

Chapter 2

Region-Guarding Problem in 3-D Areas

This chapter studies the optimal inspection of autonomous robots in a complex pipeline system. We solve a 3-D region-guarding problem to suggest the necessary inspection spots. The proposed hierarchical integer linear programming (HILP) optimization algorithm seeks the fewest spots necessary to cover the entire given 3-D region. Unlike most existing pipeline inspection systems that focus on designing mobility and control of the explore robots, this chapter focuses on global planning of the thorough and automatic inspection of a complex environment. We demonstrate the efficacy of the computation framework using a simulated environment, where scanned pipelines and existing leaks, clogs, and deformation can be thoroughly detected by an autonomous prototype robot.

2.1 Introduction

Active monitoring and frequent inspections are critical to maintaining pipeline health. As the most economical way to transport gas, oil, biofuels, water resource, sewer, and so forth, pipelines have become an indispensable part of our daily lives. However, pipelines always suffer from aging and damages, which can cause great waste of resource, environmental pollution, and many other incidence. For example, the leak of petroleum pipeline causes ocean pollution and ecocatastrophe. Regular inspections and maintenance of pipelines are essential to keep them functional.

Unfortunately, the difficulty and the cost for human inspection can be extremely high, especially with the appearance of increasingly complicated pipelines nowadays (see Fig. 2.1 for illustration). There are several reasons:

This chapter has been reprinted with permission from “On Optimizing Autonomous Pipeline Inspection,” Xin Li, Member, IEEE, Wuyi Yu, Student Member, IEEE, Xiao Lin, and S. S. Iyengar, Fellow, IEEE, *IEEE Transactions on Robotics*, Vol. 28, No. 1, February 2012.

Fig. 2.1 Complex pipeline system



- Pipeline systems are often buried/hided underground or into walls. Hiding pipelines' presence from the surrounding environment is necessary for better protection of pipelines, as well as the elegance of the architecture.
- The structure of pipelines is usually designed long, thin, and complex, in order to conduct long-distance transportation or circumvent-complicated terrain or limited space of the architecture structure.
- The environment inside pipelines can be dirty and hazardous: Sewerage water or hazardous gas can be overflowing.

These factors make direct artificial inspection oftentimes prohibitive and, therefore, significantly increase the costs for the maintenance of pipelines. For example, to inspect the pipelines, people can dig holes in different pipeline sections for the inspection; high professional competence could be necessary especially when the environment condition is severe. Indirect methods include placing sensors outside the pipeline and monitoring parameters such as pressure and temperature. However, the sensibility is easily affected by environment and the material transmitting in the pipeline. Pipeline clogs are sometimes directly penetrated with long sticks or wires, but it can be very difficult if pipes are curved or circumvented; another common approach is to blow out blockages using air pressure, which fails, however, if pipes have multiple outlets or cracks. In any case, to apply repair, the suspected area for clogging or leaking needs to be located. This step usually takes the longest time and largest cost.

2.1.1 Pipeline Inspection by Autonomous Robots

With the development of autonomous robots and imaginary sensing technologies, pipeline robots that are equipped with cameras and sensors become ideal candidates to avoid tedious artificial inspection for automatic pipeline inspection and repair.

Current robotic inspection systems (see Sect. 2.2) usually have a robot that is equipped with a camera and sensors; the robot moves around and transmits the captured images back to a remote monitor for the operator's inspection. The robot's movement and the camera direction need to be manually controlled by a skillful inspector. Such an interactive monitoring system provides easier and safer inspection. However, it can still be *labor intensive (costly)* and *time consuming*. In addition, complex pipeline environments could limit the extensive use of these remotely controlled pipeline robots. The mobility of remotely controlled robots could not meet the requirement for the inspection in complex pipeline environments, such as the urban gas pipeline and chemical pipelines. The wired connections also limit the operation range of robots. Furthermore, the thoroughness of the examination may not be guaranteed and heavily relies on expertise of the operator.

An autonomous robot that can routinely inspect the environment and report cracks, clogs, or deformation will, therefore, be highly desirable. If such an inspection system can be developed reliably and conducted routinely, it will greatly save artificial costs and prevent the abnormal situations in important pipelines. To develop an efficient inspection plan for robots, according to different environments, to ensure inspection reliability (thoroughness) is related to several challenging geometric problems.

2.1.2 Optimal Autonomous Inspection by Region Guarding

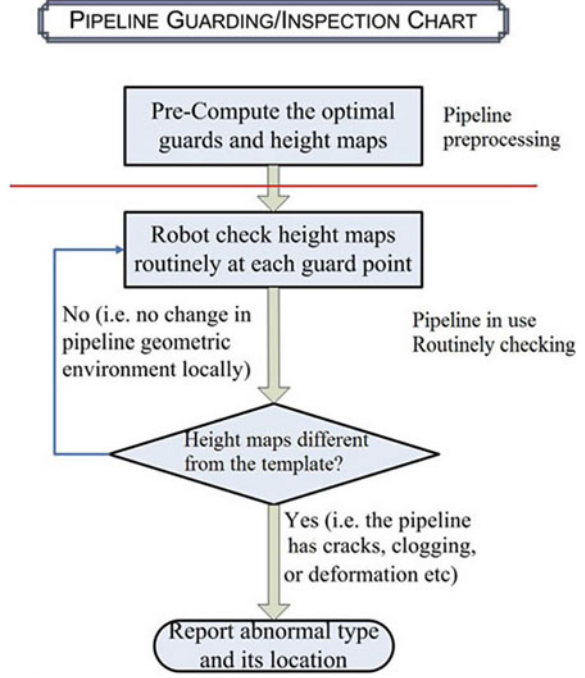
Naturally, one wants to ask the following fundamental open problem for autonomous inspection: How do we conduct the most efficient yet thorough inspection? More specifically, given an environment to inspect, how many inspection spots are necessary to visually cover the entire region? The solution directly dictates the correctness and efficiency of an autonomous inspection system and, therefore, is critical.

Visually covering a given 3-D region is an interesting geometric problem called gallery guarding, defined (see Sect. 2.3 for details) as follows: Given a region whose boundary is a surface, find the smallest set of points inside the region from which all the boundary points (i.e., the wall) are visible. This problem, having high complexity, has been actively studied in 2-D. In general, 2-D polygonal regions, this problem has been proved to be NP-hard. To the best of our knowledge, on 3-D regions approximated by polyhedra, which have much higher complexity, this problem is little explored and no efficient algorithm has been reported.

In this chapter, we design a HILP algorithm to find an approximate optimal solution for the guarding of a given 3-D region. Compared with the greedy and the optimal algorithms, HILP has a good approximation to the optimal solution (for instance, to guard a region in Fig. 2.3d, an optimal guarding needs 13 guards, the greedy approach needs 18 points, while our HILP algorithm guards it with 14 points) but is several orders of magnitude less than the direct optimization on time complexity.

Effective region guarding can greatly benefit the automatic pipeline inspection. Compared with existing manual inspection systems, the biggest advantage of the new inspection system built upon optimal guarding is its thorough (therefore, making

Fig. 2.2 Autonomous inspection based on region guarding



the system robust) inspection using fewest (therefore, making the system efficient and inexpensive) necessary checking spots. The proposed inspection framework is illustrated in Fig. 2.2, and the pipeline has two steps.

