenvector corresponding to the minimum eigenvalue is not required to describe the set of points. It is then possible to remove the dimensions corresponding to the minimum eigenvalue. Therefore, the data can be described as being on the plane represented by the remaining eigenvectors. Thus, the dimensionality of the dataset is reduced from a 3D point cloud to 2D plane.

For the problem considered in this work, a disc needs to be extracted from a point group in 3D space as shown in Fig. 5a. The dimension perpendicular to the disc contains less information than the dimensions which are parallel to the plane. This is since the points on the disc are dense along the perpendicular dimension. In contrast to conventional PCA applications, the eigenvector that corresponds to the minimum eigenvalue is selected to represent the normal vector, $\vec{n}$. That is,

$$\vec{n} \equiv \mathbf{v}_i, \tag{11}$$

where $\mathbf{v}_i$ is selected based on the corresponding minimum eigenvalue, λ_i. Each triangle in the original fused triangular mesh contains a normal that is the cross product of the first two sides of each triangle. The plus or minus direction of $\vec{n}$ is based upon the normals to the triangle mesh from which the points were extracted [31]. In the case of the 3D points, the 'surface normal' (i.e. the vector perpendicular to the surface) originates from the centre of the disc, $\hat{\mathsf{P}}_j$ (note that $\hat{\mathsf{P}}_j$ can be different from the centre of mass of the point group). Hence, a single SLD is characterised by $\{\hat{\mathsf{P}}_j, \vec{n}_j\}$. All points, P_j, within μ of the home point and within μ of the plane are registered to the jth SLD.

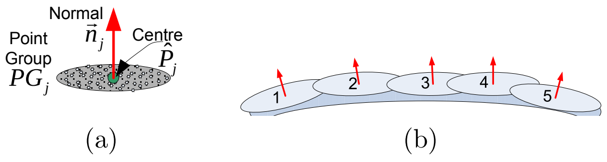


Figure 5: a) SLD (point and surface normal $\{\hat{\mathsf{P}}_j, \vec{n}_j\}$) represents point group, PG_j (a subset of the full point cloud dataset); b) Five SLDs with normal vectors shown, represent a curved surface. The SLD connectivity must be defined so as to cluster SLDs into map segments for ease-of-use by the manipulator path planner.

Once the geometry of the surfaces in the environment has been mapped, the SLDs are generated from the point cloud using the PCA-based algorithm. This reduces the complexity of the abrasive blasting task by simplifying the number of parameters required to represent a segmented surface, and by providing manageable sets of targets on the surface. Additionally, the approximate surface is smoother and the overlap of the SLDs can assist in planning continuous manipulator motions. The task-specific manipulator 'pose selection' algorithm [31][18] is used (as shown in Fig. 1) to find an acceptable manipulator pose for each SLD. This deterministic algorithm is used since it facilitates the rapid discovery of a valid joint configuration for a given SLD target, and it can incorporate the significant parameters for abrasive blasting: stream length, the angle between the blast stream and the surface normal, the proximity of the manipulator to the environment, and the physical joint limitations.

3. Map Segmentation

The map that is generated through exploration and mapping has a set of SLDs each of which has an associated manipulator pose. In order to perform the abrasive blasting task effectively the SLDs are ordered in a boustrophedon pattern [20]. Although the boustrophedon pattern produces the required coverage results, it generally results in inefficient manipulator movements and cannot efficiently cover areas where there is large variation between the surface normals [29]. Also, since the nozzle must always be pointing in a safe direction when blasting, then the blast stream must be halted when moving between two disconnected areas. By segmenting the map (i.e. grouping SLDs) there can be numerous efficiency improvements: reduce the number of times the blast stream must be halted; reduce the total time to perform a task; reduce the wastage of the abrasive (i.e. minimise excessive blast spot movements over the surface); and reduce the manipulator joint movements while blasting. In order to obtain the efficiency gains the map segmentation algorithm must consider the geometric movements of the abrasive blasting spot and the joints of the manipulator.

As presented in the previous section, the ith SLD consists of a surface normal $\vec{n}_i$ a centre point $\hat{P}_i$, and has a collision-free manipulator pose associated with it, $\vec{Q}_i$, that can point the blast stream at the SLD. An algorithm is devised that clusters SLDs into segments based on connectivity of SLDs. Graph theory is utilised where each SLD is a graph's vertex and an edge exists between two vertices only when two SLDs are connected according to a specific rule. Four distinct graphs are created by thresholding the following four parameters: the angular difference between the SLD normals; the distance between centre points of SLDs; the average distance between SLD centre points and other SLD planes; and the angular difference between the poses which enable the manipulator to act (i.e. perform a task) on a SLD target.

Consider the case shown in Fig. 5b where there are five SLDs. Even if SLDs 1 and 5 would not be connected directly, if the manipulator poses for all SLDs were known to be similar, then a manipulator trajectory could be planned from SLD 1 to 5 through SLDs 2, 3 and 4. Fig. 5b may then be regarded as a single map segment, assuming that the remaining three aforementioned conditions of SLD connectivity are met. Henceforth, the connectivity of SLDs will be formulated.

