Computers & Graphics 115 (2023) 181–190

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on CAD/Graphics 2023

# Optimizing retroreflective marker set for motion capturing props

Pedro Acevedo [a], Banafsheh Rekabdar [b], Christos Mousas [a,*]

[a] Purdue University, West Lafayette, IN, USA
[b] Portland State University, Portland, OR, USA

ARTICLE INFO

*Article history:*
Received 12 May 2023
Received in revised form 10 July 2023
Accepted 10 July 2023
Available online 13 July 2023

*Keywords:*
Motion capture
Props
Retroreflective markers
Marker placement
Optimization

ABSTRACT

One of the most widely used motion capture (MoCap) methods depends on retroreflective markers placed on an object. Through cameras, the MoCap system captures the moving prop's (3D object) motion. However, noise in MoCap data caused by the shape of the captured volume, motion between frames, ghost points, and markers' self-occlusion could impact the quality of the captured data. To improve the quality of capturing the motion of props, we tackle the problem of finding an optimal marker set configuration for a given input prop while considering various constraints. By "props," we mean any objects or handheld items of different shapes and sizes on which markers can be precisely placed to reduce MoCap errors. We propose an approach to optimize the placement of optical (retroreflective) markers over props while encountering various constraints, such as the visibility of markers, the number of makers used, the symmetry of the marker set, and markers' overlapping. We solve the marker set configuration problem using an optimization-based method, the reversible-jump Markov chain Monte Carlo. We provide marker set configurations for various props and constraints obtained through several simulations we ran to evaluate the performance of our method.



## 1. Introduction

Capturing the movement of humans or objects has been a well-studied field in the computer graphics and animation literature. Motion capture (MoCap) describes the process of capturing frames of movements of humans interacting with the environment, which are later mapped into 3D representations (e.g., virtual character) [1,2]. Nowadays, MoCap setups are relevant for applications in multiple fields, such as films and visual effects, sports, health, biomechanics, gaming, and immersive realities [3,4]. Among the optical MoCap solutions [5], one requires attaching several retroreflective markers to a human body to capture motion over time using a set of calibrated cameras [1]. These retroreflective markers are usually small spheres or blobs. The MoCap system is then used to detect a marker in at least two images simultaneously from the cameras to reconstruct the three-dimensional position of the marker [6]. Subsequently, the system uses the captured positions of the markers through sequences of images to represent the performer's movements.

Raw optical MoCap data are often erroneous due to marker occlusions or mislabeling from marker swapping during tracking, and require time-consuming manual post-processing [7]. Fixing the mistakes in labeled markers through denoising could have implications and is a highly costly computational process requiring sophisticated solutions, such as machine learning techniques [8–10]. Such algorithms require reliable training data, which can be complicated or expensive, considering the specific motion task.

Marker placement for human MoCap can be considered an easier task since: (1) all humans have the same structure, and their skeleton follows the same hierarchy and (2) there is a lot of documented knowledge resulting from years of work in human MoCap literature [1,11,12]. However, placing markers on props is challenging since each prop has its own unique shape, size, topology, and topological constraints. Thus, optimizing marker set placement is important to ensure efficient MoCap for any given prop. The props on the scenes are one of the major causes of missing markers during a MoCap session [13], making optimal marker set placement an essential factor in the MoCap process. To tackle this problem, we propose a method that finds an optimal marker set configuration for any given prop (3D object). To our knowledge, no prior work has focused on optimizing the marker set layout in props. We formulate the problem of optimizing a marker set configuration, considering various constraints as cost terms, including the number of markers, markers' visibility, markers' overlapping, and marker set symmetry. We solve the total cost function and provide optimal solutions using the reversible-jump Markov chain Monte Carlo optimization method [14]. Our method can provide multiple optimal marker set configurations for different props and constraints.

* Corresponding author.
*E-mail addresses:* paceved@purdue.edu (P. Acevedo), banafsheh.rekabdar@pdx.edu (B. Rekabdar), cmousas@purdue.edu (C. Mousas).

https://doi.org/10.1016/j.cag.2023.07.021
0097-8493/

To understand the performance of our method in optimizing a marker set for a user-defined prop, we measured its performance based on the total optimization time, and we evaluated our method by comparing the following:

- variations in the number of markers used per object;
- variations in the resolution of the tested captured area; and
- props with and without constrained areas.

MoCap specialists could use our method to provide optimal marker set configurations for any given prop. Thus, the specialist will not need to rely on empirical decisions and trial-and-error MoCap sessions to determine the performance and reliability of a defined marker set. Therefore, our method could save the time and effort required to prepare the props for a MoCap session.

The rest of the paper is structured as follows. In Section 2, we provide a review of related works on marker placement methods, camera placement, and MoCap of objects. In Section 3, we present an overview of the methods and the problem formulation. In Section 4, we present the optimization algorithm in detail. Then, we present the results from the different constraint comparisons in Section 5. In Section 6, we discuss the advantages and disadvantages of our proposed methods. Lastly, in Section 7, we conclude and present potential directions for future work.