1. *Preprocessing Stage*: Once pipeline is newly installed or when it is working well, we compute its optimal guarding, e.g., a small set of points $\{g_i\}$. By checking on these spots, the entire pipeline can be visually covered. Then, a pipeline robot only goes to these points, scans the pipe, and builds up sequential height maps as templates. These height maps characterize the original pipeline geometry.
2. *Online Stage*: The robot will go into the pipeline to conduct inspection routinely. Every time, it only needs to move to these spots $\{g_i\}$, scan the depth information of the surrounding environment, and compare these height maps with the corresponding templates. Abnormal geometry changes such as cracks, clogs, and deformation can be detected and located immediately.

The main contributions of this work lie in both efficiently finding good approximate solutions of the NP-hard 3-D guarding problem and its application on robotic inspection.

1. *Optimality*: We develop an efficient algorithm to find approximate solution to 3-D region guarding. The solution indicates a smallest set of spots from which thorough inspection can be most timely and costly efficient.

2. *Autonomy*: We design an automatic pipeline inspection system for autonomous robots. Unlike other existing systems whose inspection quality heavily relies on manual controls, in our framework, robots can inspect on these fewest guarding points autonomously yet thoroughly.
3. *Generality and Robustness*: The algorithm efficacy is demonstrated in our simulated platform. It is also generally applicable on various robot systems, such as our pipe robot prototype FAMPER [16], which is equipped with a range sensor that provides 2.5-D range image with depth information. Furthermore, combined with 2-D image reconstruction techniques, our algorithm can work well for robots that are equipped with a conventional 2-D camera.

2.2 Background and Related Work

2.2.1 Gallery Guarding

On a geometric region M , we want to find the optimal guarding, which uses the smallest number of points $\{g_i\}$ inside M so that any boundary point $p \in \partial M$ is visible to at least one guard. Here, M is a 3-D shape whose boundary ∂M is represented by a polygonal mesh. For a given guard g_i , any point $p \in \partial M$ is visible to g_i if the line segment $\overline{g_i p}$ is entirely located inside M (we consider $\partial M \subseteq M$). Various versions of this problem are generally called art gallery problems which are known to be a famous problem with high complexity. Even in the 2-D case, the problem is known to be NP-complete. “Very little is known about gallery guarding in three dimensions” [24]. To our best knowledge, no effective approximation algorithm has been proposed for 3-D regions that are bounded by general polygons, and this is the first practical algorithm that works for large free-form 3-D domains (such as complicated pipeline systems) represented by polygonal meshes.

The art gallery problem was first proposed by Klee. Guards can be restricted to boundary vertices ($p \in \partial M$), interior vertices ($p \in M$), or mobile vertices. When guards are not mobile, they are called stationary guards. If guards are restricted to the boundary, they are called vertex guards; if there is no boundary restriction, the guards are referred as point guards. In 2-D, Chavatal [2] and Fisk [13] both showed that a simple polygon $M \subset \mathbb{R}^2$ needs at most $\lfloor n/3 \rfloor$ stationary guards, based on which Avis and Toussaint [12] developed an $O(n \log n)$ time algorithm to position $\lfloor n/3 \rfloor$ guards in M . When guards are mobile, we call them mobile guards. Furthermore, mobile guards are called edge guards if they are restricted to boundary vertices. O’Rourke [5] showed that $\lfloor n/4 \rfloor$ mobile guards are sufficient for a simple polygon $M \subset \mathbb{R}^2$. More results are recapped in Table 2.1.

The aforementioned theoretic work discusses the conservative upper bounds for necessary guards on various regions. Given a specific region, we are interested in designing practical algorithm to find its optimal point guards, which depends on topology and geometry of this region. An effective algorithm to compute the optimal

Table 2.1 Upper bounds for gallery-guarding problem

Work	M type	Guard type	Optimal bound
Chvatal [5]	Simple, 2D	Stationary	$\lfloor \frac{n}{3} \rfloor$
Rourke [36, 37]	Simple, 2D	Mobile	$\lfloor \frac{n}{4} \rfloor$
Urrutia [44]	Simple, 2D	Edge	$\lfloor \frac{n}{4} \rfloor$
Kahn et al. [19]	Orthogonal, 2D	Stationary	$\lfloor \frac{n}{4} \rfloor$
and Rourke [36, 37]		Vertex	$\lfloor \frac{n+2h}{3} \rfloor$
Rourke [35]	h holes, 2D	Vertex	$\lfloor \frac{n+2h}{3} \rfloor$
Hoffmann et al. [13] and	h holes, 2D	Point	$\lfloor \frac{n+h}{3} \rfloor$
Bjorling-Sachs et al. [3]		Point	$\lfloor \frac{n+h}{3} \rfloor$
Gyori et al. [24]	Orthogonal, h holes, 2D	Mobile	$\lfloor \frac{3n+4h+4}{16} \rfloor$

guarding of a given region will benefit many geometric computing tasks. However, computation of optimal guarding is highly challenging. Finding minimal guards has been shown to be NP-hard for 2-D polygons with holes [1], 2-D simple polygons [34], and even 2-D simple orthogonal polygons [27, 39], using either vertex or point guards. Approximation algorithms have been studied in 2-D to get a close to optimal result in polynomial time complexity. Ben-Moshe et al. [24] cover 1.5-D terrain using point guards in $O(n^2)$ time, with the optimal factor $O(1)$. Efrat and Har-Peled [21] find vertex guards for 2-D simple polygonal regions and h -hole polygonal regions in $O(nc_{\text{opt}}^2 \log^4 n)$ and $O(nc_{\text{opt}}^2 \log^4 n)$ expected time, with expected $O(\log c_{\text{opt}})$ and $O(nhc_{\text{opt}}^3 \log^4 n)$ optimal factors, respectively. Lien [10] computes guarding for 3-D point cloud data, approximating visibility using ϵ -view. The algorithm is based on a randomized greedy approach.

2.2.2 Pipeline Inspection Robots

According to the degree of autonomy, pipeline inspection robots can be classified as follows [30].

1. *No Autonomy*: Robots are fully tele-operated by humans via a tether cable. While the robot is traveling through the pipe, the pipeline condition data are collected and sent back by the robot and then assessed by human operators.
2. *Semiautonomy*: Robots are partially controlled by automatic control programs [33].
3. *Full Autonomy*: Robots are fully controlled by programs and perform an automatic pipeline condition assessment. However, lacking effective technologies in efficient analysis of environment hinders the automatic assessment [30].

2.2.2.1 Manual and Semi-Autonomous Methods

There are many application inspection technologies, such as closed-circuit television (CCTV), laser surveys, sonar surveys [4], radio frequency identification [7], and mobile sensor [22]. The introduction of CCTV inspection methods in the 1960s provided an inexpensive and safe option, and they, thus, have been the most popular and widely used approaches across the industry for many years. The CCTV provides rich videos/images information, which is collected by robots for subsequent pipeline condition assessment. Various CCTV methods have similar principles. The robot is mounted by a remotely controlled tractor, carries a television camera, and illuminates the interior of the pipe. The inspector has to identify and categorize defects by the image displayed on the monitor. When a defect is noticed, the inspector stops the robot and assesses the condition.

Advances in optical survey techniques have been utilized in the sewer scanner and evaluation technology (SSET) such as in [23]. Unlike the CCTV inspection system, the SSET may not need to stop for a zooming-in defect inspection. For instance, recent work in [45] has advanced the use of automated defect detection systems for pipelines.

