To appear in an IEEE VGTC sponsored conference proceedings

Optimizing Placement of Commodity Depth Cameras for Known 3D Dynamic Scene Capture

Rohan Chabra* Adrian Ilie† Nicholas Rewkowski‡ Young-Woon Cha§ Henry Fuchs¶

University of North Carolina at Chapel Hill, USA

Input color images | Input depth images | 3D reconstruction of medical procedure | Immersive learning experience for medical students

Figure 1: 3D capture from depth cameras can be used to prepare immersive learning experiences for medical students.

Abstract

Commodity depth cameras, such as the Microsoft Kinect®, have been widely used for the capture and reconstruction of the 3D structure of room-sized dynamic scenes. Camera placement and coverage during capture significantly impact the quality of the resulting reconstruction. In particular, dynamic occlusions and sensor interference have been shown to result in poor resolution and holes in the reconstruction results. This paper presents a novel algorithmic framework and a method for off-line optimization of depth cameras placements for a given 3D dynamic scene, simulated using virtual 3D models. We derive a fitness metric for a particular configuration of sensors by combining factors such as visibility and resolution of the entire dynamic scene with probabilities of interference between sensors. We employ this fitness metric both in a greedy algorithm that determines the number of depth cameras needed to cover the scene, and in a simulated annealing algorithm that optimizes the placements of those sensors. We compare our algorithm's optimized placements with manual sensor placements for a real dynamic scene. We present quantitative assessments using our fitness metric, as well as qualitative assessments to demonstrate that our algorithm not only enhances the resolution and total coverage of the reconstruction, but also fills in voids by avoiding occlusions and sensor interference when compared with the reconstruction of the same scene using mual sensor placement.

Index Terms: G.1.6 [Numerical Analysis]: Optimization—Global optimization , Simulated annealing ; I.4.8 [Computing Methodologies]: Image Processing and Computer Vision—Reconstruction , Scene Analysis; I.6.3 [Computing Methodologies]: Simulation and Modeling—Applications , Model Development;

*e-mail: rohanc@cs.unc.edu
†e-mail: adyilie@cs.unc.edu
‡e-mail: nrewkows@live.unc.edu
§e-mail: youngcha@cs.unc.edu
¶e-mail: fuchs@cs.unc.edu

1 Introduction

3D capture of room sized dynamic scenes has many applications, such as virtual reality and telepresence [2, 22]. The primary application of our research are events about which some information is known in advance, such as surgical procedures, dance performances or motion picture shots. In this paper, we use prostate biopsies as an example event of interest that may be recorded and repeated many times, including when supervising young surgeons or recertifying experienced ones. The resulting 3D capture enables us to provide medical students with an immersive learning experience using virtual reality headsets, as shown in Figure 1. The capture of these room-sized events can be done using commodity depth cameras such as the Kinect v1. However, sensor resolution limitations, dynamic occlusions in the scene, and depth errors due to sensor interference makes the capture a challenging problem. Smooth and realistic reconstructions of dynamic scenes can be achieved by making use of temporal information in the form of tracked surfaces [13]. However, sudden or significant changes in the scene may lead to failure in surface tracking. Attempts to reduce sensor interference by making use of hardware mechanisms such as vibration [23], and time-multiplexing methods such as shuttering [3] have proven helpful, but suffer from limitations such as image blurring, vibration noise and frame rate drop. Time-of-flight depth cameras such as the Kinect v2 offer better accuracy, but their limitations are less well understood, and continue to be an active area of research [28, 32].

We propose an algorithm to optimize the placements of the depth cameras such that it not only maximizes the scene coverage and enhance the capture resolution, but also attempts to reduce the holes in the resulting reconstruction by avoiding interference and reducing dynamic occlusions. We assume that the approximate choreography of the events to be captured is known in advance, which allows our system to simulate the entire scene using virtual actors. We attempt to provide realistic animations of the virtual actors in the simulation, and analyze the effect of inaccuracies in the simulated animations on the capture results. In order to ensure that the optimized sensor placements work efficiently even if the actors in the actual capture move differently than the virtual actors in our simulations, we introduce variability in the movements of our virtual models, capture the actual events multiple times, and optimize our simulations iteratively.

Our system assigns different weights to different parts of the bodies of the captured actors, depending on the nature of the events. For example, when capturing a surgery, the surgical site of the patient

and the movements of the surgeon's hands are more important than other parts of the scene. We also analyze how a significant movement of a captured subject might either result in sensor pose adjustments around that subject, or require more sensors in order to provide full coverage. We show how making use of event-specific optimization weights leads to better capture results.

We begin by summarizing some related work in Section 2. Section 3 defines our fitness metric and details its computation. Section 4 presents our optimization process. We present our results in Section 5, envision future directions in Section 6 and conclude the paper in Section 7.

2 Related Work

While there is ample related work in optimizing camera placement, to our knowledge, there is no such prior work for depth cameras. However, many of the principles, concepts and factors explored by camera placement optimization approaches also apply to depth cameras. Camera placement has been applied in diverse domains such as tracking [6], surveillance [4, 25] and 3D reconstruction [27].

Modeling camera coverage is a useful step when trying to optimize the performance of a camera-based computer vision system. Camera coverage modeling methods are reviewed comprehensively by Mavrinac and Chen [24]. Relevant to our work are the geometric coverage models, which are concerned with the physical volume of the scene covered by the cameras, and include criteria such as field of view and resolution. The application domain of geometric coverage models that is closest to our work is camera placement, which aims to determine the camera configurations that best satisfy the task requirements, given knowledge about the cameras, the environment and the tasks to accomplish. Comprehensive reviews of camera placement methods can be found in previous work [15, 17, 19, 31]. Research closest to our method includes simulation-based approaches [16, 30], which take advantage of computer graphics to render a simulated scene and help evaluate the fitness of each camera configuration. Fleishman et al. [16] present an automatic camera placement method for image-based modeling from scenes with known geometry. They employ a visibility algorithm that starts with a large set of potential camera positions to produce a small subset that covers every visible polygon in the scene. State et. al. [30] describe an interactive software simulator that assists with the design of multi-camera setups. The simulator lets users interactively place and manipulate cameras within a pre-modeled 3D environment. It uses projected textures to show the coverage of each camera and the effective spatial resolution on the 3D surfaces.

Greedy algorithms have been commonly used in sensor placement approaches, and proven to provide good approximate solutions in polynomial time. For example, Krause et. al. [20] show that finding the sensor configuration that maximizes mutual information is NP-complete and describe how a greedy algorithm achieves a polynomial-time approximation that is within $(1-\frac{1}{e})$ of the optimum.

