

# Interpreting Thermal 3D Models of Indoor Environments for Energy Efficiency

Girum G. Demisse, Dorit Borrmann, and Andreas Nüchter

**Abstract**—In recent years, 3D models of buildings are used in maintenance and inspection, preservation, and other building related applications. However, the usage of these models is limited, because most models are pure representations with no or little associated semantics. In this paper, we present a pipeline of techniques used for interior interpretation, object detection, and adding energy related semantics to windows of a 3D thermal model. A sequence of algorithms is presented for building the fundamental semantics of a 3D model. Furthermore, a *Markov Random Field* is used to model the temperature distribution of detected windows to further label the windows as either *open*, *closed* or *damaged*.

**Key words:** Energy efficiency, 3D thermal model, Boltzmann distribution, energy function, window detection

## I. INTRODUCTION

Efficiency in energy usage is a fundamental step in adopting Green energy and conservation of natural resources: the European Commission estimates the largest and cost-effective energy saving potential lies in residential ( $\approx 27\%$ ) and commercial ( $\approx 30\%$ ) buildings [5]. Among other factors, uncontrolled air leakage, known as air infiltration, plays a significant role in energy consumption, both during heating seasons, but also in geographical locations where air conditioning is a necessity. Infrared thermometers are mainly used to detect faulty insulation in a labor intensive and time taking manner [8], [13]. Consequently, automating the process of detecting air infiltration has a significant impact on efficiency, cost and effectiveness of the leakage detection and proofing process. A high rate of air infiltration is also caused by opened windows or doors. This can easily be resolved by human intervention once detected.

Motivated by the economic and environmental impact we contribute to the efforts of fully automating the energy leakage detection process. Building on results obtained in [5], where a method for acquiring a 3D thermal model of a building is

presented, we present a sequential pipeline of algorithms for 3D scene understanding and temperature distribution modeling as given in Fig. 1. Particularly, the temperature distribution is used to model the state of a window, as either opened, closed, or damaged, i.e., not properly insulated. After describing our autonomous robot and reviewing related work, we define and formalize the problem mathematically in section II. Our solution pipeline uses probabilistic modeling and pre-processing of a 3D point cloud and is presented in section III and IV. Finally, experimental results are presented in Section V. Section VI concludes the paper.

### A. Automatic Acquisition of Thermal 3D Models

Thermal imaging is state of the art in recording energy related issues, while terrestrial laser scanning has been used for years to create 3D models. The combination of these two yield a 3D model that contains precise temperature information including the dimensions of heat and air leaks.

The setup for simultaneous acquisition of 3D laser scan data and thermal images is the robot Irma3D (cf. Fig. 2). Irma3D is built of a Volksbot RT-3 chassis. Its main sensor is a Riegl VZ-400 laser scanner from terrestrial laser scanning. The optris PI160 thermal camera has an image resolution of  $160 \times 120$  pixels and a thermal resolution of  $0.1^\circ \text{C}$ . It acquires images at a frame rate of 120 Hz and with an accuracy of  $2^\circ \text{C}$ . The laser scanner acquires data with a field of view of  $360^\circ \times 100^\circ$ . To achieve the full horizontal field of view the scanner head rotates around the vertical scanner axis when acquiring the data. We take advantage of this feature when acquiring image data. Since the thermal camera is mounted on top of the scanner, it is also rotated. We acquire 9 images with the camera during one scanning process to cover the full  $360^\circ$ .

To acquire thermal 3D point clouds of indoor environments, we have performed the intrinsic and extrinsic calibration using a special pattern. The color mapping procedure regards the inaccuracies and low resolution camera issues [5]. In all our experiments, we pay attention to general rules of thermography, such as the weather. There has to be a temperature difference between indoor and outdoor of at least  $15^\circ$  to measure a valid,

---

A. Nüchter and D. Borrmann are with the Robotics and Telematics group at University of Würzburg, Germany, andreas@nuechti.de. The work was performed while the authors were at Jacobs University Bremen gGmbH, Germany.

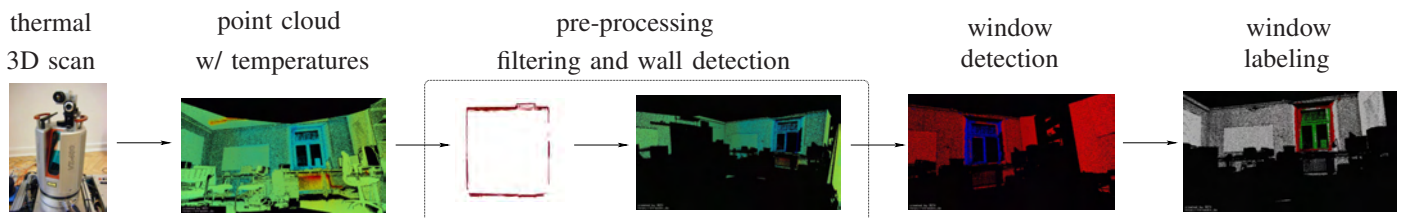


Fig. 1: Overview of window Detection and labeling pipeline.