Laser-based systems and ultrasonic-based systems are also used in pipeline inspection (see the survey in [25]). Laser-based systems are generally implemented in two ways: the whole-circle image method and the single-spot scanning method [9]. The first method projects a full ring of light onto the wall in one go, while the single-spot scanning method sends point-by-point beams in sequential. These two methods indicate a trade-off between accuracy and inspection time. The whole-ring image method allows faster data acquisition but has been found less accurate [46]. Ultrasonic-based systems use high-frequency sound waves to detect pipe properties such as thickness, shape, and presence/sizes of defects [32]. Laser-based and ultrasonic-based methods can be combined to obtain higher quality data [40].

2.2.2.2 Autonomous Methods

A few full-autonomous robots have been developed for pipeline inspection. The Kurt [28] can run in dry clean pipelines guided by maps uploaded into the robot. The Marko [15] is designed for autonomous navigation in clean pipelines with diameter ranges from 300 to 600 mm. The Kantaro [30] is used to navigate in pipelines with diameter ranges from 200 to 300 mm, but only the horizontal mobility is considered. To design autonomous pipeline inspection robots, the main challenges include their moving ability, energy, and pipe condition assessment [41].

In this chapter, we use laser range finders to detect the depth (height) information toward sets of sample directions. The captured 3-D range images provide easy measurement of the environment. We focus on designing the algorithm and architecture of the effective autonomous inspection system. The guarding and subsequent inspection can easily extend to various systems that are based on different data acquisition schemes.

2.3 3-D Gallery Guarding

The geometric abnormalities of the pipeline can be detected from the robot if the robot can see this region and measure the distance from itself to the pipe. Suppose the robot always checks at a set of same spots, and it has premeasured (template) distance information on each spot toward different directions, then it can tell whether the current pipeline is normal, i.e., preserving the same shape. These checking spots need to be intelligently selected so that fewest comparisons are necessary. Meanwhile, to guarantee that the entire pipeline is visually covered, we require that these points together can guard the entire region.

Given a point p inside the region M , suppose we represent the boundary surface of M using a triangle mesh $\partial M = T, V$, where $V = \{v_1, v_2, \dots, v_{N_V}\}$ is the vertex set, and $T = \{t_1, t_2, \dots, t_{N_T}\}$ is the set of triangles connecting them. We say that p is visible to a point q on ∂M (q is not necessary a vertex; it can be a point on a triangle from T), if the line segment \overline{pq} connecting p and q is totally inside M , namely \overline{pq} intersects ∂M only on q . We call the set of all visible points on the boundary $\{q\}$, $q \in \partial M$ the visible region $S(p)$ of p . Then, we say that a set of points $\{p\}$ can visibly guard the entire region, if the union of their visible regions is the entire ∂M . Finding a smallest guarding set $\{p\}$ that can cover the entire region is the optimal guarding problem that we want to solve. Our algorithm is based on the following intuitions.

1. As demonstrated in several medical visualization and virtual navigation applications (e.g., [14, 31]), medial axes (curve skeletons) usually have desirable visibility to boundary points (referred as the “reliability” of skeletons). An effective skeleton can guide the camera navigation, ensuring nice examination (visibly covered) of the interior of organ surfaces.
2. Hierarchical skeletons or skeletons for a progressively simplified mesh can be effectively computed and used to reduce the size of the optimization problem, leading to a computation of better numerical efficiency and stability against boundary perturbations.

Many effective skeletonization algorithms (see a survey by Cornea et al. [20]) have been developed for 3-D shapes. We use the algorithm in [6] since it efficiently generates skeletons on medial axis surfaces of the 3-D shapes. Suppose the boundary surface ∂M of a volumetric region M is represented by a triangle mesh (also denoted as ∂M) with n vertices, and the output skeleton has k nodes; the guarding problem is then converted to finding a minimal-size point set G from this k points, such that all n boundary vertices are visible to G .

2.3.1 Visibility Detection

A basic operation is to detect the visible region $S(p)$ of a given point p . Following the definition, for a point $q \in \partial M$, to check its visibility to a point $p \in M$, one should check intersection between the line segment \overline{pq} and ∂M . If the intersection is

detected on a point $q' \in \partial M$ other than q and the Euclidean distance $|\overline{pq'}| < |\overline{pq}|$, then q is not visible from p . The intersection between the line segment \overline{pq} and ∂M can be detected by checking the intersection between \overline{pq} and each triangle face $t_i \in T \subseteq \partial M$.

We need to compute the visibility of p against all the vertices of the mesh of ∂M . Simply enumerating every $\overline{pv_i}$ to check its intersections with every triangle $t \in T$ is time consuming: For a single interior point p , it takes $O(N_V \cdot N_T) = O(N_V^2)$ time to check its visibility on all boundary vertices. We develop the following sweep algorithm to improve the efficiency to $O(N_T \log N_T)$, i.e., $O(N_V \log N_V)$.

We create a spherical coordinate system which is originated at p . Each vertex $v_i \in V$ is represented as $\overline{pv_i} = (r(v_i), \theta(v_i), \phi(v_i))$, where $r(v_i) \geq 0$ and this is the case that:

$$\begin{cases} -\pi < \theta(v_i) \leq \pi \\ -\frac{\pi}{2} \leq \phi(v_i) \leq \frac{\pi}{2} \end{cases}$$

For every $i = 1 \dots N_T$, let t_i denote triangle $\Delta v_{i,1} v_{i,2} v_{i,3} \in T$. We contract the following notations:

$$\begin{cases} \theta_{\max}(t_i) = \max_{j=1}^3 \{\theta(v_{i,j})\} \\ \theta_{\min}(t_i) = \min_{j=1}^3 \{\theta(v_{i,j})\} \\ \phi_{\max}(t_i) = \max_{j=1}^3 \{\phi(v_{i,j})\} \\ \phi_{\min}(t_i) = \min_{j=1}^3 \{\phi(v_{i,j})\} \end{cases}$$

The segment $\overline{pv_k}$ cannot intersect with a triangle t unless they are adjacent:

$$\begin{cases} \theta_{\min} \leq \theta(v_k) \leq \theta_{\max}(t) \\ \phi_{\min}(t) \leq \phi(v_k) \leq \phi_{\max}(t) \end{cases} \quad (2.1)$$

The angle functions θ and ϕ are not continuously defined on a sphere. When a triangle t spans $\theta = \pi$, we duplicate it to ensure that each θ of the original t is in interval $[\theta_{\min}(t) - 2\pi, \theta_{\min}(t))$ and θ of its duplicate is in interval $[\theta_{\max}(t) - 2\pi, \theta_{\max}(t))$, by adding or subtracting θ by 2π . For each triangle t that spans $\phi = \pi$, we detect and duplicate it in the same way. Using $\theta(v_i)$ as the primary key and $\phi(v_i)$ as the secondary key, we then sort all line segments $\overline{pv_i}$. Then, we sweep all segments following the angle functions one by one, filtering out triangles not satisfying Condition 2.1. Specifically, we define a counter c_i on every triangle t_i . Initially, $c_i \leftarrow 0$, when the segment \overline{pv} for some $v \in t_i$ is being processed, $c_i \leftarrow c_i + 1$. The following two cases indicate that the sweep has not reached the neighborhood of the triangle t_i , and subsequently, we do not need to check its intersection with line segment \overline{pv} .

$$\begin{cases} c_i = 0 \rightarrow (\theta_{\min}(t_i) > \theta(\overline{ov})) \vee (\phi_{\min} > \phi(\overline{ov})) \\ c_i > 3 \rightarrow (\theta_{\max}(t_i) < \theta(\overline{ov})) \vee (\phi_{\max} < \phi(\overline{ov})) \end{cases} \quad (2.2)$$

Therefore, we maintain a list L of neighboring triangles $\{t_i\}$ whose counters have $c_i \in \{1, 2, 3\}$. When the sweep segment hits a new triangle t_j , we make $c_j \leftarrow 1$ and add t_j into L ; when a counter $c_j = 3$, after processing the current segment, we remove t_j from L .