Many sensor placement methods, including ours, employ optimization of a fitness/performance/quality metric over some domain. Various metrics have been proposed over time, in an attempt to integrate the heterogeneous factors that can affect performance into a single number. Wu et al. [33] use the 2D quantization error on the camera image plane to estimate the uncertainty in the 3D position of a point when using multiple cameras. The error is modeled geometrically, using pyramids, and the uncertainty is computed as an ellipsoid around the polyhedral intersection of the pyramids. Chen and Davis [7] improve this metric by taking into account probabilistic occlusion, and apply it to optimally place cameras for motion capture. Rahimian and Kearney [27] further improve the metric by taking into account the convergence angles of the cameras, and propose a method for dealing with dynamic occlusions when placing cameras for motion capture systems employed in a CAVE environment. They compute a quality metric using marker visibility and triangulation accuracy, and optimize it using simulated annealing. Mittal and Davis [25] introduce a framework for incorporating visibility in the presence of random occlusions into sensor planning. They compute the probability of visibility in the presence of dynamic occluders, under constraints such as field of view, fixed occluders, resolution, and viewing angle, as well as algorithmic constraints such as stereo matching and background appearance. The metric is evaluated at each location, for each orientation and each given sensor configuration, aggregated across space, and optimized using simulated annealing and genetic algorithms. In the surveillance domain, Bodor et al. [4] compute the optimal camera poses for maximum task observability given the possible target trajectories. The function being optimized is directly related to the resolution of the targets in the camera images, and incorporates the distance from the camera to each target trajectory and the angles that lead to foreshortening effects. Other authors take an information theory approach to compute their quality metrics. Denzler et al. [10] introduce uncertainty as a performance metric for selecting the optimal focal length in 3D object tracking. Uncertainty is derived from the expected conditional entropy given a particular action. Visibility is taken into account by considering whether observations can be made and subsequently using the resulting probabilities as weights. Deutsch et al. [11, 12] improve the process by predicting several steps into the future, speeding up the computation, and using sequential Kalman filters to deal with a variable number of cameras and occlusions. In his PhD thesis, Ilie [19] uses a similar approach, but substitutes a-posteriori state error covariance for conditional entropy and uses a different aggregation method over space and time. Sommerlade and Reid [29] add a Poisson process to model the potential of acquiring new targets by exploring the scene.

To the best of our knowledge, there has been no work on optimizing placement of depth cameras for dynamic scene capture. Most of the previous efforts mentioned above optimize the placement of regular cameras for 3D reconstruction. The fundamental differences in the operating principles between regular cameras that simply receive light and depth cameras that also have to emit light (usually infrared) makes their requirements when used for 3D reconstruction differ in fundamental ways as well. For example, the goal when placing regular cameras is usually to maximize both overall coverage and overlap between sets of cameras, to help with computations such as triangulation or space carving. In contrast, in the case of depth cameras, while the overall coverage is still maximized, the overlap needs to be minimized in order to reduce the interference between sensors. These fundamental differences make it difficult to directly compare our depth camera placement results with the regular camera placement results listed in this section. A comparison of the final reconstruction results needs to carefully isolate the effects of the camera placement from the effects of the reconstruction method employed, and is outside the scope of this paper.

3 Evaluating A Camera Configuration's Fitness

In our approach, we use virtual cameras to simulate the depth cameras and polygonal models to simulate known dynamic events in 3D. We define the fitness value F of a particular depth camera configuration C for a dynamic event taking place over a time interval of T frames as the average:

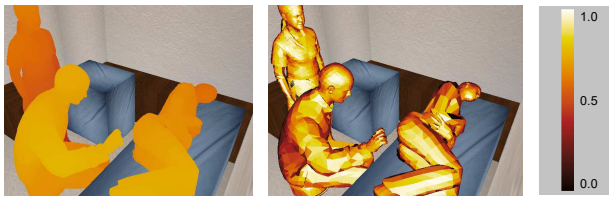
$$F(C,T) = \sum_{t=1..T} \frac{f(C,S_t,W_t)}{T} \tag{1}$$

f is the fitness function, $C = \{c_1,..,c_K\}$ is the set of K depth cameras, $S_t = \{s_{t1},..,s_{tN}\}$ is the set of N dynamic 3D surfaces in the scene, and $W_t = \{w_{t1}..,w_{tN}\}$ is the set of weights for the surfaces based on their importance in the scene, with $\sum_{i=1..N} w_{ti} = 1$. For every frame t, the surface set S_t and its associated set of weights W_t vary as the scene changes.

The fitness value depends on a number of factors, detailed in the next subsections, followed by evaluation and implementation details.

3.1 Surface Factors

The fitness function f at each frame t is based on factors taking into account the characteristics of the 3D surfaces, such as their resolution, orientation, and visibility. We describe each of these factors in detail below.



(a) Surface Resolution Factor (b) Surface Orientation Factor

Figure 2: Visualization of surface capture quality factors in simulation

We account for the *resolution* of a surface s as captured by a depth camera c using the term f_r, defined as:

$$f_r(c,s) = \frac{A_s}{||\vec{c} - \vec{s}||^2} \tag{2}$$

A_s is area of surface s, $\vec{c}$ is camera c's center of projection, and $\vec{s}$ is surface s's centroid respectively, in 3D space. An example distribution for this term is shown in Figure 2(a).

We account for the *orientation* of a surface s with respect to depth camera c using the term f_o, defined as the dot product:

$$f_o(c,s) = \frac{\vec{c} - \vec{s}}{||\vec{c} - \vec{s}||} \cdot \vec{s_n} \tag{3}$$

$\vec{s_n}$ is the normal vector of surface s. In the simulation the normal at the centroid of each mesh triangle is treated as the surface normal. An example distribution for this term is shown in Figure 2(b).

A surface s may not be visible from a depth camera c due to factors such as clipping associated with the field of view of camera c, occlusion by other surfaces in the scene and depth ambiguity due to limits in the range of camera c.

Approaches such as the one by Dou and Fuchs [13] can track and reconstruct surfaces which are temporarily not visible by using temporal integration techniques, provided that the surfaces were visible in the past and become visible again in the future. Figure 3 shows a simple experiment conducted with our simulator. The corresponding reconstruction results, obtained using the approach by Dou and Fuchs [13], are shown in Figure 4.

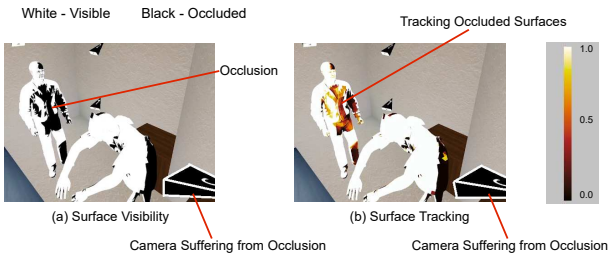


Figure 3: (a) Visualization of surface visibility. (b) Visualization of surface tracking confidence f_t.

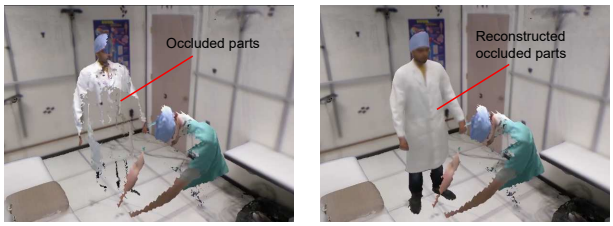


Figure 4: (a) Reconstruction without surface tracking. (b) Reconstruction enhanced with surface tracking [13].