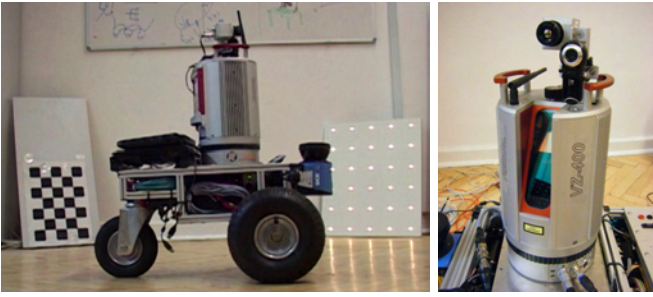


Fig. 2: The robot Irma3D, with a 3D laser scanner, a thermal camera and a webcam.

noise-reduced thermogram. Other error sources such as sun light, wind and rain, clear sky were minimized as well.

### B. Related work

According to Xiong and Huber, creating a 3D model of an indoor environment has notable advantages in maintenance, management, and architectural renovation of buildings [18]. The traditional approach to create a 3D model is based on CAD (Computer-aided design) tools and manual measurements, despite the consequential high cost and lengthy time consumption. However, recent technological advances in laser scanning technology prompted the full automation of 3D model creation [1], [16]. 3D model creation is still a fairly complex task with challenges at different levels. These challenges can be categorized in two; *lower level* problems and *higher level* problems. Lower level problems are successfully explored in the computer vision community. In 3D point cloud processing the acquisition of accurate data is solved by technological means. Computer vision methods are widely used to solve fundamental problems such as registration [3], i.e., solving simultaneous localization and mapping [6], and representation, i.e., octrees [10] and range images. The addition of semantics belongs to the second group of problems. Semantics range from primitive shape detection to higher level knowledge inference. Several shape detection methods have been proposed in the computer vision community that can cope with uncertainties and clutter in the data sets [9], [15]. Additionally, the current trend in 3D point cloud interpretation is to infer higher level knowledge [12].

Building thermal 3D models of environments received some attention recently. Borrmann et al. [5] and Vidas et al. [17] focus on co-calibrating a thermal camera and a 3D range sensor. Ham and Golparvar-Fard model and evaluate thermal models of building exteriors and the energy performance of buildings [8]. However, to the best of our knowledge, there has not been any work done in automatic temperature analysis for the understanding of an object state.

## II. PROBLEM DEFINITION

Inferring higher level knowledge about the property of an object can be seen as two problems that are highly related. First, the detection of objects that belong to a certain class. This is a problem where the emphasis is on understanding and modeling of time-invariant properties of a certain class of objects, e.g. all windows are made of glass, so that the

properties are used to recognize an object of that class. Second, the inference about the *object*, rather than the class, using specific knowledge. In the second case, the problem is recognizing properties of an object that are observed in a certain time frame under a certain condition; which, of course, gives information about the objects state rather than the class it belongs to. Consequently, each of the above problems is usually solved separately and sequentially. In fact, the solution space of the first problem is the domain/problem space of the second problem. In this paper, we will be dealing with window detection and labeling, i.e., assessing windows either as open (O), closed (C) or damaged (D), i.e., a window without the proper insulation. Apparently, window detection belongs to the first group of problems while window labeling belongs to the second group of problems.

Although, window detection from a point cloud representation of a room full of objects is a challenging task we have designed sequential pre-processing modules which essentially reduce window detection to the estimation of a mapping function  $f(\cdot)$ . Assuming we have successfully identified 3D points that represent a window, the 3D points are considered as random variables taking a temperature value, from  $\mathbb{R}$ . Thus, the labeling of windows as closed, opened or damaged is formulated as a probability distribution modeling problem. In general, probabilistic models for object recognition are categorized either as *generative models* or *discriminative models*. The former one attempts to model the *joint* probability distribution  $P(X, Y)$  between the data denoted by  $X$  and the label denoted by  $Y$ . Alternatively, the discriminative model approach is to model the *posterior* probability  $P(Y|X)$  directly from the data. A discriminative model is widely believed to be the better modeling technique with better predictive ability [11]. Consequently, we have chosen the discriminative modeling approach, and thus the *posterior* probability of a label, e.g., C, is modeled as:

$$t_s = f(x), \quad (1)$$

$$P(C|t_s) = \frac{p(t_s|C)P(C)}{p(t_s)}, \quad (2)$$

where  $x$  is the fully registered thermal 3D point cloud model of the room,  $f(\cdot)$  is a function that takes this model as an input and outputs the temperature distribution of the detected window,  $t_s \in \mathbb{R}^n$ , where  $n$  is the number of 3D points representing the window. Since  $p(t_s)$  is exactly the same for all the labels it has no effect on the label specific *posterior*, and thus is ignored. Additionally, the probability of a window being closed is assumed to be exactly the same as being open or being damaged. In fact, this might not be true but we have no prior information to assume otherwise. Therefore, the modeling task is:

$$P(C|t_s) \approx p(t_s|C)$$

In summary, labeling a window as open, closed or damaged is formulated as estimating the conditional probability distribution of every label, followed by a decision rule, where the decision will be based on MAP (Maximum *a posteriori*) to minimize the expected error [4]. MAP is summarized as:

$$\hat{y} = \arg \max_{y_i \in Y} P(y = y_i | t_s) \quad (3)$$

where  $\hat{y}$  is the final label assigned to the temperature distribution of a window  $t \in \mathbb{R}^n$ , and  $Y = \{\mathbf{C}, \mathbf{O}, \mathbf{D}\}$ . In the following sections a solution to the problem formulation given in Eq. (2) is presented, respectively.