Given a skeleton point p , for a boundary triangle mesh with N_V vertices, it takes $O(N_T \log N_T)$ to compute and sort all triangles following their segment angles. When we are sweeping a segment $\overline{pv_i}$, if the size of the active triangle list L is m , it takes $O(m)$ intersection-detecting operations. Therefore, the total complexity is $O(N_T \log N_T + N_V \cdot m)$. The incident triangle around a vertex v_i is generally very small (i.e., $m < \log N_T$). Therefore, the algorithm finishes visibility detection of p in $O(N_T \log N_T)$ time. On a skeleton containing k nodes, it takes $O(kN_T \log N_T)$ precomputation time to compute the visible region for all nodes.

2.3.2 Greedy and Optimal Guarding

Once visibility information for all skeletal nodes is computed, we want to pick a minimum sized point set that can cover all boundary vertices. It can now be converted to a set-covering problem which is also NP-complete [8]: Given the universe point set $V = \{v_i | i = 1, \dots, N_V\}$, and a family S of subsets $S_j = \{s_{j,k}\} \subseteq V$ (for every $j = 1, \dots, N_S$), a *cover* is a subfamily $C \subseteq S$ of sets whose union is V . We want to find a cover C that uses the fewest subsets in S . Here, V corresponds to the set of all vertices of ∂M ; for each skeletal node p_j , $j = 1, \dots, N_S$, S_j contains all the boundary vertices visible to p_j . Each C indicates a subset of skeletal nodes that can guard the entire region. Skeletons generated using medial axis-based methods with dense enough nodes usually ensure that S itself is a covering. This holds in all of our experiments. However, if a coarsely sampled skeleton cannot cover the entire V , we can easily include all those invisible vertices, i.e., their visible regions, into S .

A *greedy* strategy for the set covering is to iteratively pick the skeletal nodes p that can cover the largest number of unguarded vertices in V , then remove all guarded vertices $v \in S(p)$ from V meanwhile, and update S accordingly since the universe becomes smaller, until $V = \emptyset$. The greedy strategy is effective, and it yields $O(\log n)$ approximation [19] to the set-covering problem.

An *optimal* selection can be computed by 0–1 programming, also called integer linear programming (ILP). For every skeleton point p_i , $i = 1, \dots, N_S$, we assign a variable x_i such that:

$$\begin{cases} 1 & \text{if } p_i \text{ is chosen} \\ 0 & \text{otherwise} \end{cases}$$

The *objective* function to minimize is then $\sum_{i=1}^{N_S} x_i$, as we want to pick the fewest necessary points. Since every vertex $v \in V$ should be covered, for each such v_i , at least one of its visible skeletal nodes $P_i = \{p_j | v_i \in S(p_j)\}$ should be picked. Therefore, we minimize $\sum_{i=1}^{N_S} x_i$ subject to:

$$x_i = \{0, 1\} \wedge \sum_{p_j \in P_i} x_j \geq 1 \quad \forall v_i \in V$$

This objective function can be minimized using branch-and-bound algorithms. When the dimension is small (e.g., a few hundreds to a few thousands), we can use the TomLab optimization package [18] to solve it efficiently.

2.3.3 Hierarchical Guarding

Computing the optimal guarding based on ILP is highly time consuming and it limits the size of problems that we can handle: General 3-D volumetric shapes can easily have a number of vertices (20,000–200,000) on its boundary surface, which is too large for this optimization. On the other hand, the greedy algorithm generates the guards in a locally optimal manner. Furthermore, the greedy strategy is not robust against local geometric perturbations. For example, a small bump could lead to global structural variance of the guarding points. We propose a hierarchical guarding computation framework which is based on the progressive mesh [43], combining the 0–1 programming optimization and the adaptive greedy refinement.

We simplify the boundary mesh ∂M into several resolutions in the following form using a progressive mesh [43].

$$\partial M^i = \{V^i, F^i\} \quad \forall i = 0, \dots, m$$

In the coarsest level $i = m$, ILP optimization is performed on all elements $v \in \partial M^m$, and we get the coarsest level guard set $G_i = \{g_k^i\}$. Then, we progress to $i = m - 1$ level $\partial M^{m-1} = (V^{m-1}, F^{m-1})$.

1. Map existing guards $G_{i+1} = \{g_k\}$ to the closest finer level skeletal nodes $G_i = \{g_k'\}$ to locally adjust them to maximize their visible region $S(g_k')$.
2. Remove the least significant guards $\{g \mid |S(g)| < \varepsilon N_V\}$ from G^i .
3. Remove covered vertices $\{v \mid v \in S(g) \wedge g \in G^i\}$.

Then, we solve ILP again on uncovered boundary vertices. With details increase in finer levels, new guards will be inserted into G^i . Before applying the ILP optimization, we further conduct four *reduction* operations (see in the following) on uncovered regions to reduce the dimensions of the optimization. This progressive refinement ends when all boundary vertices are covered on the finest level $i = 0$.

Reduction: The dimension of the ILP optimization on each level can be reduced using the following reduction rules, without changing the size of the optimal solution. Suppose we store the visibility information in an incidence matrix A . If the skeletal node p_i can see the vertex v_j , then we let $a_{ij} = 1$, otherwise (p_i cannot see v_j), let $a_{ij} = 0$. Originally, the dimension of A is $N_S \times N_V$. The following four rules are applied to reduce it.

1. If column j has only one nonzero element at row i , we must take p_i in order to see v_j . Therefore, add p_i into G and remove column j . In addition, for all nonzero element a_{ik} , remove column k (we take p_i : all points that it sees are guaranteed to be covered, and thus now can be removed).
2. If for the couple of rows i_1 and i_2 , the following proposition holds:

$$a_{i_1 j} = 1 \rightarrow a_{i_2 j} = 1 \quad \forall j,$$

then p_{i_2} sees all vertices that p_{i_1} can see; and we can remove the entire row i_1 .

3. If for the couple of columns j_1 and j_2 , the following proposition holds:

$$a_{i j_1} = 1 \rightarrow a_{i j_2} = 1 \quad \forall i,$$

then guarding v_{j_1} guarantees the guarding of v_{j_2} , and we can remove the entire column j_2 .

4. If the matrix A is composed of several blocks, we partition A into several small matrices $\{A_k\}$.

In step 4, after removing vertices that have been seen by the adjusted guards from a coarser level, remaining boundary vertices could be partitioned to several connected components far away from each other, which can be optimized separately and more efficiently.

In our experiments, we simplify the boundary mesh to the coarsest level with 5,000 vertices for the first round ILP optimization. Generally, we make each iteration to add in another 10,000 vertices. When the size of constraints is around 5,000 and the size of variables (skeletal nodes) is around 1,000, the optimization usually takes 10–50 s to solve.

Our hierarchical scheme together with the reduction processing has the following important advantages over both the pure greedy strategy and the pure 0–1 optimization.

1. It is *much faster* than the nonlinear ILP optimization. The current framework can handle large-size geometric shapes.
2. With similar performance, it usually provides *better* guarding solutions than a pure greedy strategy.
3. It is *hierarchical* and therefore is *robust* and *stable* against geometric noise. In our HILP framework, refined local details tend to not change the global structure of the previously optimized guarding graph in coarser levels.

