INTELLIGENT HAZARD IDENTIFICATION: DYNAMIC VISIBILITY MEASUREMENT OF CONSTRUCTION EQUIPMENT OPERATORS

A Thesis
Presented to
The Academic Faculty

by

Soumitry J. Ray

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in
Computational Science and Engineering

Civil and Environmental Engineering
Georgia Institute of Technology

INTELLIGENT HAZARD IDENTIFICATION: DYNAMIC VISIBILITY MEASUREMENT OF CONSTRUCTION EQUIPMENT OPERATORS

Approved by:

Professor Reginald DesRoches,
Advisor
Civil and Environmental Engineering
Georgia Institute of Technology

Professor Yong K. Cho
Civil and Environmental Engineering
Georgia Institute of Technology

Professor Polo Chau
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Rajesh Narasimha
Senior Researcher
Metaio Inc

Professor Patricio A. Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: 27 March 2014

ACKNOWLEDGEMENTS

Foremost, I would to express my gratitude to my advisor Dr. Reginald DesRoches for the valuable guidance and support. Besides, I would like to thank my committee members: Dr. Patricio A. Vela, Dr. Polo Chau, Dr. Rajesh Narasimha and Dr. Yong K. Cho for their guidance and insightful comments which helped me thoroughly.

I would also like to take this opportunity to thank my fellow labmates: Tao Cheng, Sijie Zhang, NiPesh Pradhananga, Yihai Fang, Eric Marks, Kyungki Kim and my close friend Atish Patra for being supportive all throughout my last 5 years and helping me in various stages of my PhD study.

Finally, I would like to thank Dr. Dayal R. Parhi who encouraged me to pursue my higher studies during my undergraduate study at NIT Rourkela and had been a source of motivation for me to pursue my PhD.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Struck-by fatalities involving heavy equipment such as trucks and cranes accounted for 24.6% of the fatalities between 1997-2007 in the construction industry. Incidents related to construction equipment can result in severe injuries too. Limited visibility due to blind spots and travel in reverse direction are some of the main causes of these fatalities. Blindspots are spaces surrounding an equipment that are not in the field-of-view of the equipment operator. Thus, a hazard is posed to the ground personnel working in the blind spaces of an operating equipment.

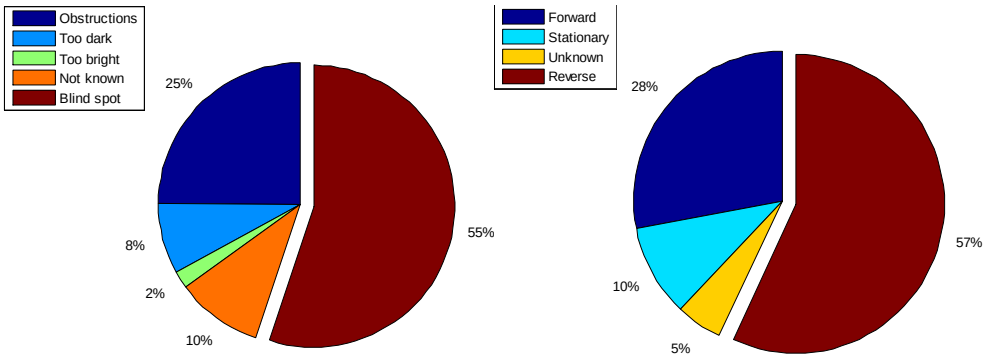
This research presents a novel approach to construct visibility maps of the equipment operator which can aid in identifying potential hazards posed to workers operating in the vicinity of an equipment in operation. The approach has two components a) *Head pose of equipment operator,* and b) *Static blindspots map.* The underlying concept is to compute the visibility of an operator at any instant of time. To measure the operator's visibility, we mount a commercial depth camera in the equipment cabin and apply machine learning technique (Random Forests) on camera frames to estimate the *head pose* of the operator. The head pose of operator yields the direction of field-of-view (FOV). To measure blindspots around an equipment, a novel computationally efficient approach using Ray Casting algorithm is run on 3D point cloud data of the equipment. This results in a *static blindspots map.* It is termed static because the map represents the visibility of an operator from a fixed location, that is the origin of the point cloud data. To turn the static map in to a dynamic one, the origin needs to be translated according the estimated location of operator's head. The location of operator's head is estimated along with the head orientation. This integration of *static blindspots map* with operator's head location is termed as *dynamic blindspots map.* Measurement of dynamic blindspots is necessary because the visibility of an operator changes with the location of operator's head.

The presented blindspots approach provides a novel way to objectively measure and visualize different facets of blindspots. An understanding of the blindspots caused due to vehicle hardware and the FOV of proximity-detection and warning technologies can provide vehicle manufacturers with more information to improve the design of the equipment or to retrofit existing equipment with proximity-detection and warning to enhance safety. Additionally, this research has the potential not only to improve safe operation of equipment on construction sites but also to record incidents.

CHAPTER I

INTRODUCTION

Equipment blindspots are those regions that are invisible to the equipment operator. Blindspots pose a significant hazard to personnel operating around construction equipment which can result in fatalities. The Occupation Safety and Health Administration (OSHA) attributes fatalities to five categories: falls, struck-by, caught-in/between, exposure to harmful substances, and others. Struck-by incidents are primarily caused due to (a) vehicles, (b) falling/flying objects, and (c) construction of masonry walls. Among vehicles, 75% of the struck-by fatalities involve heavy equipment which is primarily due to visibility-related issues [34]. Figure 1 illustrates the causes of fatalities related to travel direction. 55% of the visibility-related fatalities were caused due to equipment blindspots and 57% of the same fatalities were caused due to travel in reverse direction [34].



(a) Causes of fatalities. **(b)** Travel direction.

Figure 1: Safety analysis using historical OSHA data (1997-2007): (a) cause of fatalities and (b) travel direction [34].

OSHA considers struck-by object to be the the second leading cause of fatalities

on construction sites for the year 2011. For the year 2013, OSHA attributes 73(10%) of the 738 total deaths to struck-by incidents. Thus, a need exists for potential hazard identification posed due to the interaction of equipment and personnels working around equipment.

To address the limited visibility due to blindspots, technologies such as radar, camera system, ultra sonic sensor system and GPS have been assessed [58, 57]. Very-High Frequency (VHF) and active Radio Frequency (RF) has been used for proximity detection and warning for equipment-worker interactions [64, 43]. To perform resource tracking for productivity and safety on construction sites, performance of Ultra Wideband technology has been assessed [11]. These technologies primarily focus on detecting hazardous conditions that may arise as a result of equipment-worker interactions. Mounting of such technologies on equipment can be optimized through the knowledge of the blindspots of equipment. As such real-time pro-active technology has the potential to save lives by pro-actively monitoring the surroundings of a piece of equipment. However, the inherent limitation of such systems is that it only takes into account the proximity of the workers to the equipment and does not incorporate any knowledge of the operator's FOV. Hence, alerts need to be optimized to address issues relating to alert desensitization [43]. To address this issue, knowledge of the operator's FOV may be incorporated in to a warning or alert system to raise alerts intelligently.

1.1 System Representation

Identifying the requirements stated above, in this research a system (shown in Figure 2) is proposed. The system incorporates the information on dynamic blindspots of equipment and location of workers around an operating equipment. Knowledge of operator's FOV aids in assessing hazards posed to the workers. Figure 3 illustrates the concept of the system from top and side views. Note in Figure 3a how the worker

on the left is located in a region that is in the operator's FOV, however the worker at bottom right is located in a visible region but is outside the operator's FOV, thus exposing the worker to potential hazard due to the equipment. Similarly a hazard is posed to the worker on the right of Figure 3a.

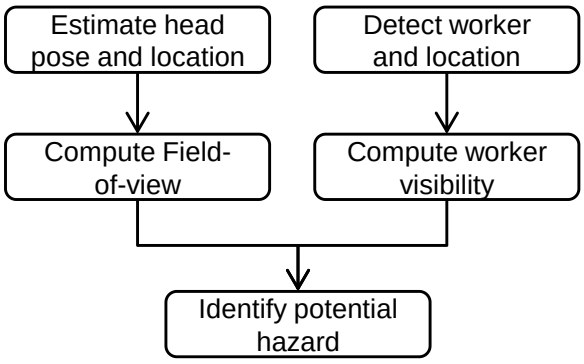
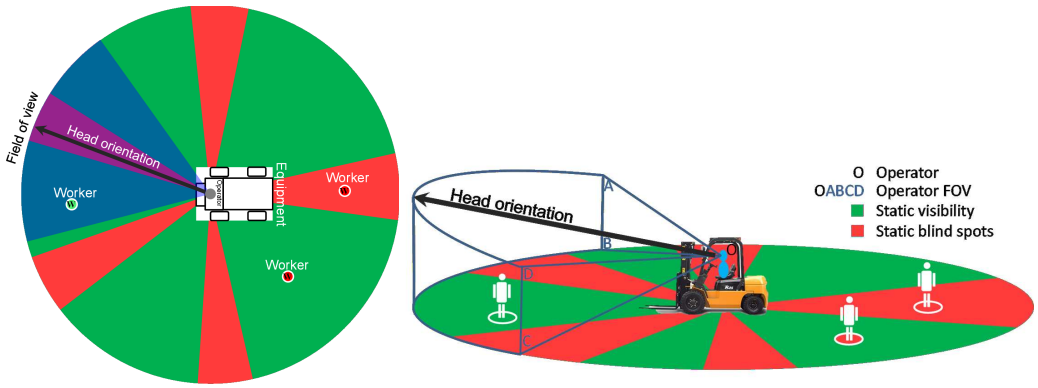


Figure 2: System architecture.



(a) Pictorial representation (top view).

(b) Pictorial representation (side view).

Figure 3: System representation.

1.2 Research Framework

The research framework for the system illustrated in figures 2 and 3 is discussed in this section. Figure 4 illustrates the research framework of this thesis. The framework has been divided into four modules and each module leads to its successor in the following

order: a) Sensor or Data sources, b) Data, c) Processing and d) Applications. The arrows indicate the flow of data or information. The green arrows indicate those flows that have been used in this research and the solid red arrows indicate data or information that is *necessary* but have not been addressed in this research. Thus, the solid red arrows indicate the pipeline for future research study. The dashed red arrows indicates *optional* studies for future research study.

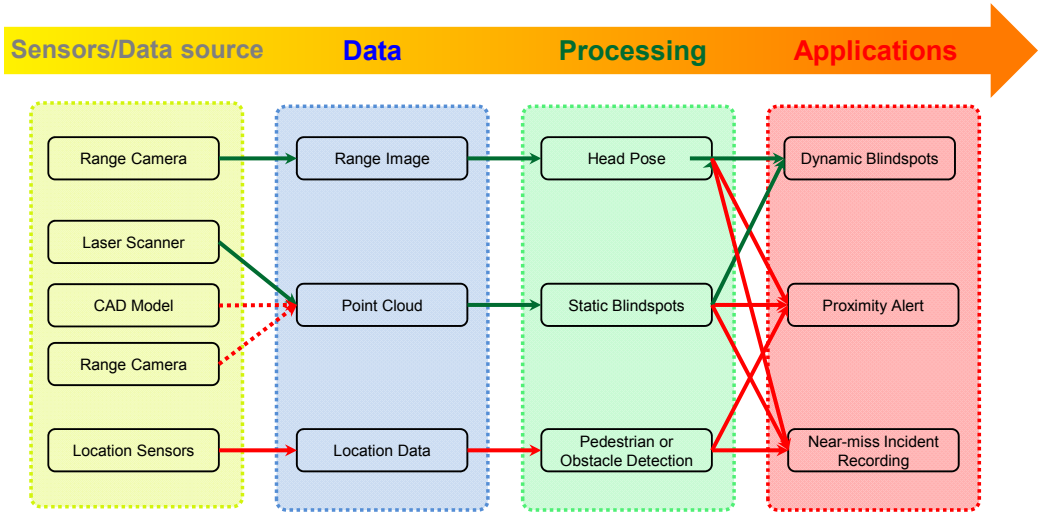


Figure 4: Research framework.

The first module deals with the sensors or data sources which are: a) Range camera b) Laser scanner/CAD Model/Range camera and c) Location sensors. A range camera is mounted inside the equipment cabin generates range images which is used to monitor the head posture of the equipment operator. The second data source is used to generate a 3D point cloud of the equipment. Such a point cloud can be generated by a) laser scanning the equipment or b) from the CAD model of the equipment or c) by scanning the equipment with a hand held scanner such as a range camera. This point cloud information is then utilized to compute the static blindspots map of the equipment. The third data source is a location sensor system that monitors the surroundings of the equipment for potential obstacles or workers. In

this research, the problem of obstacle or worker detection has not been addressed. The final module deals with the creation of knowledge using the information generated in the Processing module. In this thesis only the knowledge of visibility maps is created whereas the creation of proximity alert system and recording of near-miss incidents remains in the scope of future research. The thesis statement below summarizes the presented research framework.

Thesis Statement: *The visibility of an equipment operator can be measured by computing the head posture and the static blindspots of the equipment.*

1.2.1 Research Impact

The presented research can potentially impact multiple facets of safety issues caused due to Human-Equipment interaction. The blindspots analysis technique generates valuable information which can be used to improve the design of construction equipment to increase safety. Also existing equipment can be retrofitted with proximity sensors once the sensor's FOV information is fused with the blindspots information. By integrating an obstacle or pedestrian detection module with the presented research, a proximity alert system can be created which can potentially mitigate fatalities on construction sites while reducing nuisance alerts. Such nuisance alerts are responsible for desensitizing operators towards the alerts over a period of time. Finally, this research can potentially be used as a near-miss recording system, which can record non-fatal but hazardous incidents occurring on construction sites. Such information can then be utilized to train construction workers and plan construction tasks in a way so as to minimize the occurrence potential hazards on construction sites due to Human-Equipment interaction.

1.3 Thesis Outline

In Chapter 2, the technique for head posture estimation of an equipment operator is presented. It begins with a literature review of existing methods for estimating head

orientation using vision based technique and then discusses the methodology used in this research and then the results. In Chapter 3, the focus shifts to computing blindspots of equipment. As shown in Figure 2 our primary focus is to measure the operator's visibility, thus in Chapter 3 we discuss several aspects of blindspots measurement which are key to understanding the visibility of an equipment operator. In Chapter 4 results to computing dynamic blindspots in indoor and outdoor environments are presented and then followed by the Chapter 5 which discusses several key issues and assumptions that need to be addressed for a real-world implementation of the presented approach.

CHAPTER II

ESTIMATING THE HEAD POSTURE OF AN EQUIPMENT OPERATOR

2.1 Background

Pose estimation finds numerous applications in areas such as Human-Computer Interaction (HCI), gaze estimation, analysis of facial expression, video conferencing, driver fatigue [40, 31, 71]. Measuring the head orientation is a pose estimation problem that has been widely studied using intensity cameras. Multiple studies have focussed on studying driver attention [40, 31, 71]. However, the inherent dependence of intensity cameras on illumination makes them less effective in situations where there is insufficient or fluctuating illumination, as it is likely the case on construction sites or in outdoor environment. Furthermore, the issue of depth ambiguity is associated with intensity cameras too.

Most of the existing head pose estimation techniques either make use of intensity images or spatial (3D) data. A recent study [49] classified these techniques into eight categories. Based upon the approach used to solve the head pose estimation problem, the methods have been broadly classified into: (a) appearance based methods, (b) detector array methods, (c) non-linear regression models, (d) manifold embedding methods, (e) flexible models, (f) geometric methods, (g) tracking methods, and (h) hybrid methods. Some of their strengths and gaps of relevant techniques are presented in abbreviated form.

Appearance based methods generally consider the space containing the face. They solve the pose estimation as a classification problem [35] by training a set of faces on a Support Vector Machine (SVM) model that correspond to discrete views of poses such

as frontal view, left view, and right view. Wavelet transformation was applied and the low resolution sub-band was projected pose on to the eigenspace using Principal Component Analysis (PCA) [47]. Other approaches treat it as a regression problem and thus train a ridge regression model [59] or a support regression model [1] to estimate the head orientation. Such methods generally assume that the face has been extracted from the image frame. They then utilize a dimensionality reduction technique like Singular Value Decomposition (SVD) or Gabor wavelets [38] to extract the feature vectors using a Support Vector Machines (SVM) classifier. These feature vectors are finally trained using a supervised learning algorithm to predict the angles. Such methods exhibit poor performance when the identity of the subject changes.

Detector array methods are based upon a set of trained detectors that correspond to discrete poses and a new image is classified by the detector that gives the highest support. The detector array is based up on neural network [37] which works by first de-rotating the head into frontal view and then detects the head. Eigenspaces have also been used as a cascade of detectors where each eigenspace was related to one discrete pose [62]. These give good results as prediction is independent of the identity of the subject. However, such methods predict only few discrete poses (typically ten that vary across a single degree of freedom). Furthermore, building a cascade to predict a higher number of poses would create complexity in training these detectors.

In non-linear regression models, the image is mapped from the image space to the pose space. Generally, fitting a regression model on the high dimensional image data proves to be a challenge; therefore a face is sparsely represented by facial feature points such as eye inner corner, eye outer corner, eye center, mouth corners and their interpolated positions. A Relevance Vector Machine (RVM) was trained to learn the relationship between the sparse representation and the pose estimation angles [42]. A sparse representation is achieved by reducing the dimensionality using techniques such as PCA and Support Vector Regression (SVR) models [1].

Manifold based techniques project images corresponding to the different poses to a lower dimensional space [48]. The dimensional reduction is achieved by Kernel Principal Component Analysis (KPCA) [10], Kernel Discriminant Analysis [9], Locality Preserving Projections [54], Locally Linear Embedding [32], or Isometric Feature Mapping [56]. To estimate the pose in a new image the head frame is mapped on to a lower dimensional space. Using a distance metric the pose angle is estimated. A variation [3] in this technique uses Generalized Regression Neural Network (GRNN) for dimensionality reduction and a linear regression model functional relationship to map images from the reduced dimensional space to the pose angle space.

Other techniques include fitting an Active Appearance Model (AAM) [14] which takes into consideration the geometric features like bilateral symmetry and position of facial features such as eyes, nose tip, mouth, ears etc. These are utilized to calculate the orientation angles and track the feature points of the head using particle filters.

3D range data are independent of illumination [69] and hence can be a more favorable choice for estimating the head orientation in outdoor applications over other alternatives, such as intensity cameras. Range cameras can work in low-level illumination conditions and are color and texture invariant. Depth data simplifies the task of background subtraction which is not the case with traditional intensity cameras [61]. They have demonstrated the reliability of depth cameras to estimate human body posture for Xbox gaming device. Using range images significant improvement in human head posture estimation accuracy using random forest on kinect depth frames has also been demonstrated [22, 23].

2.2 Random Forests

Random Forests [7] is combination of decision trees that can perform classification, regression or both. It recursively splits the input feature space into axis aligned smaller regions (cuboids). The splits are performed by simple binary tests which

decide whether the input data goes to the left or the right child node. These splits are termed as *decision stumps.* By repetitively splitting the data into proper subsets, the nodes of a tree are trained. Once the forest is trained, prediction is performed by averaging (regression) or voting (classification), estimates of the leaf nodes of the trees. From performance perspective, predictions of random forest improves as the data set size grows [61]. Other advantages include ease of implementation and parallelization.

2.2.0.1 Notations

- The set of trees in the forest is represented by $\mathrm{T} = \{T_t\}$ and $|T|$ is the number of trees in the forest.
- 'c' represents classification of a patch and 'r' represents regression on orientation and location of head.
- n is any node in a tree T_t. To distinguish the leaf nodes from the non-leaf ones, the leaf nodes are represented by $l \in L$, where L is the set of all leaf nodes of a tree.
- The class label $k \in K$, where $K = \{0, 1\}$; 0 represents background and 1 represents foreground.
- The set of all patches in the training data set of a tree is represented by Π_r and a patch is represented by π.
- Traversing from the root node n_r to any leaf node l, say nodes n_1, n_2,..., n_k are encountered, then we have $|\Pi_r| < |\Pi_{n_1}| < |\Pi_{n_2}| < ... < |\Pi_{n_k}| < |\Pi_l|$.
- Offset vector $\mathbf{d} = \{d_x, d_y, d_z\}$, represents the vector joining the center of a patch π to the reference point on the head (see Figure 6). Similarly, $\boldsymbol{\theta} = \{\alpha_p, \alpha_y, \alpha_r\}$

represents the rotation vector consisting of yaw, roll and pitch angles. For negative patches, the offset vector $\mathbf{d}$ and the rotation vector $\boldsymbol{\theta}$ are set to $\mathbf{0}$.

Figure 5 illustrates Random Forests comprising of a set of trees. The root of a tree is represented by solid black circle and the leaves are represented by solid green circles. Following a bagging approach, training samples are chosen at random to train each tree separately. At each node, the training data, comprising of positive and negative patches, is split between the child nodes. The optimal split parameters are determined by maximizing a gain function. By maximizing the gain function, the *entropy impurity* due to *classification* and *regression*, at a node is reduced. Once an optimal split is found, the patches are then split between child nodes and the process continues. The split data forms two proper subsets of the data set input to their parent node. Splitting is stopped at a node n if:

a) the node receives less than a predefined number of patches or,

b) if negative patch probability $p(k = 0|\Pi_n)$ is greater than a certain threshold,

c) if the node has the maximum allowed depth.

After all the training data has been pushed down the tree to the leaves the training of the tree is complete. Using the set of patches Π_l at the leaves, the following statistics are then computed:

a) $p(k|\Pi_l)$: class probabilities for foreground and background class,

b) $\mu^{\boldsymbol{\theta}}_{\Pi_l}$: mean of the euler angles,

c) $\Sigma^{\boldsymbol{\theta}}_{\Pi_l}$: diagonal variance matrix of the euler angles,

d) $\mu^{\mathbf{d}}_{\Pi_l}$: mean of the offset vector and,

e) $\Sigma^{\mathbf{d}}_{\Pi_l}$: diagonal variance matrix of the offset vector.

For simplicity, we have assumed a unimodal Gaussian distribution with a diagonal variance matrix for the euler angles and the offset vectors.

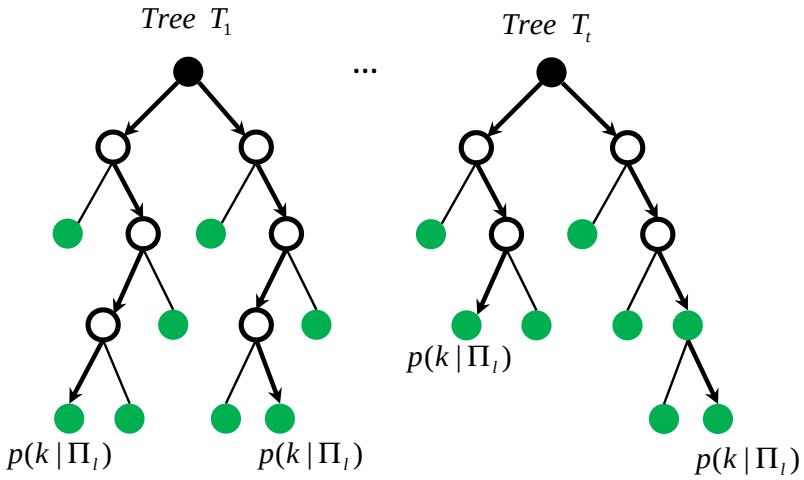


Figure 5: Random Forests.

2.2.1 Entropy

The gain function used to evaluate the quality of split is a weighted sum of the gain due to classification and regression. In performing classification, the aim is to segregate the training patches according to their class labels and in regression the objective is to minimize the variance of the offset vector $\mathbf{d}$ and the rotation vector $\boldsymbol{\theta}$ computed over the training patches. Entropy of class distributions is computed by using the class probabilities. For regression, we assume a unimodal Gaussian distribution over $\mathbf{d}$ and $\boldsymbol{\theta}$, the entropies are then computed using the standard formula of entropy for Gaussian distribution.

2.2.1.1 Classification

After the set of patches Π_n reaches a node n, the class probabilities $p(k|\Pi_n)$ are computed using equation 1. The factor r_k adjusts for any bias induced by unequal number of positive and negative patches in the training data set for the tree.

$$p(k|\Pi_n) = \frac{|\Pi_n^k|.r_k}{\sum_k (|\Pi_n^k|.r_k)};\ r_k = \frac{|\Pi_r|}{|\Pi_r^k|} \tag{1}$$

Once the class probabilities are computed, the *entropy impurity* or *information*

impurity is computed for node n using equation 2.

$$H_c(\Pi_n) = -\sum_k p(k|\Pi_n)\ \log(p(k|\Pi_n)) \tag{2}$$

2.2.1.2 Regression

Entropy due to a Gaussian distribution is given by $H[\mathbf{x}] = \frac{1}{2}\ln|\Sigma| + \frac{D}{2}(1 + \ln(2\pi))$, where $\mathbf{x} \in \mathbb{R}^D$ and Σ is the covariance matrix. Assuming, unimodal Gaussian distribution for offset vector $\mathbf{d} \in \mathbb{R}^3$ and the orientation vector $\boldsymbol{\theta} \in \mathbb{R}^3$, the *entropies* $H_{rl}[\mathbf{d}]$ and $H_{ro}[\boldsymbol{\theta}]$ for location and orientation can be estimated by replacing Σ with $\Sigma_n^{\mathbf{d}}$ and $\Sigma_n^{\boldsymbol{\theta}}$ respectively (sub-script n denotes the node). It should be noted that *entropies* for the distribution of $\mathbf{d}$ and $\boldsymbol{\theta}$ are computed only over the positive patches.

2.2.2 Features and Split Function

To split training image patches between the child nodes, a split function is employed. The split function f_ϕ is a real-valued function that determines whether a patch goes to the left or right child of a non-leaf node n. The split function operates on features to produce a binary output. The features that are used are typically low-level features, such as the pixel difference [61] or the difference of averages of two windows [22, 23, 15, 28] (see [66] for a brief description on features).