## 2. Related work

In the following subsections we discuss related work to our project.

### 2.1. Marker placements methods

For MoCap sessions, the layout of the markers is relevant for the accurate reconstruction of motion. The placement of markers could cause multiple drawbacks, such as labeling problems, unnatural animation, and pose inaccuracy. The labeling process describes how the computer interprets the marker set data through timeframes to reconstruct the corresponding captured model, such as humans or objects [15]. This issue is investigated due to the time-consuming labeling process for larger datasets and mainly because of the costs of using the commercial software and licenses (e.g., Vicon Software [16]) required for this functionality [10]. To overcome the marker labeling problem, Ghorbani and Black [10] proposed a method for labeling raw MoCap point cloud sequences of the human body into markers called Solving Optical Marker-Based MoCap Automatically (SOMA). SOMA addresses the problem of robustly labeling human motion, considering noise and variations across subjects, motions, marker placement, marker density, and MoCap quality. The authors generated synthetic MoCap point clouds with realistic noise and trained a layout-specific network that could cope with realistic variations across an entire MoCap dataset. The dataset used for model selection and validation consisted of 215 sequences across four subjects, on average, using 40 markers. They used the HMD05 [17] database, which includes human MoCap data with an average of 40–50 markers placed on the skeletal kinematic chain model. As with any learning-based method, SOMA cannot generalize its results to new motions outside the training data.

Chatzitofis et al. [9] presented an approach employing machine learning and considering the costs of the equipment required for marker-based MoCap. The authors focused on a marker-based, low-cost method called DeMoCap. Their model learns the underlying structural relationships between the human body and marker set placement. The authors trained DeMoCap on a unique dataset of human motion data using a marker set composed of 53 markers. The retroreflective markers were 14 mm in size and attached to the performer's suit. The authors did not provide details about how they placed the markers on the performers. The usage of a unique dataset implies that the model needs to be trained for new configurations of markers.

Reconstructing human motion data could lack realism, lifeless results, and unnatural motions. To overcome this problem, Loper et al. [18] proposed a method to capture the skeleton's movement and the human body's shape. They defined motion and shape capture (MoSh) to compute body shape and marker locations to estimate body pose throughout a MoCap session. Their method simulates markers during frames based on Powell's dogleg method with Gauss–Newton Hessian approximation. As a contribution, the authors designed an optimization algorithm using a greedy method to estimate the adequate number of markers that need to be placed on the human body to capture soft tissue deformation. They used a maximum number of markers (144 markers) and compared them to a dataset of human models (165 models in total). Their results showed the effectiveness of the proposed method with an optimized subset of markers (67 markers).

Liu and McMillan [13] explored another marker layout problem: the missing marker problem. They proposed a linear modeling approach to recover markers during frames for MoCap systems. Their implementation considers a global principal component analysis of the training data from the motion sequences of markers' positions. Their method employs a random forest classifier to determine the positions of the markers in incoming scenes to provide accurate human motion reconstruction. In each experiment, they randomly chose a fixed number of markers to be missing for one second (120 frames) for every testing motion sequence. Additionally, they considered a variety of motions for the Carnegie Mellon University (CMU) MoCap database [19]. This dataset is composed of 41 markers placed on actors' bodies. Their model focused on the marker's positions, and the error minimizations were calculated based on the millimeters of difference between the prediction and the real data.

MoCap systems have been used to capture detailed human motions, such as hands [20,21] or facial expressions [22,23], with applications in emotion recognition, grabbing, facial animation, and others. Zell and McDonnell's [23] proposed a novel algorithm for computing minimalistic facial landmark layouts specific to a blendshape model. Blendshape interpolation is the dominant approach to facial animation and provides the model's degrees of freedom. The optimization method categorizes potential candidates for facial landmarks and computes their quality to reconstruct captured facial motion. Despite the NP-hard nature of the problem (such as the weighted set cover problem), the authors computed a globally optimal solution using state-of-the-art mixed integer solvers. Their method considers constraints related to symmetry, distances between markers, and blendshape displacement for their marker positioning. The authors recognized the inaccuracies introduced by manual marker placement and retargeting issues between different characters from previous datasets, which they considered inappropriately placed over the blendshape.

Schröder et al. [21] considered optimizing marker layouts to capture the hand. They generated sparse marker configurations that were optimal for solving the constrained inverse kinematics problem. The optimization finds the minimum cost through the particle swarm optimization scheme. The penalties they considered for the cost's functions are related to the constrained areas of the hand to generate well-conditioned marker sets for real usage and sparse between markers, maximizing the minimum distance between them. Their results show that the reduced marker layouts (comparison between 17, 8, and 6 markers) can reconstruct hand motions robustly and accurately from sparse marker set input.