3.1. Segmentation Formulation

3.1.1. Angle Between Surface Normals

Every SLD has a surface normal which defines its orientation. Using this normal vector, $\vec{n}$, it is possible to determine the angle between neighbouring SLDs. As shown in Fig. 6a, for SLD $\mathbf{s}_i$ and $\mathbf{s}_j$, the angular difference, $\delta\theta_{ij}$, of their surface normals is calculated as,

$$\delta\theta_{ij} = \mathbf{acos}(\vec{n}_i \cdot \vec{n}_j), i = \{1, \cdots n_{ss}\}, j = \{1, \cdots n_{ss}\} \tag{12}$$

where n_{ss} is the number of SLDs. The n_{ss} by n_{ss} angular difference matrix can then be formed as $\delta\boldsymbol{\Theta} = [\delta\theta_{ij}]$ with $\delta\theta_{ij} = \delta\theta_{ji}$ and $\delta\theta_{ii} = 0$.

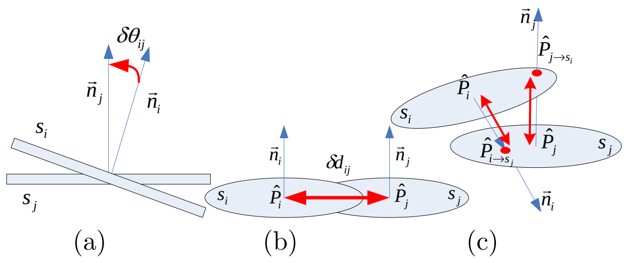


Figure 6: a) Angle difference, $\delta\theta_{ij}$, between normals on the ith and jth SLDs. b) Distance between centres of the ith and jth SLDs is $\delta d_{ij} = \|\hat{\mathsf{P}}_i - \hat{\mathsf{P}}_j\|$. c) The point of intersection between the normal of the ith SLD and the jth SLD's plane, $\hat{\mathsf{P}}_{i\to s_j}$, and the point of intersection between the jth SLD's normal and the ith SLD's plane, $\hat{\mathsf{P}}_{j\to s_i}$.

It is possible to compare this matrix of angles to a constant threshold, τ_θ, which is the maximum allowable angular difference between the normals of any two SLDs, $\mathbf{s}_i$ and $\mathbf{s}_j$. If $\delta\theta_{ii} \leq \tau_\theta$, then an edge connects the graph vertices for SLDs i and j. This will give a binary connection graph, ${}^{\delta\theta}\mathcal{B} = [{}^{\delta\theta}b_{ij}]$, which is an $n_{ss} \times n_{ss}$ matrix of binary values where,

$${}^{\delta\theta}b_{ij} = \begin{cases} 1, & \delta\theta_{ij} \leq \tau_\theta \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

3.1.2. Distance Between Centres of SLDs

Denoting the position of centres of two SLDs as $\hat{\mathsf{P}}_i$ and $\hat{\mathsf{P}}_j$, the Euclidean distance vector between them (Fig. 6b) is found by,

$$\delta d_{ij} = \|\hat{\mathsf{P}}_i - \hat{\mathsf{P}}_j\|, \tag{14}$$

where $\|\cdot\|$ is the Euclidean distance.

This can also be transformed into a binary connectivity graph, ${}^{\delta d}\mathcal{B} = [{}^{\delta d}b_{ij}]$, such that,

$$^{\delta d}b_{ij} = \begin{cases} 1, & \delta d_{ij} \le \tau_d \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

3.1.3. SLD Centre-to-Plane Distance

The normal from the centre of the jth SLD intersects with the ith SLD plane at $\hat{\mathsf{P}}_{j\to s_i}$. Conversely, the normal from the centre of the ith SLD intersects with the jth SLD plane at $\hat{\mathsf{P}}_{i\to s_j}$. The average distance between the points of intersection and the respective centres of the SLDs is denoted as δp_{ij} and calculated as,

$$\delta p_{ij} = \frac{\left\|\hat{\mathsf{P}}_{j\to s_i} - \hat{\mathsf{P}}_j\right\| + \left\|\hat{\mathsf{P}}_{i\to s_j} - \hat{\mathsf{P}}_i\right\|}{2} \tag{16}$$

The average of two SLD centre-to-plane distances have to be within a limiting threshold, τ_p. The binary graph is constructed from $^{\delta p}\mathcal{B} = [^{\delta p}b_{ij}]$, where,

$$^{\delta p}b_{ij} = \begin{cases} 1, & \delta p_{ij} \le \tau_p \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

3.1.4. Difference Between Manipulator Poses

For the ith SLD there is an associated manipulator pose, $\vec{Q}_i$ which is determined through the task-specific pose selection process. The joint angles for a manipulator pose can be determined for each SLD target [31][18]. The angular difference between joints used in a manipulator pose for SLDs i and j is another measure for map segmentation. When the angular difference for each joint is small, then the manipulator movement required from SLD i to SLD j is also small. The joint difference between poses $\vec{Q}_i$ and $\vec{Q}_j$ is formed as,

$$\delta Q_{ij} = \left|\vec{Q}_i - \vec{Q}_j\right| \tag{18}$$

where $|\cdot|$ denotes the absolute angular difference between two manipulator poses such that $\delta Q_{ij} = [\delta q_{1,ij}, \delta q_{2,ij}, \delta q_{3,ij}, \delta q_{4,ij}, \delta q_{5,ij}, \delta q_{6,ij}]^T$.