The features F_1 and F_2 are rectangular windows located inside a patch and are generated randomly. Figure 6 shows the feature windows located inside a positive and a negative patch. In this approach, scale invariance is introduced by scaling the dimensions of the patches using the depth value z of the pixel located at the center of the patch. Equation 3 shows the height and width of a patch π being scaled. For pixel located farther from the camera, the dimensions of patch extracted around it are smaller than it is for a pixel located close to the camera. The parameters ρ_{ht} and ρ_{wid} are evaluated from cross-validation.

$$\pi_{ht} = \rho_{ht}/z, \pi_{wid} = \rho_{wid}/z \tag{3}$$

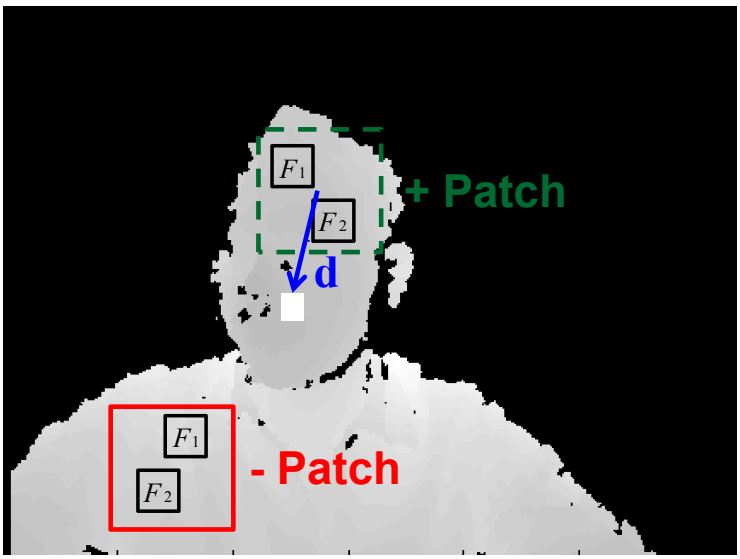


Figure 6: Features F_1 and F_2 located inside a positive (green) and a negative (red) patch.

From each patch π, two low-level features F_1 and F_2 are randomly chosen. Figure 6 shows a positive patch (dashed rectangle) and a negative patch (solid rectangle). The split function f_ϕ is a real-valued function that determines whether a patch goes to the left or right child of a non-leaf node n. The split function is parametrized by $\phi \in \Phi$ where $\phi = \{F_1, F_2, \tau\}$ and, $\tau \in \mathbb{R}$. The threshold τ determines whether a patch goes to the left or the right child node. Figure 7 illustrates how patches are split at each node based on the value of f_ϕ. To compute the split function output for depth based features, first the v_i's, average of pixels values in features windows F_i's, are computed as shown in equation 5. By comparing the difference of v_i's with the threshold τ , the binary output of split function is obtained by using the equation 4. If the binary function outputs 0, then the patch is sent to the left child, otherwise it is sent to the right child or vice-versa.

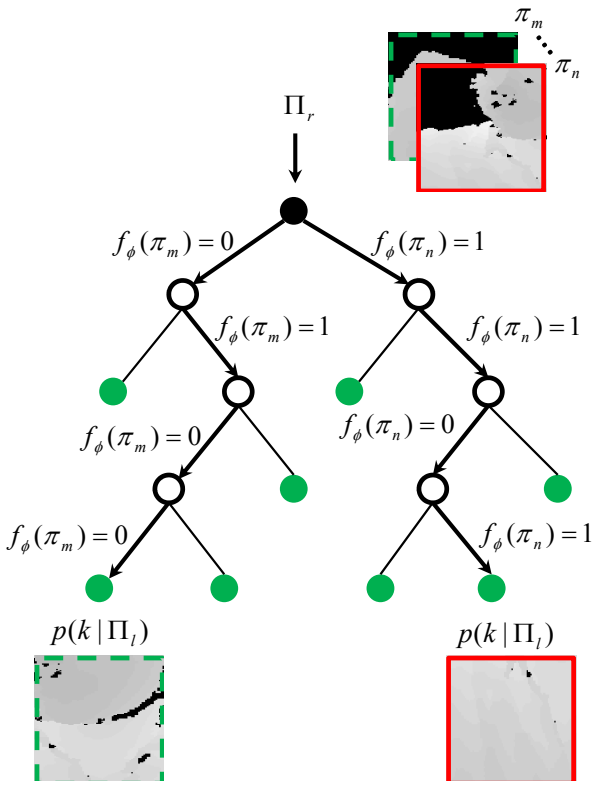


Figure 7: Split function.

$$f_\phi(\pi) = \begin{cases} 0 & if\ v_1 - v_2 < \tau \\ 1 & otherwise \end{cases} \tag{4}$$

where,

$$v_i = \frac{1}{|F_i|} \sum_{0 \le p \le |F_i|} F_i(p),\ i = \{1,\, 2\} \tag{5}$$

2.2.3 Recursive splitting

To train a tree, a recursive splitting approach is followed where the nodes are trained recursively [28, 39]. To train a node, a set of random $\phi \backslash \tau$ are generated, let this set be represented by Φ. Maximization of g by iterating through Φ gives the optimal feature

and threshold: ϕ^* (see section 2.2.5). Using ϕ^*, the patches are divided between the left and right child nodes and the steps 1 to 3 are repeated. The gain function g (discussed in section 2.2.4) is a weighted difference of entropy of the parent and the child nodes. Entropies due to classification and regression were discussed in section 2.2.1. To stop growing the sub-tree a node is not split and is set as a leaf if it meets the criteria which will be discussed in section 2.2.5.1.

1. Generate $\Phi = \{\phi_k\}$.

2. Divide the set of patches Π_n at node n in to two subsets at Π_{n_L} and Π_{n_R} for each ϕ. Patches in the set Π_{n_L} are received by the right child.

$$\Pi_{n_L}(\phi) = \{\pi \in \Pi_n | f_\phi(\pi) = 0\} \tag{6}$$

$$\Pi_{n_R}(\phi) = \{\pi \in \Pi_n | f_\phi(\pi) = 1\} \tag{7}$$

3. Select the split parameters ϕ^* that maximizes the drop in impurity at a node or the *gain function*, g:

$$\phi^* = \arg\max_{\phi \in \Phi} g(\phi, \Pi_n) \tag{8}$$

where,

$$g(\phi,\, \Pi_n) = H(\Pi_n) - \sum_{s \in L,R} \frac{|\Pi_{n_s}(\phi)|}{|\Pi_n|} H(\Pi_{n_s}(\phi)) \tag{9}$$

$H(\Pi_n)$ is the classification or regression *entropy* computed over all patches Π_n at node n.

2.2.4 Gain functions

Input samples are split at each node with an objective to reduce the uncertainty or to gain information and to make predictions accurate. From classification perspective, the gain signifies the grouping of samples of the same class together. From regression standpoint, the gain signifies reducing the uncertainty which is represented by the covariance matrix Σ. The information gain is measured by the gain function g_{cr}

where the subscript cr means the gain is the sum of the classification and regression gain. The goal is to maximize g_{cr} over a set of random samples Φ at any node. The gain function introduced in section 2.2.3 is the weighted sum of classification gain function (g_c) and the regression gain function which comprises of location (g_{rl}) and orientation (g_{ro}) estimation and is given by equation 10. The weight used in equation 10 can be either *linear*, *exponential*and *inter leaved*. *Linear* weight is of the form as shown in equation 11. It gives more importance to classification gain function, however once the class probability for positive samples $p(k = 1|\Pi_l)$ increases beyond the threshold t_p, regression gain starts to contribute to the gain function g_{cr}. Also, the contribution of regression gain function is amplified by the factor α. Recently, [22, 23] proposed *exponential* weights (in equation 12) that gives more importance to regression gain function as the depth of the node d_n increases in the tree. The parameter λ specifies the steepness of change. Unlike *linear* and *exponential* weighing schemes, in *inter leaved* mechanism the gain function is randomly selected to be either the classification gain or the regression gain, instead of a weighted combination of classification and regression gain.

$$g_{cr}(\phi, \Pi_n) = g_c(\phi, \Pi_n) + \omega_r^n \left(g_{rl}(\phi, \Pi_n) + g_{ro}(\phi, \Pi_n)\right) \tag{10}$$

$$\omega_r^n = \alpha \max\left(p(k = 1|\Pi_n) - t_p, 0\right) \tag{11}$$

$$\omega_r^n = \left(1 - e^{-d_n/\lambda}\right) \tag{12}$$

The gain functions g_c, g_{rl} and g_{ro} are computed by using the respective entropies (see section 2.2.1) in equation 9.

2.2.5 Evaluating ϕ^*

The optimal ϕ^* for each node n that maximizes the gain g_{cr} is obtained by randomized node optimization. First a set of $\phi \backslash \tau$ are generated using which the threshold τ is evaluated for all the patches in $|\Pi_n|$. The suitable τ is chosen which maximizes the

gain function g_{cr} over all the patches $|\Pi_n|$ and the set of $\phi \backslash \tau$. This is illustrated in Figure 1 below.

Algorithm 1 Evaluate ϕ^*

while $i <nSamples$ **do**
  Generate $\phi \backslash \tau$
  while $j <|\Pi_n|$ **do**
    Compute $\tau_j = v_1 - v_2$
    $j = j + 1$
  end while
  Set $\tau_i = \tau_k$ s.t $g_{cr}^k \geq g_{cr}^j \ \forall j$
  $\phi^i = \{\phi \backslash \tau, \tau^i\}$
  $g_{cr}^i = g_{cr}^k$
  $i = i + 1$
end while
set $\phi^* = \phi^k$ s.t. $g_{cr}^k \geq g_{cr}^i \ \forall i$

2.2.5.1 When to Stop Splitting?

A batch training approach is used to train the trees. For each tree, a set of patches are selected at random from the entire training data set. Recursive splitting is then performed to grow the tree and it is stopped at a node until a stopping criterion is met. Then node is set as the leaf node and then the class probabilities $p(k|\Pi_l)$ are computed along with the leaf statistics: $\mu_l^{\mathbf{d}}$, $\Sigma_l^{\mathbf{d}}$, $\mu_l^{\boldsymbol{\theta}}$ and $\Sigma_l^{\boldsymbol{\theta}}$.

At a certain stage of training the splitting needs to be stopped to prevent overfitting. By stopping the splitting, the resulting tree typically will have leaves at varying levels. For online learning approach, stopped splitting results in a phenomenon called *horizon effect*, as possible future splits can result in performance improvement. To

address this, post pruning is performed once online training is complete [20]. However for batch processing, stopped splitting does not suffer from the *horizon effect* drawback. Different criteria for stopping splitting or post pruning include setting threshold a) on gain b) minimum number of input samples at a node, or c) maximum node depth of tree. As finding a threshold on gain is difficult a more practical criteria is to use a threshold on minimum number of samples or the depth of tree [28].

2.2.6 Testing: Location and Orientation Estimation

To predict the location and orientation of head in a depth frame, patches are extracted from the frame and passed through the trained forests. Patches are sampled from the image in a strided fashion with a fixed step size. Once the patches are pushed down the trees, they end up at certain leaves. From these leaves, statistics such as the class probabilities $p(k|\Pi_l)$, and the mean and covariances $\mu_l^{\mathbf{d}}$, $\Sigma_l^{\mathbf{d}}$, $\mu_l^{\boldsymbol{\theta}}$ and $\Sigma_l^{\boldsymbol{\theta}}$ are read off. Based on the class probabilities, the leaves participate in estimating the location and orientation. In [22, 23], only those leaves that have +ve class probability $p(k = 1|\Pi_l) = 1$ in prediction. An additional threshold on the covariance matrix $\Sigma_l^{\mathbf{d}}$ is also set to reject leaves with high degree of uncertainty. Irrespective of the threshold used, the underlying idea is to collect reliable leaf estimates from all the trees and then run meanshift algorithm [12, 13] to find the mode from the leaf estimates which gives the head location in a depth frame [61, 22, 23]. Figures 8 and 9 show the leaf votes for possible head location in light green color. To estimate the head orientation, from amongst the selected leaves for voting, only those leaves are used whose voted head location is in the pre-defined neighborhood of the estimated head location obtained from meanshift algorithm. This neighborhood is a spherical ball with a radius represented by ρ_{nh}. By averaging over the leaf statistics for orientation $\mu_{\Pi_l}^{\boldsymbol{\theta}}$ in this neighborhood an estimate for the head orientation in the frame is obtained.

Extracting patches in a strided manner from the entire depth frame generates a

lot of noisy votes which can affect the estimation accuracy. Additionally, such an approach also incurs a computational cost as we need to evaluate many pixel location in the depth frame to generate hypothesis for locating the head. It is beneficial both in terms of computational cost and accuracy to sample only those regions in the frame where the head is likely to be located. Also, since the head pose of the driver is being estimated it is fairly reasonable to assume there is only one person in the camera scene. Under this assumption, evaluating the pixel locations covering the entire image is not beneficial. In this regard, the sampling region is constrained to be a bounding box region which is centered at the head location estimated from the previous frame. Figures 8 and 9 illustrate bounding box dimensions changing with the head pose obtained from the previous frame. Note that in the figures the reference head location is the temple and not the nose as is the case in the BIWI Kinect Head Pose (BKHP) database [22, 23]. The BKHP database is used only for validating the approach. Instead of a fixed size, the dimensions of the bounding box is a function of a) depth value and b) the head orientation angle, both obtained from the estimated head pose from the previous frame or using a filter (such as Kalman filter) to predict the pose for the current frame. Equations 13 and 14 below show how the bounding box dimensions are allowed to scale with the head location and head orientation. Scaling with depth value allows the bounding box to become smaller as the head moves away from the camera and vice-versa. Figure 8 shows how the bounding box scales with the distance from the camera. For head orientation angle, the pitch and yaw angles of the head pose from the previous frame are used to scale the dimensions of the bounding box. In equations 13 and 14, z^{k-1} is the z coordinate of the estimated head location in the $(k-1)^{th}$ frame. f_x and f_y are the focal lengths the depth camera along the X and Y directions and α_p^{k-1} and α_y^{k-1} are the pitch and yaw angles from the previous frame. The pitch angle affects the height of the bounding box and the yaw angles affects the width of the bounding box. The height of the bounding box is the

largest when the pitch angle is 0 degree and it becomes smaller as pitch becomes more positive or negative. Figure 9 illustrates how the bounding box dimensions change with the yaw and pitch angles. However, for the width of the bounding the box, the smallest value is obtained when the yaw angle is 0 degree and it increases as the yaw angle becomes more positive or negative. The reason behind increasing the bounding width with increasing magnitude of yaw angle is to sample over a larger facial region and which helps in reducing the jitteriness in estimation across frames. In the results section, it is shown that following such a bounding box approach gives better results as compared to following the scheme of sampling the entire depth image. The factors h_f and w_f are the face height and face width. From empirical observations these values were set to the following: $h_f = 150.0$ mm and $w_f = 120.0$ mm. Low values of these factors causes jitteriness in estimation as a result of a smaller sampling region while increasing the execution speed, vice-versa for higher values.

$$bb^k_{ht} = (h_f / z^{k-1}) f_y cos(\alpha_p^{k-1}) \tag{13}$$

$$bb^k_{wid} = (w_f / z^{k-1}) f_x / cos(\alpha_y^{k-1}) \tag{14}$$

Close to the camera

Away from the camera

Figure 8: Bounding box scaling with distance to the camera.

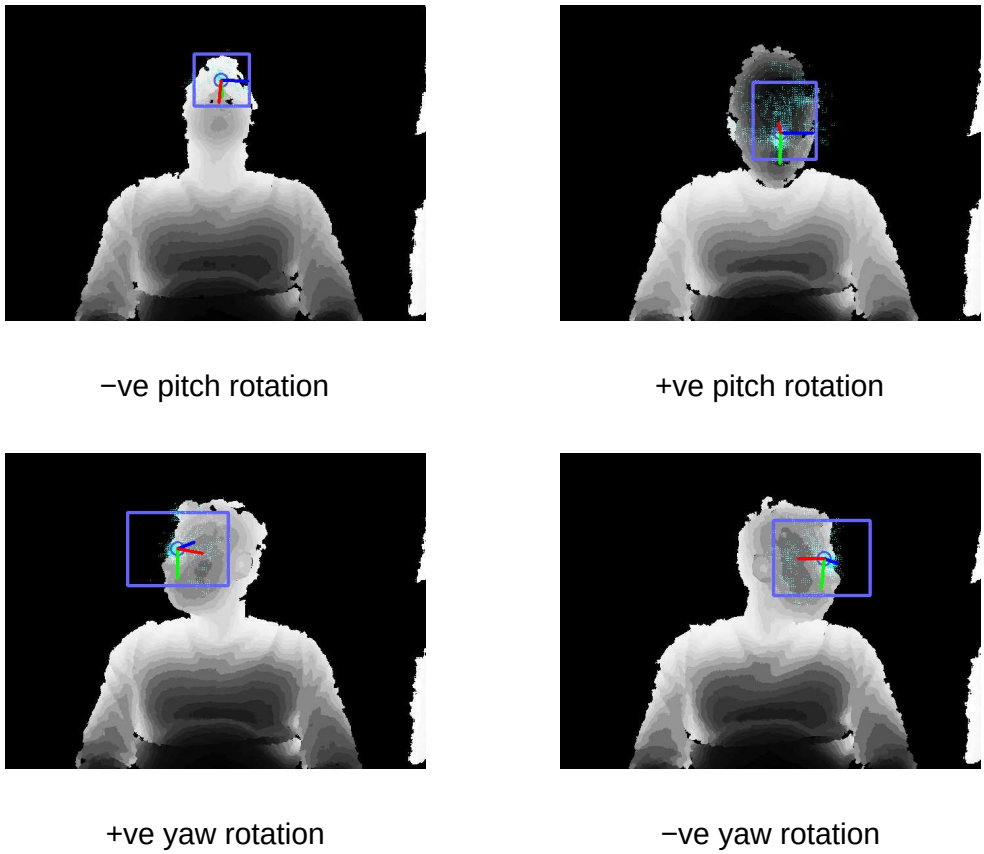


Figure 9: Bounding box height and width changing with the pitch and yaw angle.

2.3 Data Set and Results

The BKHP database [22, 23] was used to validate the approach. The database contains around $\sim$ 15000 frames (depth and rgb) collected from 20 different individuals. The data set is structured into 24 sequences, and each sequence corresponds to a subject, with some subjects having multiple sequences. The head rotations in the database are in the range $\pm 60^\circ$ for pitch (α_p), $\pm 75^\circ$ for yaw (α_y), and $\pm 50^\circ$ for roll (α_r). Table 1 shows the location and orientation estimation error reported by [24]. The nose localization error was measured in terms of the euclidean distance whereas the orientation error is measured for each of the euler angle. The timing was measured on a 2.67GHz Intel Core i7 CPU.

Table 1: Mean and standard deviation of the errors for location and orientation estimation reported by [24].

Stride	Nose (mm)	Pitch (°)	Yaw (°)	Roll (°)	Missed (%)	Time (ms)
5	12.2 ± 22.8	3.5 ± 5.8	3.8 ± 6.5	5.4 ± 6.0	6.6	44.7

2.3.1 Training

Several Random Forests, indexed f_1 to f_6, are trained, each with a different parameter set as shown in Table 2. Each forest contained 8 trees and the maximum allowed depth in a tree was 30. To prevent overfitting, splitting at a node is stopped if it received less than 10 patches from its parent node. Also, to prevent undesirable splitting at a node with high number of negative patches, a threshold is set on the value of $p(k = 0|\Pi_n)$. If the probability of negative patches, $p(k = 0|\Pi_n)$, at a node exceeds the threshold then the node is not grown any further. The first 18 sequences of the BKHP database are used for training. From each training image, three positive and six negative patches are extracted randomly. Thus, each of the Random Forests was trained with $\sim 100K$ patches. At each node in a tree, 500 random feature windows (F_1 and F_2) are generated and 100 thresholds are evaluated and thus 50K binary split functions are evaluated at each node. The minimum and maximum size of the feature windows (at 1m distance) is set to 15 and 35 respectively. In addition to the parameters stated in Table 2, the patch scaling factors: $\{\rho_{ht}, \rho_{wid}\}$ are also evaluated. To reduce the number of parameter evalulations and thus the training time, only square patches are used, that is $\rho_{ht} = \rho_{wid} = \rho_{patch}$. A coarse grid search is performed to find the best for patch scaling factor value from the set: $\rho_{patch} = \{6.1e4, 8.1e4, 1.01e4\}$ which corresponds to patches of sizes 61×61, 81×81 and, 101×101 respectively when the depth value of the pixel at the center of the patch is 1000mm. Thus, in total $6 \times 3 = 18$ forests were trained. In section 2.3.2 the performance of Random Forests

for different patch sizes has been presented.

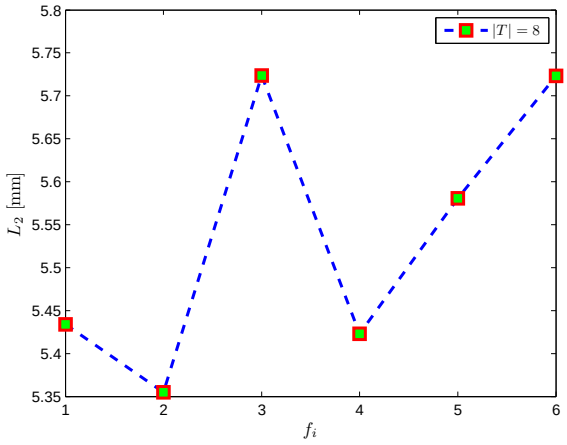
Table 2: Parameters of Random Forests used for training.

Forest Index	Weight Type	Parameters
f_1 to f_2	linear	$\alpha = 1.0$, $t_p = \{0.6, 0.8\}$
f_3 to f_5	exponential	$\lambda = \{5, 10, 15\}$
f_6	interleaved	random selection

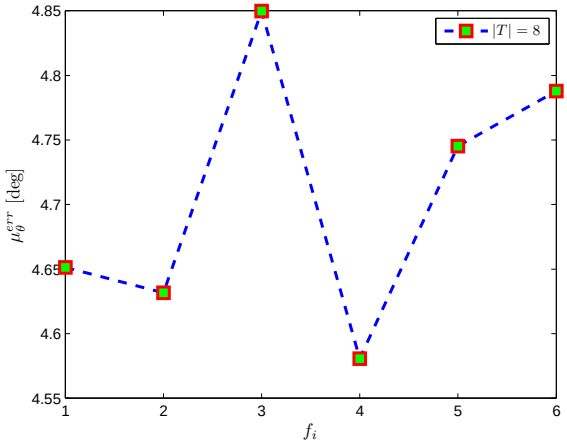
2.3.2 Testing

To select the best parameters for Random Forest, the test error was computed on sequences 19 and 20 in the BKHP database. The total number of images in these two sequences was 1058. The estimation on a frame is considered to have failed or missed: If there are no reliable leaf estimates available for performing meanshift algorithm. Reliable leaves are those that meet the criteria as defined in section 2.2.6. A threshold of 0.8 is set for $p(k = 1|\Pi_l)$, that is leaves with $p(k = 1|\Pi_l) = 1$ value more than 0.8 participate in the voting phase. Also, to reject uncertain leaves, an additional threshold is set on $\Sigma_l^{\mathbf{d}}$, such that leaves whose $tr\,\Sigma_l^{\mathbf{d}} < 500.0$ participate in the voting phase. A fixed step size of 3 is used to select pixel locations around which patches are extracted. The meanshift bandwidth is set to 50.0 mm and a maximum of 25 iterations are performed to achieve convergence. To estimate the orientation, leaves are chosen if they lie in the neighborhood defined by the radius $\rho_{nh} = 15.0$mm.

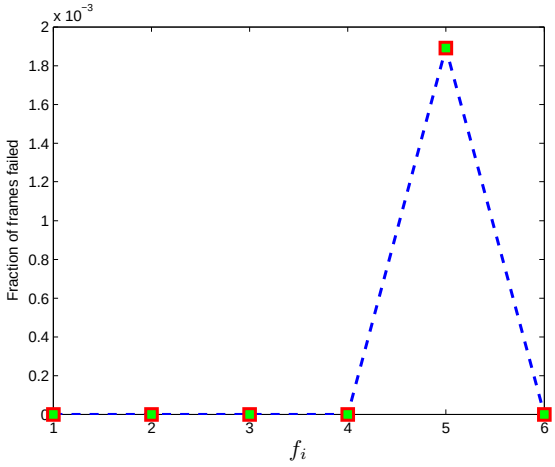
The results reported below were obtained on a 2.30 GHz Intel Core i3 CPU machine, with only one core being used (with all the frames loaded in the memory). Figures 10, 11, 12 show the results for the patch scaling factor ρ_{patch} set to 6.1e4, 8.1e4 and 1.01e4 respectively. Patches when scaled by the factor $\rho_{patch} = 6.1e4$, gave the lowest error for both location and orientation estimation.



(a) Mean L_2 nose location estimation error.

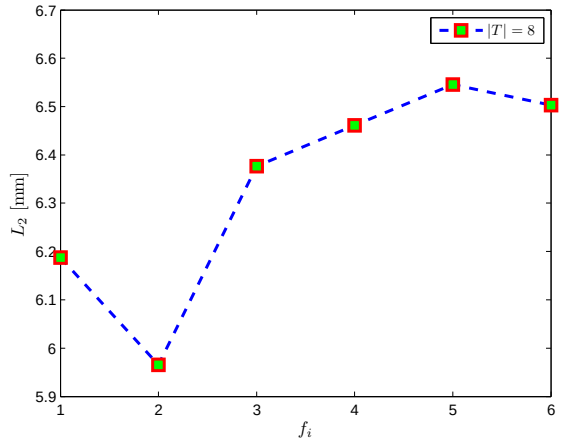


(b) Mean euler angle ($\boldsymbol{\theta}$) estimation error.