Figure 2.3 shows some examples of HILP guarding on sculpture data, and Fig. 2.1 shows the run-time statistics. We use these irregular sculpture data to demonstrate the significant efficacy of our algorithm since our prototype pipelines on hand are relatively simple. To thoroughly cover and inspect complex pipeline system such as in Fig. 2.1, using HILP, we can find its guarding point set efficiently.

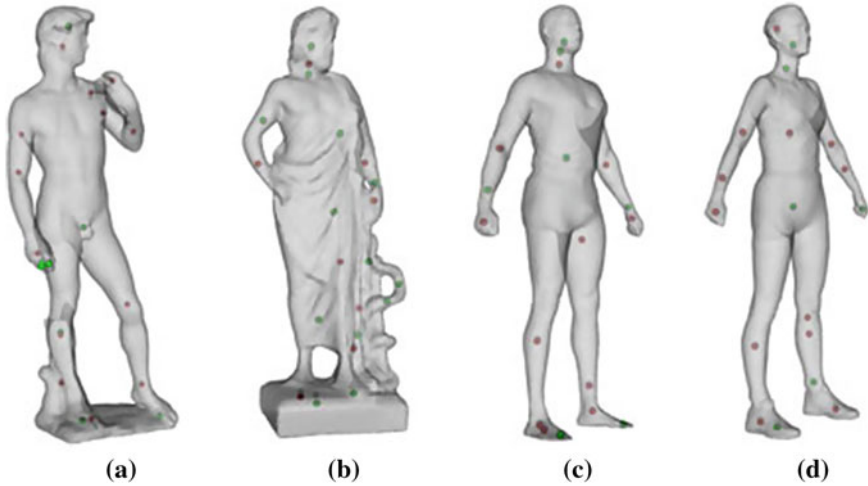


Fig. 2.3 Guarding statues using HILP. From *left to right* guarding on a few sculpture datasets. **a** Michelangelo's David. **b** Greek. **c** Cyberware Male. **d** Female. Small nodes are the guards, where *green nodes* are the latest computed guards on the finest level

2.4 Autonomous Pipeline Inspection

Our guarding algorithm computes the set of necessary checking spots for thorough inspection. The inspection robot only needs to go to each guarding point, construct the current height maps (see Sect. 4.4.1), and compare them with the precalculated templates for abnormal identifications. When geometric changes are detected, the system refines the identification of the abnormal areas (see Sect. 4.4.2) and extracts the boundary of the damaged region (see Sect. 4.4.3).

2.4.1 Height Maps Acquisition

The geometry of the pipeline environment is measured using laser range finders in our system. The distance from the robot to a point on the wall is captured and stored. A range image stores a set of depth information in a rectangle view-port along a direction, and we call it a height map.

The abnormal detection is based on the comparison between precalculated height map templates and the current height maps. We compare height maps on each inspection point. Suppose the laser scanning has two parameters, the scan range angle α and the sampling rate $s \cdot \alpha$ decide the field of view of the scanner, and s indicates the sampling resolution inside the field of view. As Fig. 2.4a shows, given a shooting direction, the scanner takes a snapshot of the environment, which produces a height map on a planar square region R , uniformly sampled with $s \times s$ points P . Since

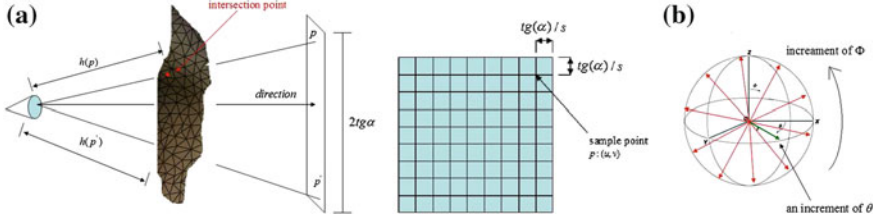


Fig. 2.4 **a** Shot of the laser scanning. The sample grid density is decided by the sample rate and the scan range of the laser scanner. Every sample point has a planar coordinate (u, v) corresponding to the actual intersection point in 3-D space. The laser scanner rotates to a direction and does a snapshot to project the surface to a planar with a range of $2 \tan \alpha$. The red point indicates the intersection result of p in the 3-D region, and the corresponding height data $h(p)$ are stored for detection. Scanning at every sample point, the height map is constructed. **b** Motion plan of the scanner. The scanner rotates to cover the whole spherical area. The red arrowheads are the rotated direction of ϕ , and the green is one step of rotation increment of θ

one snapshot can only cover an area within the current view angle, a planned motion sequence is necessary for the laser scanner to rotate and take pictures to cover the entire 360° . The laser scanner with the scan range α and the sample rate s is placed at a guarding position o which points toward an initial direction L ; the height map acquisition processes can be simply conducted as follows.

1. Using the local spherical coordinate system which is defined at o , given a direction $L(r, \phi, \theta)$, L takes a snapshot and gets a depth image $P(\alpha, s)$ (see Fig. 2.4a). Depth is defined on every point on the image $p \in P$, whose 2-D coordinate can be defined as (u, v) , for every $(u, v) \in [0, 1]^2$.
2. 3-D position of each boundary point v can be derived from the depth on its projection $p(u, v)$. The transformation can be represented by a rotation matrix.
3. After taking one depth image, the laser scanner rotates for another height map. The motion plan of the scanner follows the rotation sequences: first fix ϕ , rotate θ for N times, increase $\pi\alpha$ iteratively, and then increase ϕ to perform the θ rotation again until the whole spherical region is covered. The rotation path is shown in Fig. 2.4b.
4. Finally, we get the set of height maps $\{H_L\}$ collected in the aforementioned steps and save them together with the starting projection direction on every guarding point.

In practice, during height map acquisition, the geometry of regions far away from a guard can be visible but captured less accurately because of the precision of the range finder or the sampling resolution. To tackle this issue, before guarding computation, we add a simple parameter d_p for each skeleton node p : If a boundary point $q \in \partial M$ is visible to p but far away, i.e., distance $|\overline{pq}| > d_p$, we consider q to be invisible. In other words, a guard only sees points in a bounded distance. The whole optimization algorithm can be applied exactly in the same way. In long and thin environments, more guards may be necessary, but each guarding region will have less long antenna, and the inspection accuracy will be improved. A heuristic

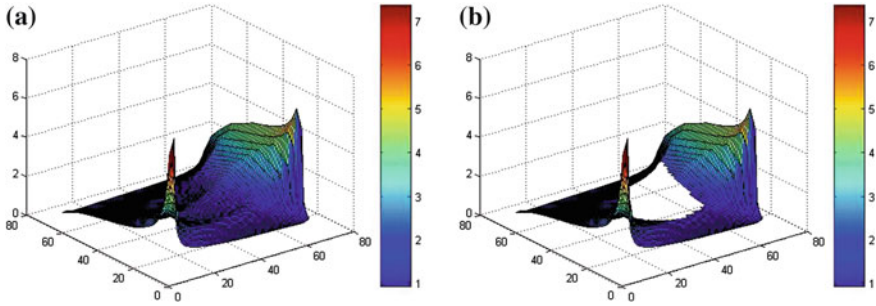


Fig. 2.5 Height maps comparison. **a** Template height map. **b** Height map of an abnormal region. The missing region indicates a hole

setting for d_p can be $d_p = \alpha \cdot d_p^0$, where α is a constant parameter, and d_p^0 is the distance from p to its nearest visible boundary point.