Where τ_{q_k} is the constraint on the kth manipulator joint, the binary graph $^{\delta q}\mathcal{B} = [^{\delta q}b_{ij}]$, is constructed as,

$$^{\delta q}b_{ij} = \begin{cases} 1, & \delta q_{k,ij} \le \tau_{q_k}, k \in \{1, \dots 6\} \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

3.2. SLD Clustering Algorithm

Based upon the four binary graphs, one single connectivity binary graph (i.e. an $n_{ss} \times n_{ss}$ binary matrix) is generated.

$$\mathbf{B} = \left\{^{\delta\theta}\mathcal{B} \wedge {}^{\delta d}\mathcal{B} \wedge {}^{\delta p}\mathcal{B} \wedge {}^{\delta q}\mathcal{B} \wedge (\neg\mathbf{I})\right\} = [b_{ij}] \tag{20}$$

where $\wedge$ is the intersection between the binary graphs such that $[1] \wedge [0] = [0]$ and $[1] \wedge [1] = [1]$. Also, $\neg\mathbf{I}$ is the logical negation of the identity truth matrix, which forms an n_{ss} by n_{ss} matrix of ones (true) with the diagonal as zeros (false). Therefore, the self-connectivity is disabled by this Boolean combinatorial process. This is notionally similar to removing edges between the same graph vertices, thus $b_{ii} = 0$. The edges in the graph, b_{ij}, represent the connection between two SLDs. The combined graph then must be searched for clusters of vertices which correspond to map segments. Algorithm 1 shows how a breadth-first search is used recursively in $\mathbf{B}$ to find the n_s clusters of SLDs (i.e. the segments), such that $n_s \geq 1$. Each SLD in a segment has a connection to at least one other SLD in that segment.

Algorithm 1 Clustering SLDs into map segments

```
 1: Set the number of segments, $n_s$, to one
 2: Enqueue an unregistered root SLD
 3: for each SLD in the current queue do
 4:    Dequeue a SLD and examine it
 5:    Check for connected SLDs in combined graph $\mathbf{B}$
 6:    Enqueue these SLDs to the stack of the $n_s$th segment
 7:    if there are no more SLDs to dequeue then
 8:       Segment is complete
 9:       Increment $n_s$
10:    end if
11: end for
12: if all $n_{ss}$ SLDs are registered to segments then
13:    FINISH
14: else
15:    Repeat from Step 2
16: end if
```

4. Results

This section presents experimental results from the surface representation and map segmentation process. Geometry data is gathered in three different bridge maintenance environments using the exploration and mapping approach [18]. Experiment 1 presents the details of the segmentation approach in a straightforward environment. The environment consists of a portion of a replica bridge I-beam channel section. Experiment 2 shows a second, more complex environment, modelled on a different bridge environment. Experiment 3 uses data from a main I-beam section on an actual bridge. In all experiments SLDs are generated from the geometric data based upon consideration from the maintenance operation. Corresponding robot manipulator poses are determined for each SLD and the segmentation algorithm is used to cluster the SLDs into segments. The resulting differences are presented in terms of both abrasive blasting point travel (indicative of the abrasive blasting material usage and the time taken) and manipulator joint movement. The segmentation parameters that are used in the experiments were determined empirically by laboratory and field trials [19], and are shown in Table 1. The manipulator joint limitation $\tau_{\mathbf{q}}$ is determined based upon μ and τ_d. Using forward kinematics the maximum acceptable joint movements are found such that the end effector location does not deviate by μ when moving τ_d between two SLDs. Note the lower the joint index, the smaller the joint limitation is since lower joints have more affect on the end effector location, while joint 6's affect is negligible.

Table 1: Map Segmentation Demonstration Constraint Parameters

Param.	**Purpose**	**Value**
μ	Radius of the SLDs	40mm
n_{min}	Ensure SLDs contain enough points	8 points
τ_θ	Min. angle between surface normals	$5\times\frac{\pi}{180}$ rads
τ_d	Min. distance between SLD centres	200mm
τ_p	Min. average SLD centre-to-plane dist.	50mm
$\tau_{\mathbf{q}}$	Min. angle between manipulator poses	9,10,13,20,20,120°

4.1. Experiment 1

Fig. 7 shows a section of I-beam channel consisting of two I-beams and a flat roof. The robot is placed underneath the structure and tasked with the maintenance of the inside section. Exploration and mapping processes were performed and a number of vertices, $n_p = 10758$, were extracted from the fused mesh map. Using the points which are inside the manipulator's work envelope, the SLD generation algorithm created $n_{ss} = 391$ SLDs. For each SLD a pose is generated using the pose selection algorithm so as to meet the application-specific requirements. The segmentation algorithm is then applied. The individual binary graphs (${}^{\delta\theta}\mathcal{B}$, ${}^{\delta d}\mathcal{B}$, ${}^{\delta p}\mathcal{B}$, ${}^{\delta q}\mathcal{B}$) are shown in Fig. 8. It is noted that for each of the four conditions the connectivity of the graphs is different. Fig. 8b and Fig. 8c are also observed to be similar but not identical. All graphs can be seen to be symmetric about the diagonal (i.e. $\mathcal{B}_{i,j} = \mathcal{B}_{j,i}$).

The graph of angle differences between surface normals, Fig. 8a, contains several clusters. The first cluster contains most SLDs with index 1 to 50 and indexes around 200, 270 and 360. The second cluster is from 50 to 200 and the third is from 270 to 360. Similarly, for the surfaces on the same plane, ${}^{\delta p}\mathcal{B}$, in Fig. 8c there are three clusters which is intuitive since Fig. 7 contains three predominantly planar surfaces. Fig. 8b shows the graph of distances between the centres of the SLDs, ${}^{\delta\theta}\mathcal{B}$ which, as expected for the proximity constraint, only has local connectivity. Conversely to the first three graphs, the configuration space constraint (Fig. 8d) results in a completely different pattern, since the poses of the manipulator may be similar for completely different SLDs. At the bottom right of Fig. 8d there is clustering around the SLD index of 350.