We model surface tracking in our framework as follows: if the surface s is not visible for a time Δt that is less than a threshold τ, then it is marked visible with a tracking confidence f_t.

$$f_t = 1 - \frac{\Delta t}{\tau} \tag{4}$$

Incorporating tracking into our fitness function, we account for the *visibility* of a surface s from depth camera c using the term f_v, defined as:

$$f_v(c,s) = \begin{cases} 1 & \text{if } s \text{ is visible by } c \\ 0 & \text{if } s \text{ is not visible by } c \text{ and not tracked} \\ f_t & \text{if } s \text{ is tracked for a time } \Delta t < \tau \end{cases} \tag{5}$$

3.2 Interference Between Depth Cameras

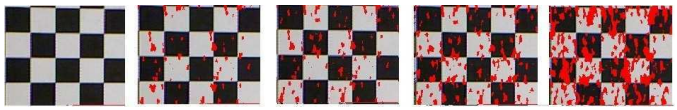
In this subsection, we describe how our framework attempts to minimize the interference between depth cameras. We limit our investigation here to structured-light depth cameras such as the Kinect v1 and the Intel RealSense. Time-of-flight sensors such as Kinect v2 also suffer from interference that can, in theory, be mitigated using time multiplexing or different modulation frequencies. However, modeling and minimizing interference is much more difficult, since the Kinect v2 currently does not allow direct control of its camera and projector shutters, or modulation frequency.

A Kinect v1 sensor continually projects a predetermined pattern of infrared dots, which is then captured by an infrared camera. Depth is calculated using the disparity between the captured infrared image and the projected pattern. Correlating the observed and expected infrared images can fail in areas where there are overlapping dots from another Kinect, resulting in missing data, or "holes", in the depth map. There have been attempts to measure the interference between multiple Kinects. For example, Limin et. al. [21] model the interference between two Kinect sensors and conclude that the probability of interference is directly proportional to the overlap area.

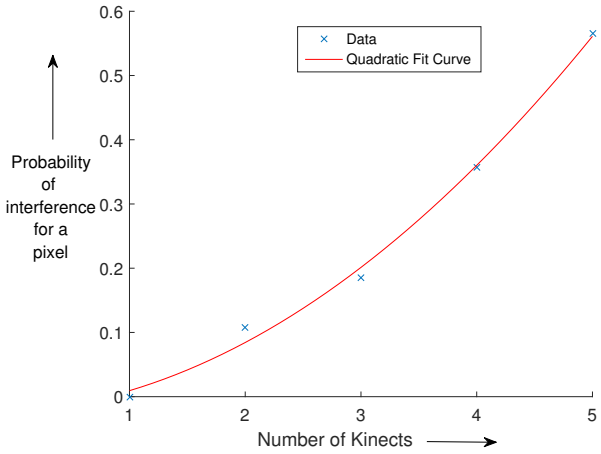
We do not attempt to reduce sensor interference using the method by Maimone [23] because the vibrations involved would produce sounds that may distract in a medical procedure. Multiplexing methods such as the one by Berger et. al. [3] are inappropriate in our case, as they significantly degrade the quality of the reconstruction results by dropping frames, which interferes with temporal integration, tracking, and synchronization between sensors. Instead, we model the interference and use the model in our optimization framework.

To model sensor interference, we conducted a simple experiment with five Kinect sensors aimed at a flat checkerboard placed on a wall approximately $1m$ away from the sensors. The setup is shown in Figure 5(a). We successively switched the Kinect units on, one by one, and measured at each step the percentage occupied by holes in the area of the first sensor's depth map corresponding to a $1m^2$ area on the checkerboard. Figure 5(b) shows the number of holes

(a) Setup for interference modeling



(b) Holes in Kinect1 depth sensor's capture after switching on other Kinects one by one



(c) Inter-depth-sensor interference probability vs number of kinects

Figure 5: Multiple Kinects setup for interference modeling.

increases with the number of sensors switched on. The relationship between the number of overlapping Kinects and the interference probability is shown in Figure 5(c). A quadratic function provides a close fit, so we define interference probability for a particular surface as:

$$p(I) = min(x \cdot I^2 + y \cdot I + z, 1) \tag{6}$$

where p is the interference probability, I is the the number of overlapping sensors that can view the surface s, and x , y and z are constants determined experimentally as $x = 0.021$, $y = 0.012$ and $z = -0.0236$. $p(I)$ models the worst case probability of inter-depth-sensor interference.

We quantify the effect of sensor interference due to overlap on a surface s of area A_s seen by I depth cameras, including camera c, as:

$$f_i(c,s,I) = (1 - p(I-1)) \frac{A_s}{||\vec{c} - \vec{s}||^2} = (1 - p(I-1)) \cdot f_r(c,s) \tag{7}$$

where $\vec{c}$ is the position of the closest sensor and $\vec{s}$ is the position of the surface.

While we have not conducted a similar experiment using Kinect v2 sensors, we envision that our probabilistic framework for interference minimization can also be used in their case, by measuring the variance of depth readings from ground truth instead of measuring the percentage of holes in the depth readings.

3.3 Fitness Function Evaluation

We define the fitness function f at each time t as the weighted sum of the product between the worst possible combined factors $f_c = f_o \cdot f_v$ and the sensor interference factor f_i:

$$f(C,S_t,W_t) = \sum_{i=1..N} w_{ti} \cdot \min_{j=1..K} (f_c(c_j, s_{ti})) \cdot f_i(s_{ti}, I_t) \tag{8}$$

By substituting Eq. (8) in Eq. (1), our fitness value F for a camera configuration C becomes:

$$F(C,T) = \sum_{t=1..T} \sum_{i=1..N} \frac{w_{ti} \cdot \min_{j=1..K} (f_c(c_j, s_{ti})) \cdot f_i(s_{ti}, I_t)}{T} \tag{9}$$

3.4 Implementation Details

To speed up the computation of our fitness function, we use GPU computations and precompute the parts of the fitness function that do not change during the optimization process.

3.4.1 GPU Acceleration

To compute $f_c(c,s) = f_o(c,s) \cdot f_v(c,s)$, we render the scene from the point of view of camera c. We compute $f_c(c,s)$ at each camera pixel in a fragment shader, and store its value in a texture along with the identifier of the surface it belongs to. Then, we compute the average f_c from all pixels belonging to surface s. This is much faster than a CPU computation of f_c, as graphics computations such as occlusions and clipping are performed in parallel on the GPU. Our algorithm is dependent on the resolution of the rendered textures, which we can use to trade-off speed vs. precision.

One shortcoming of this approach is that it treats partially visible surfaces as completely visible when averaging the score of rendered pixels belonging to a surface. We try to minimize this problem by keeping the size of the surfaces very small (about 1-2 cm^2).

3.4.2 Fitness Precomputation

We also use a preprocessing step to avoid computing the values of the combined factors f_c at each iteration step of our optimization (see Sections 4.1 and 4.2): we compute and store f_c for surfaces that are visible in frame t by camera c, and pre-multiply each f_c by the corresponding weight w_{ti}. The result is the individual fitness value f_p of a camera pose $c \in C$:

$$f_p(c,S_t,W_t) = \sum_{i=1..N} w_{ti} \cdot f_c(c, s_{ti}) \tag{10}$$

To keep the size of the precomputed data manageable, we only keep the f_p values for the cameras poses whose individual fitness is higher than a threshold. We store the corresponding f_p values in a hash map $H(c,t)$ data structure with the surface identifier s_{ti} as the key.