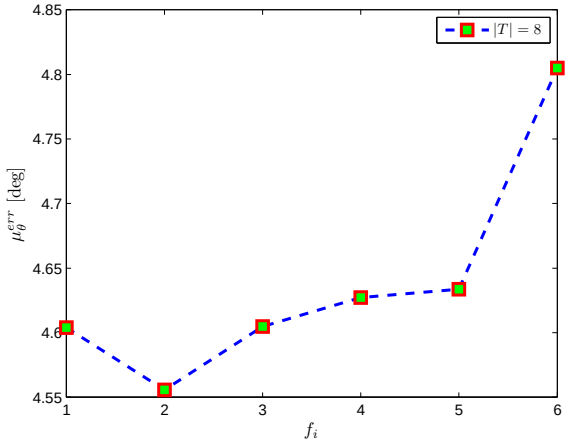


(c) Fraction of frames failed.

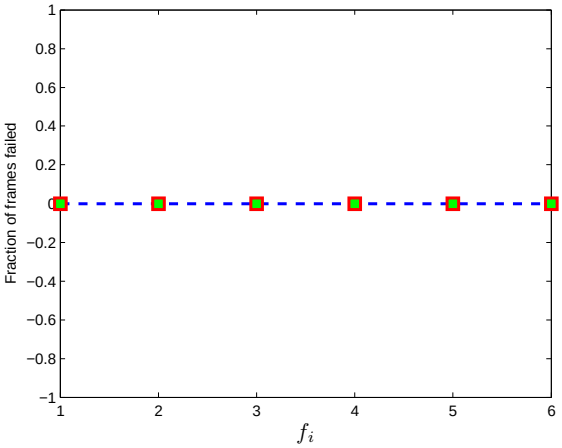
Figure 10: Test results for the patch scaling factor $\rho_{patch} = 6.1e4$.



(a) Mean L_2 nose location estimation error.

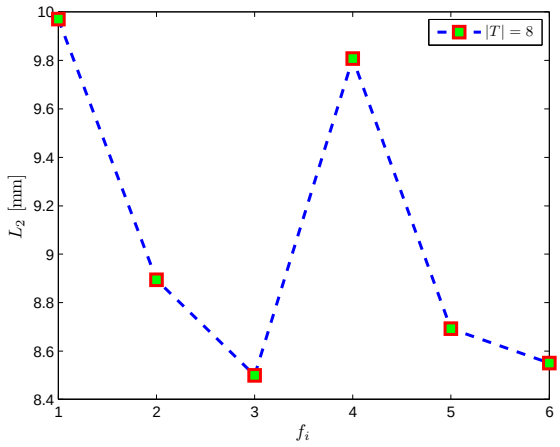


(b) Mean euler angle ($\boldsymbol{\theta}$) estimation error.

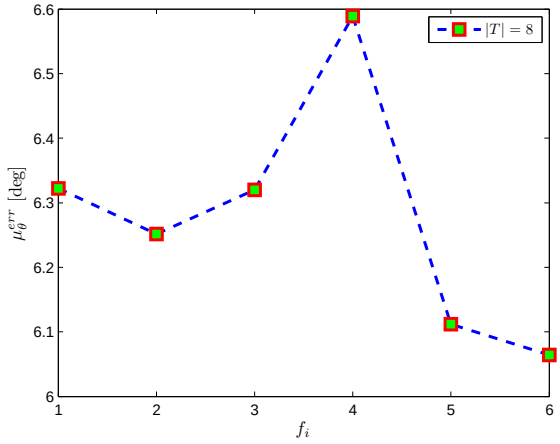


(c) Fraction of frames failed.

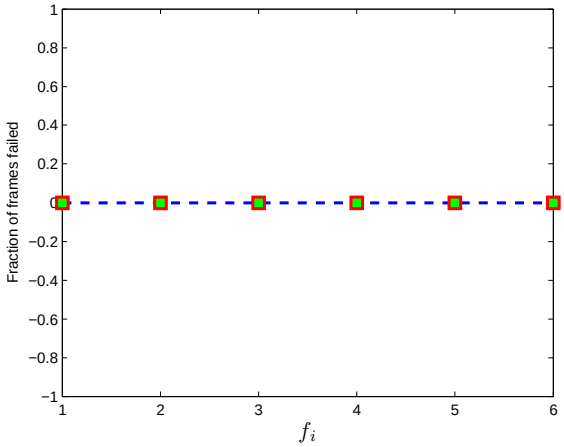
Figure 11: Test results for the patch scaling factor $\rho_{patch} = 8.1e4$.



(a) Mean L_2 nose location estimation error.



(b) Mean euler angle ($\boldsymbol{\theta}$) estimation error.



(c) Fraction of frames failed.

Figure 12: Test results for the patch scaling factor $\rho_{patch} = 1.01e4$.

From Figure 10, it can be concluded that both the *linear* weighing mechanism with $\alpha = 1$ and $t_p = 0.8$ and the *exponential* weighing mechanism with $\lambda = 5$ are the best performing forests. Table 3 shows the location and orientation estimation of the selected forest f_2 (*linear* weighing mechanism with $\alpha = 1$ and $t_p = 0.8$). Comparing Table 3 with Table 1, it can be observed that the nose localization error has reduced significantly while euler angle error is only marginally more. However, the number of missed frames is 0 as compared to 6.6% as in Table 1. In addition, the presented approach runs at $\sim$ 125 fps as compared to $\sim$ 25 fps as reported in Table 1. The speed-up is due to the bounding box approach due to which only a small region in a depth image is evaluated.

Table 3: Test error of forest f_2 (linear weighing mechanism with $\alpha = 1$ and $t_p = 0.8$).

Forest	Patch Size	Location Error (mm)	Angle Error (°)	Missed	Time (ms)
f_2	61×61	$l_x^{err} = 3.4 \pm 4.5$ $l_y^{err} = 3.6 \pm 3.9$ $l_z^{err} = 2.0 \pm 2.5$	$\alpha_p^{err} = 2.8 \pm 3.6$ $\alpha_y^{err} = 4.5 \pm 7.0$ $\alpha_r^{err} = 6.5 \pm 7.6$	0/1058	8.3

CHAPTER III

COMPUTING STATIC BLINDSPOTS OF EQUIPMENT

3.1 Background

Industrial powered equipment is becoming more popular in construction as construction material has to be frequently moved. In the construction sector, forklifts are more frequently being used on construction sites as many elements that are used in construction are pre-fabricated. Thus, forklifts have become an important lifting and hauling resource that is essential to carry out modularized construction successfully. However, as is known in other industries such as the transportation sector, forklifts are associated with a lot of accidents. NIOSH [53] has reviewed the number of fatalities and traumatic injuries in forklift-related incidents. From 1980 to 1994, 1021 workers died. 20% of these related to "worker on foot struck by forklift" incidents. In 1995, 94 fatal injuries were associated with forklifts.

Several approaches have been developed to measure the blindspots of equipment. These approaches can be divided into (a) manual and (b) computer simulation methods. Using a manual approach, an artificial light source was mounted at operator's seat using a Seat Index Point (SIP) Apparatus (based on ISO 5353 standard) and the visibility of a test body or test screen was measured following the ISO 13564-1 standard for powered industrial vehicles [5]. Such approaches are time-consuming and require extensive set-up to measure visibility. As an alternative, the National Institute of Occupational Safety and Health (NIOSH) proposed a "low-tech alternative" that does not require any infrastructure set-up or computer design drawings [52]. The procedure involves preparing a polar grid test bed around the equipment with the operator's seating location at the center of the grid. The visible areas are then

marked around the equipment on this polar-grid manually depending on operator's perception of the grid's visibility. This approach was primarily proposed for "construction companies, labor unions, and training organizations to better understand the blind areas around their own equipment." However, this approach is "subjective" as it involves the "human element" in measuring visible regions around equipment.

Using computer simulation methods blindspots diagrams are developed using CAD models of equipment. Software based artificial lighting is used to determine the blindspots [51, 33]. The blindspots include indirect visibility due to mirrors. The measurements can develop blindspots diagrams on ground plane, and on planes at 900mm and 1500mm vertical distance direction from the ground. This approach allows measuring direct visibility and indirect visibility (visibility due to mirrors). However, on construction sites equipment may undergo modification in which case the blindspots map may change after it has been purchased by a user.

Recently, an automated blindspots measurement tool was developed by [65] to measure the static blindspots of construction equipment. The approach followed the standards related to earth-moving machinery-operator's field of view (FOV) [36]. It utilized a three-dimensional (3D) point cloud of equipment generated by a laser scanner. Blindspots maps were produced by running Ray Casting algorithm [2] on the point cloud. It provides a fast alternative to measure blindspots "objectively", without involving "human perception" in measuring the blind areas manually. Furthermore, changes made to the equipment cabin can be accounted easily as the approach just requires rescanning the equipment. However, the results provided are empirical without any validation [65].

As articulated in [65], any proposed method must at least comply with existing blindspots measurement techniques or international standards. As these currently focus primarily on 2D drawings, the following blindspots calculations and analyses (from any viewing perspective) should be refined or added:

- Volumetric blindspots
- Blindspots map, 12m circumference visibility,
- Rectangular 1m boundary visibility, and
- Worker visibility

From existing approaches it is not clear whether or not computer simulation methods can perform the above measurements subjectively or objectively. Furthermore, to help construction operators understand the visibility of an equipment, a laser scanning based approach may be more feasible. Point cloud data of an equipment can be obtained by using a laser scanner. A scan typically yields millions of points and thus a need to develop a fast and efficient approach to compute blindspots exists.

3.2 Methodology

Figure 13 illustrates the research methodology. Point cloud data is obtained by laser scanning the equipment. The laser scan may be performed by (a) taking multiple scans and registering them or (b) mounting the scanner at the driver's seat of the equipment and performing the scan once [65]. Approach "*a*" typically aids in capturing the equipment surface better as compared to interior scan approach "*b*". There are two reasons why scans from the outside yield better point clouds: (1) many dark or black surfaces exist inside cabins and (2) wind shields or windows might be dirty. In both cases, range measurements vary or can be inaccurate by producing artifacts. Equipment with no enclosed cabin may not have such issues. In general, range point clouds may get significantly affected by glass windows on equipment and thus fail to capture the interior surface of the cabin accurately. Approach "*b*" takes less time and can capture the interiors better but however it may fail to capture the exterior surfaces accurately (which can also obstruct the field-of-view of an operator; for example,

an exhaust pipe or blade mounted at the front of the equipment). Thus, integrating these two methods can produce an accurate point cloud representation of equipment.

A manual cleaning procedure is typically followed to remove artifacts from point cloud using commercially available point cloud manipulation software. This step has only to be performed once though. These artifacts may be caused by artificial lighting and reflectivity of certain cabin surfaces (such as glass windows). The point cloud data is then binned into a 3D histogram. The 3D histogram is constructed in a cache-friendly way which results in speeding up the computations. By running Ray Casting algorithm on the histogram, the different blindspots analyses such as: (a) volumetric blindspots, (b) blindspots map, (c) rectangular 1m boundary, (d) 12m circumference visibility, and (e) worker visibility are performed.

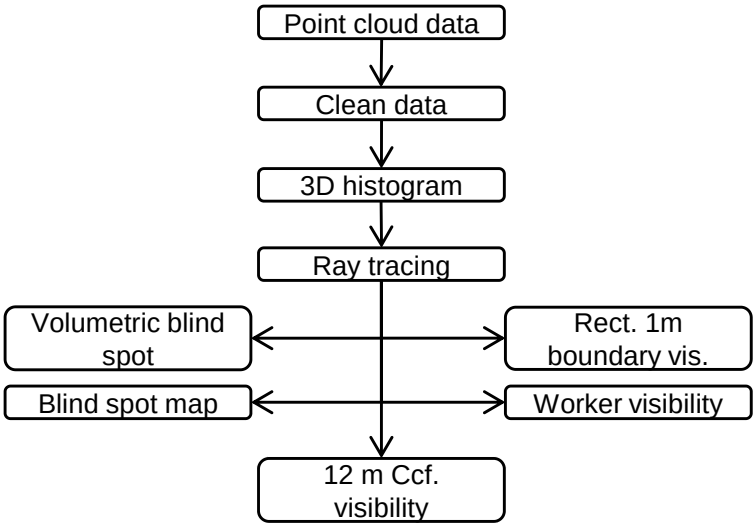
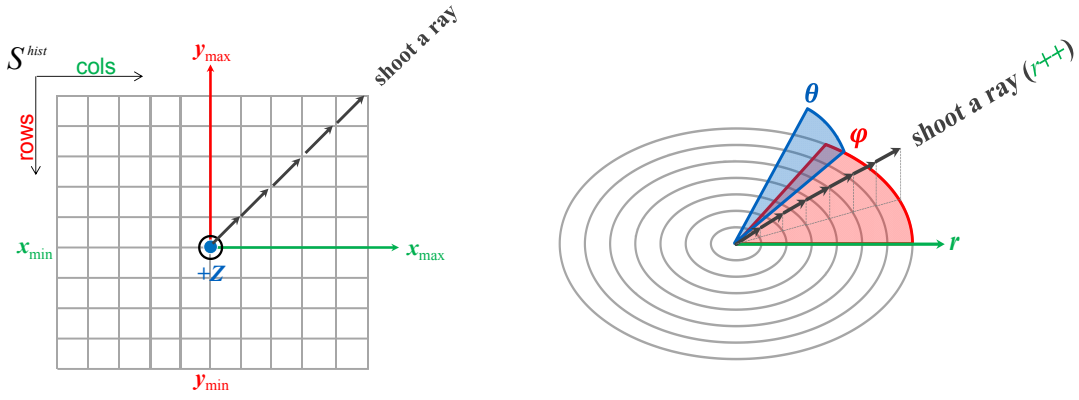


Figure 13: Methodology diagram of blindspots measurement.

3.2.1 3D Histogram (S^{hist}) Construction

The input laser scan data, denoted by P_{in}, can be either in cartesian or in spherical coordinate system. It is assumed that the input laser scan data is in cartesian coordinate system. The collected laser scan data is then binned in to a 3D histogram. The 3D histogram can be constructed in (a) cartesian, (b) spherical $\langle r, \theta, \phi \rangle$, or (c)

cylindrical $\langle z, \rho, \phi \rangle$ coordinate system. An appropriate choice of the coordinate system can significantly reduce the computational cost. The most significant factor that contributes to the computational cost is fetching the 3D histogram data from memory to the processor while performing Ray Casting. By making efficient use of the cache, the number of fetches from memory can be reduced. Performing Ray Casting algorithm in cartesian coordinate system caused strided memory access as shown in figure 14a and thus makes inefficient use of the cache memory. A similar argument can be made against cylindrical coordinate system. Choosing spherical coordinate system for constructing the 3D histogram allows contiguous memory accesses in the Ray Casting algorithm which increases the cache hit ratio and thus results in reduced "time to solution". The 3D histogram matrix, constructed in the spherical coordinate system, is denoted by $S^{hist}[t][p][r]$, where t, p and r are the bin indices along θ, ϕ and r dimensions. An increment in the bin index corresponds to a change of $\Delta\theta$, $\Delta\phi$, Δr and respectively. For the ease of notation, $S^{hist}[t][p][r]$ is represented by S^{hist}. It is noted here that θ and ϕ are also known as the azimuth and elevation respectively. Figure 14b illustrates Ray Casting in spherical coordinate system.



(a) Ray Casting in cartesian coordinate system. **(b)** Ray Casting in spherical coordinate system.

Figure 14: Illustration of Ray Casting algorithm in cartesian and spherical coordinate system.

3.2.2 Ray Casting

The *Ray Casting* algorithm forms the backbone of the blindspots computations. The principle behind Ray Casting algorithm is to shoot a ray from the origin along a direction by incrementing the radius index r of S^{hist} while keeping θ and ϕ constant. If the ray hits a bin that has a frequency greater than a specified threshold then that bin is termed as a "blocking bin". All subsequent bins lying behind a blocking bin, along that ray, lie in the blindspots region. This process is repeated for different values of θ and ϕ.

It can be observed that if the histogram is constructed in the spherical coordinate system then casting path of a ray will correspond to moving along the row of the S^{hist} matrix. The row ordering layout in memory of matrix (implementation was performed in C) allows accessing contiguous memory locations. This increases the cache hit ratio and thus results in a reduced "time to solution". On the other hand, constructing histogram in cartesian or cylindrical coordinate system can cause random or strided memory accesses which can be computationally costly as the number of bins is typically high ($\sim 10^7$).

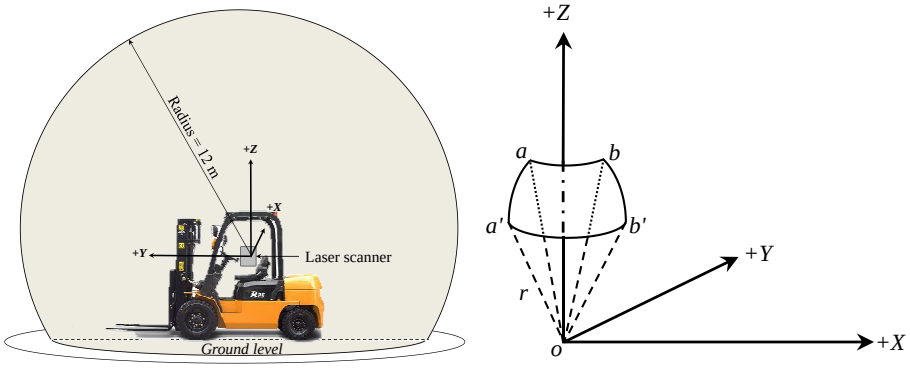
As discussed above, the Ray Casting algorithm needs a choice of origin. The origin is located at "approximately" the same location as the equipment operator's head. Only an "approximate" location is necessary as the approach has the capability to perform analyses from other "viewpoints". Viewpoints reflect the head location of an operator. To perform the analyses from different viewpoints, the constructed 3D histogram S^{hist} is converted into a coarser point cloud (denoted by P_0) in the cartesian coordinate system. However, all bins in S^{hist} are not added to P_0. Bins whose frequency is higher than a certain threshold are added to the point cloud P_0. This allows construction of a coarser representation of the original input laser scan data P_{in}. Assuming the viewpoint has changed by v (in the k^{th} frame), then the point cloud is translated by $-v$ to form a new point cloud P_k. By constructing a new 3D

histogram S_k^{hist} from the point cloud P_k and then performing Ray Casting on S_k^{hist}, all blindspots analyses can be performed from this new viewpoint. Construction of the histogram S_k^{hist} is sped up since the P_k is a reduced representation of P_{in} with significantly less number of points. In implementation, S^{hist} is reused as S_k^{hist}, so that no extra memory allocation is required for the storing S_k^{hist} memory.

3.3 Blindspots Analyses

3.3.1 Volumetric Blindspots

The *percentage volumetric blindspots* is defined as the ratio of total blind areas on the surface of a 12m radius sphere to the total area of the same sphere lying above the ground plane. The sphere is assumed to be centered at the origin or the head of the operator. The surface of the sphere lying above the ground plane is only considered in the computation. Figure 15a illustrates this imaginary sphere. Even though the blindspots are measured as a ratio of the blindspots areas to the total area on the sphere, it can still have a volumetric interpretation if the visibility of an operator inside the equipment cabin is disregarded. In figure 15b, the surface patch $abb'a'$ can be interpreted volumetrically by considering the origin 'o' as the vertex of the cone whose base is the surface patch $abb'a'$.



(a) **(b)**

Figure 15: Illustration of volumetric blindspots measurement.

The pseudo code for volumetric blindspots computation is shown in Algorithm 2.

Algorithm 2 Ray Casting for computing volumetric blindspots.

```
1: totalNumRays = nBins_ThtGrnd* nBins_Phi
2: nBins_R= floor (r'/Δr +0.5)
3: rayCount = 0
4: for t = 0 to nBins_ThtGrnd do
5:     for p = 0 to nBins_Phi do
6:         bool blindspotFlag = false
7:         for r = 0 to nBins_R do
8:             if S^hist[t][p][r] >binThresh then
9:                 blindspotFlag = true
10:                break
11:            end if
12:        end for
13:        if blindspotFlag == true then
14:            rayCount += 1
15:        end if
16:    end for
17: end for
18: volBlindspot = 100.0*(rayCount/totalNumRays)
```

$nBins_{Phi}$ and $nBins_{Rad}$ represent the number of bins in S^{hist} along with ϕ and r dimensions. $nBins_{ThtGrnd}$ corresponds to the bin index of discretized elevation angle of the ground plane measured with respect to +Z axis. $binThresh$ is the frequency threshold parameter and its value is at the discretion of the user. If a encounters a "blocking bin" while proceeding along the radial direction, then the *boolean* variable $blindspotFlag$ is set to $true$, otherwise $false$. Thus, the volumetric blindspots is the ratio of the number of rays ($rayCount$) that have hit "blocking bins" to the total

number of possible rays (denoted by $totalNumRays$).

3.3.2 Blindspots Map

A *blindspots map* is defined as the mapping of visible and blind areas contained in a 12m radius circle lying on planes parallel to XY, YZ or XZ planes. The operator's head is located at the center of the map. While computing blindspots on the XY plane, the equipment's foot print area is discarded. The equipment's foot print area is defined as the area enclosed by the smallest rectangle that can be placed around the vertical projection of the equipment on the test floor (ground plane) on which the equipment is located (see figure 20).

From geometry it is known that area of a sphere can be computed by integrating the area over smaller patches constructed over the surface of the sphere. Such a patch is referred to as an "elementary patch" here and has been illustrated in the figure 15b. In figure 17, pq represents the YZ view of such a patch P'' and similarly for ab which represents patch P. Let be the area of the P''. Using geometric principles,

$$\Delta A_E = R^2 sin(\theta_n)\Delta\theta_n\Delta\phi_n \tag{15}$$

where, R is the radius of the sphere and θ_n is the angle formed by the normal to P'' with the +Z axis. ϕ_n is the angle formed by the projection of the normal on XY plane with the +X axis. $\Delta\theta_n$ and ϕ_n are the step sizes (or bin widths) for θ_n and ϕ_n, respectively. Thus, ΔA_E gives the area of an elementary patch on the surface of the sphere. However, our interest is to compute the total area of blindspots regions on a circular section plane parallel to X, Y, or Z axis. Figure 17 shows a circular section plane (shaded) of radius R' and parallel to the XZ plane. It should be noted that by projecting the surface area of the sphere above the section plane on to the section plane, the area of the section plane is obtained. The projection of the surface area of the sphere above the section plane on to the section plane is achieved by projecting

patches such as P'' to form "projection patches" P', as illustrated in figure 17. This is the key concept that is used in computing the blindspots area on a section plane.

Computing blindspots on a section plane:

- Shoot a ray from the origin until it hits the section plane. In figure 17, ray oa and ob hit the section plane at points m and n respectively.

- If before hitting the section plane, the ray hits any "blocking" bin, then the point at which it hits the section plane is a part of the blindspots region or else it is visible.

- Now, project points m and n on to the sphere (of radius R), as shown in figure 17. In the figure the projected points are p and q respectively. The arc ab is then formed by $\Delta\theta$ change in θ, and let the arc pq be formed by $\Delta\theta_n$ change in θ where $\angle aoZ = \theta$ and $\angle poZ = \theta_n$. Thus, the area of "projection patch" P' can be computed by projecting P'' on to the section plane.

Figure 16 illustrates the process of computing blindspots map on XY map. The gray points shown in the figure (left side) represent equipment point cloud. Rays are shot in directions as required to construct the blindspots map. Depending on whether the rays get obstructed or not, regions on the ground plane are determined as blindspots (represented by red colored pixels) and visible regions (represented by green pixels) respectively.

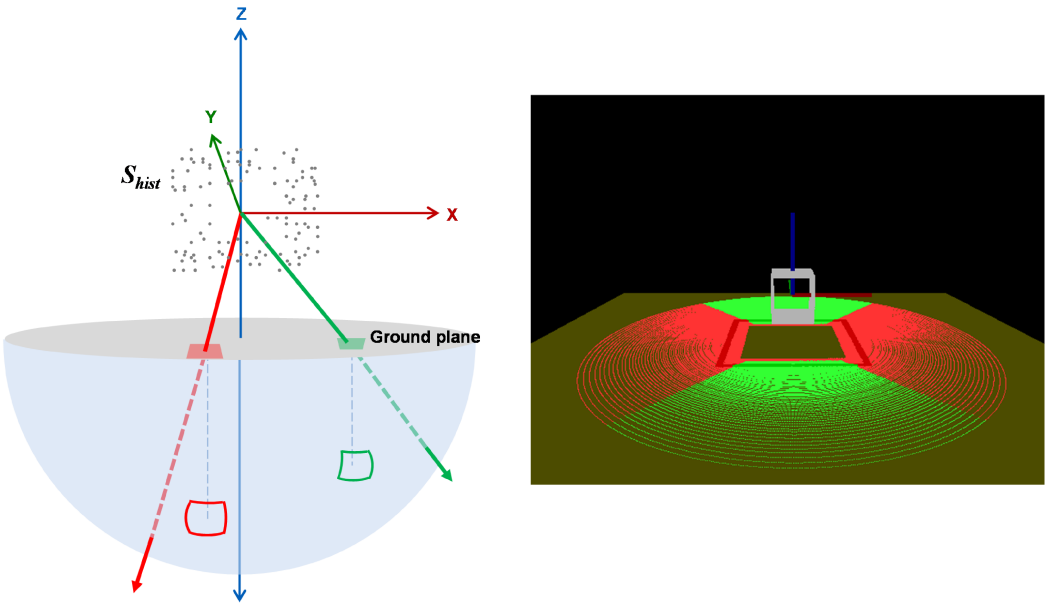


Figure 16: Illustration of Ray Casting algorithm for computing blindspots map on XZ plane.