2.4.2 Abnormal Boundary Detection

After all height maps are obtained, on every guarding point, we compare the precollected template height maps $\{H_L\}$ (see Fig. 2.5a) and the current height maps $\{H'_L\}$ (see Fig. 2.5b). If the height information changes; i.e., $|h(u, v) - h'(u, v)| > \varepsilon$, we consider the region around (u, v) as a defective region and report (u, v) as an abnormal point. ε is a similarity threshold. In practice, the acquired depth data could have geometric and topological noise. By adjusting ε , the system can tolerate small deviations because of certain acquisition noise. On the other hand, a local efficient data preprocessing step in topological denoise or geometric completion/fairing [17, 38] could also be helpful in cleaning environment noise.

Compared with reporting simply a set of sampled abnormal points, an accurate estimation of the bad region's shape is desirable. When a defective region is detected, in order to “zoom in” to see the shape of the defective region, we need to examine more sampling points by performing denser scanning around this region. During the initial scanning of the original pipeline, we may not do very dense sampling; therefore, with the same number of depth acquisition, we can capture a larger region in every shot for better efficiency. Regions among sampling points are approximated using bilinear interpolation $h(E)$:

$$\begin{aligned}
 h(E) \approx & \frac{h(u_1, v_1)}{(u_2 - u_1)(v_2 - v_1)}(u_2 - u)(v_2 - v) + \frac{h(u_2, v_1)}{(u_2 - u_1)(v_2 - v_1)}(u - u_1)(v_2 - v) \\
 & + \frac{h(u_1, v_2)}{(u_2 - u_1)(v_2 - v_1)}(u_2 - u)(v - v_1) + \frac{h(u_2, v_2)}{(u_2 - u_1)(v_2 - v_1)}(u - u_1)(v - v_1)
 \end{aligned}$$

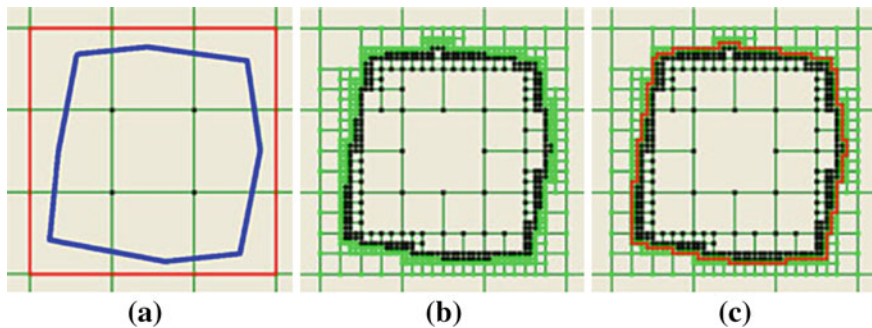


Fig. 2.6 Quadtree splitting and boundary extraction. **a** Polygon colored in blue indicates a defective region on the pipe wall. The green nodes are normal points, and the black ones indicate the abnormal height data; the boundary extracted from the quadtree without quadtree splitting, colored in red, is coarse. **b** Quadtree cells split. **c** Extracted boundary (red curves) after quadtree refinement

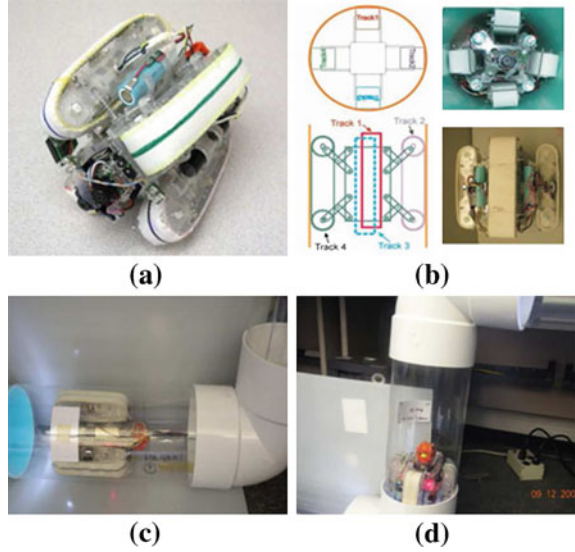
To report a refined boundary shape of an abnormal region, we use a quadtree [29] subdivision scheme: Near such a region, boundary cells split accordingly to get the refined geometry. For each new point (u', v') , we detect its height $h'(u', v')$ and compare it with the interpolated height on the template height map $h(u', vv')$. Figure 2.5 shows an example. The max level of the subdivision is determined by the resolution of the laser range finder. The boundary of holes, clogs, and deformations can be detected/refined using this paradigm, since these geometric changes always lead to changed height maps.

2.4.3 Boundary Extraction

When the refinement is done, the boundary of the abnormal region can be extracted and reported. We conservatively link the normal points on boundary cell to form the boundary: Starting from a normal point on a boundary cell, linking the normal points in the cell along the edge, getting a neighboring boundary cell, and repeating this process until all boundary cells are traversed.

Figure 2.6 illustrates the process of quadtree splitting and extraction. In a snapshot, the grids are the cells, and the green polygon indicates a hole on the pipeline wall. Normal points are colored in green, while abnormal points are colored in black. In the coarsest resolution, we get the red boundary shown in (a). We can get finer boundary shape (b) and the extracted boundary is depicted in red in (c).

Fig. 2.7 Prototype robot FAMPER. **a, b** Robot and its design. **c, d** Robot inspecting a pipe



2.5 Result of Simulated Experiments

Our guarding-driven pipe inspection system can be implemented on the prototype pipeline robot [16], which can be used for the inspection of pipelines. This robot consists of four wall-press caterpillars that are operated by two dc motors each to provide steering capability to go through 45° , 90° elbows, T -branches, and Y -branches and make a superior performance in all types of complex networks of pipelines (see Fig. 2.7). The robot is also equipped with a powerful computing system that makes it extendable to various sensing and actuating devices, such as in our experimental system, for localization and laser scanning. The height information can be obtained using a multislit laser range scanner [11], which has the size of about $110 \text{ m} \times 90 \text{ mm}$ and includes a laser projector and a charge-coupled device (CCD) camera (the laser projector in [11] is StockerYale Mini-715L, which projects 15 slits, with an adjacent slit angle 2.3° ; the CCD camera is Point Grey Research Flea2, which measures 330 points). The height information can also be obtained by laser scanners in [15, 30] or other range cameras (e.g., [26, 42]).

The rotation of the sensor can be controlled by a small mechanical platform installed in the rear part of the robot; therefore, the scan can be conducted radially inside the pipe toward different directions. Several effective sensing platforms with similar mechanism have been developed [30].

We develop a simulated platform for testing our algorithm (see Fig. 2.8), which simulates the process of inspection. We apply our procedure on this platform using several complicated 3-D virtual pipelines. The simulated results are convincing and show the effective inspection on pipeline geometry.