The computation for each frame t is independent of other frames in set T. If the pose of the camera is changed, we recompute the hash map $H(c,t)$ for that pose. Algorithm 1 details the computation of the fitness function F.

Algorithm 1: Evaluation of Fitness of a Configuration

Input: C,S,W,H,T
Output: F
forall *frames t in set T* **do**
  $F_t = 0$
  for *each surface* s_{ti} *in set* S_t **do**
    $x_i = 0$
    for *each camera pose* c_j *in set* C **do**
      if *find* s_{ti} *key in* $H(c_j,t)$ **then**
        $x_{ij} = H(c_j,t)$.valueAtKey(s_{ti})
      else
        $x_{ij} = f_p(c_j, s_{ti})$
      end
      if $x_{ij} > x_i$ **then**
        $x_i = x_{ij}$
      end
    end
    F_t += $F_t + x_i \cdot f_i(s_{ti}, I)$
  end
  F += F_t
end

4 Optimization of Camera Placements

To solve the problem of finding the optimal camera configuration for a known dynamic event, we employ a discrete optimization technique. We discretize the space of possible depth camera poses as the set $P = \{c_1, ..., c_M\}$, where M is the number of possible camera poses. Our goal is to find a set $O = \{c_1, ..., c_K\}$ such that $O \subseteq P$ represents the optimized camera configuration set and K is the corresponding number of cameras used. We initialize O using a greedy algorithm and improve it using simulated annealing.

4.1 Greedy Initialization

We use a greedy algorithm to derive an initial configuration set O. We start with $O = \emptyset$. At each iteration k, we select the camera pose c_k from set P which increases the fitness value of set O the most when added to it. The algorithm stops adding cameras when the incremental gain g achieved in the fitness value by adding a new selected camera pose c_k to set O is less than a tolerance value v, or when no suitable camera pose was found. We describe this algorithm in detail in Algorithm 2.

Algorithm 2: Greedy Algorithm

Input: P,τ
Output: O,K
$g = \infty$, $X = 0$, $K = 0$, $O = \emptyset$
repeat
  $c_k = \emptyset$
  $x_k = 0$
  for *each camera pose* c_j *in set* P **do**
    x_j = fitness($O + \{c_j\}$)
    if $x_j > x_k$ **then**
      $c_k = \{c_j\}$
      $x_k = x_j$
    end
  end
  $g = x_k - X$
  $X = x_k$
  $O = O + \{c_k\}$
  $K = K + 1$
until $(g < v)$ *or* $(X = 0)$

4.2 Refinement Optimization by Simulated Annealing

Simulated Annealing (SA) is an effective method for finding a solution close to the global optimum [5]. SA iteratively explores the solution space by varying components of the solution vector. It has been shown to produce very good, but not provably optimal results in a reasonable time for a variety of problems [18].

We start with the initial configuration set O produced by the greedy algorithm. In each iteration, SA changes the camera configurations either by changing the pose of cameras in set O or by selecting new camera poses from set P. If the fitness value of the new configuration is better than the best configuration found so far, the algorithm accepts the new configuration. If it is worse, SA accepts the new configuration with some probability, depending on the current temperature and Boltzmann selection [1]. SA accepts a configuration with a quality metric less than the best known configuration with higher probability at the beginning of the run, to escape potential local maximums. As the algorithm proceeds, the probability of accepting a configuration with a fitness value that is lower than that of the best known configuration is reduced, until the algorithm only accepts solutions that are better than the best current configuration.

5 Results

5.1 Experimental Setup

The primary motivation of our work is to reconstruct dynamic events such as surgeries. These reconstructions can be displayed on Virtual Reality headsets and used by medical students for immersive learning experiences as shown in Figure 1. We started by capturing a real procedure in a clinic, using 3 Kinect sensors (Figure 6). This initial experiment provided valuable insights into the procedure and its capture, including the need to better understand the minimum number of cameras required and their optimal placement.

(a) (b)

Figure 6: (a) 3D capture of a prostate biopsy procedure; (b) simplified simulation of the procedure.

Our next step was to set up a space where we could conduct experiments: a capture lab whose dimensions, layout and equipment closely match the clinic room ($3.7m \times 2.8m \times 2.6m$). The next hurdle was repeatability: testing many different camera configurations with live actors (or medical professionals) would be prohibitively expensive and time-consuming. Therefore, we initially simulated the procedures using virtual models and cameras. We used our optimization to select the best configurations for further testing in our capture lab, with real cameras and actors. This way, when we arrive at the best camera configuration for a procedure, we can use it to set up our cameras in the clinic and capture actual procedures involving doctors and patients.

We used the approach by Dou and Fuchs [13] to reconstruct the real capture room. First, we obtained a scan of the static background by capturing a sequence of RGB-D frames with a moving Kinect, and used the plane-based and visual feature-based bundle adjustment system by Dou et. al. [14] to align all frames. We then used a combination of a volumetric depth map fusion algorithm [9] and the Marching Cubes algorithm [8] to generate a triangle mesh of

the empty room. The result is shown in Figure 7(a). To make use of this high-quality background during live capture, we segmented out the parts that belonged to the static background by subtracting images of the empty room taken before starting the dynamic event, and combined the remaining 3D data with the high-quality static background captured beforehand. To speed up the optimization process, instead of the reconstructed 3D model shown in Figure 7(a), we used a simplified model, shown in Figure 7(b).

(a) (b)

Figure 7: (a) 3D reconstruction of capture room; (b) simplified geometric model of capture room.

We simulated the surgery using 3D models for the doctor, patient and nurse. The polygonal models for these actors have about 10,000 triangles each. We made sure that these triangles are about similar in size and are evenly distributed over different body parts of the synthetic models (for example, our synthetic models had about 2200 triangles on the chest and about 1500 triangles on the face). We use these polygonal surfaces to compute the surface factors of our fitness function. We animated the 3D models using simple animations such as walking, sitting and laying on a bed, following the general choreography of the previously-recorded procedures.

We defined the space of possible camera locations by constraining the camera positions to the top half of the walls, 30 cm apart in a regular grid, as shown in Figure 8. Candidate possible camera angles were between -60° to 60° in roll, pitch and yaw, with an angle increment of 10 degrees. The resulting search space consists of about 8000 different camera poses. We used Eq. (10) in a preprocessing step to compute the individual camera fitness values f_p for each camera pose, ranked the poses in order of their fitness, and eliminated about 48 percent of them based on a threshold.