### III. WINDOW DETECTION

Window detection is a difficult task with many associated problems. Here, in this paper, we will describe a simple but effective detection technique exploiting our hardware. Intuitively, the core idea is the difference in material property, i.e., all windows are made of glass and, walls are made of some other material. This means, given a thermal 3D point cloud of a room that contains only windows and walls, there is a thermal conductivity difference between the wall material and the window material. There will always be a temperature difference which is used to recognize one from the other. However, a 3D point cloud representation of a room contains in addition other objects and clutter. Consequently, a mandatory pre-processing has to be done to detect and remove these other objects from the 3D point cloud.

#### A. Pre-processing

Filtering objects inside rooms and points that are scanned through windows (see Fig. 3), is a challenging task that is simplified with practical and realistic assumptions. The assumptions taken are:

- A room has a rectangular shape.
- The scanner is located inside a room.
- The walls and windows are not *completely* occluded, i.e., some part of the wall is always visible.
- The thermal 3D point clouds are registered to a single co-ordinate system.

Assuming the above conditions are true, a sequential procedure is proposed for filtering out objects from the scene, such that the point cloud consists then only of walls and windows.

1) *From 3D to 2D*: Assuming that windows are located on the walls and not on the ceiling, neither the height of the room nor points representing the floor and ceiling are important for window detection. Hence, points representing the floor and ceiling need to be filtered out from the scan. Floors and the ceilings are, again, almost always parallel to each other and perpendicular to the walls. Thus, the normal vector of a 3D point representing the ceiling is parallel to a wall. Given the above, we conclude that a 3D point is representing the floor or ceiling if its normal vector is perpendicular to the  $x-y$ -plane, or parallel to the  $x-z$ -plane, where  $z$  is the vertical axis. Let  $\mathbf{a} = (1, 0, 0)$  be a vector on the  $x$ -axis, and  $\mathbf{n}_i$  be the normal vector of the  $i^{th}$  point, then the following is true if the 3D point is on the floor or ceiling:

$$\mathbf{a} \cdot \mathbf{n}_i = \|\mathbf{a}\| \|\mathbf{n}_i\| \cos \theta = \{80^\circ, \dots, 100^\circ\}$$

We set the threshold to  $\geq 10^\circ$ , due to noise and inaccuracy in calculating normals.

Inspired by the work of Xiong and Huber [18] our software further simplifies the 3D representation of the room to a 2D representation, since the height of the room is not really needed

in wall detection. Thus, the 3D points are transformed to a 2D plane as follows:

$$\begin{pmatrix} \mathbf{p}' \\ 0 \end{pmatrix} = \mathbf{p} - (\mathbf{p} \cdot \mathbf{n})\mathbf{n}$$

where  $\mathbf{n}$  is the normal vector of the projection plane,  $\mathbf{p}' \in \mathbb{R}^2$  is the projected point, and  $\mathbf{p} \in \mathbb{R}^3$ , is the point to be projected. Now that the room is projected onto a 2D plane, walls are represented with lines instead of planes (cf. Fig. 3).

2) *Wall Detection*: The projection of the room to a 2D space enables us to work with lines as walls instead of planes. We have assumed that the room has a rectangular shape, that the origin of the scan is inside the room, and that the walls are not completely occluded. Hence we can conclude that the  $x$ - and  $y$ -axes of the 2D plane with the origin at the scanner location intersect with each all four wall lines independent of the orientation and the position of the scanner, as long as it is in the room. The only exception is when the axes intersect with the corners. As a line equation is defined by two points that lie on the line we need to detect two points on each wall (line) to detect all the walls of a room.

The *first* four points are determined by selecting the farthest point on the  $\pm x$ -axis and  $\pm y$ -axis. To determine a *second* point on each line, the registered scans are rotated by a given angle  $\theta_2$ . Again the farthest point on each  $\pm x$ - and  $y$ -axes are selected. Rotating the selected points back by  $-\theta_2$  yields the second set of points needed for defining the four line equations.

The algorithm fails in cases where the furthest point on one of the axes is not a wall point, e.g. due to occlusions, noise or windows. Further problems occur when one of the intersecting points lies in a corner of the room. To ensure an accurate and robust wall detection the following conditions are introduced:

- Each detected line has to be perpendicular to two lines, the ones it is intersecting with, and parallel to the other, the one it is not intersecting with. This is a hard constraint that has to be fulfilled in order to detect the walls with a reasonable accuracy; in other words, the slope of two intersecting lines should satisfy  $S_1 = \frac{-1}{S_2}$ .
- To avoid outliers there should be a considerable number of points at all four intersection points of the four detected lines. This is more of a soft constraint, especially, in a heavily occluded scan. Consequently, this constraint is mainly used to measure the confidence level, i.e.,  $error = 1 - confidence$ .

The confidence of acceptance is quantified by counting the intersection points that have a considerable number of points on and around them, e.g., if of the four intersection points three has a significant number of points on and around them the confidence is 3/4 or 75%.