3.3.2.1 Computing blindspots on a section plane parallel to XZ

Let ΔA_P be the area of any projection patch P' on the section plane. It may be recalled that the area of any elementary patch P'' is denoted by ΔA_E. If ΔA_P forms a part of the blindspots region then it is annotated as $\Delta A_P^{blind\,spot}$. In figure 18, $m'n'$ represents the XY view of the projection patch P' and similarly, $p'q'$ is the XY view of P''. The relationship between ΔA_P and ΔA_E is given by equation 16.

$$\Delta A_P = \Delta A_E \cos(\beta) \tag{16}$$

where β is the angle formed by the normal to the pq with the +Y axis. Summing

the area ΔA_E for all elementary patches P'' above the section plane (if C is on the +Y axis) or below the section plane (if C is on the -Y axis) gives the area of the section plane by using equation 16. This summation results in a summation over all the projection patches on the section plane. Consequently, summation over those projection patches that form the blindspots region yields the total area of blindspots region on the section plane. Thus, after computing ΔA_P for all projection patches, the percentage blindspots on the circular section plane can be computed using equation 17.

$$\%age\ blindspots = \frac{\sum \Delta A_P^{blindspots}}{\sum \Delta A_P} \times 100 \tag{17}$$

where, the summation is over all the elementary patches that lie on or above the section plane or on or below the section plane depending on the location of C. Mathematically, the denominator $\sum \Delta A_P$ in equation 17 equals $\pi R'^2$ (with the inclusion of equipment foot print area on XY plane). While performing Ray Casting, rays oa and ob are shot from which θ, ϕ, r, $\Delta\theta$, $\Delta\phi$ and Δr are evaluated. However, to compute area of P', that is ΔA_P, P' is projected back to the sphere to form P''. After projection, ΔA_P is evaluated as using equation 16. Thus, ΔA_P can be evaluated from ΔA_E by using equation 15. To compute ΔA_E using equation 15, the values of θ_n, ϕ_n , $\Delta\theta_n$, and $\Delta\phi_n$ are required. ϕ_n is used to compute β by using the relation: $\beta = |90 - \phi_n|$. And finally, the values of θ, ϕ and, r, can be computed from the indices of S^{hist}.

3.3.2.2 Computing $\Delta\theta_n$ and θ

Observing the following from figure 17, we obtain equation 18.

$$\angle aoZ = \theta,\ \angle aob = \Delta\theta,\ \angle poZ = \theta_n,\ \angle poq = \Delta\theta_n$$
$$|om| = r,\ |lm| = |mn|\sin(\theta) = r\Delta\theta$$
$$|pq| = R\Delta\theta_n,\ |sq| = |pq|\sin(\theta_n),\ |sq| = |mn|$$

$$\Delta\theta_n = \frac{r\Delta\theta}{R\sin(\theta_n)\sin(\theta)} \tag{18}$$

Where, $R = \sqrt{R'^2 + d^2}$ and d is the distance of the section plane from the origin 'o' along the Y axis (oC) (see figure 17). Furthermore, using $z = r\cos(\theta)$, we obtain $\theta = \cos^{-1}(z/R)$.

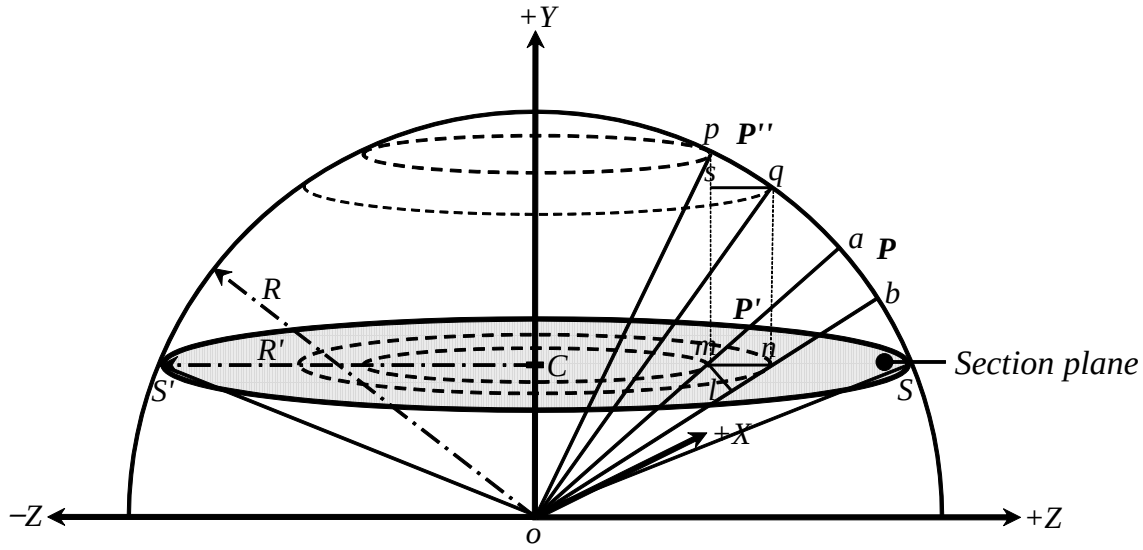


Figure 17: Computing $\Delta\theta_n$ and θ_n for measuring blindspots map on XZ plane.

3.3.2.3 Computing $\Delta\phi_n$ and ϕ_n

Observing the following from figure 18, we obtain equation 19.

$$\angle a'oX = \phi,\ \angle a'ob' = \Delta\phi,\ \angle p'oX = \phi_n,\ \angle p'oq' = \Delta\phi_n$$
$$|om'| = |om|\sin(\theta) = r\sin(\theta),\ |l'm'| = |m'n'|\sin(\phi) = r\sin(\theta)\Delta\phi$$
$$|p'q'| = R\sin(\theta_n)\Delta\phi_n,\ |s'q'| = |p'q'|\sin(\phi_n),\ |s'q'| = |m'n'|$$

$$\Delta\phi_n = \frac{r\sin(\theta)\Delta\phi}{R\sin(\theta_n)|\sin(\phi_n)|.|\sin(\phi)|} \tag{19}$$

ϕ_n is computed by noting that the projection of point m' on the sphere is point p'. The point p' has the same x and z coordinates as m' but a different y coordinate.

Let the y coordinate of point be called as y_n. Then, $y_n = sign(y)\sqrt{R^2 - x^2 - z^2}$ and $\phi_n = \cos^{-1}(x/\sqrt{(x^2 + z^2)})$.

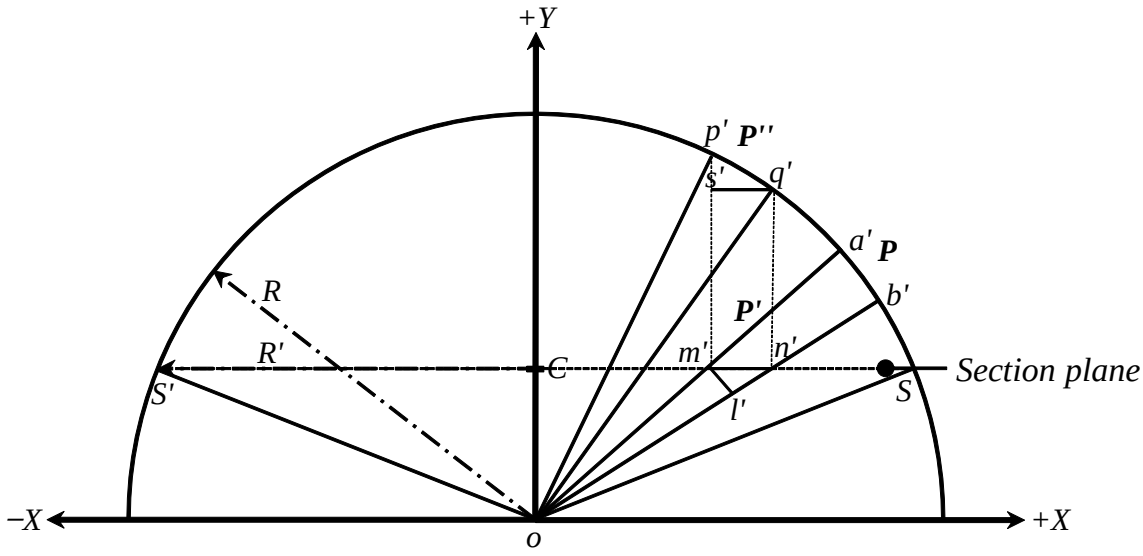


Figure 18: Computing $\Delta\phi_n$ and ϕ_n for measuring blindspots map on XZ plane.

Using equation 15 in equation 16, ΔA_P can be evaluated as:

$$\begin{aligned}
\Delta A_p &= R^2 \sin(\theta_n)\Delta\theta_n\Delta\phi_n \cos(\beta) \\
&= R^2 \sin(\theta_n) \cos(\beta)\frac{r\Delta\theta}{R\sin(\theta_n)\sin(\theta)}\frac{r\sin(\theta)\Delta\phi}{R\sin(\theta_n)|\sin(\phi_n)|.|sin(\phi)|} \\
&= \cos(\beta)\frac{r^2\Delta\theta\Delta\phi}{sin(\theta_n)|sin(\phi_n)|.|sin(\phi)|} \qquad (20)
\end{aligned}$$

The technique to compute the blindspots region on a circular section plane parallel to the XZ plane was described above. A similar technique can be employed to compute the blindspots region on circular section planes parallel to YZ plane. Computing the blindspots on section planes parallel to XY plane is simpler as only the elevation angle θ changes due to projection on to the sphere. Therefore, only θ_n and $\Delta\theta_n$ need to be computed. Furthermore, $\Delta\phi_n = \Delta\phi$ and $\phi_n = \phi$. The method to compute $\Delta\theta_n$ and θ_n for blindspots map parallel to XY plane is described next.

3.3.2.4 Computing $\Delta\theta_n$ and θ_n

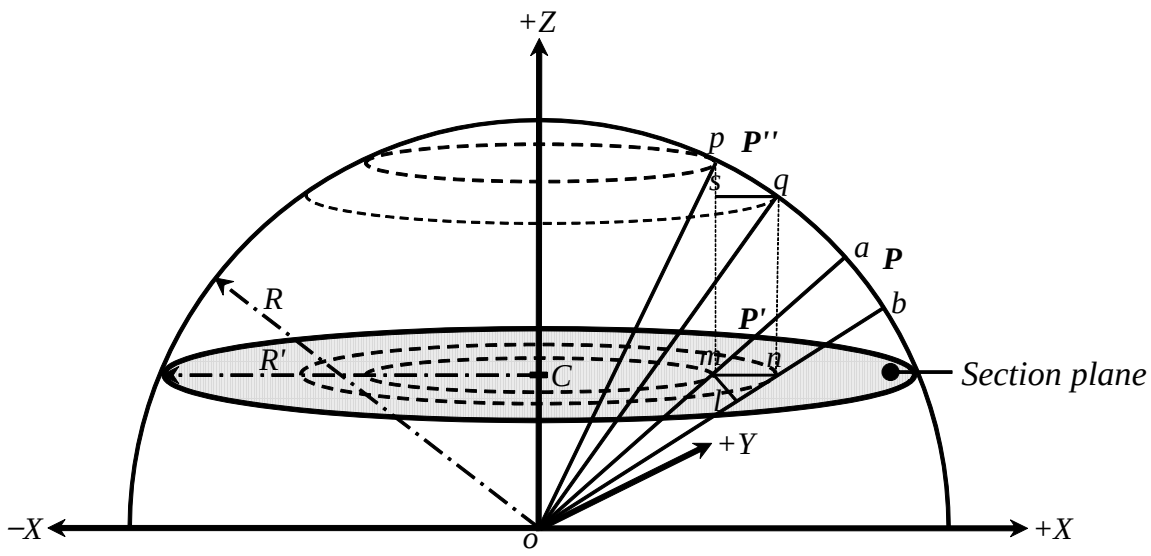


Figure 19: Computing $\Delta\theta_n$ and θ_n for measuring blind spots map on XY plane.

Observing the following from figure 19, we obtain equation 21.

$$\angle aoZ = \theta,\ \angle aob = \Delta\theta,\ \angle poZ = \theta_n,\ \angle poq = \Delta\theta_n$$

$$|om| = r,\ |lm| = |mn|\cos(\theta) = r\Delta\theta$$

$$|pq| = R\Delta\theta_n,\ |sq| = |\cos(\theta_n),\ |sq| = |mn|$$

$$\Delta\theta_n = \frac{r\Delta\theta}{R\cos(\theta_n)\cos(\theta)} \tag{21}$$

θ_n is computed by noting that projection of the point m on the sphere is the point p. The point p has the same x and y coordinates as m but a different z coordinate. Let the z coordinate of point be called as z_n. Then, $z_n = sign(z)\sqrt{R^2 - x^2 - y^2}$ and $\theta_n = \cos^{-1}(z_n/R)$.

3.3.3 12m Circumference Visibility

The 12m *circumference visibility* measurement is similar to the blindspots map measurement on XY plane as discussed in section 3.3.2. However in this case, visibility

is measured only along the edge of a 12m radius circle. The circle is located on the ground plane (parallel to the XY plane) with the operator at the center of the circle. The visibility is measured in terms of length. To measure the visibility, ray casting is performed with the value of θ fixed (value corresponds to the elevation angle of the ground plane) and ϕ, the azimuth, changed from 0 to 360^o in steps of $\Delta\phi$. From section 3.3.1, it is known that ϕ is discretized into $nBins_{Phi}$ steps. In ray casting for computing blindspots map on XY plane (with value of θ equal to elevation angle of ground plane), say N_{CV} rays hit "blocking" bins, then the edge of 12m radius circle corresponding to these N_{CV} rays form the blindspots. Thus, the visible length on this 12m radius circle can be measured using equation 22.

$$Circumference\ Visibility = \left(1 - \frac{N_{CV}}{nBins_{Phi}}\right) \times 2\pi \times 12.0 \tag{22}$$

3.3.4 Rectangular 1m Boundary Visibility

For *rectangular* 1m *boundary visibility* analysis, visibility is measured along the circumference of a rectangular 1m boundary around the equipment. A rectangular 1m boundary is constructed at an offset distance of 1m from the smallest rectangle that can be placed around the vertical projection of the equipment on the test floor (ground level) on which the equipment is located (see figure 20) [36]. From implementation perspective it is not possible to compute the visibility strictly along the rectangular 1m boundary, owing to discretization of the 3D space. Therefore, two rectangles are constructed around the rectangular 1m boundary at a distance of $w/2(w = 1.5\Delta r)$ on the inner and outer side of the boundary. Visibility is then computed over the enclosed region. In figure 20 this enclosed region is the area occupied between the two dashed rectangles. Let the visible area be represented by A_{RB} and let the visible length along this rectangular boundary be denoted by L_{RB}. Therefore, L_{RB} can be computed as A_{RB}/w. The visible area A_{RB} is computed using the same approach as

is used to compute blindspots map on XY plane.

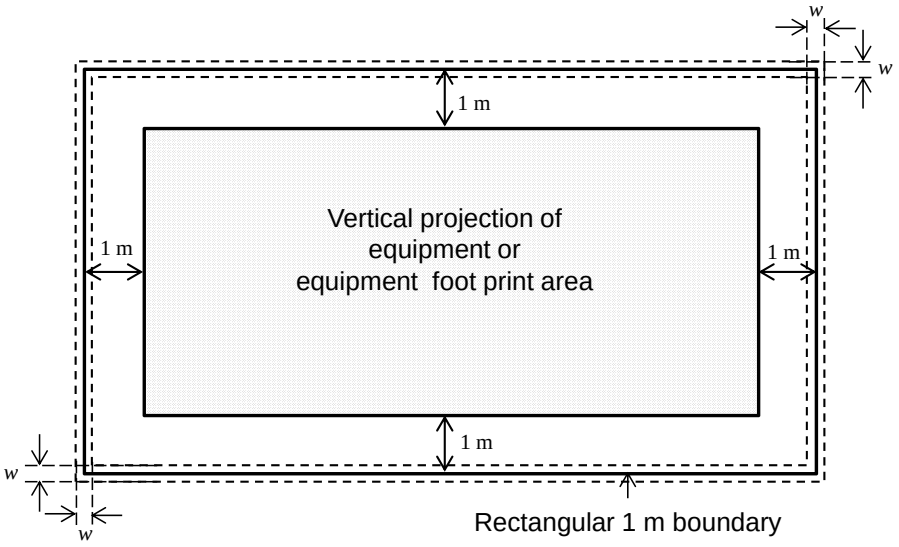


Figure 20: Illustration of rectangular 1m boundary visibility (XY plane).

3.3.5 Worker Visibility

To compute worker visibility, a worker is represented by a curved surface unlike in [65] where a worker is represented by a cylinder. The representation by a curved surface simplifies the computation as Ray Casting is performed in spherical coordinate system. Figure 21 illustrates this representation. The height and the width of the surface are at the discretion of the user. Based on this representation, worker visibility is defined as the percentage of the visible curved surface area to the total area of the curved surface. Using Ray Casting, the visible and blind areas on this curved surface are obtained from which worker visibility is finally computed.

Figure 22 illustrates the process of computing worker visibility by Ray Casting algorithm. The gray points in the figure (left side) represent the point cloud data of an equipment. In the figure on the right side the visible and invisible regions of the worker are represented by green and red pixels respectively.

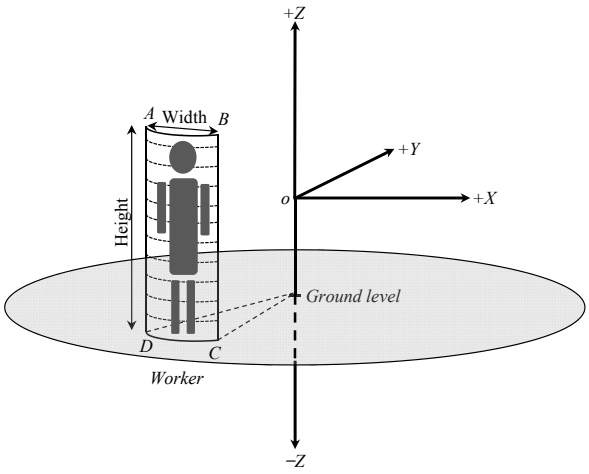


Figure 21: Worker representation. Ground level is parallel to XY plane.

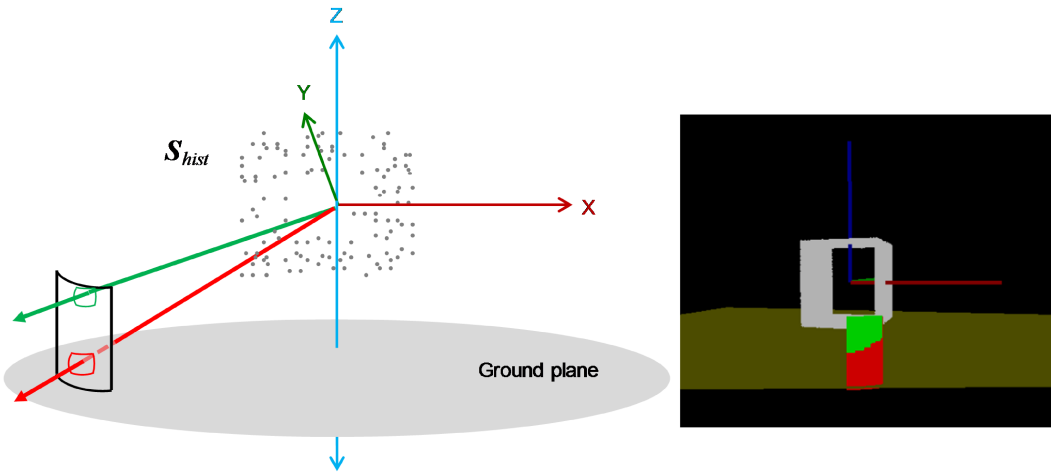


Figure 22: Illustration of Ray Casting algorithm for worker visibility computation.

3.4 Data Sets

3.4.1 Synthetic Data Sets

To validate the approach, 36 synthetic point clouds were generated. The data set is based on three hollow geometries: cube, cylinder, and sphere. Each of these geometries is longitudinally cut to generate four different geometries. These longitudinal cuts have been illustrated in figure 23. This resulted in 12 point clouds. Gaussian noise with σ= 5 and 10mm is then added to this synthetic data set to generate 24 more point clouds as illustrated in figure 4. The choice of Gaussian noise is based on the "*ranging noise (defined as a standard deviation of values about the best-fit plane)*" of the commercial laser scanner used for scanning. For each of these data sets the ground truth is trivial and is known a-priori. The ground truth for each analysis is mathematically and graphically illustrated and then compared against the results of computations in the sections below.

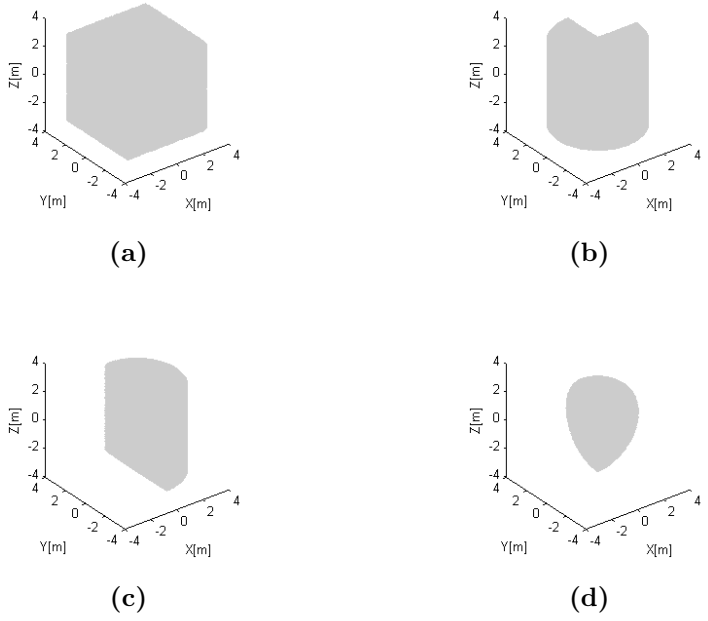


Figure 23: Closed cube, 1/4 longitudinally cut cylinder, 1/2 longitudinally cut cylinder, 3/4 longitudinally cut sphere.

Table 4: Synthetic data set: σ (mm) represents the standard deviation of added Gaussian noise.

Cuts	Closed	1/4 cut	1/2 cut	3/4 cut
Cube	σ =0, 5, 10	σ =0, 5, 10	σ =0, 5, 10	σ =0, 5, 10
Cylinder	σ =0, 5, 10	σ =0, 5, 10	σ =0, 5, 10	σ =0, 5, 10
Sphere	σ =0, 5, 10	σ =0, 5, 10	σ =0, 5, 10	σ =0, 5, 10

3.4.2 Real-World Data Set

The real world data set is a forklift's point cloud [6]. The data set was generated by registering 11 scans together. Point cloud generated by registering multiple laser scans typically captures the surface of equipment more accurately than one scan taken from the inside of the equipment cabin.

3.5 Results and Discussion

3.5.1 Validation on Synthetic Data Set

The blindspots computation on the synthetic data set is essentially affected by two factors (a) step sizes (or bin widths): Δr, $\Delta\theta$, $\Delta\phi$ and (b) noise. The error rate is monotonic function of the step sizes. Increase in step sizes results in reduction in "time to solution" and a decrease in step size requires more memory be allocated for storing the S^{hist} matrix and increases the accuracy. Owing to the memory constraints, the step sizes are set to the following minimal possible values: $\Delta r = 0.05m$, $\Delta\theta = 0.3°$, $\Delta\phi = 0.3°$. In the next sections, the accuracy of the algorithm is analyzed for the addition of random Gaussian noise to the synthetic point cloud data set. Finally, the results of the algorithm's performance are presented which establish its accuracy and speed.