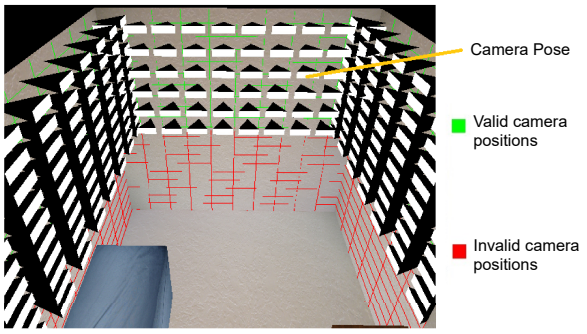


Figure 8: Camera placements in grids.

To quickly obtain a rough estimate for the number of cameras needed, we used the greedy algorithm in Section 4.1. As an experiment, we first set the tolerance value v very high and let the algorithm run until it could find no additional camera pose that would result in an improved fitness value. The algorithm took 320 minutes to complete and stopped after finding 35 camera poses. Figure 9 shows a plot of the fitness value of the optimized configuration vs. the number of cameras, truncated at 15 cameras for clarity. We then set the tolerance value v to 2%. The resulting number of depth cameras was 9, and the computation took about 12 minutes to complete.

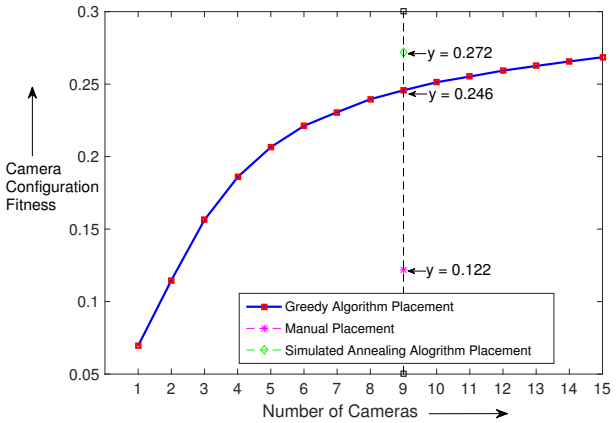


Figure 9: Fitness scores for camera placements using the greedy algorithm, the simulated annealing algorithm, and manual placement.

Next, we used the simulated annealing algorithm in Section 4.2 to improve the solution. The algorithm performed over 50000 iterations (initial temperature $T_i = 1.0$, accepting temperature $T_{accept} = 0.0001$, cooling constant $\alpha = 0.99$ and 50 neighbor cost iterations at each temperature) and took about 3 hours, resulting in about 11% improvement over the greedy solution, as shown in Figure 9.

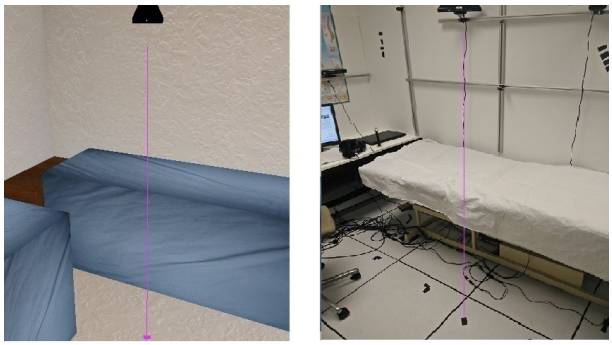
After obtaining the optimized camera configuration from our algorithm, we tested its performance in our capture lab using real actors. To match the optimized camera poses, we first placed each of the 9 Kinects at the optimized positions in our lab. Then, in order to match their orientations, we used fiducial markers. In our simulation we extend rays from the virtual camera centers to intersect with a wall, the floor or the ceiling, yielding positions for markers which we matched by placing physical markers in the actual capture room using a tape measure. We then adjusted the orientation of each Kinect so that its corresponding marker appeared in the optical center of the image taken from the Kinect's camera. Finally, we rotated each Kinect along its optical axis to approximately match the optimized roll value. We plan to improve this process in the future by placing markers at two diagonally opposed corners of the camera image. Figure 10 shows an example of our current process.

5.2 The Impact of Weights

Using virtual models for the moving surfaces in our scene allows us to specify to the optimization procedure the subset of surfaces which the users (e.g., teachers) consider most important, such as, in our case, the faces and hands of the medical personnel and the surgical site. Weights can vary over time, when the importance of parts of the scene changes due to factors such as the events taking place there. Additionally, weights can also be changed as part of a feedback loop, based on experimental observations that parts of the scene were given more attention at particular times by users viewing the reconstruction results in VR.

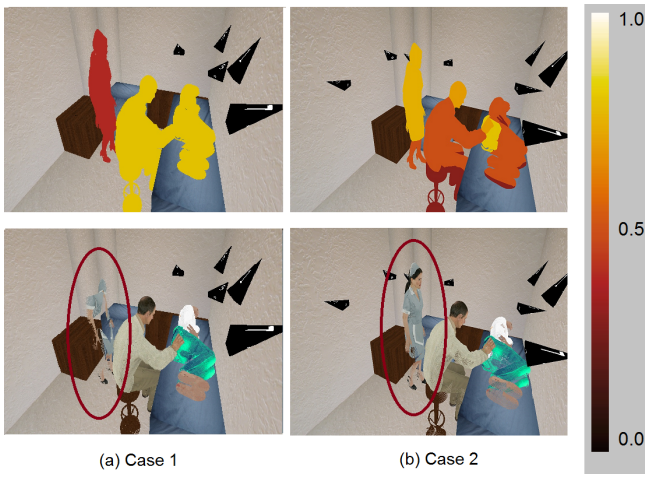
To demonstrate the impact of weights on the optimized camera configuration and the reconstruction, we apply different weights to different body parts of the virtual actors in a surgery captured by 9 cameras. The resulting optimized camera configuration is different in each case, which leads to significantly distinct reconstructions. Figure 11 shows a single frame from the same sequence in two cases.

In the first case, shown in Figure 11(a), we applied higher weights to the patient and doctor's bodies. This leads to improved resolution



(a) (b)

Figure 10: (a) marker position of projected camera center on the floor in the simulation; (b) marker position of projected camera center on the floor in the capture room. The pink lines represent the optical axes of the corresponding cameras, extended to the marker on the floor.



(a) Case 1 (b) Case 2

Figure 11: Choosing different weights (top row) leads to different reconstructions (bottom row).

and completeness in the reconstructions of the patient and doctor, but the nurse was not properly reconstructed. In the second case, shown in Figure 11(b), we assigned higher weights to the nurse's face and hands, and the reconstruction fully showed the nurse without a large impact on the doctor or patient. We kept the latter weights for the remainder of our experiments. The weights were 0.9 for the surgical site, 0.8 for the nurse's upper body, 0.75 for the surgeon's hands and head, 0.4 for the surgeon's legs and the part of the patient's body touching the bed, and 0.5 for all remaining body parts.

5.3 Accounting for Variability

When we observe and record a real surgical procedure, we make note of its general choreography, and use the reconstruction to guide our subsequent modeling and simulation. However, even if we were to try and reproduce the procedure exactly in our simulation, the optimized camera configuration we obtain might fail to reconstruct a subsequent procedure, since participants may move or orient themselves slightly differently than before, and the participants themselves can change as well. In order to account for the differences between different instances of the same procedure and obtain a more robust optimization result, we employ multiple animations that span likely variations of the event. For example, we can give our virtual models different heights, as shown in Figure 12.