If the constraints are not met the scan is rotated by an angle  $\theta_1$  and two new sets of intersection points are calculated. The procedure is outlined in Algorithm 1. After a successful detection of the lines representing the walls all points that are further than a threshold away from the wall are removed. An example is given in Fig. 3.



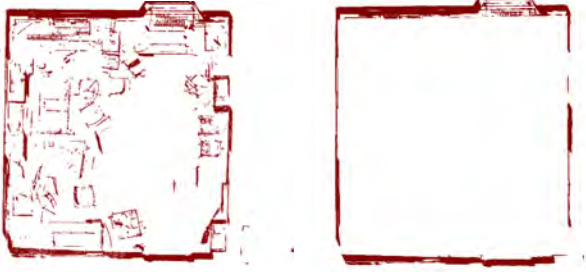


Fig. 3: Top: A 3D thermal model of the Automation Lab at Jacobs University Bremen. Bottom left: 2D projection of the room with the x-y axis colored in red. Bottom right: The detected outer rectangular shape, or walls of the room.

---

#### Algorithm 1: Wall Detecting Algorithm

---

**Data:** Registered and pre-processed scan  
**Result:** Equation of four lines  
initialization;  
 $\theta_1, \theta_2, \varepsilon$ ;  
**while** search **do**  
    calculate the first axis-line intersections;  
    rotate scan by  $\theta_2$ ;  
    calculate the second axis-line intersections;  
    rotate scan by  $-\theta_2$ ;  
    calculate the lines equation;  
    **if**  $(S_1 + \frac{1}{S_2}) < \varepsilon$  **and**  $(S_3 + \frac{1}{S_4}) < \varepsilon$  **then**  
        count = # of corner points where significant #  
        of points are found;  
        confidence = count/4;  
        search = false;  
    **else**  
        └ rotate scan by  $\theta_1$ ;

---

#### B. Window Detection

The result of the pre-processing step is a 3D model of the walls of the room. Now, the remaining point cloud is dominated by points from the wall, which means most points have an almost similar temperature value. However, 3D points of a window and points around it show a considerable temperature difference from the wall points, cf. Fig. 4. Thus, the window detection technique aims to exploit these temperature differences as a main feature.

Since there are significantly more 3D points that represent the walls than those that represent the window area a typical temperature distribution in a room takes a bell curved shape,

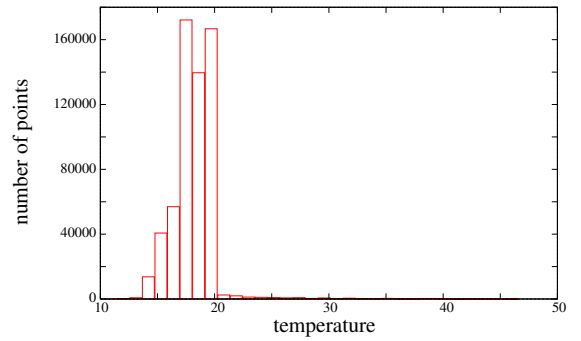


Fig. 4: Typical temperature distribution of a 3D scan with walls and windows only.

cf. Fig. 4. The rare ends of this temperature distribution correspond to temperature peaks. On the lower end these are objects that are cooler than the room temperature, i.e., mostly windows. The upper end represents hotter objects that are close to the wall, e.g., computers or heaters. Thus, filtering points with a constant threshold that is dependent on the standard deviation and mean of the temperature distribution enables us to detect points with uncommon temperature values, regardless how expressed the bell in the temperature distributions is. Although, it must be noted that for sensitive thresholding one has to consider the possible asymmetry of the distribution, e.g., heaters might be turned off. Furthermore, in the warm season when air conditioning is used windows will contribute to the upper end of the temperature scale. The thresholding constant is given as follows:

$$thres = \frac{1}{N} \sum_{j=1}^N t_j \pm \sqrt{\frac{1}{N} \sum_{j=1}^N (t_j - \mu)^2}$$

where  $t_j$  represent the temperature values of the room with walls, windows and other nearby objects.

---

#### Algorithm 2: Clustering Algorithm

---

**Data:** Potential 3D window points  $P$   
**Result:** Set of clusters  $S$   
initialization;  
 $P, S, \varepsilon$ ;  
**for**  $\forall p_i \in P$  **do**  
    **for**  $\forall S_a \in S$  **do**  
        **for**  $\forall p_j \in S_a$  **do**  
            dist =  $\|\vec{p}_i - \vec{p}_j\|$ ;  
            **if** dist  $\leq \varepsilon$  **then**  
                └ add  $p_i$  to cluster  $S_a$ ;  
                └ break out of loop;  
        **if** above loop broke out **then**  
            └ break out of loop;  
    **if** above loops did not break out **then**  
        create new cluster  $S_n$ ;  
        add  $p_i$  to  $S_n$ ;  
        add  $S_n$  to the cluster set  $S$ ;  
    remove  $p_i$  from  $P$ ;

---

Using this threshold we detect points with lower tempera-

ture values and conclude that these points represent windows. In the next step the potential window points are clustered according to their spatial distance from each other. The clustering is done based on a simplified version of  $k$ -means clustering, i.e., the clusters emphasize compactness and connectedness. See Algorithm 2 for summary. A final filtering procedure removes clusters with a small number of points.