Fig. 9a shows the overall connectivity binary graph. Using the map segmentation algorithm, four clusters (i.e. map segments) are found with the resulting

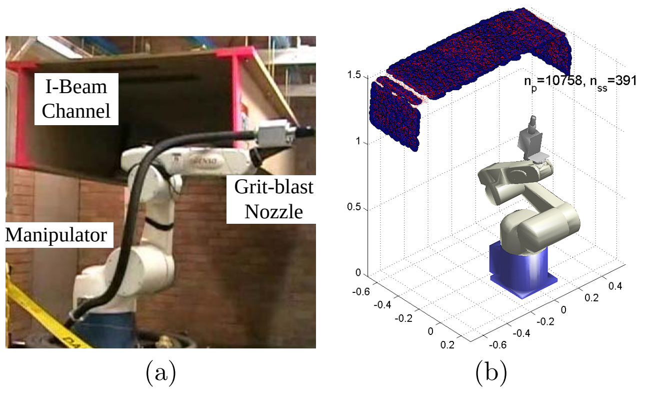


Figure 7: *a*) Bridge maintenance environment; *b*) Corresponding point cloud, P, with $n_p = 10758$ vertices represented by $n_{ss} = 391$ overlayed SLDs requiring maintenance.

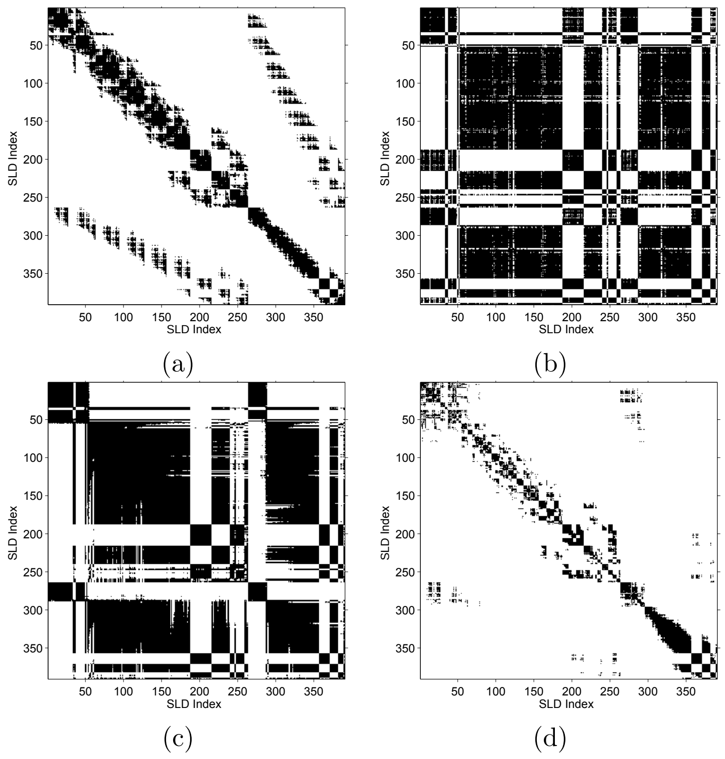
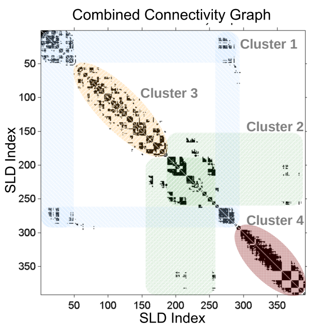
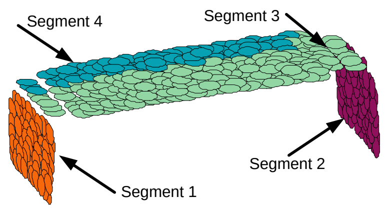


Figure 8: SLD connectivity Binary graphs: *a*) Angle between normals, ${}^{\delta\theta}\mathcal{B}$; *b*) Distance between centres, ${}^{\delta d}\mathcal{B}$; *c*) Average distance from an SLD centre to another SLD plane, ${}^{\delta p}\mathcal{B}$; *d*) Manipulator pose angular difference, ${}^{\delta q}\mathcal{B}$.



(a)



(b)

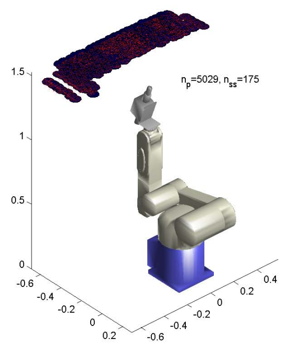
Figure 9: *a*) Clusters highlighted in the combined connectivity graph, **B**. Graphs are binary (black=connection, white=no connection) and symmetric. *b*) Resulting surface segmentation corresponding to the clusters in (*a*).