(a) Shorter Virtual Actors (b) Taller virtual Actors

Figure 12: Height variation for virtual actors.

We also vary the virtual actors' poses. Figure 13 shows the doctor and nurse placed in different positions for the same frame of an event. This makes the optimized configuration more robust to variation in the actors' body shapes and movements.

(a) Pose A (b) Pose B

Figure 13: Pose variation for virtual actors.

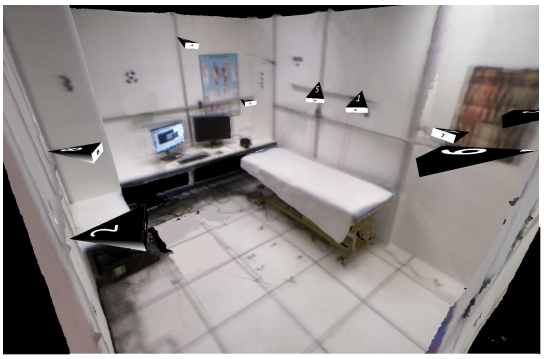
5.4 Comparison

We compare our optimization result with a camera configuration selected manually by a person with knowledge about the choreography of the event both in simulation and in our capture lab. Figure 14 shows both manually-designed and optimized camera placements. Figure 16 shows a visual comparison of virtual and real reconstructions using both manual and optimized camera placement results for the same frame. The reconstruction results for the manual camera placements show holes in place of most of the body parts of the nurse. Also, some areas near the patient's head are not covered due to occlusions by the nurse.

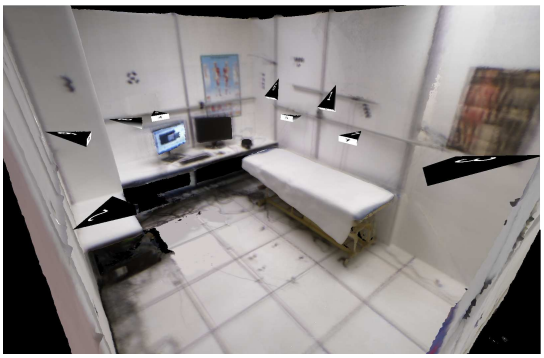
Figure 17 shows point clouds generated from the capture with depth cameras placed in both the manually-designed and the optimized configuration. The reconstruction from the manually-designed configuration capture exhibits low resolution, voids and depth inaccuracies.

The simulated procedure took about 62 seconds. We plotted the percentage of surfaces visible in both our optimized configuration and the manual configuration over the entire interval in Figure 15(a). Since visibility is only one of the terms of the camera configuration fitness, we also plotted the fitness scores for both the configurations in Figure 15(b). The mean configuration fitnesses of the manual and optimized camera placements were 0.122 and 0.272, respectively.

We observed that in the manual camera configuration most of the cameras were placed at higher locations and aimed downward in an attempt to provide more coverage. This led to lower resolution and more depth inaccuracies (as precision of depth cameras usually diminishes with increasing depth) in the reconstructions. In contrast, in the optimized camera configuration, only the cameras covering the patient were aimed downward. The rest of the cameras were placed lower and aimed towards the center of the regions where the doctor and nurse were moving for most of the time interval, and resulted in higher resolution and fewer depth inaccuracies in the reconstructions.



(a) Manual camera placement



(b) Optimized camera placement

Figure 14: Comparison between manual and optimized placements of 9 depth cameras.

6 Future Work

In the future, we would like to test our method on other types of events such as classrooms and group meetings, which feature many actors and rigid objects such as chairs and tables.

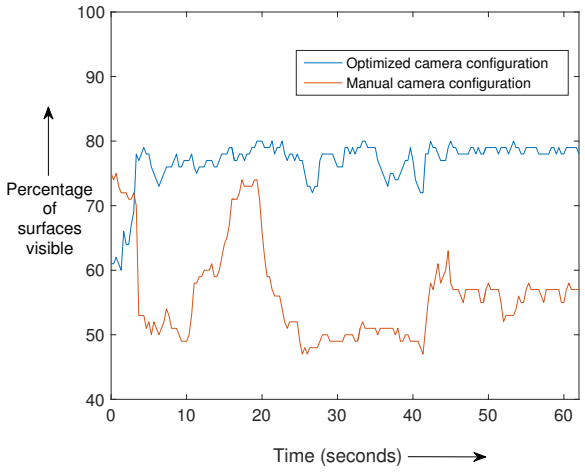
We would also like to model the impact of materials on the reconstruction results, as depth cameras are known to be sensitive to the materials the objects in the scene are made of. For example, black surfaces such as black hair are known to absorb IR light, causing drop-outs in the depth camera's results. Also, highly specular surfaces which reflect IR light are harder to resolve, and often result in holes.

Time-of-flight sensors such as the Kinect v2 provide better accuracy, but have been shown to suffer from problems such as temperature drift [28] and multi-path interference [26].We plan to explore the use of such sensors in detail in our future work.

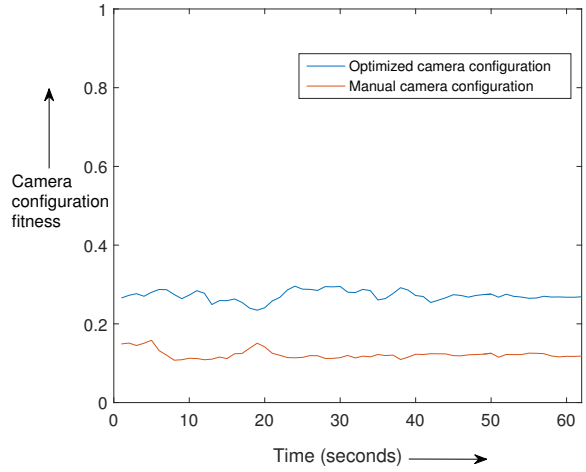
Another future direction is to perform multiple captures of similar scenes. More data from multiple recordings will expose our system to an increased variety of possible sizes and movements of the people in the scene. We would also like to explore the use of moving cameras (such as PTZ cameras) to obtain even better results.

7 Conclusion

In this paper, we proposed a novel method that can significantly improve the 3D capture of room-sized dynamic scenes, by using an optimized depth camera configuration. We demonstrated how our optimized camera placement for a simple medical procedure enhanced the resolution of the important surfaces in the scene, and was also able to reconstruct most of the moving surfaces in the scene by avoiding occlusions and providing improved coverage of the scene.



(a) Surface visibility comparison.



(b) Configuration fitness comparison.

Figure 15: Comparison between manual and optimized camera placements for a sequence of 62 seconds in terms of (a) surface visibility and (b) configuration fitness score.

We plan to make the datasets, reconstructions and our framework publicly available at `http://www.cs.unc.edu/~rohanc/OCP.html`.

8 Acknowledgments

This research is supported by the BeingTogether Centre, a collaboration between Nanyang Technological University (NTU) Singapore and University of North Carolina (UNC) at Chapel Hill. The BeingTogether Centre is supported in part by the National Research Foundation, Prime Minister's Office, Singapore under its International Research Centres in Singapore Funding Initiative.