Each of the remaining clusters is processed individually in case there is more than one window in the room. We approximately determine the width and height of each window from the cluster. This has the major advantage that we can control the number of points on the window. The boundaries of the window are approximated by first determining the plane the window lies on by removing the axis with the smallest variance. Second, we select the extreme  $\pm$  of each axis, i.e., the distance from the respective component of the mean vector of the clusters. Finally, the extreme points are used to determine the boundary points. To achieve an identical number of points on each detected window we use an octree based sub-sampling (see [7]). The procedure takes into account the size of the window by creating an octree with a variable minimum voxel size and taking only one point from each voxel.

#### IV. WINDOW LABELING

A Markov Random Field (MRF) is a modeling technique for a graph of random variables. A MRF was first introduced by Besag in 1974 as a parametric model for spatial data analysis [2]. It is mainly used to express the statistical dependencies between several random variables arranged in some form of spatial configuration or *graph*. Alternatively, Gibbs (Boltzmann) distribution is a probability distribution for spatially arranged random variables where the joint probability distribution can be factorized as:  $P(z_1, \dots, z_n) = \prod_{j=1}^n \phi_j(\psi_j)$ , where  $\phi_j$  is any real valued function defined for each clique set. A clique is a set of nodes where every node is directly connected with each other. The equivalence between MRF and Boltzmann distribution is shown in [2].

We will use the labeling of windows into either **Closed**, **Opened** or **Damaged**. The Boltzmann distribution is used to model the likelihood of each label, and in effect the *posterior*. Let  $\{x_1, \dots, x_n\}$  be the random variables, or 3D points, spatially arranged in a window shape, and take a temperature value  $t_s = \{t_1, \dots, t_n\}$ , where  $t_s \in \mathbb{R}^n$ . The Boltzmann distribution is then defined as follows:

$$p(t_s) = \frac{1}{Z} \exp\left(\frac{-E(t_s)}{T}\right) \quad (4)$$

where  $Z$  is the normalization constant,  $T$  is a controlling constant, and  $E(t_s)$  is a label specific energy function.

##### A. Energy functions

Temperature plays the main role in the labeling of windows. Apparently, the temperature distribution of a *closed* window exhibits a very small variance, regardless of the peak temperature. On the contrary, the temperature distribution of an *opened* window has a much higher variance, comparatively. This has shown to be a very robust feature, i.e., invariant to peak temperature value. *Damaged* windows are particularly detectable because of the NOT smooth temperature distribution. Intuitively, the roughness in temperature value is the

main feature of a leaky window. Therefore, roughness in the temperature distribution is the emphasized feature in the energy function of a damaged window. However, it must be noted that the temperature distribution of both *opened* and *closed* windows are smooth, i.e., small temperature difference between neighboring points.

Based on the noted above features the energy function of each label is given as follows:

$$E_C(t_s) = \alpha_1 \sum_{j=1}^N \frac{(t_j - \mu)^2}{N} + \alpha_2 \sum_{j=1}^N \frac{d(x_j, K)}{N} \quad (5a)$$

$$E_O(t_s) = \alpha_1 \frac{N}{\sum_{j=1}^N (t_j - \mu)^2} + \alpha_3 \sum_{j=1}^N \frac{d(x_j, K)}{N} \quad (5b)$$

$$E_D(t_s) = \alpha_1 \frac{N}{\sum_{j=1}^N d(x_j, K)}, \quad (5c)$$

where  $\alpha_1, \alpha_2, \alpha_3$  are weighting constants of each term,  $N$  is the number of random variables (3D points) representing the window.  $d$  calculates the average temperature difference of the  $K$  closest points in the spatial neighborhood of  $x_j$ .

The first term encourages a small variance in case of the closed window energy function (Eq. (5a)), and a high variance in case of an open window (Eq. (5b)). The second second term in both cases, open and closed window, emphasizes smoothness. However, the damaged window energy function (Eq. (5c)) encourages a rough temperature distribution.

##### B. Monte Carlo integration

So far, we have defined the energy function of each class but the normalization constant still needs to be calculated. The normalization constant for  $n$  continuous random variables defined on the same probability space is calculated as:

$$Z = \int_{\mathbb{R}^n} \exp\left(\frac{-E(t)}{T}\right) dt \quad (6)$$

The above high dimensional integration is very difficult for a straight forward computation and has to be approximated. Consequently, we have to resort to the numerical method of *Monte Carlo integration*, which is typically used for higher dimensional integrals [14]. The first step of Monte Carlo is to express Eq. (6) as follows:

$$E\left[\frac{\exp\left(\frac{-E(t)}{T}\right)}{p(t)}\right] = Z = \int_{\mathbb{R}^n} \frac{\exp\left(\frac{-E(t)}{T}\right)}{p(t)} p(t) dt, \quad (7)$$

where  $p(t)$  can be any probability distribution;  $t \in \mathbb{R}^n$ , where  $n$  is the number of random variables or 3D points of the window.  $E[\cdot]$  is used to represent the expectation value. The law of large numbers assures the approximation of the expected value with a large number of samples and thus Eq. (7) is reduced to

$$E\left[\frac{\exp\left(\frac{-E(t)}{T}\right)}{p(t)}\right] = \frac{1}{M} \sum_{j=1}^M \frac{\exp\left(\frac{-E(t_j)}{T}\right)}{p(t_j)},$$

where  $t_j \in \mathbb{R}^n$  is sampled according to  $p(t)$  for  $M$  times. The larger  $M$  is the better the approximation will be.