Fig. 2.8 Simulation environment. The 3-D model of the FAMPHER robot in the pipeline; *red points* denote guarding points

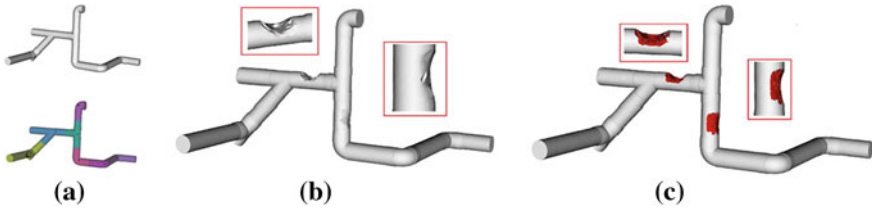


Fig. 2.9 Inspection process on a simulated pipeline. **a** *Upper* simulated model. *Lower* regions guarded by different guard points encoded in different colors. **b** Damaged pipe with some holes. **c** Damages are detected, whose boundaries are extracted and shown in *green*

2.5.1 Hole Detection

If a hole appears on the pipeline, it can be identified online when the robot reaches the guarding point that covers this region and matches the captured range depth images with the stored templates. We simulate this on pipeline meshes M by randomly generating some missing regions. An experiment is shown in Fig. 2.9. A pipeline model and the necessary guards are shown in (a), where regions covered by each guard are rendered in a specific color for the visualization purpose. Any given region of the pipeline is covered by at least one guard and, therefore, is colorized. The height maps can then be generated as templates, measuring the “correct” distance from each guarding site to the pipe wall toward specific directions. This simulates range images obtained by a laser scanner. Now, we simulate the appearance of defected regions on the pipeline by generating some missing regions as shown in (b). When the robot checks height maps on guarding points, these holes can be immediately detected and illustrated in (c).

Another example is shown in Fig. 2.10; this pipe is guarded by 12 points (a). In addition, the damaged region of the pipe is big and with complex topology (b). In this case, the robot should check from more than one guarding points in order to detect the entire shape of such a big hole. The entire defect geometry is extracted by composing boundaries detected from different guarding sites. The merged boundary loop is illustrated in (c).

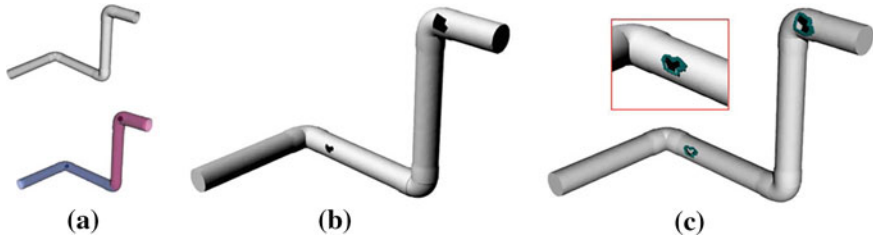


Fig. 2.10 Guarding and inspection process on a more complicated pipeline. **a** *Upper* simulated model. *Lower* guarding points and guarded regions rendered in different colors, respectively. **b** Damaged pipe with big and concave holes. **c** Large holes are detected/extracted from more than one guarding points; the whole big boundary is composed of several extracted subboundaries and identified separately from different guarding points, as shown in *green*

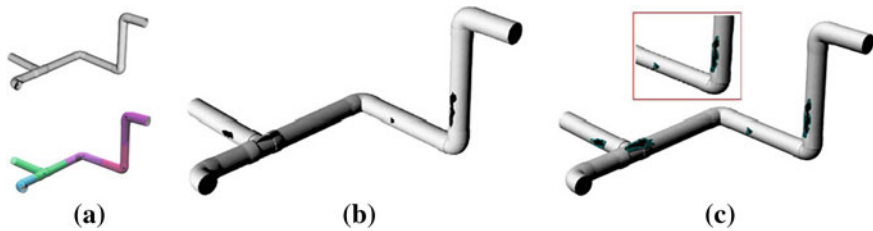


Fig. 2.11 Inspection process on a deformed pipeline. **a** *Upper* simulated model. *Lower* regions guarded by different guard points encoded in different colors. **b** Deformed pipe. **c** Deformations are detected, whose regions are extracted (and refined) and shown in *red*

2.5.2 Deformation Detection

Small deformation such as bending and erosion can also be detected in our system as shown in Fig. 2.11. The detected deformed region is colored in red.

2.5.3 Clogging Detection

Clogging also changes the scanned geometry of the pipeline and can be detected. Figure 2.12 shows an example. The clogged solid (green) is detected, and its boundary geometry is reconstructed using height maps as illustrated in dark red. The robot will report the clogs when it is detected. In this example, the reconstruction merges the geometry of the blocking stuff from two aspects (from two guarding points) using their corresponding height maps.

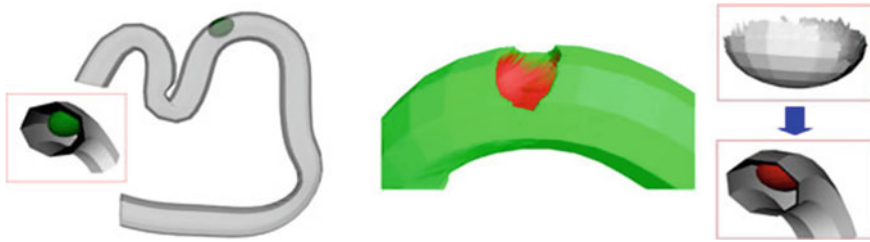


Fig. 2.12 *Left* pipeline blocked by a solid (in green). *Middle* clog detection on the pipeline. Height maps constructed and merged by detected views from different guarding points. *Green* regions indicate normal status, while the *red* regions indicate the abnormal height information corresponding to the clog geometry. *Right* inspection result, which is the reconstructed surface colored in *dark red*, describes the location and geometry of the blocking region

2.6 Summary and Outlook

We have proposed an efficient 3-D guarding algorithm that can cover a given complicated environment using as few as possible points. Finding an efficient solution to the fundamental problem of how to inspect on fewest points yet thoroughly covering the entire environment can greatly benefit the autonomous design of inspection and exploration robots. We have developed a simulation system of pipeline inspection and conducted experiments to evaluate the efficacy of our system. With our optimal guarding, abnormal geometric changes of the pipeline such as holes, clogs, and deformation can be thoroughly detected online. The remaining challenging issue for the current system is the dynamic environment mapping. The initial geometry of the pipeline system needs to be scanned to a digital model before the region-guarding computation. In addition, after the guarding spots are computed, the pipeline wall needs to be marked so that the robot can localize itself to know whether it is on the spot. The development of a system that a robot without this prior knowledge can do simultaneous 3-D mapping (i.e., reconstructing the map of the environment) and localization will be highly desirable. In the near future, we will study these localization and dynamic environment mapping problems by exploring effective partial matching of range images.

References

1. D. Avis, G. Toussaint, An efficient algorithm for decomposing a polygon into star-shaped polygons. *Pattern Recogn.* **13**, 395–398 (1981)
2. B. Ben-Moshe, M.J. Katz, J.S. Mitchell, A constant-factor approximation algorithm for optimal terrain guarding, in *Proceedings of ACM/SIAM Symposium on Discrete Algorithms*, (2005), pp. 515–524
3. I. Bjorling-Sachs, D. Souvaine, An efficient algorithm for guard placement in polygons with holes. *Discr. Comput. Geom.* **13**, 77–109 (1995)