The authors would like to thank Dr. Eric Wallen, MD and Chris Paterno at Department of Urology, University of North Carolina at Chapel Hill for allowing us to capture a prostate biopsy procedure and describing to us the important phases in the procedure.

The authors acknowledge Andrei State for his constructive comments and suggestions on improving the text in the paper.

References

[1] T. Bäck, D. B. Fogel, and Z. Michalewicz. *Evolutionary computation*. Institute of Physics, 2000.

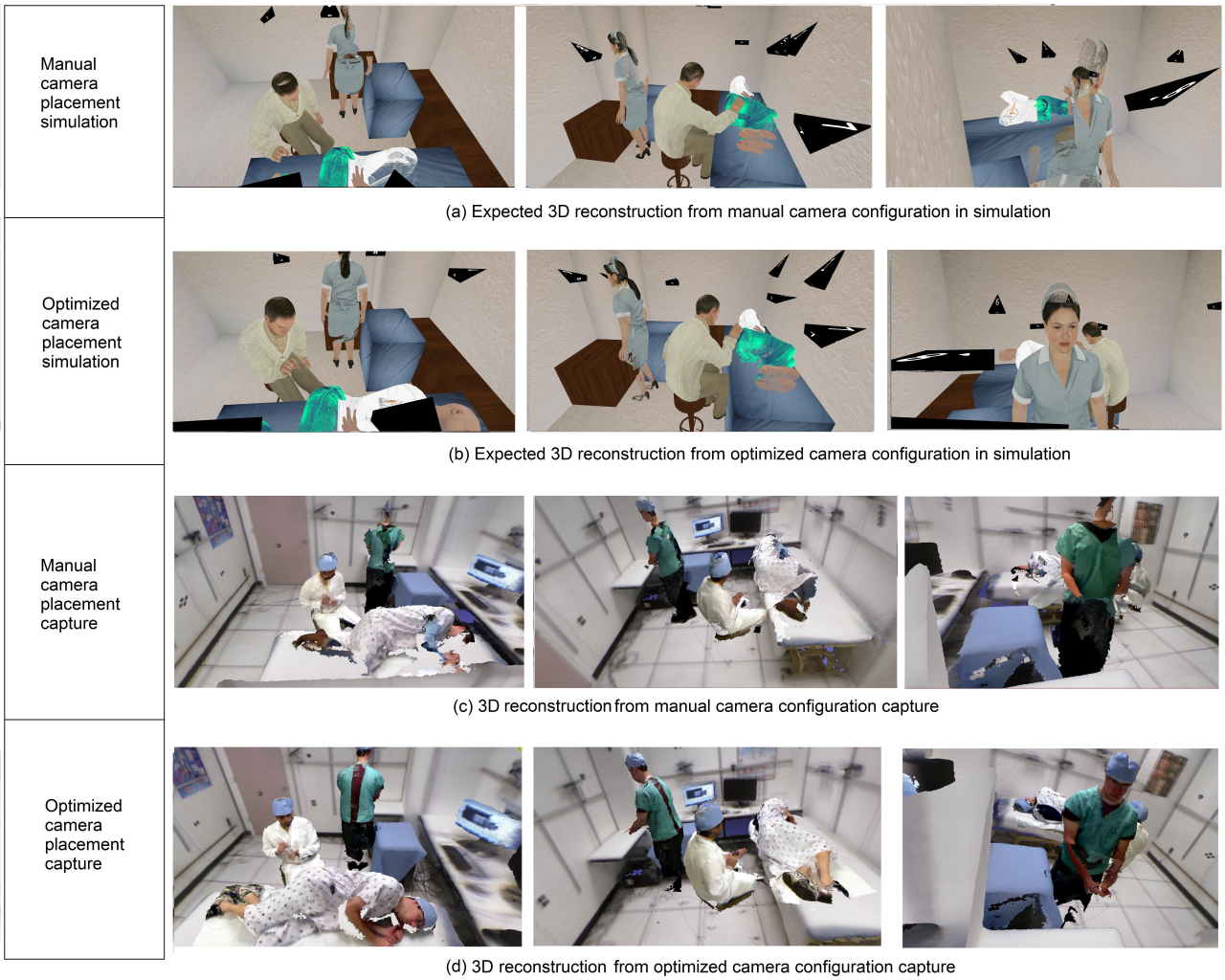


Figure 16: (a) and (b) show simulated reconstruction results; (c) and (d) show actual reconstruction results.

[2] S. Beck, A. Kunert, A. Kulik, and B. Froehlich. Immersive group-to-group telepresence. *Visualization and Computer Graphics, IEEE Transactions on*, 19(4):616–625, 2013.

[3] K. Berger, K. Ruhl, Y. Schroeder, C. Bruemmer, A. Scholz, and M. A. Magnor. Markerless motion capture using multiple color-depth sensors. In *VMV*, pages 317–324, 2011.

[4] R. Bodor, P. Schrater, and N. Papanikolopoulos. Multi-camera positioning to optimize task observability. In *IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 552–557, September 2005.

[5] S. P. Brooks and B. J. Morgan. Optimization using simulated annealing. *The Statistician*, pages 241–257, 1995.

[6] X. Chen. *Designing Multi-Camera Tracking Systems for Scalability and Efficient Resource Allocation*. PhD thesis, Stanford University, 2002.

[7] X. Chen and J. Davis. An occlusion metric for selecting robust camera configurations. *Machine Vision and Applications Journal*, 19(4):217–222, July 2008.

[8] M. Cubes. A high resolution 3d surface construction algorithm/william e. *Lorensen, Harvey E. Cline–SIG '87*, 1987.

[9] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996.

[10] J. Denzler, M. Zobel, and H. Niemann. Information theoretic focal length selection for real-time active 3-d object tracking. In *International Conference on Computer Vision*, volume 1, pages 400–407, October 2003.

[11] B. Deutsch, H. Niemann, and J. Denzler. Multi-step active object tracking with entropy based optimal actions using the sequential kalman filter. In *IEEE International Conference on Image Processing*, volume 3, pages 105–108, 2005.

[12] B. Deutsch, M. Zobel, J. Denzler, and H. Niemann. Multi-step entropy based sensor control for visual object tracking. *Lecture Notes in Computer Science*, 3175:359–366, 2004.

[13] M. Dou and H. Fuchs. Temporally enhanced 3d capture of room-sized dynamic scenes with commodity depth cameras. In *Virtual Reality (VR), 2014 iEEE*, pages 39–44. IEEE, 2014.

[14] M. Dou, L. Guan, J.-M. Frahm, and H. Fuchs. Exploring high-level plane primitives for indoor 3d reconstruction with a hand-held rgb-d camera. In *Computer Vision-ACCV 2012 Workshops*, pages 94–108. Springer, 2012.

[15] U. M. Erdem and S. Sclaroff. Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements. *Computer Vision and Image Understanding*, 103(3):156–169, 2006.

[16] S. Fleishman, D. Cohen-Or, and D. Lischinski. Automatic camera placement for image-based modeling. *Computer Graphics Forum*, 19(2):101–110, 2000.