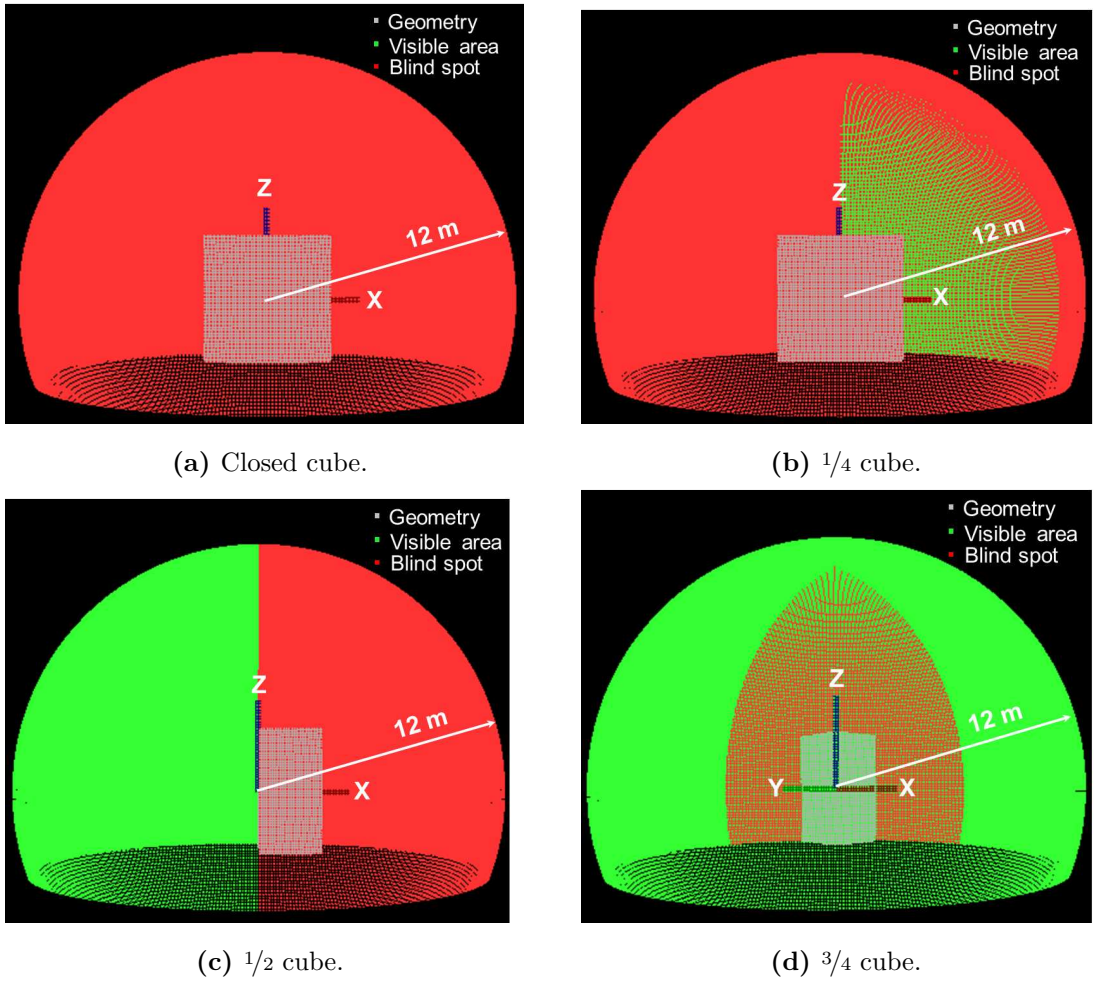
3.5.1.1 Volumetric Blindspots

In Table 5, computed volumetric blindspots are shown in rows 3 to 5. The ground truth for each data set is in the last row. The percentage volumetric blindspots was computed using the approach discussed in section 3.3.1.

Table 5: Computed volumetric blindspots and the ground truth for the synthetic data set.

Cut	Closed (%)			1/4 cut			1/2 cut			3/4 cut		
Noise (σ in mm)	0.00	5.00	10.00	0.00	5.00	10.00	0.00	5.00	10.00	0.00	5.00	10.00
Cube	99.73	99.77	99.66	74.67	74.77	74.67	50.03	50.07	50.01	25.10	25.14	25.12
Cylinder	99.93	99.90	99.85	74.95	74.90	74.84	50.13	50.13	50.11	25.15	25.17	25.16
Sphere	100.0	99.90	99.84	74.98	74.89	74.84	50.17	50.14	50.11	25.17	25.16	25.16
Ground truth	100.00			75.00			50.00			25.00		

A maximum error rate of 0.34% was observed for closed cube synthetic data with $\sigma = 10$mm. The generated point cloud data after volumetric blindspots analysis is shown in figure 24. The volumetric blindspots are 100%, 75%, 50% and 25% respectively. The reader should be convinced that the visibility (green region) is due to the longitudinal cuts in the geometries.



(a) Closed cube. **(b)** 1/4 cube.

(c) 1/2 cube. **(d)** 3/4 cube.

Figure 24: Illustration of volumetric blindspots for cube ($\sigma = 0$) with different longitudinal cuts. The %age volumetric blindspots is 100%, 75%, 50% and 25%.

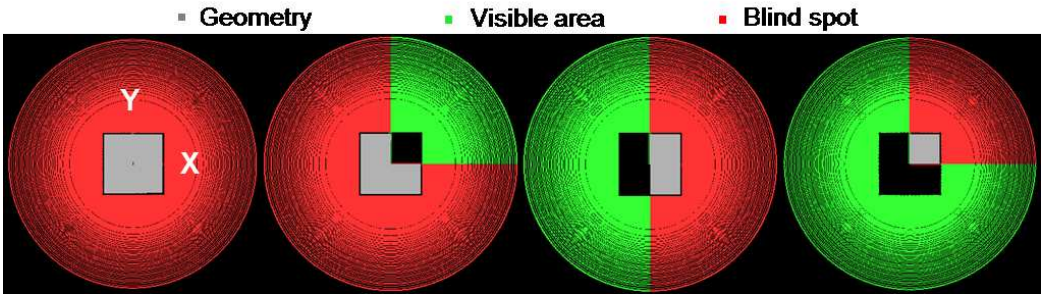
3.5.1.2 Blindspots Map

The blindspots map were computed on three planes $x = 5.0$m, $y = 5.0$m, and $z = 5.0$m. Table 6 reports the ground truth of the blindspots map for the 36 point clouds. The percentage of blindspots is based on the discussion in section 3.3.2. In figure 25 the ground truth of the blindspots map has been illustrated graphically. Tables 7, 8, and 9 show the computed blindspots percentage for $\sigma = 0$, 5 and 10mm respectively. Comparing the values of the three tables it can be observed that the computed blindspots percentage is not significantly affected by noise. A maximum

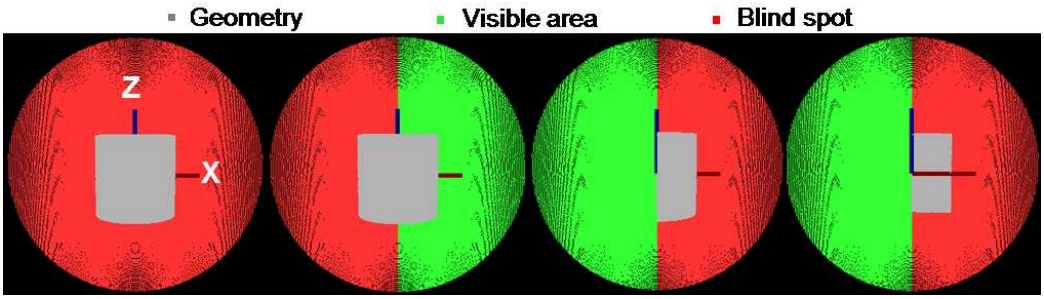
error of 0.21% is observed for the XZ blindspots map on 1/2 cut and 3/4 cut data sets. As the 12m circumference visibility is evaluated during the computation of the blindspots map on the XY plane, therefore no additional validation results for 12m circumference visibility analysis have been reported.

Table 6: Ground truth of %age blindspots on blindspots map for the 36 point cloud data sets.

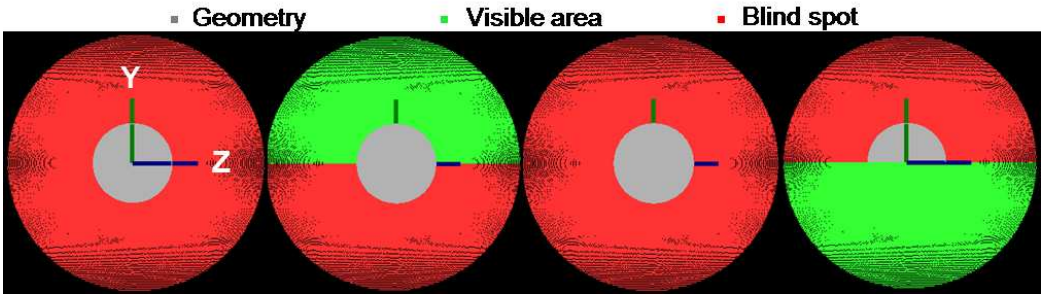
Geometry	Map location (m)	Closed (%)	1/4 cut (%)	1/2 cut (%)	3/4 cut (%)
	$z = 5.0$	100.00	75.00	50.00	25.00
Cube	$y = 5.0$	100.00	50.00	50.00	50.00
	$x = 5.0$	100.00	50.00	100.00	50.00
	$z = 5.0$	100.00	75.00	50.00	25.00
Cylinder	$y = 5.0$	100.00	50.00	50.00	50.00
	$x = 5.0$	100.00	50.00	100.00	50.00
	$z = 5.0$	100.00	75.00	50.00	25.00
Sphere	$y = 5.0$	100.00	50.00	50.00	50.00
	$x = 5.0$	100.00	50.00	100.00	50.00



(a) XY blindspots map of cube ($\sigma = 0$) at z=5.0m with different longitudinal cuts.



(b) XZ blindspots map of cylinder ($\sigma = 0$) at y=5.0m with different longitudinal cuts.



(c) YZ blindspots map of sphere ($\sigma = 0$) at x=5.0m with different longitudinal cuts.

Figure 25: Illustration of ground truth of the blindspots map for the three geometries: (a) cube, (b) cylinder, and (c) sphere. 1st column - closed, 2nd column -1/4 cut, 3rd column - 1/2 cut, 4th column - 3/4 cut).

Table 7: Without noise (σ = 0mm)

Geometry	Map location (m)	Closed (%)	1/4 cut (%)	1/2 cut (%)	3/4 cut (%)
	$z = 5.0$	100.00	75.01	50.09	25.09
Cube	$y = 5.0$	100.00	50.00	50.21	50.21
	$x = 5.0$	100.00	50.00	100.00	50.00
	$z = 5.0$	100.00	75.01	50.09	25.09
Cylinder	$y = 5.0$	100.00	50.07	50.21	50.21
	$x = 5.0$	100.00	50.00	100.00	50.00
	$z = 5.0$	100.00	75.00	50.09	25.09
Sphere	$y = 5.0$	100.00	50.04	50.21	50.21
	$x = 5.0$	100.00	50.00	100.00	50.00

Table 8: With Gaussian noise (σ =5.0mm).

Geometry	Map location (m)	Closed (%)	1/4 cut (%)	1/2 cut (%)	3/4 cut (%)
	$z = 5.0$	100.00	75.01	50.09	25.09
Cube	$y = 5.0$	100.00	50.05	50.20	50.19
	$x = 5.0$	100.00	50.00	100.00	50.00
	$z = 5.0$	100.00	75.01	50.09	25.09
Cylinder	$y = 5.0$	100.00	50.07	50.21	50.21
	$x = 5.0$	100.00	50.00	100.00	50.00
	$z = 5.0$	100.00	75.01	50.09	25.09
Sphere	$y = 5.0$	100.00	50.07	50.21	50.20
	$x = 5.0$	100.00	50.00	100.00	50.00

Table 9: With Gaussian noise (σ =10.0mm).

Geometry	Map location (m)	Closed (%)	1/4 cut (%)	1/2 cut (%)	3/4 cut (%)
	$z = 5.0$	100.00	75.01	50.09	25.09
Cube	$y = 5.0$	100.00	50.04	50.18	50.18
	$x = 5.0$	100.00	50.00	100.00	50.00
	$z = 5.0$	100.00	75.00	50.09	25.09
Cylinder	$y = 5.0$	100.00	50.07	50.21	50.21
	$x = 5.0$	100.00	50.00	100.00	50.00
	$z = 5.0$	100.00	75.01	50.08	25.08
Sphere	$y = 5.0$	100.00	50.06	50.20	50.20
	$x = 5.0$	100.00	50.00	100.00	50.00

3.5.1.3 Rectangular 1m Boundary Analysis

The rectangular 1m boundary visibility analysis is computed on the plane z=5.0m. Table 10 shows the ground truth percentage for the point clouds. The percentage visibility is based on the discussion in section 3.3.4. The ground truth has been illustrated graphically for cube point cloud ($\sigma = 0$) with different longitudinal cuts. The percentage visibility in 26 is 0%, 25%, 50% and 75%, respectively. Tables 11, 12 and 13 show the computed visibility for σ = 0, 5 and 10mm, respectively. A maximum error of 0.07% is observed for 1/2 cut and 3/4 cut point clouds.

Table 10: Ground truth for rectangular 1m boundary visibility for the 36 point cloud data sets.

Geometry	Map location (m)	Closed (%)	1/4 cut (%)	1/2 cut (%)	3/4 cut (%)
Cube	$z = 5.0$	0.00	25.00	50.00	75.00
Cylinder	$z = 5.0$	0.00	25.00	50.00	75.00
Sphere	$z = 5.0$	10.00	25.00	50.00	75.00

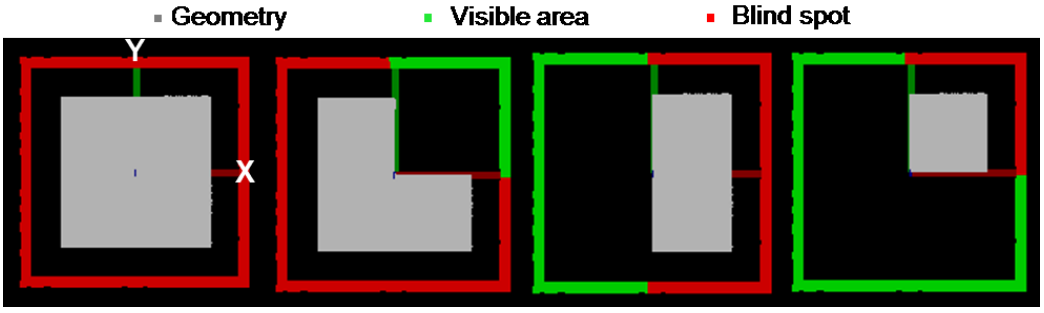


Figure 26: Illustration of the rectangular 1m boundary visibility analysis on cube (σ=0mm) with different longitudinal cuts (left to right: Closed, 1/4 cut, 1/2 cut, and 3/4 cut).

Table 11: Without noise (σ=0mm).

Geometry	Map location (m)	Closed (%)	1/4 cut (%)	1/2 cut (%)	3/4 cut (%)
Cube	$z = 5.0$	0.00	25.00	49.93	74.93
Cylinder	$z = 5.0$	0.00	25.00	49.93	74.93
Sphere	$z = 5.0$	10.00	25.00	49.93	74.93

Table 12: With Gaussian noise (σ=5.0mm).

Geometry	Map location (m)	Closed (%)	1/4 cut (%)	1/2 cut (%)	3/4 cut (%)
Cube	$z = 5.0$	0.00	25.00	49.93	74.93
Cylinder	$z = 5.0$	0.00	25.00	49.93	74.93
Sphere	$z = 5.0$	10.00	25.00	49.93	74.93

Table 13: With Gaussian noise (σ=10.0mm)

Geometry	Map location (m)	Closed (%)	1/4 cut (%)	1/2 cut (%)	3/4 cut (%)
Cube	$z = 5.0$	0.00	25.00	49.93	74.97
Cylinder	$z = 5.0$	0.00	25.00	49.93	74.93
Sphere	$z = 5.0$	10.00	25.00	49.93	74.94

3.5.1.4 Worker Visibility

To validate worker visibility, an arbitrary location $\langle 3.5m,\ 3.5m,\ -4.0m \rangle$ was selected and the computations were verified on the 36 point clouds. In the developed approach, the height and width values are arbitrary and are at the discretion of the user. In the following, the height of the worker was selected to be 1.5m and the width 0.5m. Table 14 shows the ground truth for the above specified location, height and width. Tables 15, 16 and 17 show the computed result on data set for σ = 0, 5 and 10mm respectively. Figure 27 illustrates the visibility of worker of the cube point cloud data for the above specified location, height and width of a worker.

Table 14: Ground truth for worker visibility for the 36 point cloud data sets.

Geometry	Closed (%)	1/4 cut (%)	1/2 cut (%)	3/4 cut (%)
Cube	0.00	100.00	0.00	0.00
Cylinder	0.00	100.00	0.00	0.00
Sphere	0.00	100.00	0.00	0.00

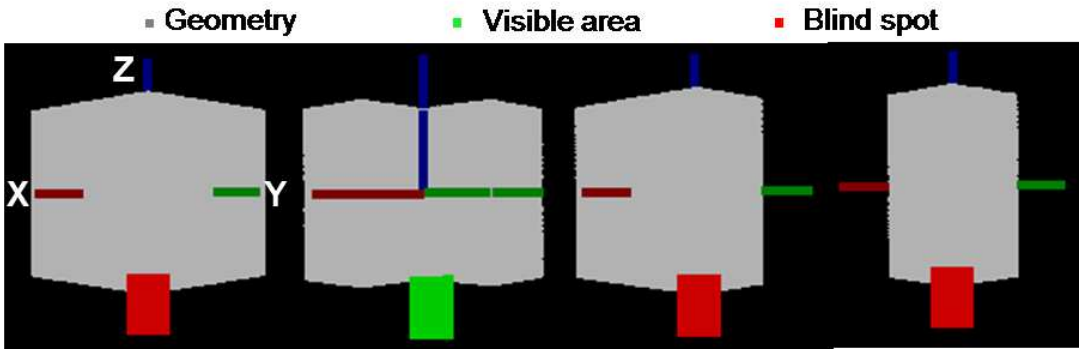


Figure 27: Illustration of worker visibility for cube geometry ($\sigma = 0$) with different longitudinal cuts (left to right: 1/4 cut,1/2 cut, and 3/4 cut).

Table 15: Without noise (σ =0mm).

Geometry	Closed (%)	1/4 cut (%)	1/2 cut (%)	3/4 cut (%)
Cube	0.00	100.00	0.00	0.00
Cylinder	0.00	100.00	0.00	0.00
Sphere	0.00	100.00	0.00	0.00

Table 16: With Gaussian noise (σ =5.0mm).

Geometry	Closed (%)	1/4 cut (%)	1/2 cut (%)	3/4 cut (%)
Cube	0.00	100.00	0.00	0.00
Cylinder	0.00	100.00	0.00	0.00
Sphere	0.00	100.00	0.00	0.00

Table 17: With Gaussian noise (σ =10.0mm).

Geometry	Closed (%)	1/4 cut (%)	1/2 cut (%)	3/4 cut (%)
Cube	0.00	100.00	0.00	0.00
Cylinder	0.00	100.00	0.00	0.00
Sphere	0.00	100.00	0.00	0.00

3.5.1.5 Computational Performance

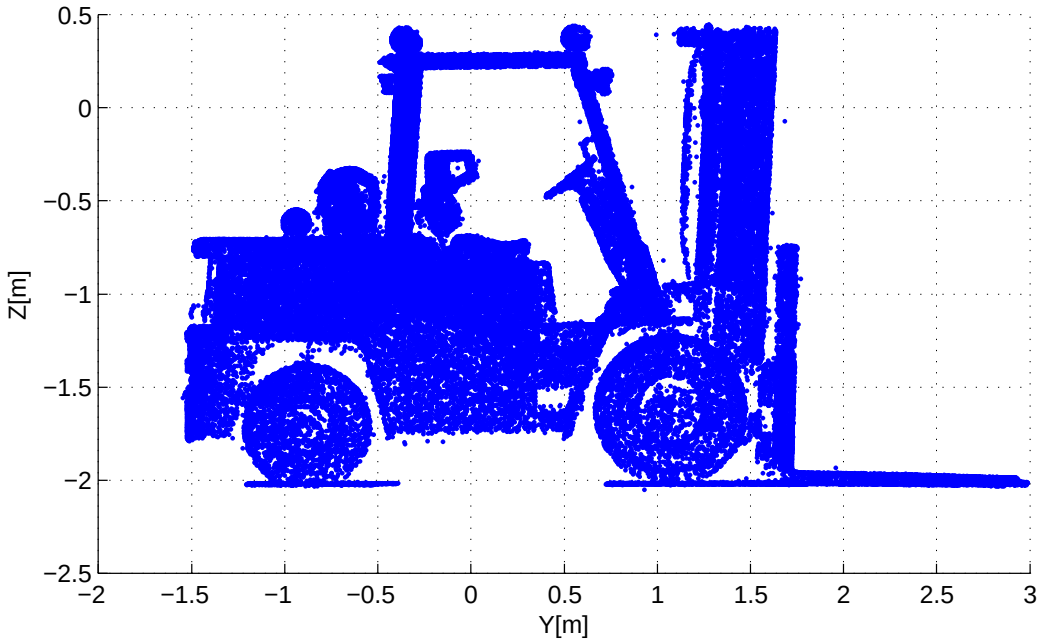
Table 18 illustrates the time taken for computing blindspots or visibility on three synthetic point clouds. Considering the nature of a Ray Casting algorithm, the advantage of having a cache friendly data structure for S^{hist} matrix becomes clear as the computations are significantly fast. The analysis in table 18 does not include time for file input/output (I/O). Furthermore, the computation time for XY blindspots map includes 12m circumference visibility and rectangular 1m boundary visibility analysis.

Table 18: Computational time (in seconds) for analyses.

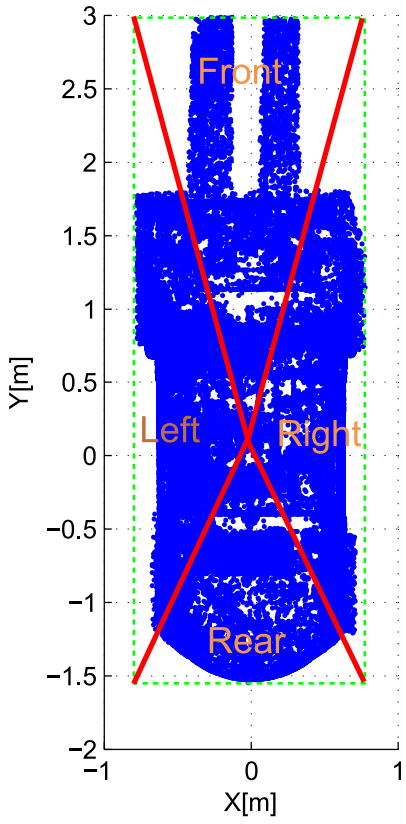
Geometry	# points	Vol. blindspots	Blindspots map			Worker visibility
			XY	YZ	XZ	
Cube	13×10^6	0.160	0.550	0.534	0.523	0.016
Cylinder	17×10^6	0.175	0.546	0.531	0.523	0.016
Sphere	16×10^6	0.187	0.542	0.531	0.519	0.020

3.5.2 Real-world (Forklift) Data Set

The forklift point cloud data set was obtained by registering 11 laser scans. The top and side views of the raw point cloud is shown in figure 28. The point cloud was binned into a 3D grid in steps of size Δr=0.05m, $\Delta\theta = 0.3^\circ$ and $\Delta\phi = 0.3^\circ$. The numbers of bins along the three directions were: 416 along r, 1200 along ϕ and 600 along θ. The number of bins was computed from the value (step sizes) the user defined. Owing to memory constraints for storing the 3D grid, the step sizes were set to the above minimal possible values.



(a)

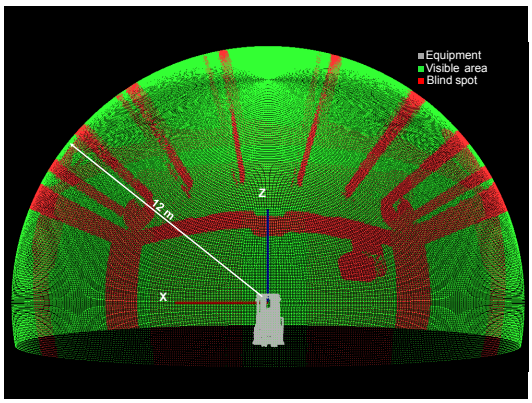


(b)

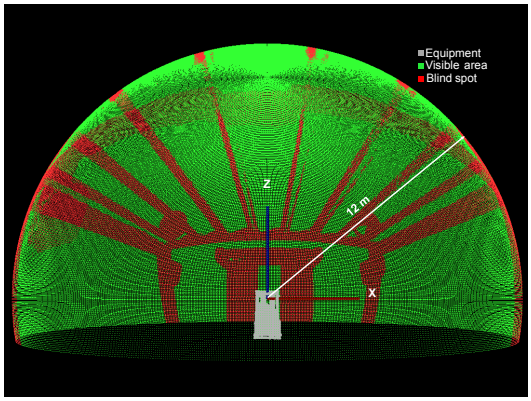
Figure 28: Point cloud of forklift generated by registering multiple scans and categorization of area surrounding forklift into: front, left, rear and right.

3.5.2.1 Volumetric Blind Spots

For the forklift shown in figure 28, the volumetric blindspots are illustrated graphically in figure 29. Figure 29 shows the visible and blind areas on a 12m sphere centered at the origin of the equipment cabin. The visible areas are shown in green whereas the blind areas are represented by red color. The percentage volumetric blindspots was 19.48%. The time taken for computing the volumetric blindspots was 1.19 seconds (includes file output).



(a) Front view



(b) Rear view

Figure 29: Volumetric blindspots of a forklift on a 12m radius sphere.

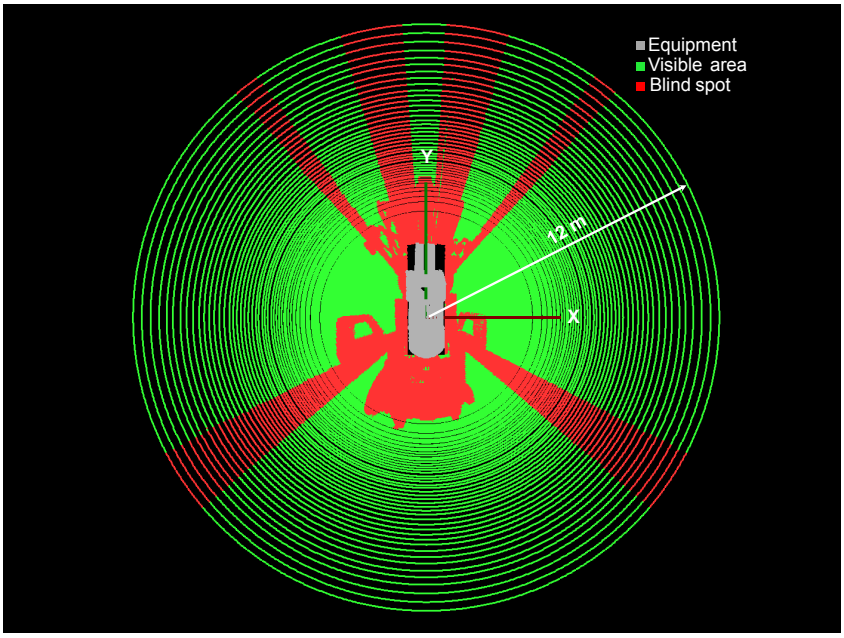
3.5.2.2 Blindspots Map

The percentage blindspots (on XY plane) was computed to be 21.20%. The time taken for computation was measured to be 0.83 seconds (includes file I/O and computation of 12m circumference visibility and rectangular 1m boundary). Table 19 shows the detailed results of the analysis. The visible area inside the circle was computed to be $360.15m^2$ (3^{rd} row).