segmented map shown in Fig. 9*b*. Segment 1 is made up of $n_{ss} = 69$ SLDs. These SLDs represent $n_p = 1623$ points to the left of the manipulator and Segment 2 has $n_{ss} = 66$ SLDs generated from $n_p = 1534$ points. Fig. 10 shows segment 3 and segment 4 relative to the manipulator along with the histogram of manipulator poses. Fig. 10*b* shows the angular histogram of joint angles for the manipulator poses which direct the nozzle on the end-effector at the SLD targets in segment 3. These angular histograms show that the poses for all the SLD on the segment are similar with q_1 to q_6 concentrated around $[30, 15, -60, -20, -30, 15]$ degrees which indicates that moving over the segment requires relatively few motions of the robot joints. Fig. 10*c, d* shows that even though segment 4 is similar to segment 3 in terms of geometry since it lies on the same plane, the poses are significantly different and hence is it advantageous to separate the surface into two segments. In Fig. 10*d* the manipulator poses for q_1 to q_6 corresponding to the SLDs are concentrated around $[-30, -30, -30, -20, 80, 60]$ degrees. This is significantly different to the angular histograms for segment 3.

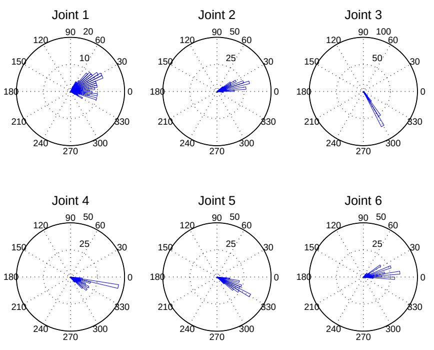
Experiment 1 showed that segments are not necessarily regular geometric shapes due to the poses considered being in the configuration space of the manipulator. Also, the blasting motions for a segment can occur with relatively small total joint changes. Finally, surfaces that are in close proximity may still require significantly different manipulator poses.

4.2. Experiment 2

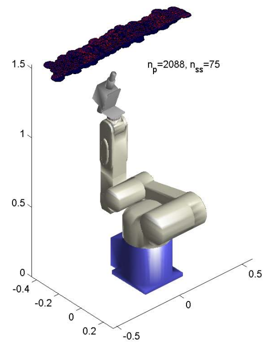
The second experiment uses a Schunk manipulator in a more complex laboratory environment that includes part of an I-beam, and a corrugated shaped roof as shown in Fig. 11*a*. The robot is placed underneath the structure and tasked with the maintenance of the inside section. Exploration and mapping processes were performed and a number of vertices, $n_p = 39104$, were extracted from the



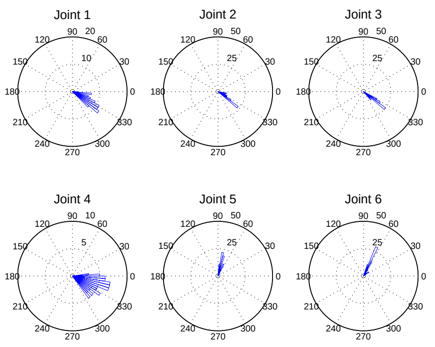
(a)



(b)



(c)



(d)

Figure 10: *a*) The third segment in relation to the manipulator; *b*) Angular histogram for each joint of the poses for the third segment, showing that manipulator poses are similar to each other. *c*) The fourth segment in relation to the manipulator; *d*) Angular histogram for each manipulator joint with different angular histograms to (*b*).

generated mesh map. Using the points which are inside the manipulator's work envelope, the SLD generation algorithm created $n_{ss} = 1736$ SLDs. For each SLD a pose is generated and then the segmentation algorithm is applied.

Fig. 12 shows the results of the segmentation algorithm, 37 segments are found. The segments are labelled and both the top view (Fig. 12*a*) and part of the isometric view (Fig. 12*b*) are shown. It is clear from the top view that, although there is geometric connectivity between SLDs on the roof, the poses to point at the roof are significantly different to each other, and they tend to cluster for SLDs where the normals are similar in a local geometric region. In the lower part of Fig. 12*a*, nearby segment 31, there are several uncoloured SLDs. These SLDs could not be clustered with other SLDs due to the large differences between the associated manipulator poses. The uncoloured SLDs are not disregarded. For completeness, these SLDs must be blasted individually. The joint movements and blast stream movements required to blast these SLDs are still added to the cumulative sum of movements for all segments, and are

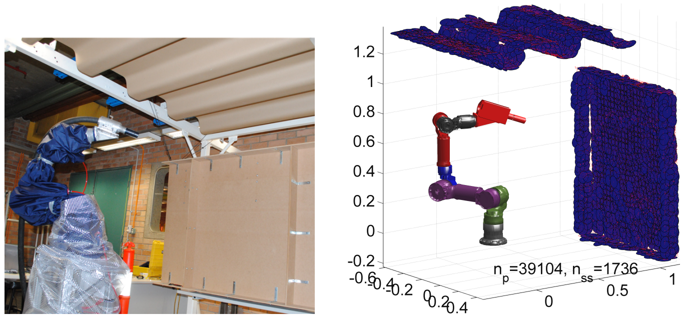

(a) (b)

Figure 11: *a*) Second bridge maintenance environment; *b*) Corresponding point cloud, P, with $n_p = 39104$ vertices represented by $n_{ss} = 1736$ SLDs to be maintained in the environment overlayed.

presented in the results.