The major practical difficulty in using the Monte Carlo integration is the design of  $p(t)$  especially for a function

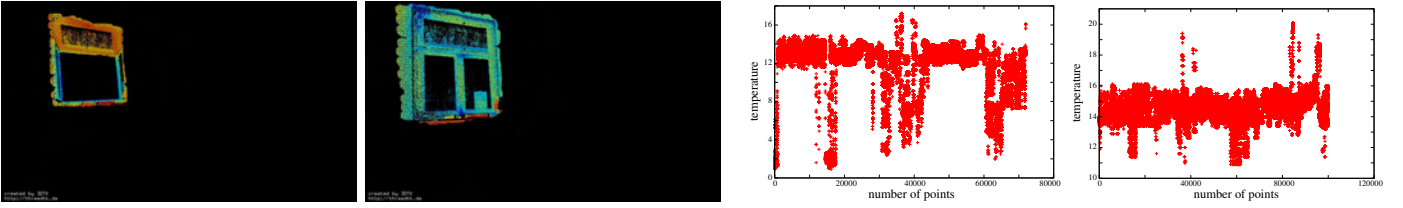


Fig. 5: Left: Open and closed windows colored according to the thermal distribution. Right: the temperature distribution of each window, respectively.

defined in a large space. For example, if we assume  $p(t)$  to be uniformly distributed the fraction  $\frac{\exp(\frac{-E(t_j)}{T})}{p(t_j)}$  becomes almost infinity; since  $p(t)$  will be extremely small due to the high dimensionality of the space. Theoretically, this can be solved if we take an infinite amount of sample from  $p(t)$ . The issue can only be dealt with a hand designed probability distribution  $p(t)$  that tracks the energy function very well, which means, if  $\exp(\frac{-E(t_j)}{T})$  is small then  $p(t_j)$  is small and vice-versa. So, the fraction  $\frac{\exp(\frac{-E(t_j)}{T})}{p(t_j)}$  is a more pragmatic number that contributes to the estimation significantly. Moreover, we estimate a more accurate expectation and in effect a better estimate with a small number of samples, comparatively, if a well-behaved probability distribution is designed, cf. Fig. 6.

### C. Designing a probability distribution

Since the energy function of each label is different, the design of the probability distribution is also label specific. Our goal is to design a probability distribution that tracks the energy function well. This means  $K \cdot p(t) \propto \exp(\frac{-E(t)}{T})$ . Therefore, we have set two conditions to be satisfied by a probability distribution to be considered as a well designed probability distribution: First, good tracking capability and second,  $p(x)$  should NOT be negligibly small regardless of its tracking capability, which is the case for most probability distributions defined in a vast space. An attempt is made to meet the first condition by designing a Markov chain as follows:

$$\begin{aligned} X_{i+1} &= X_i + W \cdot \mathcal{N}(0, \sigma^2) \\ X_0 &= \mathcal{N}(k, \sigma^2), \end{aligned} \quad (8)$$

where  $i$  counts the steps starting from 0 to the number of random variables, and  $X$  is a random variable taking a temperature value from  $\mathbb{R}$ .  $X_0$  is the initial state of the process.  $W$  is a weight that can be used to control the rate of variation and  $k$  is a constant that can be tuned accordingly.  $\sigma$  is the standard deviation of the Gaussian distribution.

The probability of a temperature distribution instant,  $t = \{x_n, \dots, x_0\}$ , that is proposed using Eq. (8) iteratively until  $i$  reaches a given number  $n$  is given as follows. Note that the Markov property is being used to simplify the computation of the joint distribution:

$$\begin{aligned} p(x_n, \dots, x_0) &= p(x_n, \dots, x_1 | x_0) p(x_0) \\ &= p(x_n, \dots, x_2 | x_1, x_0) p(x_1 | x_0) p(x_0) \\ &\vdots \\ &= p(x_n | x_{n-1}) \cdots p(x_2 | x_1) p(x_1 | x_0) p(x_0) \end{aligned} \quad (9)$$

And from Eq. (8):

$$p(x_i | x_{i-1}) = \frac{1}{\sigma \sqrt{2\pi}} e^{-r^2 / 2\sigma^2} \quad (10)$$

where  $r$  is a number sampled from  $\mathcal{N}(0, \sigma^2)$  at the  $i^{th}$  step. Thus, any sample  $t = \{x_n, \dots, x_0\}$  generated using Eq. (8) can be made to track the energy function of each label by tuning the free parameters. The second condition for the probability distribution is NOT to be negligibly small. We have approached this problem by clustering points, with their spatial location, and treat each cluster as a random variable, i.e., every point in the cluster will have the same temperature value, and thus reducing the domain space to the number of clusters from the number of points, which means the joint distribution is very much higher than the same distribution in the original space.