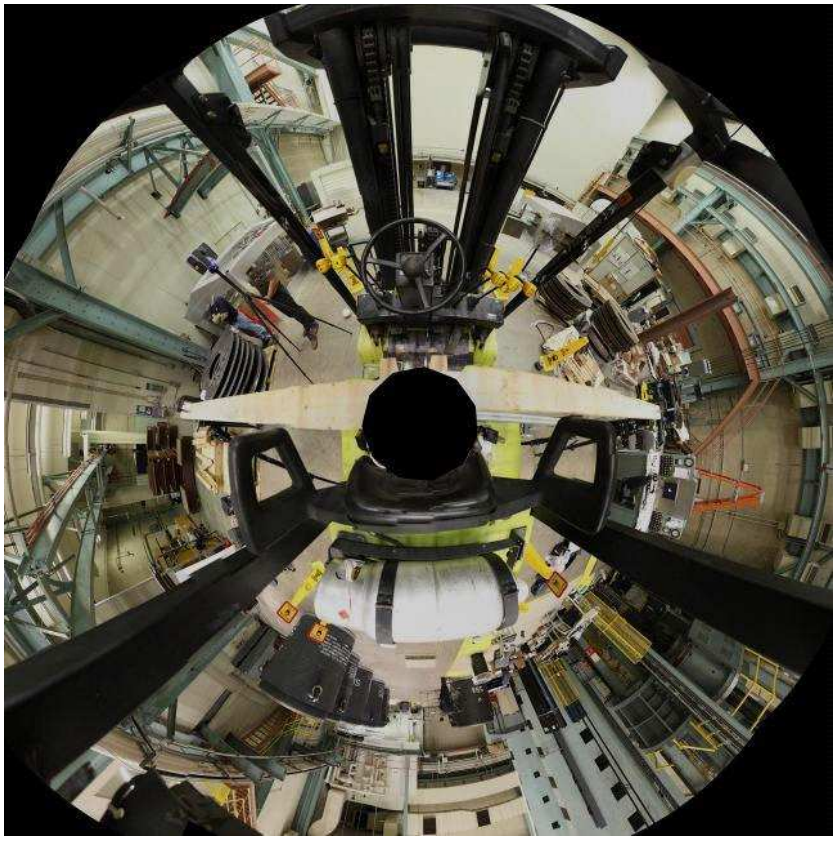
Table 19: Blindspots map measurement.

Entity	Area (m^2)	Ratio (%)
Total area	456.86	100
Blindspots area	96.71	21.20
Visible area	360.15	78.83
Front blindspots	27.08	5.93
Right blindspots	28.90	6.33
Rear blindspots	7.13	1.56
Left blindspots	33.76	7.39

The circular area was divided in to four regions: front, right, rear and left as shown in figure 28b and the blindspots area in these four regions are shown in Table 19 (bottom four rows). Figure 30a shows the blind and visible areas contained in the 12m radius circle lying on the ground plane, with the operator position at the center. To visually validate the blindspots map, a Giga Pan camera was mounted inside the cabin to capture a panoramic view of the forklift [6]. The similarity of between the blindspots map can be observed. The red pixels (blindspots regions) in figure 30a are caused due to the occlusion caused by the black structural components and the tank (at the rear) of the forklift as shown in figure 30b.



(a) Graphical illustration of blindspots map on XY plane.



(b) Panoramic view of the forklift.

Figure 30: Blindspots map of the forklift on the ground plane.

3.5.2.3 Rectangular 1m Boundary Analysis

Figure 31 shows the rectangular 1m boundary visibility surrounding the forklift. The rectangular 1m boundary visibility measurement is shown in Table 20. The actual length was computed by manual calculation whereas the computed value was reported by the developed blindspots measurement method. The discrepancy in value arises from the fact that the 3D space is discretized. The visible length was computed to be 9.66m which constituted 46.78% of the length of the rectangular 1m boundary.

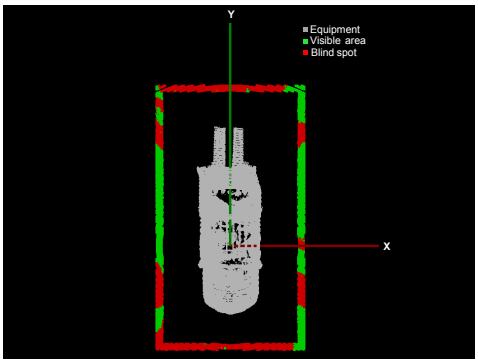


Figure 31: Visibility along the rectangular 1m boundary.

Table 20: Visibility along the rectangular 1m boundary.

Entity	Length (m)	Ratio (%)
Total (actual)	20.32	-
Total (computed)	20.65	100
Visible	9.66	46.78
Front blindspots	2.54	12.28
Right blindspots	2.32	11.21
Rear blindspots	2.59	12.54
Left blindspots	3.55	17.20

3.5.2.4 12m Circumference Visibility

The *12m circumference visibility* was computed to be 62.71m (83.17%). The total length of the circumference ($2\pi r$, r = 12.0m) was 75.40m. Additionally, the developed method reports the arcs along the circumference that are invisible as shown in Table 21. Figure 32 below is an annotated graphical representation of the arcs in Table 21.

Table 21: Invisible arcs along the circumference of 12m radius circle.

Arc #	From (°)	To (°)	Arc length (m)	Arc angle (m)
1	50.10	55.20	1.07	5.10
2	73.80	86.10	2.58	12.30
3	94.80	107.10	2.58	12.30
4	125.40	130.50	1.07	5.10
5	207.60	220.50	2.70	12.90
6	319.80	332.70	2.70	12.90

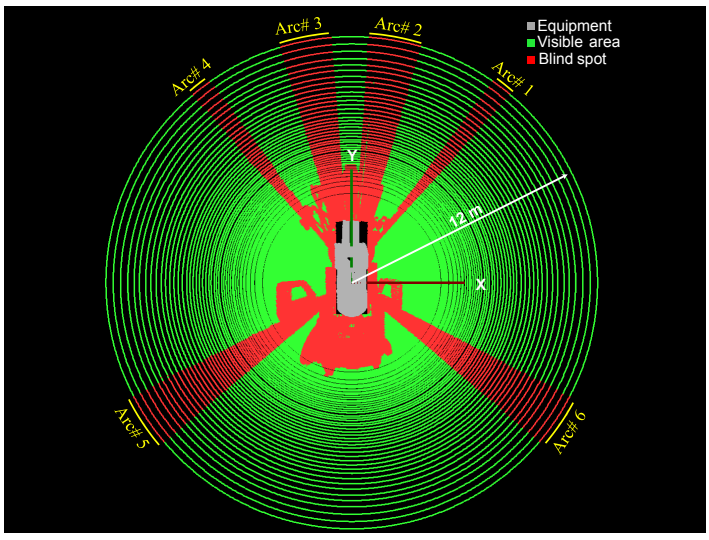
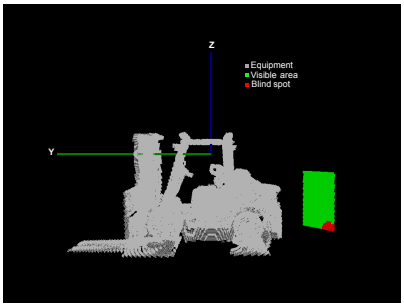


Figure 32: Graphical illustration of 12m circumference visibility.

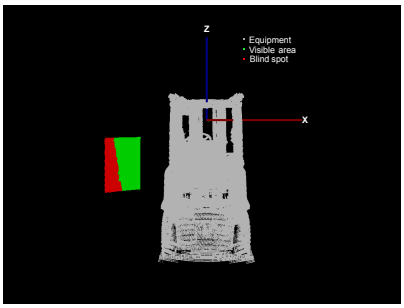
3.5.2.5 Worker Visibility

Two random worker locations L_1 and L_2 were selected with $L_1 = \langle 2.0m, -3.0m, -2.02m \rangle$, and $L_2 = \langle -2.5m, 4.0m, -2.02m \rangle$. The two locations had z=-2.02m which was the ground plane. Figure 33, graphically illustrates the visibility at these two locations.

The height and width values are arbitrary and are at the discretion of the user. In the results shown below the height and width were assumed to be 1.5m and 0.5m respectively. The percentage visibility at L_1 was 95.67% and 66.80% at L_2. The time taken for computing visibility at each of these locations was 0.033 sec (including file I/O).



(a) Worker located at L_1



(b) Worker located at L_2

Figure 33: Graphical illustration of worker visibility at two arbitrary locations L_1 and L_2.

CHAPTER IV

RESULTS

In Chapter 2 results to head posture estimation was presented and Chapter 3 presented results to computing static blindspots. In this chapter, dynamic blindspots map is formulated by integrating the head posture of the driver or equipment operator with the static blindspots map. This results in measuring the visibility of the driver in real-time. It should be noted that the visibility measurements presented here assume a monocular vision instead of binocular vision to simplify computation and only direct visibility is taken into consideration. Indirect visibility due to mirrors is not taken into consideration. To extend this approach to allow for a binocular vision model and to incorporate indirect visibility of the operator a brief discussion has been provided in Chapter 5.

In Chapter 2 the BKHP dataset was used to train Random Forests algorithm. In that data set, the reference head location is the nose tip. However, owing to the monocular vision model assumption, this approach requires the temple (located midway between the eyes) to be the reference point on the head. Therefore, a data set was created using markers to capture the ground truth such that the temple is the reference point on the head. The results presented in this chapter were obtained after training Random Forests algorithm on this new data set.

To demonstrate the feasibility of the approach results to simulation and field experiments are presented. Simulation experiment was conducted in an indoor environment with the camera mounted in front of the subject. Field experiments were conducted with a driver driving a commercial passenger car with a kinect camera

mounted on the dashboard. Before presenting results to dynamic blindspots measurement, first we need to discuss the required camera registration to integrate head posture information and static blindspots map to construct dynamic blindspots map. Camera registration is necessary because the estimated head pose is in the kinect camera coordinate system, however visibility measurement of the driver is performed in vehicle coordinate system. Thus, the estimated head pose needs to be transformed to the vehicle coordinate system to construct the dynamic blindspots or visibility maps.

4.1 Camera Registration

As discussed above the head posture of the driver (the head location is denoted by $\boldsymbol{l}_k^c$ and the orientation is denoted by the euler angles $\boldsymbol{\theta}_k^c$) is estimated in the camera coordinate system. The superscript 'c' denotes camera coordinate system and subscript 'k' denotes the frame number. The camera coordinate system is located at the optical center of the kinect depth camera as shown in figure 34. However, the visibility measurement of the driver is performed in vehicle coordinate system (denoted by superscript 'v'). The origin of the vehicle coordinate system is located at a reference point on the driver's head and it changes as the driver's head moves. This reference location is taken to be the temple. Thus, the translation of the vehicle coordinate system's origin, denoted by t_k^v, is simply the vector joining the head location in the 0^{th} frame to the head location in the k^{th} frame, where the head location measurements are in vehicle coordinate system. The axes of the vehicle coordinate system are aligned with the principal axes of the car as shown in figure 43, at all instants of time irrespective of the head orientation. Thus, only the rotation between the camera coordinate system and the vehicle coordinate system is required to compute dynamic blindspots map. It is also assumed here that the origin of the equipment point cloud is at the same position as the reference location on the driver's head in the first frame

where the head location measurement is in the vehicle coordinate system. For the purposes of illustration of result, this assumption is not restrictive. However, for a real-world application it will be necessary to ensure that the driver's head location is located exactly at the same position as the input point cloud data of the equipment/vehicle. This can be achieved by using Iterative Closest Point (ICP) algorithm on the laser scan point cloud taken from inside the equipment and the point cloud generated by the depth camera. This approach will yield an accurate estimate of the rotation and translation parameters to register the camera with the point cloud of the equipment. Algorithm 3 below illustrates computation of dynamic blindspots using the input point cloud P_{in} and the driver's head posture. Additionally, since our interest is to compute visibility on the ground plane thus we only take in to account the rotation of camera coordination system around the Z-axis of the vehicle coordinate system as illustrated in figure 35.

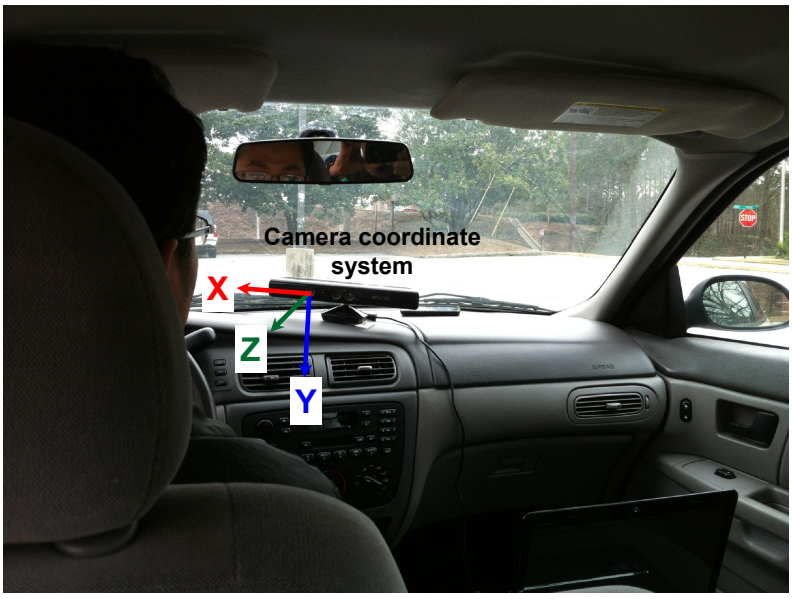


Figure 34: Camera coordinate system of the kinect mounted of the dashboard of a passenger car.

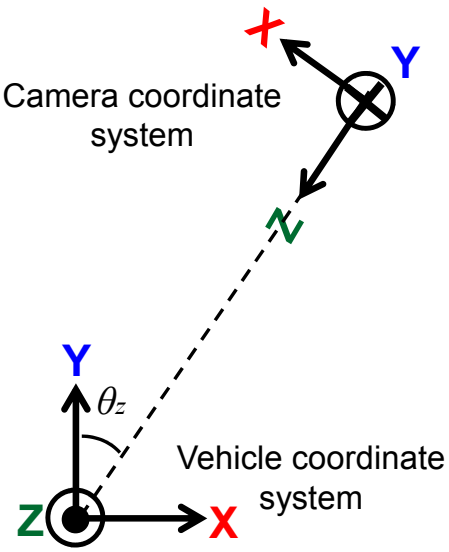


Figure 35: Rotation between camera coordinate system in vehicle coordinate system.

Algorithm 3 Computing dynamic blindspots

1: Convert input point cloud P_{in} to a 3D histogram S^{hist}

2: Convert S^{hist} to coarser point cloud P_0

3: **while** $k < nFrames$ **do**

4: Compute head pose parameters: $\boldsymbol{l}_k^c$ and $\boldsymbol{\theta}_k^c$

5: $\boldsymbol{l}_k^v = R_c^v \, \boldsymbol{l}_k^c$

6: $t_k^v = \boldsymbol{l}_k^v$ - $\boldsymbol{l}_0^v$

7: $P_k = P_0 + t_k^v$

8: Construct S_k^{hist} from point cloud P_k

9: Compute XY blindspots map

10: $k = k + 1$

11: **end while**

4.2 Simulation Experiment

To simulate dynamic blindspots, a kinect camera was mounted in front of the subject in an indoor environment as shown in the figure 36. The subject was asked to perform different head motions and then the head posture information was used to compute the dynamic blindspots in real-time in a virtual-world. It should be noted here that the subject involved in this experiment was not a part of the new data set used for training the head posture estimation algorithm with temple as the reference location on the head. To compute the dynamic blindspots, the point cloud of the fork lift shown in figure 28 was used as the virtual equipment. As in the simulation environment the subject was not operating any equipment, this allowed it to perform specific head motions in an controlled environment. The main idea behind the simulation experiment is to create dynamic blindspots map in a controlled environment and visually validate the approach. Also, it is assumed that the origin of the input point cloud, P_{in}, is at the same location as the head of the subject in the first frame as discussed in section 4.1. All the depth maps shown have been thresholded with a value of 1.5m to reject the background pixels.

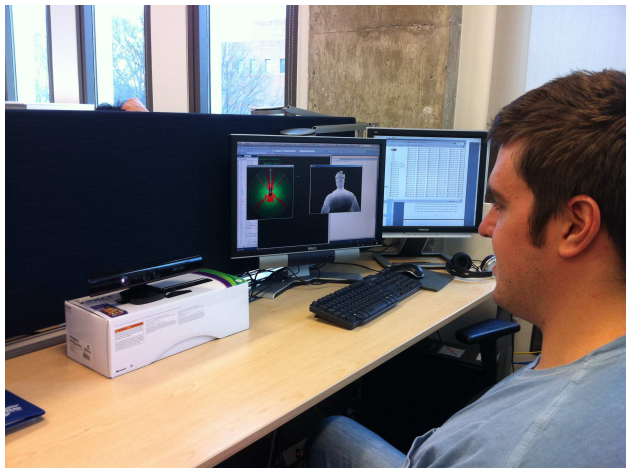


Figure 36: Camera mounted in front of the subject for simulation in an indoor environment.

Figure 37 below shows the visibility and the head pose of the subject in the first frame. The left column shows the visibility map and the column on the right shows the thresholded depth map and the rgb image captured. The gray arrow drawn the visibility represents the head orientation of the subject along the +Z axis (of the vehicle coordinate system). In the subsequent results, visibility maps are presented for some head poses and should be compared with figure 37 to validate the blindspots map generated due to changing head posture.

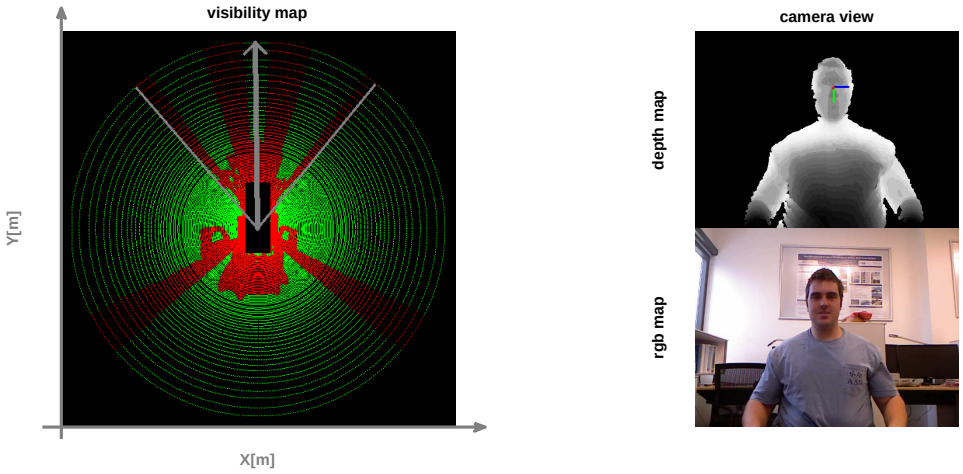
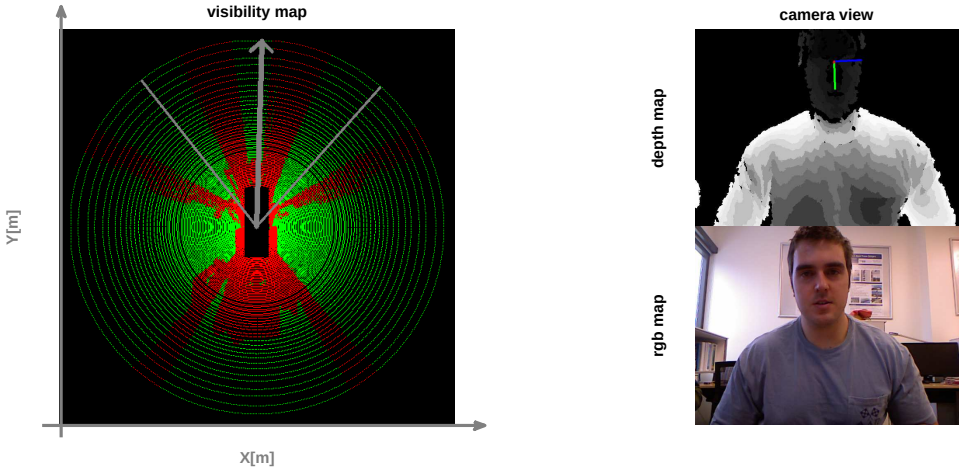
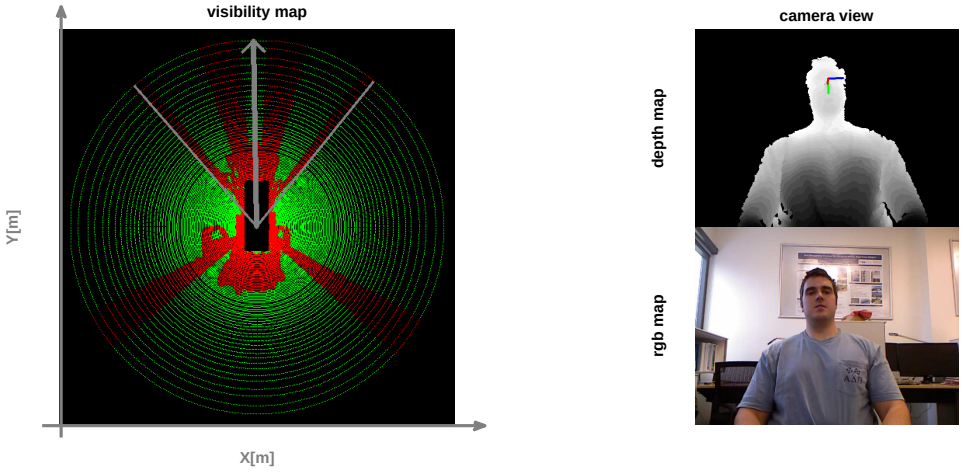


Figure 37: Visibility in the first frame. Green pixels represent visible region and red pixel represent the blindspots region.

- *Head moving forward and backward*: Figure 38a shows the visibility map generated as the head moves forward, that is moving towards the camera along the +Y axis (of vehicle coordinate system). As the head moves forward the visibility in the frontal region increases and decreases in the rear region (compare it with figure 37). Notice how the angle between the red regions increases in the frontal region and decreases in the rear region. As the user moves its head backward (away form the camera) the visibility in the rear region increases and the decreases in the forward region (see figure 38b).



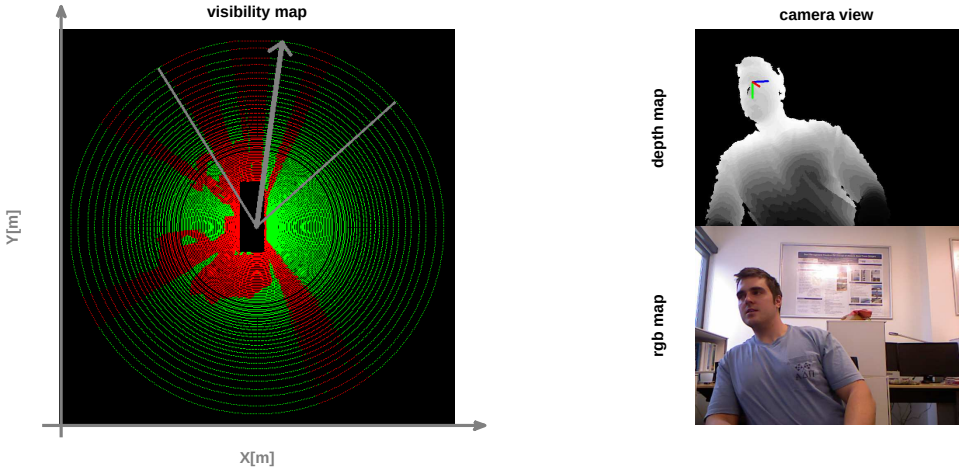
(a) Head moving forward along the +Y axis.



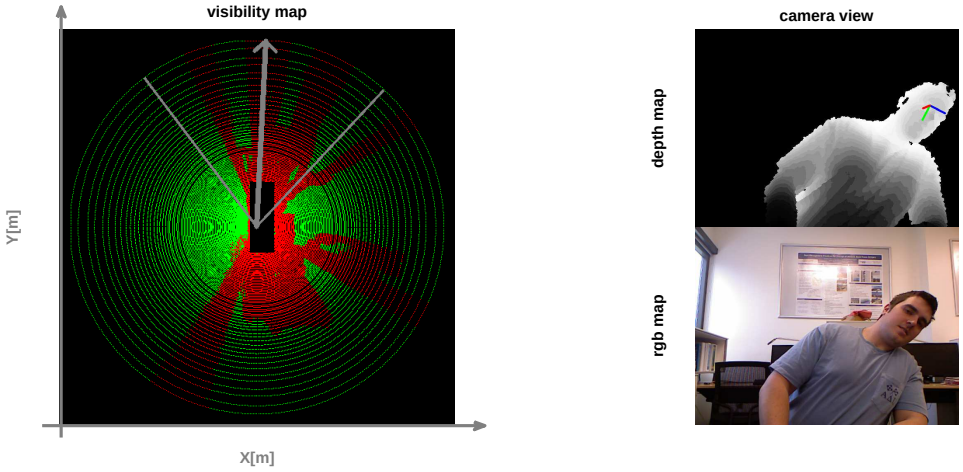
(b) Head moving forward along the -Y axis.