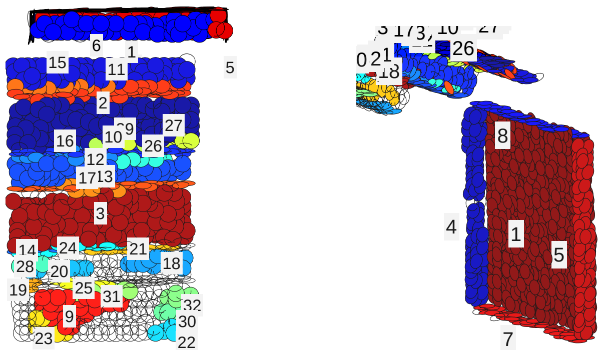

(a) Top view (b) Isometric View (partial)

Figure 12: Resulting surface segmentation corresponding to the clusters.

4.3. Experiment 3

Fig. 13 shows a dataset from the side I-beam of an actual bridge during field trials. The robot is placed in front and is tasked with the maintenance of the surfaces. Exploration and mapping processes were performed and $n_p = 90136$ vertices, were extracted from the map. The SLD generation algorithm created $n_{ss} = 681$ SLDs, a manipulator pose is generated for each SLD, and then the segmentation algorithm is applied.

Using the map segmentation algorithm, 6 segments are found as shown in Fig. 14. It is noted that segment 5 is separated from segment 1 because of a difference in joint angles. There are 7 SLDs in the bottom left hand corner which are not in any segment and must be blasted individually. A limitation

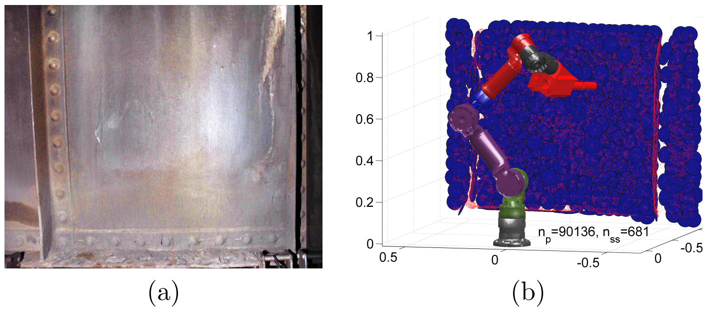


Figure 13: *a*) Real bridge maintenance environment; *b*) Corresponding point cloud, P, with $n_p = 90136$ vertices represented by $n_{ss} = 681$ SLDs to be maintained.

to the segmentation approach is also observed in this experiment which has led to segment 1 being on either sides of the vertical ribs (e.g. segments 2 and 3). Although it is possible to blast segment 1, this will probably lead to the inadvertent partial blasting of segment 2 and/or segment 3. A solution to this could be to add an additional segmentation rule that separates SLDs if there is a perpendicular surface between them.

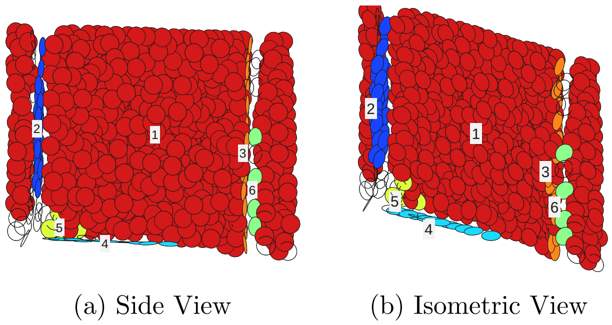


Figure 14: Resulting surface segments.

The blast stream spot travel results (in meters) for the three experiments are shown in Table 2. For clarity all distances and percentages are rounded to the nearest meter. In a typical scenario as demonstrated in [20], with a mid-range nozzle the required blast spot velocity is approximately 30mm/s. Hence a reduction in blast spot travel of 1m equates to efficiency gain of 30 seconds and saves 0.5kg of abrasive blasting material. Therefore in the cases of experiment 2, by segmenting the surface and moving the blast nozzle more efficiently over the SLDs, there is a reduction of 81m which equates to saving approximately 40 minutes and 40kgs of garnet.

Table 3 shows the robot joint movement for each joint, and in total, for the three experiments. In experiments 1 and 2, where the environment contains

Table 2: Blast spot travel (in meters) before and after segmentation.

	Experiment 1	**Experiment 2**	**Experiment 3**
	$\sum \delta d_{ij}$	$\sum \delta d_{ij}$	$\sum \delta d_{ij}$
Before (m)	39	170	53
After (m)	31	89	47
Improvement (%)	12%	48%	10%

segments that are both above and to the side of the manipulator, there is a significant reduction in the joint movements (i.e. an improvement). In experiment 2 where the SLD are significantly different and spread out in the robot's workspace, the total joints movement after segmentation is only 63% of the motion before, a 37% improvement. Note that this also includes covering all SLDs not just those in segments. In experiment 3, there is a 10% improvement in the blast spot travel. However since the poses for the SLDs in front of the robot are in similar configuration space, joints 1, 4 and 6 have a net increase in joint movement (i.e. a negative improvement percentage). Therefore, for experiment 3 there is only a small overall improvement in joint movement by segmenting the surfaces.

Table 3: Total rounded joint movement (in radians) before and after segmentation.