### 2.2. Camera placement

Camera placement for the tracking system is a widely studied area. The well-known *art gallery problem* establishes the problem of finding the minimum number of cameras that can monitor a fixed number of paintings in a gallery [24]. Researchers have also explored the placement of cameras around the captured area or environment based on the position, angle, field view, and other properties of the camera's configurations [25]. Rahimian and Kearney [26] proposed an automatic method for placing cameras in cave automatic virtual environments (CAVE) [6]. The authors focused on two main errors in the camera layout: marker visibility and triangulation accuracy. Their method maximizes the visibility of points with minimal 3D reconstruction errors in the presence of dynamic occlusion. The simulated annealing algorithm was employed to find a globally optimal solution. The occlusion-based method was tested with synthetically generated target point sets and point sets from a MoCap session of a person with a body suit covered with reflective markers walking in their CAVE. According to the authors, for a setting with 17 cameras, their method would need 2–5 h to produce results.

Another approach exploring camera positioning was demonstrated by Ercan et al. [27], who focused on multiple camera placement and performance over the sensor network approach for tracking single objects. Instead of tracking all objects in the environment, which is computationally very costly, the cameras track only the target object and treat others as occluders. The camera network's task is to track an object on the ground plane in the presence of static occluders and other moving objects. The authors explored the trade-offs involving the occluders' known positions, the number of occludes, the tracking accuracy, and the number of cameras. They focused on multiple simulations. One of the simulations consisted of a captured area of 100 $\times$ 100 units with known occludes and eight cameras placed around its periphery. They simulated moving occluders and found metrics related to the sensors' accuracy and energy consumption. The authors validated multiple algorithms, among which greedy selection performed close to the brute force method and outperformed the other selected heuristics. The authors did not provide details about the camera placement.

Aissaoui et al. [25] proposed another optimization approach to obtain an optimal camera placement. They used a guided genetic algorithm (GGA) to simulate the best camera configuration for MoCap applications. Their approach examined capturing markers attached to virtual humanoids performing movements similar to those in the real world. Regarding markers, they employed 54 markers or "tags" for the model and did not provide details about the chosen marker set and its placement. Their algorithm considers the camera's positions, angles, rotations, and error metrics, such as occlusion, resolution, and field of view. The GGA approach reaches a maximum rate of recovered tags (100%) by using only four or six cameras with a runtime of 15 min. Their defined camera placement influenced the final capture of the markers. Considering the previously discussed work, the cameras' positions and the visibility of the markers are essential factors that could influence the MoCap process. Thus, in our implementation, we decided to use an arrangement of eight cameras, which we considered to be a usual MoCap setup [2].

### 2.3. Motion capture on objects

Prior research relating to the manipulation of objects has focused on how humans interact with objects applied to animation and robotic fields [28]. The grab interaction made during human–object interactions is useful for graphics and robotics to help 3D characters or robots interact with their surroundings [29]. On objects, there are datasets focused on improving human–robot interaction in household settings. Thus, Brahmbhatt et al. [29] explored the phenomenon of thermally observable contact with household objects. They proposed a novel dataset of contact maps for household objects that captures the contact handled from a thermal camera record. A contact map is a thermal representation of the area occupied by the human hand during grasping. Furthermore, they used a generative adversarial network-based image-to-image translation to predict contact maps from object shapes. For the data collection, they used RGB-D and a thermal camera calibrated rig. They defined an object representation oriented to real-world robotics scenarios in which mobile manipulators are often required to grasp objects after observing them from a single view.

Considering object manipulation for human animations, Taheri et al. [30] described the creation of a new dataset with full-body motion for human–object interaction called GRAB. They used the extended version of the MoSh++ algorithm to estimate the 3D shape and motion of the body and hands from MoCap markers, including facial performance. For the MoCap setting, they used small hemispherical markers on the objects and showed that these did not affect grasping behavior. They attached 99 retroreflective markers per subject: 49 for the body, 14 for the face, and 36 for the fingers. They used spherical 4.5 mm radius markers for the body and hemispherical 1.5 mm radius markers for the hands and face. Also, they attached 1.5 mm radius hemispherical markers directly on the object surface to capture object motion. Per objects, they considered place at least eight markers. Additionally, the marker's size and chosen placement made them unobtrusive for the capturing task. The authors empirically distributed the markers on the object, considering that the cameras could observe at least three of them. The authors could capture the full-body motion during different grasping tasks with the previously described configurations. The GRAB dataset could be employed for learning human–object interaction models, robotic grasping mimics, mapping MoCap markers to meshes, and other applications.