4. J.Y. Choi, H. Lim, B.-J. Yi, Semi-automatic pipeline inspection robot systems, in *Proceedings of SICE-ICASE International Joint Conference*, (2006), pp. 2266–2269
5. V. Chvatal, A combinatorial theorem in plane geometry. *J. Comb. Theory* **18**, 39–41 (1975)
6. N. Cornea, D. Silver, P. Min, Curve-skeleton properties, applications, and algorithms. *IEEE Trans. Vis. Comput. Graph.* **13**(3), 530–548 (2007)
7. S. Costello, D. Chapman, C. Rogers, N. Metje, Underground asset location and condition assessment technologies. *Tunn. Undergr. Space Technol.* **22**(5–6), 524–542 (2007)
8. T.K. Dey, J. Sun, Defining and computing curve-skeletons with medial geodesic function, in *Proceedings of 4th Eurographics Symposium on Geometry Processing*, (Eurographics Association, Aire-la-Ville, Switzerland, 2006) pp. 143–152
9. O. Duran, K. Althoefer, L. Seneviratne, State of the art in sensor technologies for sewer inspection. *IEEE Sens. J.* **2**(2), 73–81 (2002)
10. A. Efrat, S. Har-Peled, Guarding galleries and terrains. *Inf. Process. Lett.* **100**(6), 238–245 (2006)
11. R. Finkel, J. Bentley, Quad trees: a data structure for retrieval on composite keys. *Acta Inf.* **4**, 1–9 (1974)
12. S. Fisk, Ashort proof of Chvatal's watchman theorem. *J. Comb. Theory Ser. B* **24**(3), 374 (1978)
13. E. Gyor, F. Hoffmann, K. Kriegel, T. Shermer, Generalized guarding and partitioning for rectilinear polygons. *Comput. Geom.* **6**(1), 21–44 (1996)
14. T. He, L. Hong, D. Chen, Z. Liang, Reliable path for virtual endoscopy: ensuring complete examination of human organs. *IEEE Trans. Vis. Comput. Graph.* **7**(4), 333–342 (2001)
15. J. Hertzberg, F. Kirchner, Landmark-based autonomous navigation in sewerage pipes, in *Proceedings of 1st Euromicro Workshop on Advance Mobile Robot*, (Oct 1996), pp. 68–73
16. Hogeschoolzeeland, (2007), Pipes, tubes, machinery and steam turbine at a power plant [Online]. Available <http://goo.gl/fRp3c>
17. H. Hoppe, Progressive meshes, in *Proceedings of SIGGRAPH*, (New York, 1996), pp. 99–108
18. D.S. Johnson, Approximation algorithms for combinatorial problems, in *Proceedings of 5th Annual ACM Symposium On Theory of Computing*, (New York, 1973), pp. 38–49
19. J. Kahn, M. Klawe, D. Kleitman, Traditional galleries require fewer watchmen. *SIAM J. Algebraic Discr. Methods* **4**(2), 194–206 (1983)
20. D.-G. Kang, J.B. Ra, A new path planning algorithm for maximizing visibility in computed tomography colonography. *IEEE Trans. Med. Imag.* **24**(8), 957–968 (2005)
21. M.J. Katz, G.S. Roisman, On guarding the vertices of rectilinear domains. *Comput. Geom.: Theory Appl.* **39**(3), 219–228 (2008)
22. J.-H. Kim, G. Sharma, N. Boudriga, S. Iyengar, Ramp system for proactive pipeline monitoring, in *Proceedings of International Conference on Communication System, Network*, (2010), pp. 1–2
23. J. Kim, G. Sharma, N. Boudriga, S. Iyengar, Spams: Asensor-based pipeline autonomous monitoring and maintenance system, in *Proceedings of International Conference on Communication System, Network*, (2010), pp. 1–10
24. J.-H. Kim, G. Sharma, S. Iyengar, FAMPER: a fully autonomous mobile robot for pipeline exploration, in *Proceedings of 2010 IEEE International Conference on Industrial Technology*, (2010), pp. 517–523
25. D.-H. Koo, S.T. Ariaratnam, Innovative method for assessment of underground sewer pipe condition. *Autom. Construct.* **15**(4), 479–488 (2006)
26. T. Kuroki, K. Terabayashi, K. Umeda, Construction of a compact range image sensor using multi-slit laser projector and obstacle detection of a humanoid with the sensor, in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Oct 2010), pp. 5972–5977
27. D.T. Lee, A.K. Lin, Computational complexity of art gallery problems. *IEEE Trans. Inf. Theory* **32**(2), 276–282 (1986)
28. D. Levesque, M. Ochiai, A. Blouin, R. Talbot, A. Fukumoto, J.-P. Monchalain, Laser-ultrasonic inspection of surface-breaking tight cracks in metals using SAFT processing. *Proc. IEEE Ultrasound Symp.* **1**(8–11), 753–756 (2002)

29. X. Li, Z. Yin, L. Wei, S. Wan, W. Yu, M. Li, Symmetry and template guided completion of damaged skulls. *Comput. Graph.* **35**, 885–893 (2011)
30. J.-M. Lien, *Approximate star-shaped decomposition of point set data*, in *Eurographics Symposium on Point-Based Graphics* (Czech Republic, Prague, Presented at the Eurograph, 2007)
31. J.M. Mirats Tur, W. Garthwaite, Robotic devices for water main in-pipe inspection: a survey. *J. Field Robot.* **27**, 491–508 (2010)
32. J. Moraleda, A. Ollero, M. Orte, A robotic system for internal inspection of water pipelines. *IEEE Robot. Autom. Mag.* **6**(3), 30–41 (1999)
33. A. Nassiraei, Y. Kawamura, A. Ahrari, Y. Mikuriya, K. Ishii, Concept and design of a fully autonomous sewer pipe inspection mobile robot “KANTARO”, in *Proceedings of IEEE International Conference on Robotics and Automation*, 10–14 Apr 2007, pp. 136–143
34. J. O’Rourke, K. Supowit, Some NP-hard polygon decomposition problems. *IEEE Trans. Inf. Theory* **29**, 181–190 (1983)
35. J. O’Rourke, *Art Gallery Theorems and Algorithms* (Oxford University Press, London, 1987)
36. J.O. Rourke, Galleries need fewer mobile guards: a variation on Chvatal’s theorem. *Geom. Dedicata* **14**, 273–283 (1983)
37. J.O. Rourke, An alternate proof of the rectilinear art gallery theorem. *J. Geom.* **21**, 118–130 (1983)
38. H. Schoner, B. Moser, A.A. Dorrington, A.D. Payne, M.J. Cree, B. Heise, F. Bauer, A clustering based denoising technique for range images of time of flight cameras, in *Proceedings of International Conference on Computational Intelligence for Modelling Control and Automation*, (2008), pp. 999–1004
39. D. Schuchardt, H.-D. Hecker, Two np-hard art-gallery problems for ortho-polygons. *Math. Logic. Quart.* **41**, 261–267 (1995)
40. M. Silk, The determination of crack penetration using ultrasonic surface waves. *NDT Int.* **9**(6), 290–297 (1976)
41. H. Streich, O. Adria, Software approach for the autonomous inspection robot MAKRO. *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3411–3416 (Apr. 2004)
42. J. Thielemann, G. Breivik, A. Berge, Pipeline landmark detection for autonomous robot navigation using time-of-flight imagery, in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, (23–28 Jun 2008), pp. 1–7
43. TOMLAB v3.0 User’s Guide, Technical report IMA-TOM-2001-01, Department of Mathematics and Physics, Malardalen University, Vasteras, Sweden, 2001
44. J. Urrutia, *Art gallery and illumination problems*, in *Handbook of Computational Geometry*, Amsterdam (North-Holland, The Netherlands, 2000)
45. R. Wirahadikusumah, D.M. Abraham, T. Iseley, R.K. Prasanth, Assessment technologies for sewer system rehabilitation. *Autom. Construct.* **7**(4), 259–270 (1998)
46. W.W. Zhang, B.H. Zhuang, Non-contact laser inspection for the inner wall surface of a pipe. *Meas. Sci. Technol.* **9**(9), 1380 (1998)

Mathematical Theories of Distributed Sensor Networks

Iyengar, S.S.; Boroojeni, K.G.; Balakrishnan, N.

2014, XIV, 153 p. 39 illus., 31 illus. in color., Hardcover

ISBN: 978-1-4419-8419-7