Experiment 1							
	Manipulator joint(s)						
	1	2	3	4	5	6	**all**
Before	39	43	29	43	74	47	**275**
After	33	28	19	32	40	39	**192**
Improvement	16%	34%	32%	24%	46%	17%	**30%**

Experiment 2							
	Manipulator joint(s)						
	1	2	3	4	5	6	**all**
Before	437	281	304	333	364	632	**2352**
After	268	206	184	222	234	364	**1478**
Improvement	39%	27%	39%	33%	36%	42%	**37%**

Experiment 3							
	Manipulator joint(s)						
	1	2	3	4	5	6	**all**
Before	16	46	93	24	45	35	**259**
After	22	41	85	30	40	39	**257**
Improvement	-37%	11%	9%	-24%	11%	-11%	**1%**

4.4. Discussion

In all the experiments conducted, a reduction is seen in both the blast spot movement and the manipulator joint movement. When a continuous trajectory is executed over the surfaces there are many inefficient manipulator movements, due to the different points and normals that must be pointed at. In the case where surfaces are curved rather than smooth it can be difficult to obtain continuous manipulator trajectories.

The manipulator joint motions that are recorded in Table 3 actually includes two components: the blasting joint motions and the joint movements between segments. The blasting motions are performed slowly so as to blast with the required tool speed. However, the movements between segments (i.e. to link the segments together) can be performed quickly and without the additional load of the abrasive blasting stream (since it can be halted during this time). By examining the robot joint motion control logs (not shown here) it was observed that more than 20% of joint movements in experiment 3 consisted of these linking joint motions.

The segmentation algorithm has only been tested using one pose selection algorithm. Attempts to calculate a trajectory prior to segmentation were unsuccessful with the current pose selection algorithm. An alternative trajectory planner may be able to determine valid poses for a group of SLDs. Additionally, a possible way to improve surface segmentation improvement would be to determine numerous manipulator poses for each SLD, and then use another level of optimisation to determine the best poses out of the set for each SLD that would lead to the best segmentation. This would be more time consuming but could lead to a better segmentation result with improved movement efficiency.

It is noted that currently segments are not formed by dividing surfaces based upon perpendicular partitions that may exist. In experiment 3, where there were two vertical rib dividers that were perpendicular to the main surface, the segmentation algorithm found a segment (i.e. index = 1) that exists on either side of these. The manipulator poses and the SLD positions and orientations were similar enough to result in the main wall consisting of one large segment, even though this is not an ideal segmentation. It would therefore be advantageous to augment the current algorithm with the consideration of perpendicular surface partitions so as to perform more logical segmentation. Additionally, complementary sensor data could be included, such as material-type information and image data, so as to improve segmentation.

5. Conclusions

This paper has presented a novel segmentation approach which can be used in maintenance-operations that utilise a robot manipulator. It has been shown that map segmentation can occur autonomously. Additionally, a segmented environment is shown to be beneficial to maintenance-operation planning when segmentation is done based upon the task considerations, such as the distance and angle between successive targets, and the difference in manipulator joint angles required to perform the task. Initially, a geometry map of an environment such as a triangle mesh is converted to a group of task-specific SLD targets with associated manipulator poses. These SLDs can be grouped to form segments based upon the SLDs' proximity/ orientation and the associated manipulator poses. The segmentation algorithm presented simplifies trajectory planning by enabling map segments to be handled separately. This has been shown to reduce the movement of a blasting spot over a surface and thus make operations more efficient and reduce the amount of abrasive material wastage. The segmented surfaces have also been shown to reduce the manipulator joint motion while blasting, which reduces the stress on the joints caused by the labour-intensive blasting operation.

Future work will see further testing of this approach as part of the maintenance system, including measuring the improvements in effective blast coverage due to the map segmentation. Further pose selection optimisation testing, and trajectory optimisation will also be undertaken so as to increase the independence of the approach. The future incorporation of additional information such as image data as well as surface material-type data is also predicted to complement this surface segmentation approach.

Acknowledgment

This work is supported by the ARC Linkage Grant (ARC-LP0776312), by the ARC Centre of Excellence for Autonomous Systems (CAS), the Roads and Traffic Authority (RTA) NSW, and the University of Technology (UTS), Sydney.

References

[1] Adan, A., Hube, D., 2011. 3d reconstruction of interior wall surfaces under occlusion and clutter. In: Proc. International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission. Hangzhou, pp. 275 – 281.

[2] Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., Silva, C. T., 2003. Computing and rendering point set surfaces. IEEE Transactions on Visualization and Computer Graphics 9 (1), 3–15.

[3] Atkar, P. N., Conner, D. C., Greenfield, A., Choset, H., Rizzi, A. A., 2009. Hierarchical segmentation of piecewise pseudoextruded surfaces for uniform coverage. IEEE Transactions on Automation Science and Engineering 6 (1), 107–120.

[4] Chitta, S., Cohen, B., Likhachev, M., 2010. Planning for autonomous door opening with a mobile manipulator. In: International Conference on Robotics and Automation. Anchorage, Alaska, pp. 1799 – 1806.

[5] Corporation, C. I., 1994. Blast Off: Your Guide to Safe and Efficient Abrasive Blasting. Clemco Industries, Washington.

[6] Curless, B., Levoy, M., 1996. A volumetric method for building complex models from range images. In: Computer graphics proceedings, annual conference series. Vol. 2006. Association for Computing Machinery SIGGRAPH, New Orleans, pp. 303–312.

[7] Gissler, M., Dornhege, C., Nebel, B., Teschner, M., 2009. Deformable proximity queries and their application in mobile manipulation planning. Advances in Visual Computing, Lecture Notes in Computer Science 5875, 79–88.