## 3. Method overview

### 3.1. Preliminaries

We developed a pipeline composed of several steps. We illustrate our pipeline in Fig. 1. In the initial step in our pipeline, we asked users to provide a prop (3D mesh) of the desired object they wanted to capture later during the MoCap process. We predefined an experimentation setup by defining the number of cameras, their positions in the room setting, and the expected capture area. We preprocessed the 3D mesh by sampling it—to retrieve potential positions so that our method could place the markers—using the constrained Poisson disk sampling method [31]. For all results we present later in this paper, we sampled all meshes by requesting the Poisson disk sampling to generate 1000 sample vertices. We considered this number to provide enough resolution for the mesh for our experiments. However, a developer could request a different sample size based on the size and complexity of the input prop. Given the sampled vertices of the input 3D mesh and an initial random marker set, the proposed optimization approach searches for the optimal marker set configuration by updating the marker set configuration using one of the moves: add, remove, or modify. The proposed marker set is then evaluated by the total cost function taking into account the defined cost terms. Our algorithm computes the score of the current marker set layout and verifies if it is a better solution. If the current marker set is not better than the previously proposed one, it rejects the proposed configuration and proposes a new one by choosing one of the defined moves. We provide more details about the optimization procedure in Section 3.2. As an output, our algorithm provides an optimal marker set configuration for the input 3D prop.

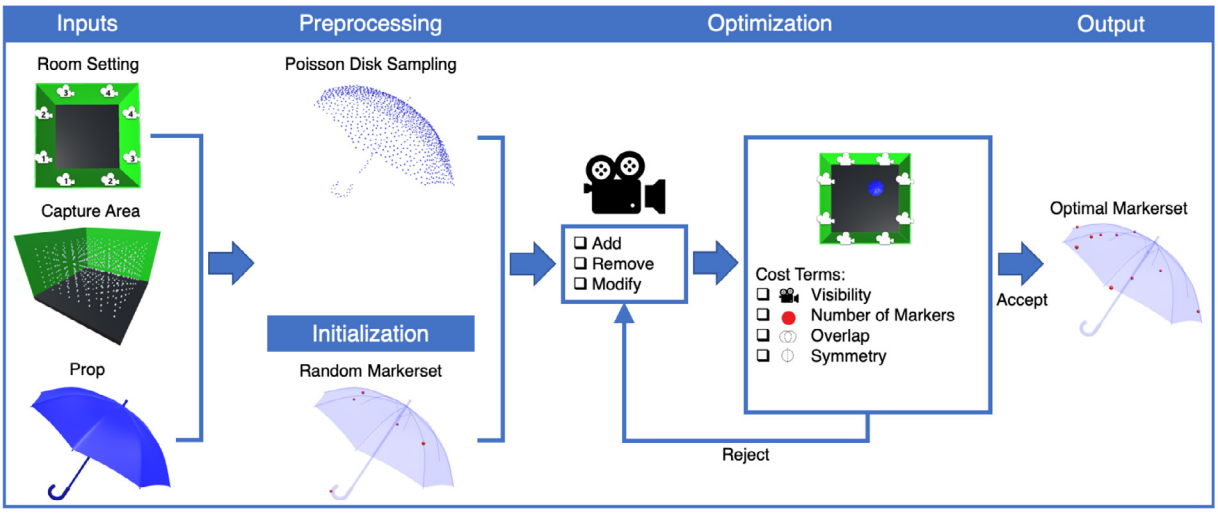


**Fig. 1.** Our pipeline includes the following steps: (1) **Input:** the required inputs for our method that include room settings parameters, configured capture area, and 3D object model; (2) **Preprocessing:** the sampling of the 3D mesh by user-specified number of vertices; (3) **Initialization:** configuration of an initial marker set over the 3D mesh; (4) **Optimization:** we use the MCMC algorithm to optimize the marker set configuration according to three moves (add, remove, and modify), and we evaluate the acceptance of the generated marker set based on cost terms (visibility, number of markers, overlap, and symmetry); and (5) **Output:** the optimal marker set for the given input prop.

### 3.2. Problem formulation

Let $M = \{m_1, m_2, \ldots, m_D\}$ denote a set of $D$ retroreflective markers consisting of the markers placed around the object. Each marker $m_i$ is placed in one of the sampled vertices of the input prop, so a single marker $m_i$ is represented by a position $P(m_i) = (x_i, y_i, z_i)$. Our method evaluates the quality of the marker set $M$ by a total cost function $C_{Total}\,(M)$:

$$C_{Total}\,(M) = \mathbf{C}_L\mathbf{w}_L^T + \mathbf{C}_P\mathbf{w}_P^T, \tag{1}$$

where $\mathbf{C}_L = [C_L^v, C_L^n]$ is a vector of the layout cost and $\mathbf{w}_L = [w_L^v, w_L^n]$ is a vector of weights. $C_L^v$ and $C_L^n$ encode the marker layout configuration: the camera's visibility of the marker arrangement and the number of markers placed on the object. $\mathbf{C}_P = [C_P^o, C_P^s]$ is a vector of penalty decisions over the marker placement configuration, and $\mathbf{w}_P = [w_P^o, w_P^s]$ is a vector for the weights of those costs. $C_P^o$ and $C_P^s$ encode the penalties considered in the optimization process, which are the overlapping cost and the symmetry of the marker set respectively.