Figure 38: Visibility of the user changing due to its head moving forward and backward.

- *Head moving right and left*: Figure 39 shows the visibility maps generated as the subject moves its head to its right and left. In figure 39a the head moves to the right which results in increased visibility on the right side as shown in the visibility map where as in figure 39b the visibility on the left side of the map increases as the user moves to the left.



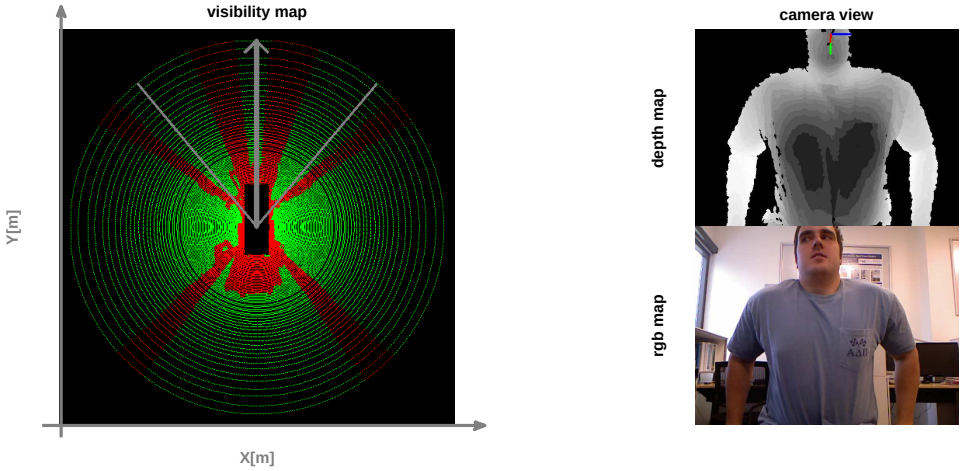
(a) Head moving to the right along the +X axis.



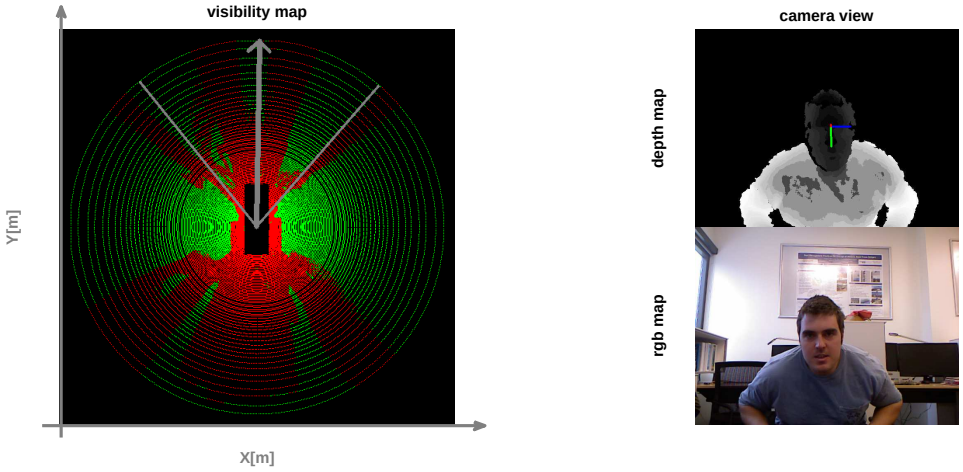
(b) Head moving to the left along the -X axis.

Figure 39: Visibility of the user changing due to its head moving to the right and left.

- *Head moving up and down*: Figure 40 shows the visibility maps generated due to the user moving its head along the +Z and -Z axis respectively. As the head moves upward the subject's visibility of the equipment's surroundings increases (see figure 40a), whereas as the user moves its head towards the ground, that is along the -Z axis, the visibility decreases all around the equipment as shown in figure 40b.



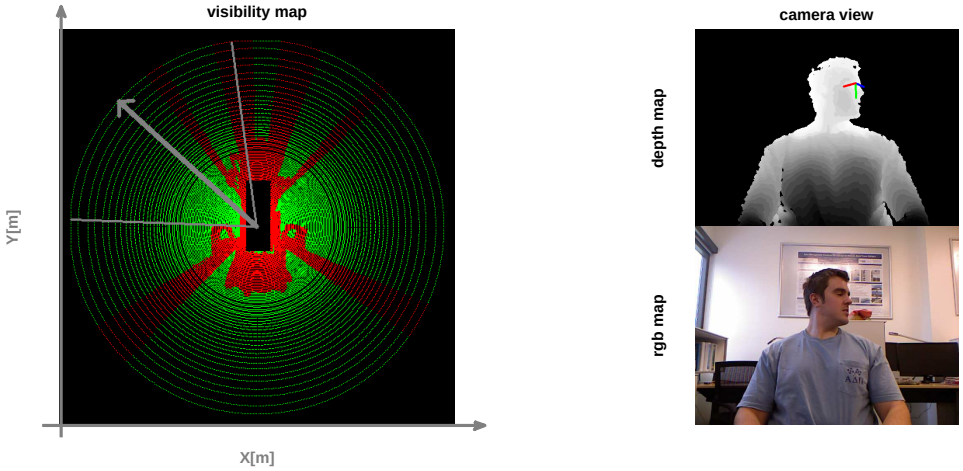
(a) Head moving up along the +Z axis.



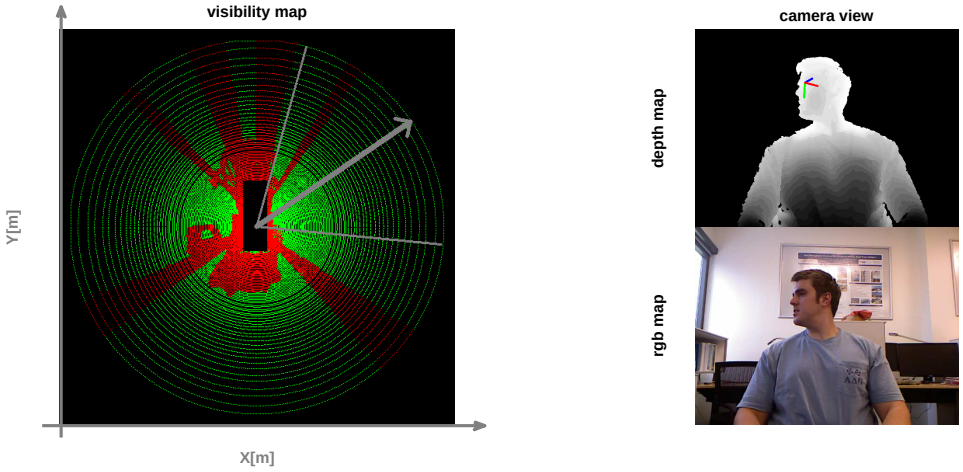
(b) Head moving down along the -Z axis.

Figure 40: Visibility of the user changing due to up and down head motion.

- *Head turning left and right*: Figure 41 shows the FOV of the user changing due to the subject rotating its head around the +Z axis of the vehicle coordinate system. Additionally, it can be observed that the visibility changes on left and the right side (+X and -X directions respectively) of the visibility maps as the user leans to the left and right hand side while turning its head.



(a) Head turning left in CCW direction around the +Z axis.



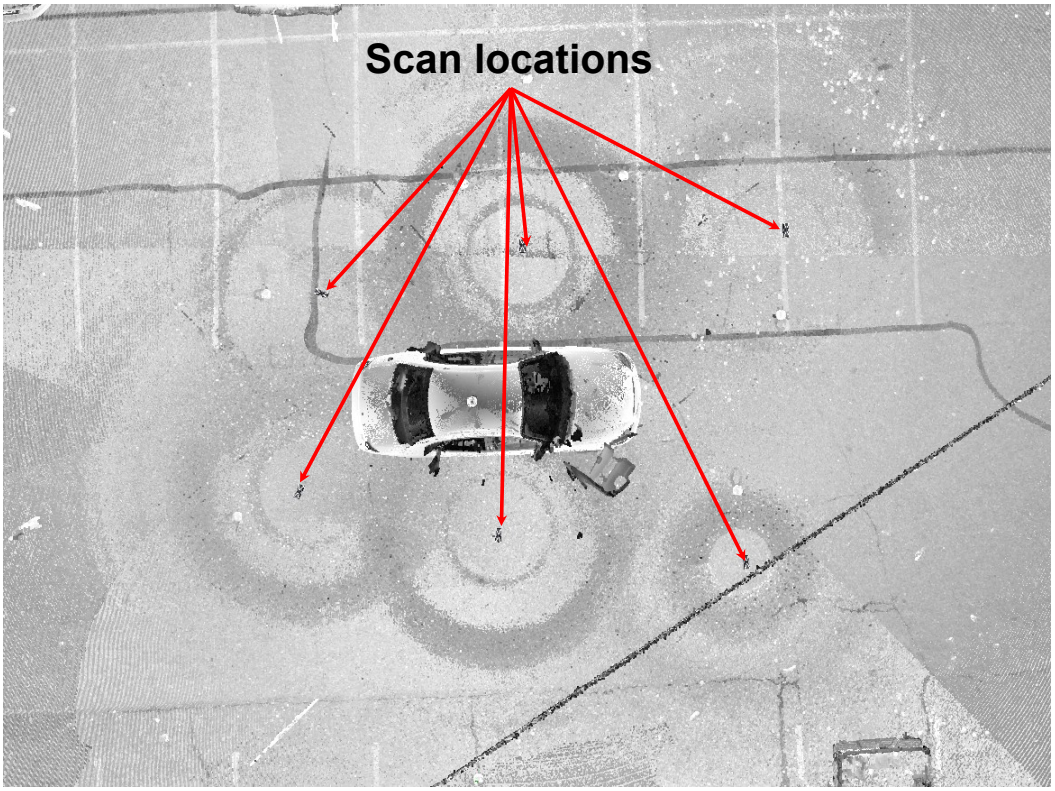
(b) Head turning right in CW direction around the +Z axis.

Figure 41: Visibility of the user changing due to rotation around the +Z axis of the vehicle coordinate system.

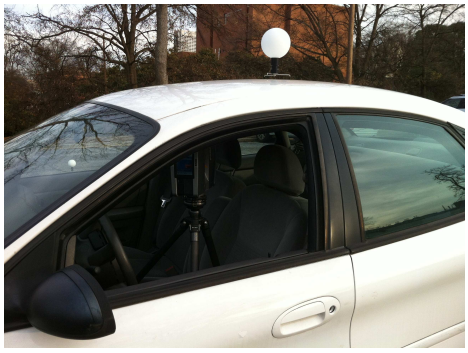
4.3 Field Experiment

Field experiments were performed to demonstrate the approach in an uncontrolled environment. The field experiments were performed in a passenger car as shown in figure 42. The point cloud of the car was obtained by scanning the car from seven different locations (with varying elevations) using a FARO laser scanner. 6 of the scans were taken from the outside as shown in figure 42a and one scan was taken from the

inside as shown in figure 42b. Each of the scan has its own associated coordinate system and the scans need to be registered together so that they agree to a common coordinate system which is the vehicle coordinate system. The origin of this common coordinate system is chosen to be the origin of scan taken from the inside of the car as this positions the origin at approximately the driver's head position. Thus, the 6 outside scans are registered with the inside scan. To register the 7 scans together, markers (white spheres) were placed on the ground and one on the roof of the car as shown in figures 42b and 42a. These markers help in establishing correspondences between the scans and thus help in registering the scans together. After registration, a manual cleaning operation was performed to remove noisy artifacts (specifically on the glass windows). This resulted in a point cloud containing $\sim 33.3 \times 10^6$ points. Figure 43 shows the top and side views of the resulting point cloud respectively. The point cloud shown in the figure 43 is a sparsely sampled version of the actual point cloud obtained after registration.



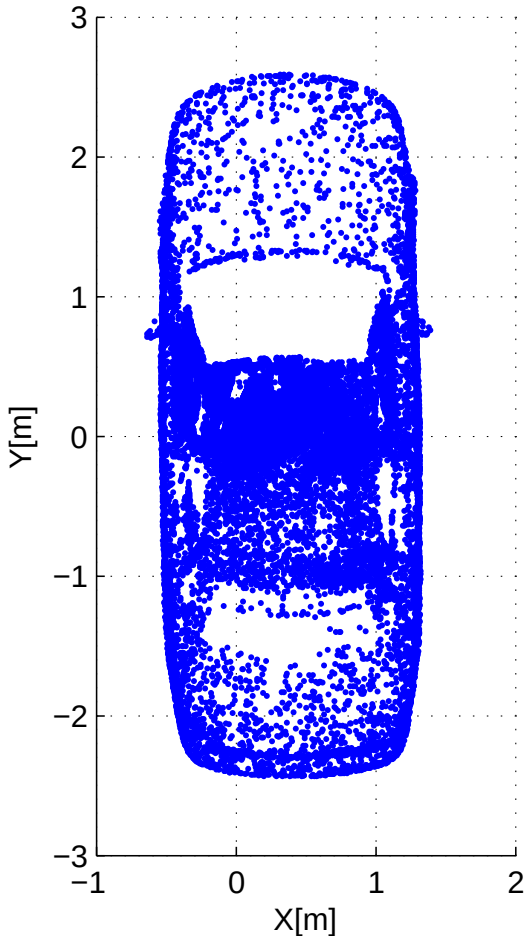
(a) Six scans taken from the outside.



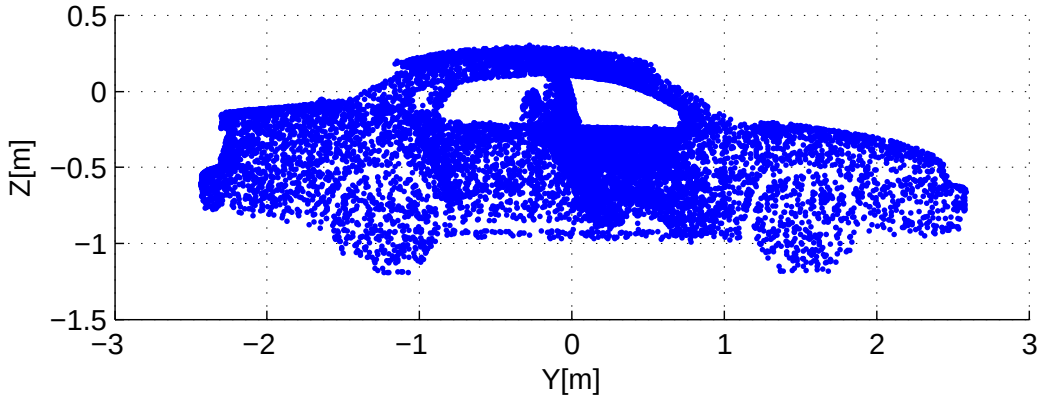
(b) One scan from the inside.

(c) Markers used for registering the 7 scans.

Figure 42: Laser scanner locations.



(a) Top view



(b) Side view

Figure 43: Point cloud generated after registering the 7 laser scans. The point cloud shown here is a sparse representation of the point cloud obtained from after registration.

To measure the dynamic blindspots of the driver while driving the car, a kinect camera was mounted on the dashboard as shown in figure 34. The route through which the car was driven is shown in figure 44 with many turns which allowed the driver to perform various head motions. The driver in the experiment does not belong to the data set used for training the head posture estimation algorithm.

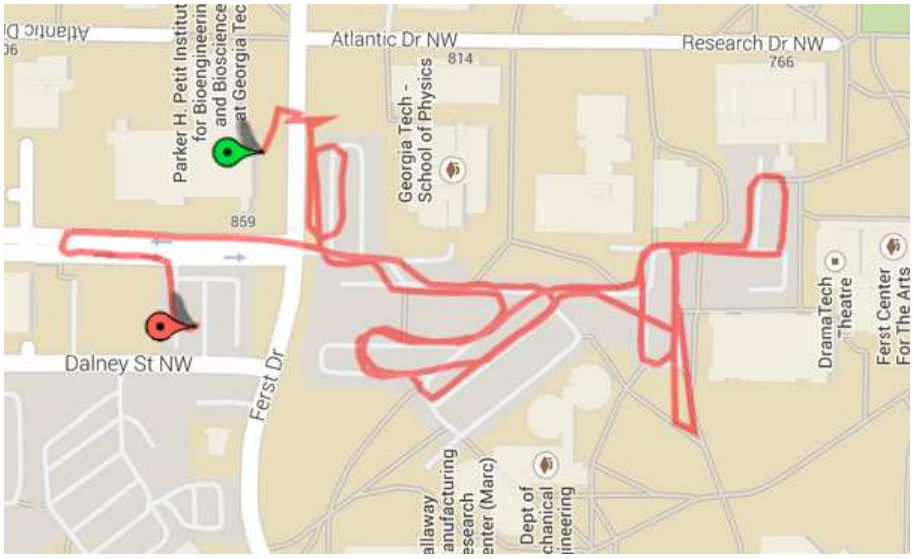
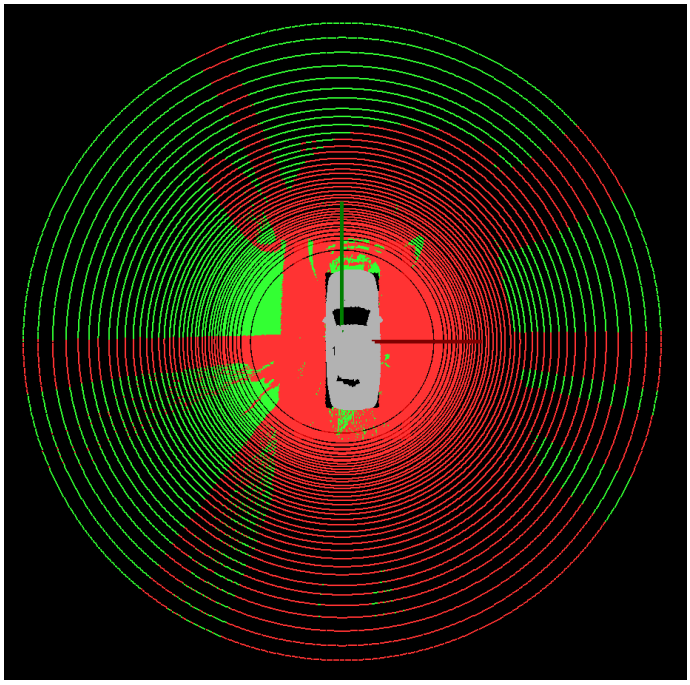
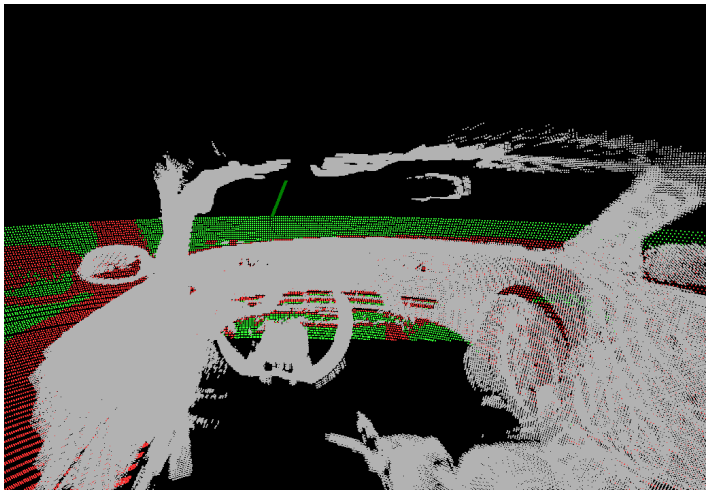


Figure 44: The map of the car route.

Before presenting results to dynamic blindspots, the results to static blindspots are first presented. Figure 45 shows the static blindspots map generated from the point cloud shown in figure 43. Notice the region close the frontal and rear region of the car's point cloud marked green. These visible regions are the result of the scans failing to capture the geometry of the dashboard and the car floor. Thus, multiple scans must be taken from the inside and from locations in the immediate vicinity of the car windows to capture the interior geometry. Figure 45b shows the visible region caused to due failure of the scans to capture the car geometry (notice the region under the steering wheel).



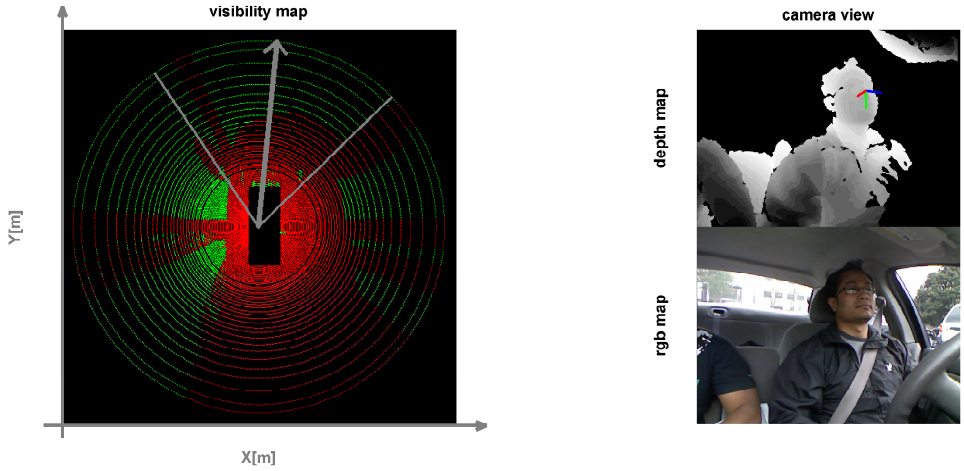
(a) XY blindspots map of the car.



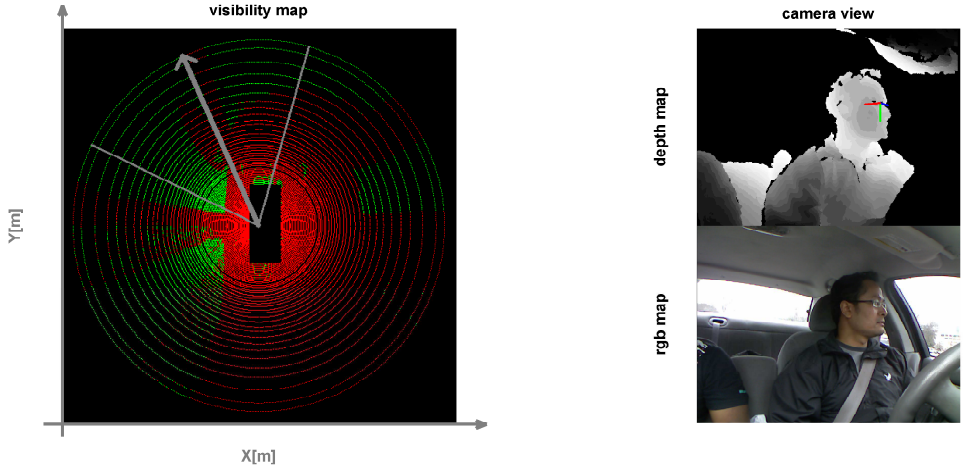
(b) Visible region in front of the front bumper from the driver's head position.

Figure 45: Blindspots map of the car on XY plane (ground plane). +Y axis: bottom to top and +X axis: left to right.

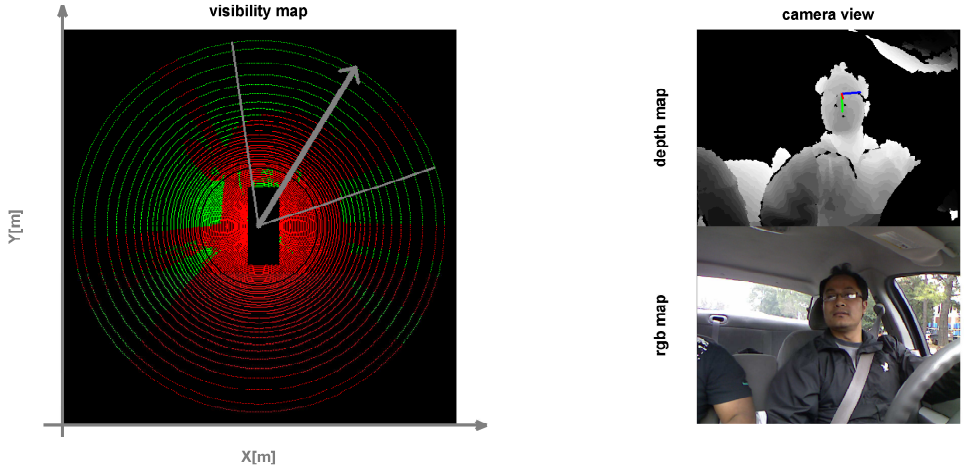
In figure 46, results to dynamic blindspots have been illustrated for three different head poses: frontal, left-ward and right-ward. The column on the left hand side shows the visibility map of the driver. The green pixels denote the visible region and the red pixels denote the invisible region caused due to the structure of the car. The gray arrow shows the head orientation of the driver along with the FOV represented by two gray lines on either side of the arrow. The column on the right hand side shows the depth and the rgb image of the driver of the corresponding visibility maps on the right column. The depth images also show the head posture estimated in camera coordinate system. All the depth maps have been threshold with a value of 0.9m to reject background pixels.



(a) Frontal viewing head pose.



(b) Leftward viewing head pose.



(c) Rightward viewing head pose.

Figure 46: Visibility of the driver for different head postures.