[8] Godil, A., Ressler, S., Grother, P., 2005. Face recognition using 3d surface and color map information: Comparison and combination. Biometric Technology for Human Identification, SPIE 5404, 351–361.

[9] Howarth, B., Katupitiya, J., Guivant, J., Whitty, M., 2011. Extraction and grouping of surface features for 3d mapping. In: Proc. Australasian Conference on Robotics and Automation. Melbourne, pp. 1–6.

[10] Jiang, X., Bunke, H., 1994. Fast segmentation of range images into planar regions by scan line grouping. Machine Vision and Application 7 (2), 115–122.

[11] Kawata, H., Ohya, A., Yuta, S., Santosh, W., Mori, T., 2005. Development of ultra-small lightweight optical range sensor system. In: Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS. Alberta, Canada, pp. 1078–1083.

[12] Ko, K. H., Maekawa, T., Patrikalakis, N. M., 2005. Algorithms for optimal partial matching of free-form objects with scaling effects. Graphical Models 67 (2), 120–148.

[13] Larsson, S., Kjellander, J. A. P., 2006. Motion control and data capturing for laser scanning with an industrial robot. Robotics and Autonomous Systems 54 (6), 453–460.

[14] Levinski, K., Sourin, A., 2002. Interactive function-based artistic shape modeling. In: Proc. IEEE First International Symposium on Cyber Worlds. Tokyo, pp. 521–528.

[15] Mederos, B., Amenta, N., Velho, L., de Figueiredo, L. H., 2005. Surface reconstruction from noisy point clouds. In: Proceedings of the third Eurographics symposium on Geometry processing. Eurographics Association, Aire-la-Ville, Switzerland.

[16] Nielson, G. M., 2003. On marching cubes. IEEE Transactions on Visualization and Computer Graphics 9 (3), 283–297.

[17] Paul, G., 2010. Autonomous exploration and mapping of complex 3d environments by means of a 6dof manipulator. Ph.D. thesis, University of Technology, Sydney.

[18] Paul, G., Kirchner, N., Liu, D. K., Dissanayake, G., 2009. An effective exploration approach to simultaneous mapping and surface material-type identification of complex 3d environments. Journal of Field Robotics, Special Issue on Three-Dimensional Mapping 26 (11-12 SI), 915–933.

[19] Paul, G., Liu, D. K., Kirchner, N., 2007. An algorithm for surface growing from laser scan generated point clouds. In: Tarn, T., Chen, S., Zhou, C. (Eds.), Robotic Welding, Intelligence and Automation. Springer-Verlag, Berlin, pp. 481–491.

[20] Paul, G., Webb, S., Liu, D. K., Dissanayake, G., 2010. A robotic system for steel bridge maintenance: Field testing. In: Proc. Australasian Conference on Robotics and Automation. Brisbane, pp. 1–8.

[21] Paul, G., Webb, S., Liu, D. K., Dissanayake, G., 2011. Autonomous robot manipulator-based exploration and mapping system for bridge maintenance. Robotics and Autonomous Systems 59, 543–554.

[22] Peters, J., Wu, X., 2000. Optimized refinable surface enclosures. Tech. rep., University of Florida.

[23] Ren, T. R., Kwok, N. M., Liu, D. K., Huang, S. D., 2008. Path planning for a robotic arm sand-blasting system. In: Proc. International Conference on Information and Automation, ICIA. Hunan, China, pp. 1067–1072.

[24] Rusu, R. B., Marton, Z. C., Blodow, N., Dolha, M., Beetz, M., 2008. Towards 3d point cloud based object maps for household environments. Robotics and Autonomous Systems 56 (11), 927–941.

[25] Smith, L. I., 2002. A tutorial on principal component analysis. Tech. rep., University of Otago, New Zealand.

[26] Sujan, V. A., Dubowsky, S., 2005. Efficient information-based visual robotic mapping in unstructured environments. The International Journal of Robotics Research 24 (4), 275–293.

[27] Taylor, C., Cowley, A., 2011. Fast scene analysis using image and range data. In: Proc. IEEE International Conference on Robotics and Automation (ICRA). Shanghai, pp. 3562 – 3567.

[28] Teng, Z., Feng, H. Y., Azeem, A., 2006. Generating efficient tool paths from point cloud data via machining area segmentation. The International Journal of Advanced Manufacturing Technology 30 (3-4), 254–260.

[29] To, W. K., Paul, G., Kwok, N. M., Liu, D. K., 2009. An efficient trajectory planning approach for autonomous robots in complex bridge environments. International Journal of Computer Aided Engineering and Technology 1 (2), 185–208.

[30] Vasudevan, S., Siegwart, R., 2008. Bayesian space conceptualization and place classification for semantic maps in mobile robotics. Robotics and Autonomous Systems 56 (6), 522–537.

[31] Webb, S. S., 2008. Belief driven autonomous manipulator pose selection for less controlled environments. Ph.D. thesis, University of New South Wales Australia.

[32] Weingarten, J. W., Gruener, G., Siegwart, R., 2004. Probabilistic plane fitting in 3d and an application to robotic mapping. In: Proc. IEEE International Conference on Robotics and Automation, ICRA. Vol. 1. New Orleans, pp. 927–932.

[33] Zender, H., Mozos, O. M., Jensfelt, P., Kruijff, G.-J., Burgard, W., 2008. Conceptual spatial representations for indoor mobile robots. Robotics and Autonomous Systems 56 (6), 493–502.