#### 3.2.1. Layout costs

Our method evaluates each marker set $M$ over a camera configuration. In a MoCap studio, the cameras should be placed in a way that ensures: (1) maximization of the captured volume and (2) overlap between pairs of cameras' captured volumes to allow reconstruction of the marker's position through triangulation. Consider the set of camera pairs $C = \{cp_1, cp_2, \ldots, cp_{cn}\}$ in which $cp_i = (c_j, c_k)$ represents a camera pair placed in the captured area, $c$ is a single camera, and $cn$ the total number of camera pairs. In this case, we consider $cn = 4$, so the configuration comprises eight cameras. In our project, we arranged the camera pairs based on the suggestion of Rahimian and Kearney [26] and the findings of Olange and Mohr [32], which mentioned that placing camera pairs in angles with view vectors that are sufficiently non-parallel, as illustrated in Fig. 2, avoids triangulation error propagation and provides a correct triangulation calculation.

***Visibility cost:***. To reconstruct the position of the marker through the cameras from world space coordinates, we considered the visibility of the marker set $M$. The visibility of the marker set refers to the average visibility of markers (obtained through triangulation) per camera pair in a user-defined sampled resolution in the captured volume (see Fig. 3 for different resolutions of the captured area). Specifically, we denote as $P$ the set of sampled points $[p_1, p_2, \ldots, p_D]$ in the captured volume in which an examined prop iterates over. For example, suppose the user specifies four sample points per axis ($4 \times 4 \times 4$). In that case, our method will sample the capture volume by $D = 64$ points. We then used each of these points to evaluate the visibility of the marker set by testing the props in different positions in the 3D room to approximate the object's coverage from multiple camera perspectives. We expressed this visibility coverage by the following cost:

$$C_L^v = \left| \frac{1}{|P|} \sum_{p_i \in P} \frac{1}{|M|} \sum_{m_i \in M} \frac{1}{|C|} \sum_{cp_i \in C} \Delta\ (p_i, m_i,\ cp_i) - \rho_v \right|, \tag{2}$$

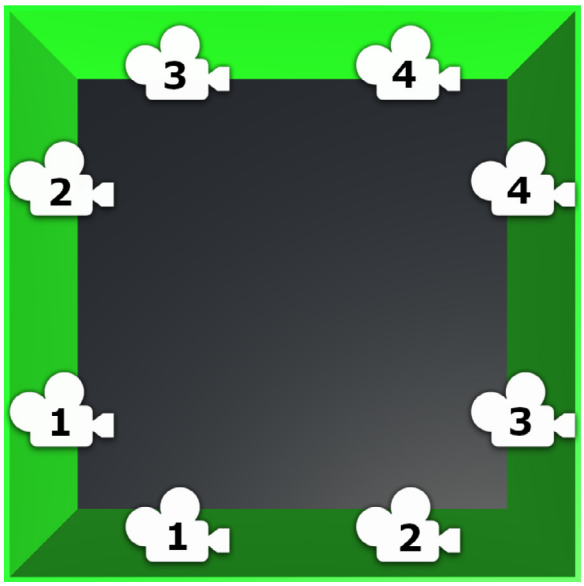


**Fig. 2.** The placement of cameras and the pairs between them in the examined captured area.

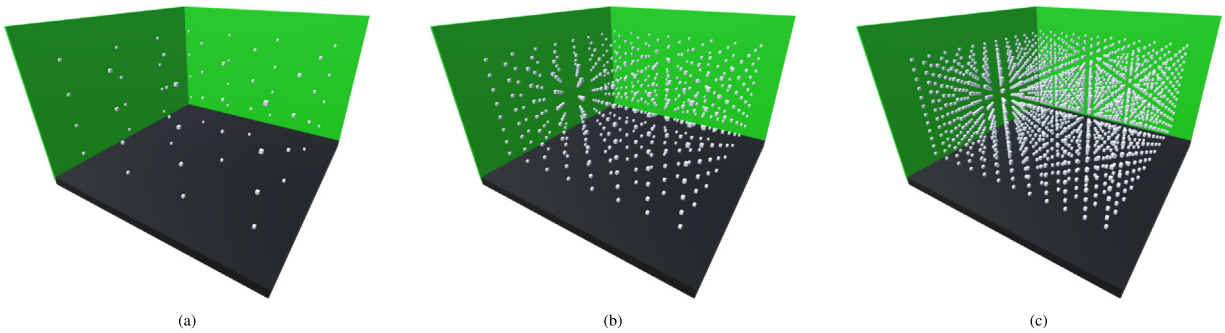


**Fig. 3.** Examples of different capture area volumes: (a) 4 × 4 × 4, (b) 8 × 8 × 8, and (c) 12 × 12 × 12 sampled resolutions.