The free parameters of Eq. (8) are assigned as  $W = 1$  and  $\sigma^2 = 0.3$  for the label *closed window*. Consequently, Eq. (10) and (9) are high valued when there is less variance between consecutive points, i.e.,  $x_{i-1}$  and  $x_{i+1}$ , and small when there is high variance. This by itself encourages smoothness. On the contrary, the variance of the temperature distribution for *open window* is much higher than the variance of *closed window*, but the smoothness should be exactly the same. Hence, setting  $W = 20.0$ , aims to cause a higher variance between clusters, NOT between points, and will cause Eq. (10) and (9) to be high valued for states with high variance yet a smooth temperature distribution. Note that increasing  $W$  achieves proposal states with higher variance without causing the probability distribution to shrink which would happen if we simply increase the variance. Thus, the variance is left as  $\sigma^2 = 0.3$ .

As noted in the previous sections, the most distinctive feature of a damaged window is the roughness of the temperature distribution, unlike for *closed* or *opened window*. The design of the probability distribution for the energy function for the label *damaged window* is based on Eq. (8) with minor but basic modifications on the handling of points inside a cluster. For open and closed windows each point in a cluster is assigned exactly the same temperature. But, in case of damaged windows the assignment is done as follows:

$$p_j = X_{i-1} + W_2 \cdot \mathcal{U}(0, 1) \cdot r, \quad (11)$$

where  $p_j$  is a point in a cluster  $X_i$ ,  $X_{i-1}$  is the value of the previous cluster,  $r$  is a sample generated from  $\mathcal{N}(0, \sigma^2)$  at the  $i^{th}$  step, and  $\mathcal{U}(0, 1)$  is a uniform distribution that is sampled iteratively  $\forall p_j \in X_i$ . Finally,  $W_2$  is a weighting constant that is used to amplify roughness. As can be seen, each point in a cluster will have different value, unlike the points in a closed

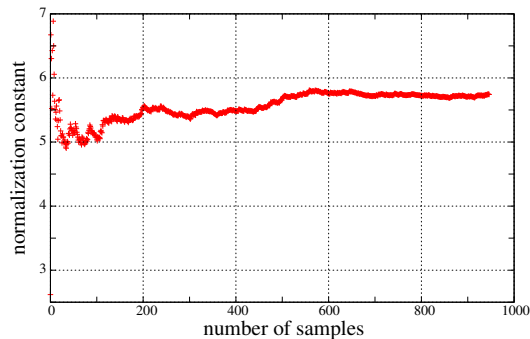


Fig. 6: The convergence of the mean sequence; the approximation of the normalization constant of a closed window with 1000 samples.

or open window. But, most importantly, the probability for a point is exactly the same as for a cluster, since every value is sampled from  $\mathcal{U}(0, 1)$  with a probability equal to 1. And this property, enables the proposed distribution to propose a very rough temperature distribution, which is expected, without flattening the probability distribution.

As a result from the octree based sub-sampling, the 3D points on a typical window are assumed to be constant. Thus, the normalization constant is approximated offline, which otherwise would have to be computed for each labeling task. For applications where speed is not an issue the normalization constant can be calculated online. However, we have not seen any significant difference on the final performance except the apparent overhead in the later case.

## V. EXPERIMENTAL RESULTS

The test data set is acquired with a high precise laser scanner, the Riegl VZ-400, and an Optris PI160 thermal camera. The pre-processing, e.g., registration, visualization and mapping of the thermal image is done with *3DTK – The 3D Toolkit* (<http://threedtk.de/>). The window detection is tested on acquired data sets and has been proven to be adequate, see Fig. 7. Open, semi-open or closed windows are correctly detected. However, as discussed in the previous sections there are free parameters, apart from the normalization constant, that need to be estimated for the final labeling of windows.

The normalization constant for each label is approximated with 1000 samples randomly taken according to the designed probability distribution, see subsection IV-B. Since the approximation of the mean gets closer to the true mean as  $N \rightarrow \infty$ , where  $N$  is the number of samples, the error is estimated with the variance of the following sequence  $M_h$ , that gets smaller

TABLE I: Summary of the approximated normalization value and error range.

Label	Number of Sample	Normalization constant	Error range
<i>Closed</i>	1000	5.79517	0.0333378
<i>Open</i>	1000	5.26492	0.0231288
<i>Damaged</i>	1000	4.64347	0.0379469

TABLE II: The value of free parameters for each labels energy function.

Label	$\alpha_1$	$\alpha_2$	$\alpha_3$	$T$
<i>Closed</i>	0.05	1	N/A	1
<i>Open</i>	2	1	N/A	1
<i>Damaged</i>	N/A	N/A	0.1	1

and smaller as  $N \rightarrow \infty$ , cf. Fig. 6:

$$M_h = \sum_{i=1}^h \frac{f_i}{h},$$

where  $h$  goes from 1 to  $N$  and  $f_i = \frac{\exp(-E(t_i)/T)}{p(t_i)}$ . As shown in Table I the error range of the normalization constants  $Z$  for each label is very low. Despite the hand tuning of the free parameters the algorithm performed as expected. The experimentally determined parameters  $\alpha_1$ ,  $\alpha_2$ ,  $\alpha_3$ , and  $T$  are given in Table II. Exemplary results achieved with these parameters are detailed in Table III. All examples are correctly labeled by the algorithm. The probabilities for each label and also the probability to choose a wrong label are given. Closed windows are reliably detected. The probability to mislabel an open window is much higher. This is due to the fact that in these cases the roughness increases the probability that the window is damaged. This suggests that the roughness function is not optimal for window labeling. Nevertheless, also in these cases the correct label was chosen.