4.4 Computational Performance

For a real-world implementation it is critical that the dynamic blindspots computation is real-time. In this regard, below the results to the computational performance are presented. The blindspots computation timings reported in this section are different from those reported in section 3.5 as a modified implementation was used for dynamic blindspots computation. Table 22 below reports that the dynamic blindspots computation runs at ~ 7 fps when the user is at approximately 0.7m distance. The user distance from the camera only affects the HPE because of the size of the bounding box as discussed in the section 2.2.6. However, PCT and BM are not affected by user distance from the camera. The results presented below were obtained on a 3.07 GHz Intel Xeon machine (only one core was used). It can be observed that the bottle neck of the computation arises from translating the point cloud and re-binning the translated point cloud into the 3D histogram matrix S^{hist}. Further speed up can be achieved by increasing the bin width of S^{hist} matrix as it would result in a smaller point cloud thus speeding up the computation of PCT.

Table 22: Computational performance.

Module	Fps (user at 0.7m distance)
Head Pose Estimation (HPE)	~ 38
Point Cloud Translation (PCT)	~ 9
Blindspots Map (BM)	~ 27
Dynamic blindspots (HPE + PCT + BM)	~ 7

4.5 Summary

In this chapter, results to dynamic blindspots were presented. Experiments were conducted both in indoor and outdoor environment. The performance of the approach indicates that such a system can run real-time, however an optimized implementation

can speed up the implementation. Finally, in the figure 47 below shows some of the frames from the simulated and field experiments in which head pose estimation algorithm failed.

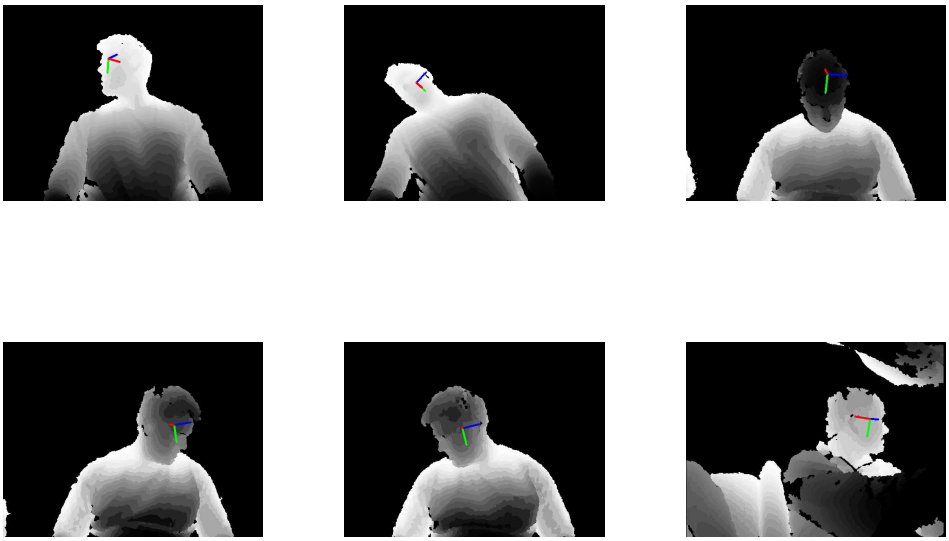


Figure 47: Some of the failed frames in simulated and field experiments.

CHAPTER V

DISCUSSION

In this thesis, an approach to compute visibility of equipment or vehicle operator was presented. Results presented in the previous chapter suggest this approach is feasible for real-world applications. However, for any real-world implementation several key issues and assumption that have been made need to be addressed. This Chapter discusses these key issues and assumptions briefly below.

5.1 Binocular Vision Model

The visibility maps or blindspots map were generated from a point source assuming a monocular vision model, however human visibility is binocular by nature. While such an assumption simplifies the approach and reduces the code runtime but however it is not representative of the human vision system. The presented approach can be easily extended from a monocular vision to binocular vision. The first step in this approach will be to estimate the eye locations. The presented head posture approach in Chapter 2 can be easily extended to estimate multiple fiducial locations [24, 61]. After estimating the eye locations, two independent blindspots map can be constructed and merged together to generate a visibility map for a binocular vision model. A more accurate representation of driver visibility can be obtained by measuring the gaze of the driver from the movement of eyes [30]. Such an approach would however increase the computational time by affecting the PCT phase in Table 22 the most.

5.2 Indirect Visibility

All the results presented in this thesis are measurements of *direct* visibility. It does not take into account the *indirect* visibility offered by mirrors. Thus, any real-world implementation must take into account the *indirect* visibility of an operator. One approach to measure the *indirect* visibility of an operator is by shooting rays from the head location in the constructed visibility maps towards the mirrors. Such, an approach will only take into consideration the head rotation about the Z axis in the vehicle coordinate system. By computing direction and orientation of the reflected ray, the visible and blindspots region can then be marked.

5.3 Visibility due to Loads

Another factor that affects the visibility of the operator are the loads lifted by the equipment. Loads carried by the equipment can have a varying degree of effect on the visibility of the operator. Consider the example of a dozer and a forklift, in a dozer the load is held in the bucket whereas in a forklift the loads sits in the fork and can potentially obscure the visibility of the operator in forward direction. In the case of a dozer the operator's visibility is not significantly affected due to a loaded bucket. Thus, the point cloud of the dozer may be enough to measure the visibility of the operator. Whereas in case of a forklift, a point cloud alone is not enough to measure the visibility of the operator. It will be necessary the measure the size and the location of the load to get an accurate measurement of the visibility of the operator. This measurement can be performed by using computer vision based technique. Thus, it can be realized that the obscurity caused due to loads should be taken into account to measure the visibility of the operator.

5.4 Point Cloud Generation

In the experiments a commercial laser scanner was used to generate point cloud data of the skid steer loader and the passenger car. Recent developments in scene reconstruction suggest range cameras can be used to generate point cloud data of equipment [8, 50]. KinectFusion is one of the most popular techniques for scene reconstruction [50]. In [44], it is reported that KinectFusion approach when used indoors in a volume of $(7m)^3$ results in an accuracy drop to $\sim 80mm$. Thus, it will be interesting to study and compare the generated blindspots map as the bin width values: $\Delta r = 0.05$m, $\Delta\theta = 0.3°$, $\Delta\phi = 0.3°$ as reported in section 3.5 are only marginally higher. Additionally, scanning with a hand held device such as kinect will allow to scan unreachable parts in a vehicle which can help in avoiding the problem shown in Figure 45. Another way to generate point cloud data of equipment or vehicle is to use CAD model of the equipment. These CAD models can be used in proprietary software packages to generate point cloud data. Such, a CAD generated point cloud is an accurate representation of equipment unlike scanning generated ones which are typically get affected due various factors such as reflectivity, proximity and orientation of the surfaces of equipment with respect to the scanner.

5.5 Articulated Machines

In Chapter 4 the visibility maps were constructed using point cloud data of a skid steer loader and a passenger car. It is inherently assumed that these machines are not articulated, that is, there is no relative motion between the different structures of the equipment. However, this is not true of most construction equipment such as skid steers, bull dozers and, telehandlers. The pose of such equipment at any instant of time need to be taken into consideration as they affect the visibility of the operator. This equipment articulation can be addressed by assigning rotation and translation parameters to the point cloud of each the movable structure of the

equipment. By either determining the equipment pose using vision-based techniques or the equipment hardware, appropriate rotation and translation can then be applied to the point cloud of the respective structure, thus yielding a complete point cloud of the articulated equipment for every pose. Then, as discussed, the Ray-Casting algorithm can be applied to construct visibility maps for these articulated machines.

5.6 Pedestrian Detection System

Vision based systems detect humans from a video sequence or individual frames. Such video or frames may be generated by a) Intensity or RGB cameras, or b) Night vision cameras. Night vision cameras can be based on the principle of a) intensification of light emitted from natural sources (such as moon light at night), b) active illumination by an infrared emitter (kinect), or c) thermal imaging by detecting difference in temperature of background and foreground objects. These cameras (except thermal cameras) operate by measuring the amount of light (visible or infra red) reflected by the scene. However, thermal cameras rely on the heat emitted by the entities in the scene and convert it to images.

Night vision cameras have seen limited use for detecting and tracking pedestrians. Algorithms used for detecting humans in Intensity or RGB cameras can essentially be used for detecting humans in Night vision cameras. Following [16] approach, [63] performed pedestrian detection with infra red camera for night-time applications. SVM was utilized to detect pedestrians in the scene and using Kalman filter prediction and mean shift algorithm pedestrian's were tracked.

Human detection using vision based approaches is a difficult problem owing to the variabilities induced by the non-rigid structure of human body and the changes in appearance. The complexity of the problem increases further when environmental factors such as illumination, occlusion due to surrounding comes in to play. Keeping these issues in focus various approaches have been formulated to perform human

detection using vision. These approaches can be broadly classified into a) global feature based and b) local feature based methods.

One way to incorporate illumination and scale invariance is to use covariance matrices as object descriptors. In [67] human detection was performed by using covariance matrices as object descriptors. These object descriptors are then represented as a connected Riemann Manifold and then classification is performed in the manifold by using LogitBoost. As discussed above, robust features invariant to scale, rotation and illumination are critical to the performance of learning algorithms. SIFT features [41] were shown to address these issues. A significant work on human detection was by [16] which used HoG descriptors with SVM. To improve the run time performance of [16], [70] combined a cascade of rejector approach with HoG features. These techniques essentially use a single frame based approach and further performance improvement is achieved by incorporating motion information [17]. As discussed above issues pertaining to partial occlusion due to non-rigid structure, illumination invariance can be addressed by [41, 16]. However, occlusion due to surrounding can degrade the performance of methods which rely on global descriptors [18, 19]. To improve performance under partial occlusion conditions, HoG descriptors were combined with cell structured Local Binary Pattern [68].

The other alternative to global feature, is to utilize local level features for object detection. Local features can be high-level or low-level. High level local features may represent parts of an object which encode local appearance properties of an object. These parts are then assembled in a deformable configuration and then object detection is performed by maximizing a score function [45, 25, 26]. The low-level features can be based on pixel difference [61] or difference of averages of two windows [22, 23, 15, 28]. Such low-level local features essentially vote for local regions in an image which are then combined together to detect objects. This is particularly desirable as it can identify objects in presence of occlusion due to surrounding as voting

takes place for based on image patches sampled from the entire image. In addition to being simple in nature, typically computational cost incurred in evaluating low-level features is less. Random forests [7] algorithm is suitable for utilizing low-level features for the training the *decision stumps.* In [27] a random forest was trained to classify image patches as foreground and background. Then patches sampled from a query image are evaluated using the trained forest. Each patch votes in Hough space and objects are then identified by finding maxima or peaks in Hough space. Additionally, patch based feature approach with Random Forests have shown good results in presence of occlusion [21] but without scaling. In [4] pedestrians were detected by transforming conventional Hough Transform based approach into a probabilistic model. Hough Transform based approaches identify object of interest by locating peaks through non-maxima suppression in the Hough Space which requires tuning of several parameters. In [4] energy minimization (MAP inference) was performed over the probabilistic model to find optimal hypothesis.

The automotive industry has also been researching on the development of pedestrian protection system [29, 60]. A few notable commercial pedestrian detection systems are due to Mercedes Benz [55] and Mobileye (Volvo)[46] . To integrate the pedestrian detection system into the proposed approach it is required that in addition to detecting, the spatial coordinates of the pedestrians(in vehicle coordinate system) be also estimated. The spatial locations can then be used as discussed in Section 3.3.5 to compute the pedestrian or worker visibility of the operator.

CHAPTER VI

CONCLUSION

Injuries and fatalities on construction sites due to Human-Equipment interaction can be mitigated by the use of proximity detection technologies. While there exists a number of different technologies that can serve as warning systems, they also raise nuisance alerts as they do not raise alerts intelligently. In this thesis, an approach was proposed to measure the visibility of equipment operators in real-time using vision and ranging based technologies which can potentially be used to create an intelligent proximity alert system. The generated visibility maps of equipment operator or drivers can be used to identify potential hazards posed to workers due to operating equipments. To estimate the head posture of the driver, Random Forests algorithm was used on range images. Proposed approach for head pose estimation reduced the head location error while increasing the execution speed. The proposed blindspots computation methodology allows measuring different facets of blindspots which were previously not possible using automated technique. From a computational viewpoint, the visibility maps were constructed in real-time at 7fps. As it is critical that alerts be raised instantaneously with the occurrence of a hazardous situation, thus it is necessary to optimize the code to improve the execution speed to realize a real-world implementation.

In addition to potentially serving as a proximity alert system, the proposed approach can be utilized to record near-miss incidents. Collection of such information can be used to train the workers and plan construction tasks to minimize potential hazards. Additionally, the blindspots map information generated through this

approach can potentially be used along with the knowledge of sensors FOV to effectively monitor obscure regions surrounding a vehicle or help equipment manufacturers improve the design of equipment to increase safety.

REFERENCES

[1] Ando, S., Kusachi, Y., Suzuki, A., and Arakawa, K., "Appearance based pose estimation of 3d object using support vector regression," in *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, vol. 1, pp. I–341, IEEE, 2005.

[2] Appel, A., "Some techniques for shading machine renderings of solids," in *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, AFIPS '68 (Spring), (New York, NY, USA), pp. 37–45, ACM, 1968.

[3] Balasubramanian, V., Ye, J., and Panchanathan, S., "Biased manifold embedding: A framework for person-independent head pose estimation," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pp. 1–7, IEEE, 2007.

[4] Barinova, O., Lempitsky, V., and Kholi, P., "On detection of multiple object instances using hough transforms," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 9, pp. 1773–1784, 2012.

[5] Bostelman, R. and Lang, L., "Measurement and evaluation of visibility experiments for powered industrial vehicles," *National Institute of Standards and Technology(NIST)*, December 2011.

[6] Bostelman, R., Teizer, J., Ray, S., Agronin, M., and Albanese, D., "Improvement methods for evaluation of visibility for industrial vehicles towards improved safety standards," *In review*, 2012.

[7] Breiman, L., "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[8] Bylow, E., Sturm, J., Kerl, C., Kahl, F., and Cremers, D., "Real-time camera tracking and 3d reconstruction using signed distance functions," in *Robotics: Science and Systems (RSS) Conference 2013*, vol. 9, 2013.

[9] Chen, L., Zhang, L., Hu, Y., Li, M., and Zhang, H., "Head pose estimation using fisher manifold learning," in *Proceedings of the IEEE International Workshop on Analysis and Modeling of Faces and Gestures*, p. 203, IEEE Computer Society, 2003.

[10] Cheng, P., Li, W., and Ogunbona, P., "Kernel pca of hog features for posture detection," in *Image and Vision Computing New Zealand, 2009. IVCNZ'09. 24th International Conference*, pp. 415–420, IEEE, 2009.

[11] Cheng, T., Venugopal, M., Teizer, J., and Vela, P., "Performance evaluation of ultra wideband technology for construction resource location tracking in harsh environments," *Automation in Construction*, vol. 20, no. 8, pp. 1173 – 1184, 2011.

[12] Cheng, Y., "Mean shift, mode seeking, and clustering," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 17, no. 8, pp. 790–799, 1995.

[13] Comaniciu, D. and Meer, P., "Mean shift: A robust approach toward feature space analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 5, pp. 603–619, 2002.

[14] Cootes, T., Edwards, G., and Taylor, C., "Active appearance models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 6, pp. 681–685, 2001.

[15] Criminisi, A., Shotton, J., Robertson, D., and Konukoglu, E., "Regression forests for efficient anatomy detection and localization in ct studies," *Medical Computer Vision. Recognition Techniques and Applications in Medical Imaging*, pp. 106–117, 2011.

[16] Dalal, N. and Triggs, B., "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, IEEE, 2005.

[17] Dalal, N., Triggs, B., and Schmid, C., "Human detection using oriented histograms of flow and appearance," *Computer Vision–ECCV 2006*, pp. 428–441, 2006.

[18] Dollár, P., Wojek, C., Schiele, B., and Perona, P., "Pedestrian detection: A benchmark," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 304–311, IEEE, 2009.

[19] Dollár, P., Wojek, C., Schiele, B., and Perona, P., "Pedestrian detection: An evaluation of the state of the art," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 4, pp. 743–761, 2012.

[20] Duda, R., Hart, P., and Stork, D., "Pattern classification. 2nd," *Edition. New York*, 2001.

[21] Fanelli, G., Dantone, M., Gall, J., Fossati, A., and Van Gool, L., "Random forests for real time 3d face analysis," *International Journal of Computer Vision*, pp. 1–22, 2012.

[22] Fanelli, G., Gall, J., and Van Gool, L., "Real time head pose estimation with random regression forests," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 617–624, IEEE, 2011.

[23] Fanelli, G., Weise, T., Gall, J., and Van Gool, L., "Real time head pose estimation from consumer depth cameras," *Pattern Recognition*, pp. 101–110, 2011.

[24] Fanelli, G., Dantone, M., Gall, J., Fossati, A., and Van Gool, L., "Random forests for real time 3d face analysis," *Int. J. Comput. Vision*, vol. 101, pp. 437–458, February 2013.

[25] Felzenszwalb, P., Girshick, R., and McAllester, D., "Cascade object detection with deformable part models," in *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pp. 2241–2248, IEEE, 2010.

[26] Felzenszwalb, P., Girshick, R., McAllester, D., and Ramanan, D., "Object detection with discriminatively trained part-based models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 9, pp. 1627–1645, 2010.

[27] Gall, J. and Lempitsky, V., "Class-specific hough forests for object detection," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 1022–1029, IEEE, 2009.

[28] Gall, J., Razavi, N., and Van Gool, L., "An introduction to random forests for multi-class object detection," *Outdoor and Large-Scale Real-World Scene Analysis*, pp. 243–263, 2012.

[29] Gavrila, D., Giebel, J., and Munder, S., "Vision-based pedestrian detection: The protector system," in *Intelligent Vehicles Symposium, 2004 IEEE*, pp. 13–18, IEEE, 2004.

[30] Guestrin, E. D. and Eizenman, E., "General theory of remote gaze estimation using the pupil center and corneal reflections," *Biomedical Engineering, IEEE Transactions on*, vol. 53, no. 6, pp. 1124–1133, 2006.

[31] Guo, Z., Liu, H., Wang, Q., and Yang, J., "A fast algorithm face detection and head pose estimation for driver assistant system," in *Signal Processing, 2006 8th International Conference on*, vol. 3, IEEE, 2006.

[32] Hadid, A., Kouropteva, O., and Pietikainen, M., "Unsupervised learning using locally linear embedding: Experiments with face pose analysis," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 1, pp. 111–114, IEEE, 2002.

[33] Hefner, R., "Construction vehicle and equipment blind area diagrams," *National Institute for Occupational Safety and Health*, 2004. (Accessed May 11, 2008).

[34] Hinze, J. W. and Teizer, J., "Visibility-related fatalities related to construction equipment," *Safety Science*, vol. 49, no. 5, pp. 709 – 718, 2011.

[35] Huang, J., Shao, X., and Wechsler, H., "Face pose discrimination using support vector machines (svm)," in *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, vol. 1, pp. 154–156, IEEE, 1998.

[36] ISO-5006:2006(E), "Earth-moving machinery-operator's field of view-test method and performance criteria," *International Organization for Standardization*, 2006.

[37] Kanade, T., Baluja, S., and Rowley, H., "Rotation invariant neural network-based face detection," 1997.

[38] Lablack, A., Zhang, Z., and Djeraba, C., "Supervised learning for head pose estimation using svd and gabor wavelets," in *Multimedia, 2008. ISM 2008. Tenth IEEE International Symposium on*, pp. 592–596, IEEE, 2008.

[39] Lepetit, V., Lagger, P., and Fua, P., "Randomized trees for real-time keypoint recognition," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 775–781, IEEE, 2005.

[40] Liu, K., Luo, Y., Tei, G., and Yang, S., "Attention recognition of drivers based on head pose estimation," in *Vehicle Power and Propulsion Conference, 2008. VPPC'08. IEEE*, pp. 1–5, IEEE, 2008.

[41] Lowe, D., "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[42] Ma, Y., Konishi, Y., Kinoshita, K., Lao, S., and Kawade, M., "Sparse bayesian regression for head pose estimation," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 3, pp. 507–510, IEEE, 2006.

[43] Marks, E. and Teizer, J., "Proximity sensing and warning technology for heavy construction equipment operation," in *9th European Conference on Product and Process Modeling*, pp. 25–27, 2012.

[44] Meister, S., Izadi, S., Kohli, P., Hämmerle, M., Rother, C., and Kondermann, D., "When can we use kinectfusion for ground truth acquisition?," in *Proc. Workshop on Color-Depth Camera Fusion in Robotics*, 2012.

[45] Mikolajczyk, K., Schmid, C., and Zisserman, A., "Human detection based on a probabilistic assembly of robust part detectors," *Computer Vision-ECCV 2004*, pp. 69–82, 2004.

[46] Mobileye, "Mobileye pedestrian collision warning (pcw)," (Accessed January 20, 2014).

[47] Motwani, M. and Ji, Q., "3d face pose discrimination using wavelets," in *Image Processing, 2001. Proceedings. 2001 International Conference on*, vol. 1, pp. 1050–1053, IEEE, 2001.

[48] Murase, H. and Nayar, S., "Illumination planning for object recognition using parametric eigenspaces," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, no. 12, pp. 1219–1227, 1994.

[49] Murphy-Chutorian, E. and Trivedi, M., "Head pose estimation in computer vision: A survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 4, pp. 607–626, 2009.

[50] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., and Fitzgibbon, A., "Kinectfusion: Real-time dense surface mapping and tracking," in *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pp. 127–136, Oct 2011.

[51] NIOSH, "Highway work zone safety, construction equipment visibility - computer simulation," *National Institute for Occupational Safety and Health*.

[52] NIOSH, "Highway work zone safety, construction equipment visibility - manual method," *National Institute for Occupational Safety and Health*. (Accessed July 20, 2012).

[53] NIOSH, "Preventing injuries and deaths of workers who operate or work near forklifts," *NIOSH alert, Center for Disease Control*, 2001. (Accessed March 15, 2012).

[54] Niyogi, P., "Locality preserving projections," *Advances in neural information processing systems*, vol. 16, pp. 153–160, 2004.

[55] Pfeiffer, D. and Franke, U., "Efficient Representation of Traffic Scenes by Means of Dynamic Stixels," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, (San Diego, CA), pp. 217–224, June 2010.

[56] Raytchev, B., Yoda, I., and Sakaue, K., "Head pose estimation by nonlinear manifold learning," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 4, pp. 462–466, IEEE, 2004.

[57] Ruff, T. M. and Holden, T. P., "Preventing collisions involving surface mining equipment: a gps-based approach," *Journal of Safety Research*, vol. 34, no. 2, pp. 175 – 181, 2003.

[58] Ruff, T., "Evaluation of devices to prevent construction equipment backing incidents," *National Institute for Occupational Safety and Health*, 2004.

[59] Saito, M. and Kitaguchi, K., "Appearance based object pose estimation using regression models," in *SICE Annual Conference, 2008*, pp. 1926–1929, IEEE, 2008.

[60] Shashua, A., Gdalyahu, Y., and Hayun, G., "Pedestrian detection for driving assistance systems: Single-frame classification and system level performance," in *Intelligent Vehicles Symposium, 2004 IEEE*, pp. 1–6, IEEE, 2004.

[61] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A., "Real-time human pose recognition in parts from single depth images," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 1297–1304, IEEE, 2011.

[62] Srinivasan, S. and Boyer, K., "Head pose estimation using view based eigenspaces," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 4, pp. 302–305, IEEE, 2002.

[63] Suard, F., Rakotomamonjy, A., Bensrhair, A., and Broggi, A., "Pedestrian detection using infrared images and histograms of oriented gradients," in *Intelligent Vehicles Symposium, 2006 IEEE*, pp. 206–212, IEEE, 2006.

[64] Teizer, J., Allread, B. S., Fullerton, C. E., and Hinze, J., "Autonomous pro-active real-time construction worker and equipment operator proximity safety alert system," *Automation in Construction*, vol. 19, no. 5, pp. 630 – 640, 2010.

[65] Teizer, J., Allread, B. S., and Mantripragada, U., "Automating the blind spot measurement of construction equipment," *Automation in Construction*, vol. 19, no. 4, pp. 491 – 501, 2010.

[66] Tuzel, O., Porikli, F., and Meer, P., "Region covariance: A fast descriptor for detection and classification," *Computer Vision–ECCV 2006*, pp. 589–600, 2006.

[67] Tuzel, O., Porikli, F., and Meer, P., "Human detection via classification on riemannian manifolds," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pp. 1–8, IEEE, 2007.

[68] Wang, X., Han, T., and Yan, S., "An hog-lbp human detector with partial occlusion handling," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 32–39, IEEE, 2009.

[69] Wientapper, F., Ahrens, K., Wuest, H., and Bockholt, U., "Linear-projection-based classification of human postures in time-of-flight data," in *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pp. 559–564, IEEE, 2009.

[70] Zhu, Q., Yeh, M., Cheng, K., and Avidan, S., "Fast human detection using a cascade of histograms of oriented gradients," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2, pp. 1491–1498, IEEE, 2006.

[71] Zhu, Y. and Fujimura, K., "Head pose estimation for driver monitoring," in *Intelligent Vehicles Symposium, 2004 IEEE*, pp. 501–506, IEEE, 2004.

VITA

Soumitry Jagadev Ray was born in Jatni (near Bhubaneswar), Orissa, India. After completing his high schoolwork at Prananath Autonomous College, Khurdha, Orissa, Soumitry entered the National Institute of Technology (NIT), Rourkela, Orissa in the year 2004. He graduated with a Bachelor's degree in Mechanical Engineering from NIT Rourkela in the year 2008. After working in Maruti Suzuki India Ltd. for a year (2008-2009), he attended the Georgia Institute of Technology, Atlanta, U.S.A from Aug 2009 to May 2014 to pursue his doctorate in Computational Science and Engineering with a concentration in Computational Data Analysis.