where $\rho_v$ is the target of the average visibility of the marker set, and $\Delta\,(cp_i, m_i, p_i)$ represents the computation of the triangulation of each camera pair in $C$ to locate a single marker $m_i$ in the translated object position $p_i$, which we defined by the following statement:

$$\Delta\,(cp_i, m_i, p_i) = \begin{cases} 1 & m_i \text{ is visible at position } p_i \text{ from a camera pair } cp_i \\ 0 & \text{otherwise.} \end{cases}$$

Specifically, our method computes this triangulation by projecting a camera ray $c_j$ and $c_k$ from a camera pair with $j \neq k$ and $j, k \leq |C|$ to the current marker $m_i$. We expect that a marker $m_i$ should be located in the 2D projection of the captured area for a pair of cameras $c_j$ and $c_k$.

***Number of markers cost:***. We allow the user to specify the maximum number of markers placed on a prop. Thus, we defined the number of makers' costs as follows:

$$C_L^n = 1 - \exp\left(-\frac{1}{2\sigma^2}\ (|M|\ -\ \rho_n)^2\right), \tag{3}$$

where $\rho_n$ is the target number of markers to be placed on the prop, $|M|$ denotes the total number of markers currently placed on the prop, and $\sigma$ controls the spread of the Gaussian penalty function, which we empirically set as $\sigma = 1.0$.

### 3.2.2. Penalty costs

Failures during marker labeling could happen due to marker swapping in mid-sequence or introducing new misplaced markers. These errors could occur when markers pass within a small distance from each other and, as a result, swap paths due to numerical inaccuracies in the system. For the penalty costs, we primarily focus on markers' placement to avoid overlapping (possible marker swapping) and the occurrence of symmetry between the resulting marker set positions (MoCap system errors).

***Overlap cost:***. The optical markers attached to the object must not overlap. To avoid the placement of markers in the same or adjacent vertices of the mesh, we defined the overlapping penalty as follows:

$$C_P^o = \left| \frac{1}{N} \sum_{i=1}^{|M|} \sum_{j=1}^{|M|} OVL(m_i,\ m_j) - \rho_o \right|, \tag{4}$$

where $N$ represents the total number of evaluated marker comparisons without considering the case $i = j$; so, in this case, $N = \frac{|M|!}{(|M|-2)!}$. $\rho_o$ is the target overlap between the markers, where $\rho_o = 0$ means that we expect no overlapping between markers, whereas a higher value means that we have less restriction on the overlap between markers. Moreover, the *OVL* operation detects whether there is an overlap between pairs of markers in the marker set $M$ as follows:

$$OVL(m_i, m_j) = \begin{cases} 1 & \text{overlapping between a pair of markers } m_i \text{ and } m_j \\ 0 & \text{otherwise.} \end{cases}$$

For the *OVL* operation, we used a sphere–sphere intersection algorithm[1] to evaluate whether the markers $m_i$ and $m_j$ intersect. For the intersection algorithm calculation, a marker is represented as a sphere of a 2 mm radius.

***Symmetry cost:***. As we stated before, the set of markers $M$ contains the marker set configuration on the 3D mesh. Suppose there is evidence of symmetry in the marker set configuration. We consider this a drawback due to possible ambiguities [33], since the MoCap system, in these cases, is not able to accurately understand the direction of the captured object (might consider by mistake the symmetrical representation of the object).[2] Therefore, we defined a penalty cost for the set of $P\,(m_i)$ points resulting from marker set $M$ as:

$$C_P^s = |\ SMC\ (M) - \rho_s|\,, \tag{5}$$

where $\rho_s$ is the target symmetry for the marker set, and *SMC* (Symmetry Marker Calculator) is an operation that determines if there is a symmetry in the evaluated marker set $M$ and it defines the following condition:

$$SMC(M) = \begin{cases} 1 & \text{symmetry found } \forall P(m_i) \text{ points} \\ 0 & \text{otherwise.} \end{cases}$$

The mentioned operation detects the symmetry of the current marker set configuration by rotational transformations per 3D axis. For symmetry detection, we used the theory of rotational symmetry [34]. For a marker set (set of 3D points), we determined a symmetry expressed by $k$ rotations, named the "$k$-fold line of rotational symmetry" [35]. This method considers a $k$-fold rotation if there is a way to rotate the marker set $(360/k)$ degrees around an axis where we can match the exact positions of the markers. To find this property, we defined a $k$ constant that establishes the number of rotations to evaluate over $x$, $y$, and $z$ axis to look for possible symmetries between the markers.