## VI. CONCLUSIONS

In this paper, we presented a thermal information analysis for object detection and labeling, which is shown to be a robust feature. In effect, we presented a reasonable approach for understanding the structure of a room, and we have, sequentially, shown usage of temperature as a main feature for object detection and, furthermore, modeling of temperature distribution to infer object related semantics. Although, the main aim of this work is to contribute to the efforts of automating energy leak detection and prevention, the approaches can be adopted for object detection and modeling task in general, especially in cases where there is a small data set to learn from. In future work we plan to further assess and try to improve the methods presented here. First, this includes a thorough evaluation of the performance under changing temperature distributions. Especially transferring the approach to examine rooms with air condition rather than heating systems is a goal. Second, different energy functions should be evaluated in order to improve the reliability of the labeling. Third, we would like to extend the processing pipeline to label other heat sources and to detect poor insulation in buildings.

## VII. ACKNOWLEDGMENTS

This work was partially supported by the SEE-ERA.NET project ThermalMapper under the project number ERA 14/01.

## REFERENCES






- [1] A. Adan and D. Huber. 3D reconstruction of interior wall surfaces under occlusion and clutter. In *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, pages 275–281. IEEE, 2011.





Fig. 7: Top: Three original thermal 3D point clouds with temperature values. Bottom: The detected windows under different circumstances, i.e., closed, semi-open, and fully open are shown in green.

TABLE III: A summary of experimental windows labeling and probability of making an error

segmented window	Probability of				Final Label
	closed	opened	damaged	making error	
	0.6427	0.3032	0.0540	0.3573	<b>Closed</b>
	0.7315	0.1544	0.0248	0.1568	<b>Closed</b>
	0.3080	0.4232	0.1569	0.5768	<b>Opened</b>
	0.2712	0.4773	0.2514	0.5227	<b>Opened</b>
	0.3474	0.4669	0.1857	0.5331	<b>Opened</b>

[2] J. E. Besag and P. AP. Moran. On the estimation and testing of spatial interaction in Gaussian lattice processes. *Biometrika*, 62(3):555–562, 1975.

[3] P. Besl and N. McKay. A method for registration of 3–D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239 – 256, February 1992.

[4] C. M. Bishop. *Pattern recognition and machine learning*. Springer, New York, 2006.

[5] D. Borrmann, H. Afzal, J. Elseberg, and A. Nüchter. Mutual calibration for 3D thermal mapping. In *Proceedings of the 10th International IFAC*

*Symposium on Robot Control (SYROCO '12)*, volume 10, Dubrovnik, Croatia, September 2012.

[6] D. Borrmann, J. Elseberg, K. Lingemann, A. Nüchter, and J. Hertzberg. Globally consistent 3D mapping with scan matching. *Journal Robotics and Autonomous Systems (JRAS)*, 56(2):130–142, February 2008.

[7] J. Elseberg, D. Borrmann, and A. Nüchter. One billion points in the cloud—An octree for efficient processing of 3D laser scans. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2012.

[8] Y. Ham and M. Golparvar-Fard. An automated vision-based method for rapid 3D energy performance modeling of existing buildings using thermal and digital imagery. *Advanced Engineering Informatics*, 2013.

[9] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke. Real-time plane segmentation using RGB-D cameras. In *Proceedings of the 15th RoboCup International Symposium*, Istanbul, Turkey, July 2011.

[10] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. Software available at <http://octomap.github.com>.

[11] A Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 14:841, 2002.

[12] D. Pangercic, M. Tenorth, B. Pitzer, and M. Beetz. Semantic object maps for robotic housework - Representation, acquisition and use. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '12)*, Vilamoura, Portugal, October 2012.

[13] C. Pedersen and K. Hellevang. Insulation and air infiltration levels. Tech. Report, 2008.

[14] C. P. Robert and G. Casella. *Monte Carlo statistical methods*, volume 319. Citeseer, 2004.

[15] R. Schnabel, R. Wessel, R. Wahl, and R. Klein. Shape recognition in 3D point-clouds. In *The 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2008*, February 2008.

[16] H. Surmann, A. Nüchter, and J. Hertzberg. An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments. *Journal Robotics and Autonomous Systems (JRAS)*, 45(3–4):181–198, December 2003.

[17] S. Vidas, P. Moghadam, and M. Bosse. 3D thermal mapping of building interiors using an RGB-D and thermal camera. In *Proceedings of the IEEE Conference on Robotics and Automation (ICRA '13)*, Karlsruhe, Germany, May 2013.

[18] X. Xiong and D. Huber. Using context to create semantic 3D models of indoor environments. In *British Machine Vision Conference, Aberystwyth*, pages 45–1, 2010.