[17] E. Hörster and R. Lienhart. On the optimal placement of multiple visual sensors. In *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, pages 111–120. ACM, 2006.

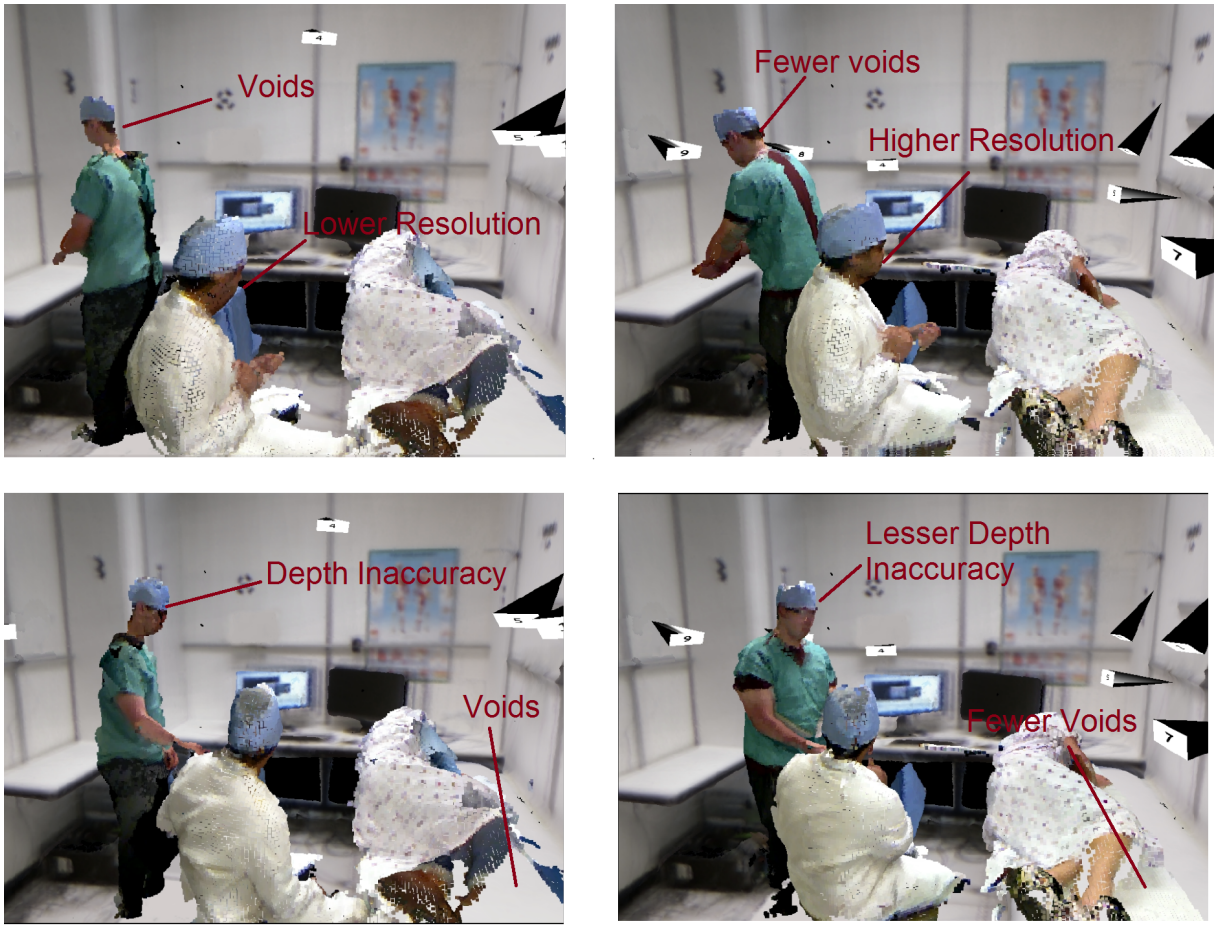


Figure 17: Two examples of reconstructed point clouds generated from captures with manually-designed and with optimized camera placements: (left column) manual camera placement; (right column) optimized camera placement.

[18] C.-R. Hwang. Simulated annealing: theory and applications. *Acta Applicandae Mathematicae*, 12(1):108–111, 1988.

[19] D. A. Ilie. *On-Line Control of Active Camera Networks*. PhD thesis, The University of North Carolina at Chapel Hill, 2010.

[20] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(Feb):235–284, 2008.

[21] C. Limin, C. Mingyu, X. Shizhe, and L. Yin. Analysis of interference between multiple kinect sensors and a noise reduction method for mobile robots. *Journal of Convergence Information Technology*, 7(21), 2012.

[22] A. Maimone and H. Fuchs. Encumbrance-free telepresence system with real-time 3d capture and display using commodity depth cameras. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 137–146. IEEE, 2011.

[23] A. Maimone and H. Fuchs. Reducing interference between multiple structured light depth sensors using motion. In *Virtual Reality Short Papers and Posters (VRW), 2012 IEEE*, pages 51–54. IEEE, 2012.

[24] A. Mavrinac and X. Chen. Modeling coverage in camera networks: A survey. 101(101):205–226, 2013.

[25] A. Mittal and L. S. Davis. A general method for sensor planning in multi-sensor systems: Extension to random occlusion. *International Journal of Computer Vision*, 76(1):31–52, January 2008.

[26] N. Naik, A. Kadambi, C. Rhemann, S. Izadi, R. Raskar, and S. Bing Kang. A light transport model for mitigating multipath interference in time-of-flight sensors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 73–81, 2015.

[27] P. Rahimian and J. K. Kearney. Optimal camera placement for motion capture systems in the presence of dynamic occlusion. In *Proceedings of the 21st ACM Symposium on Virtual Reality Software and Technology*, VRST '15, pages 129–138, New York, NY, USA, 2015. ACM.

[28] H. Sarbolandi, D. Lefloch, and A. Kolb. Kinect range sensing: Structured-light versus time-of-flight kinect. *Computer Vision and Image Understanding*, 139:1 – 20, 2015.

[29] E. Sommerlade and I. Reid. Information theoretic active scene exploration. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, May 2008.

[30] A. State, G. Welch, and A. Ilie. An interactive camera placement and visibility simulator for image-based vr applications. *Engineering Reality of Virtual Reality 2006 (3D Imaging, Interaction, and Measurement; IS&T/SPIE 18th Annual Symposium on Electronic Imaging Science and Technology)*, January 2006.

[31] K. A. Tarabanis, R. Y. Tsai, and P. K. Allen. A survev of sensor planning in computer vision. *IEEE Transactions on Robotics and Automation*, 11(1):86–104, February 1995.

[32] O. Wasenmueller and D. Stricker. Comparison of kinect v1 and v2 depth images in terms of accuracy and precision. In *Asian Conference on Computer Vision Workshop (ACCV workshop)*, 2016.

[33] J. J. Wu, R. Sharma, and T. S. Huang. Analysis of uncertainty bounds due to quantization for three-dimensional position estimation using multiple cameras. *Optical Engineering*, 37:280–292, January 1998.