To validate symmetry, we used the Knuth Morris Pratt (KMP) string-searching algorithm [36]. Specifically, we considered $L = l_1 l_2 l_n$, with $l_i$ as a string representation of a 3D coordinate and $c_k$ as a rotation operation. We then affirmed that $L'$ is the rotation of $L$ by $k$ through $L' = c_k(L)$. Given both $L$ and $L'$, we found a rotation $k \in \{1, 2, \ldots, 360\}$ such that $L' = c_k\,(L\,)$. To find the rotation, we constructed the string $LL$ by concatenating $L$ with itself and then reporting all occurrences of $L'$ in $LL$. If we find an $L'$ in $LL$, we find

[1] https://mathworld.wolfram.com/Sphere-SphereIntersection.html.
[2] https://docs.optitrack.com/motive/markers#marker-placement.

a $k$-fold rotational symmetry. To represent the marker set $M$ as a string, we transformed the 3D coordinates of a marker $P(m_i)$ to a cylindrical coordinate $(r, \varphi, z)$. Each marker's new representation was then sorted by the radial distance $r$ and concatenated in a long string $L$. Each $k$ rotation generates an $L'$, rotating the original marker set positions $P(m_i)$ and then transforming them into the string. If we find one of those transformations' strings with the KMP algorithm in $LL$, then the $SMC$ output will be set as stated in the condition mentioned previously ($SMC(M) = 1$).

## 4. Optimization

We found an optimal configuration of marker set for a given prop by optimizing the total cost function $C_{Total}(M)$. We employed a Markov chain Monte Carlo (MCMC) method called simulated annealing (SA) with a Metropolis–Hastings state searching step. As an input object could be composed of an arbitrary number of markers in the arbitrary mesh vertices position, our optimization searches in a trans-dimensional solution space to find the optimal solution. To effectively sample the solution space of marker configurations, we employed the reversible-jump MCMC method [14]. Thus, we defined a Boltzmann-like objective function:

$$f(M) = exp\left(-\frac{1}{t} C_{Total}(M)\right), \tag{6}$$

where $t$ is the temperature parameter of SA. SA randomly selects a move at each algorithm step to refine the current marker set $M$. The movements used by the SA algorithms are as follows:

- **Add a Marker:** A marker is placed on a randomly chosen vertex of the sampled prop and added to the proposed marker set $M$.
- **Remove a Marker:** A marker is randomly selected and removed from the proposed marker set $M$.
- **Modify a Marker:** A marker is randomly selected from the current marker set $M$ and placed in a randomly chosen vertex of the prop to create a new proposed marker set $M$.

We defined the add, remove, and modify moves selection probabilities as $p_a$, $p_r$, and $p_m$ respectively. For this implementation, we set $p_a = .30$, $p_r = .30$, and $p_m = .40$. This means that the modified move is more likely to be selected as the next step during the optimization process. The new configuration $M'$ is evaluated using the proposed cost function, $C_{Total}(M')$. If the total cost $C_{Total}(M')$ of the new configuration is better than the best configuration found so far $C_{Total}(M)$, then the algorithm proceeds to use it as a new configuration. Further, if the proposed configuration $M'$ has a higher total cost than the best configuration $M$, SA accepts the new configuration with some probability, depending on the current temperature. In this solution space exploration, considering the trans-dimensionality of the problem, our approach accepts the proposed marker set $M'$ with the following acceptance probability $\Pr(M'|M)$ specified based on the Metropolis criterion: for the Add a Marker move:

$$p_a(M'|M) = \min\left(1, \frac{p_r}{p_a}\frac{Z - |M|}{|M'|}\frac{f(M')}{f(M)}\right); \tag{7}$$

for the Remove a Marker move:

$$p_r(M'|M) = \min\left(1, \frac{p_a}{p_r}\frac{|M|}{Z - |M'|}\frac{f(M')}{f(M)}\right); \tag{8}$$

for the Modify a Marker move:

$$p_m(M'|M) = \min\left(1, \frac{f(M')}{f(M)}\right). \tag{9}$$

**Table 1**
The default configurations for the optimization.

| Parameters | Default values |
|---|---|
| Target average visibility of marker set ($\rho_v$) | 1 |
| Target number of markers ($\rho_n$) | 7 |
| Target overlap between markers ($\rho_o$) | 0 |
| Target symmetry of the marker set ($\rho_s$) | 0 |
| Marker interval | [4, 20] |
| Evaluated positions | 64 |
| Captured area dimensions | $2 \times 2 \times 2$ m |
| $k$-fold symmetry | 8 |
| Camera field of view | 60° |
| Marker size | 2 mm |

We assumed that we could only use $Z$ markers rather than an infinite number so that the dimensionality of the solution space has an upper limit. Thus, we restricted the current marker set configuration $M$ by a maximum number of elements by $Z$. We set a maximum number of markers as $Z = 20$ for our experiments' possible marker set.

We used this Markov chain-based algorithm to explore the solution space to find a minimum efficiently. The temperature parameter $t$ of the simulated annealing helps our algorithm overcome local minima and find an optimal solution. At the beginning of the optimization, we set $t$ to be high, so the optimizer aggressively explores the solution space, accepting multiple configurations that do not necessarily meet the requested constraints. Throughout the optimization, the temperature $t$ is lowered until it reaches zero. We initialized the temperature $t = 1.00$ at the beginning of the optimization process and then reduced every 50 interactions by .25. When the algorithm reaches 200 iterations, the temperature value is restricted to zero, constraining the algorithm from accepting less optimal solutions. The algorithm terminates after the cost difference of the last 50 iterations is less than 3%.

Unless otherwise specified, we set the weights as $w_L^v = 1.00$ and $w_L^n = .80$ in our optimization. We set the weights of all penalty terms as $w_P^o = 1.00$ and $w_P^s = 1.00$. Moreover, we established the interval limits for the markers used in our method and defined an interval between 4–20 markers for optimal placement. In addition, we set the radius of the marker to 2 mm. The designer can modify these parameters to emphasize confident design choices or constraints by adjusting the weight values.

## 5. Evaluations and results

We implemented our approach on an Alienware PC with an Intel Core i7-8700K CPU and 32 GB of memory. We implemented the optimization framework in C# through the Unity game engine. Our interest in the algorithm implementation is to place the optical markers optimally over the input props. We tested 10 props with different configurations and constraints defined for the optimization process. We used these props as inputs in our method, and we ran 10 optimization tests per prop to collect data related to the algorithm's performance. We describe the results in the following subsections. The default configurations of our method is provided in Table 1.

In the default setting, the algorithm performed a single iteration in 2.17 s, and to provide an optimal marker set configuration, it required 585.44 s on average. The optimization required time to compute the costs due to the per iteration validation of the resolution of the capture area used to calculate the visibility costs. We evaluated the number of markers used, the capture area resolution, and the constraints over the meshes. For the 3D meshes used in the analysis, we focused on props that humans could manipulate during a MoCap session (e.g., umbrella, box). In

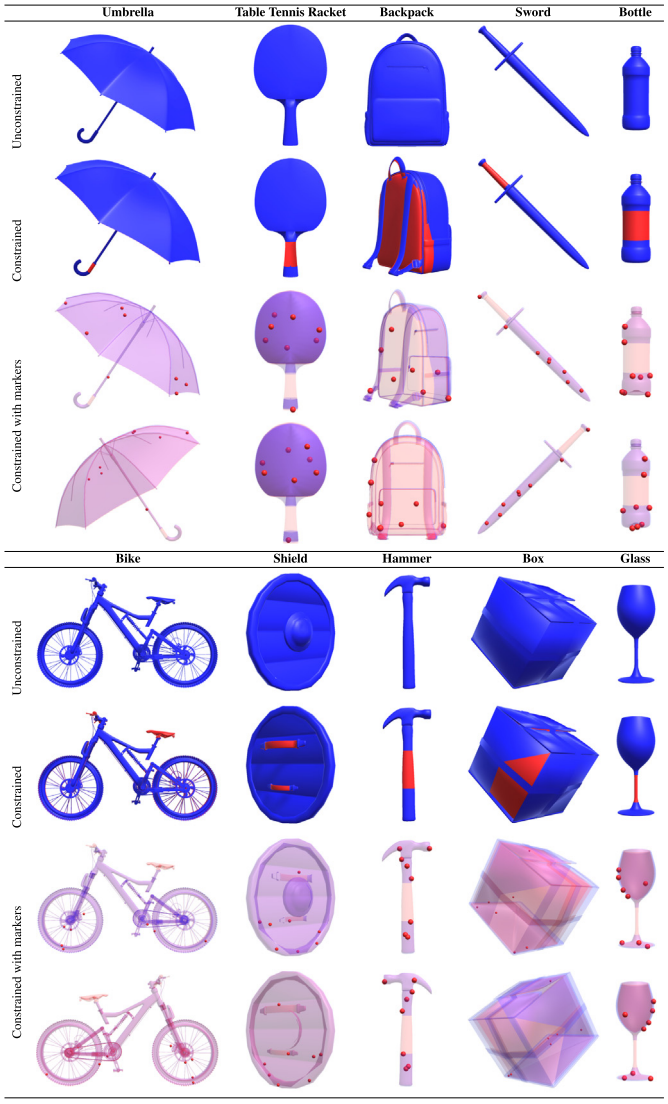


**Fig. 4.** The input props, the props with constrained areas, and an optimal marker set configuration proposed by our method.

Fig. 4, we provide a list of the input props with the corresponding constraint mesh layout and the optimal marker placement in the constrained props (props with areas where markers cannot be placed).

### 5.1. Evaluating the number of markers

For this evaluation, we considered the number of markers targeted for optimization. We used three marker targets (5, 10, and 15 markers) based on the defined interval limits (see Table 1) for optimal marker placement. We summarize the obtained marker sets for different numbers of marker targets in Fig. 5, and Fig. 6 provides the average cost for each object in the evaluated marker number.

We used a one-way repeated measures analysis of variance (ANOVA) to explore potential differences among the targeted input variants. The analysis indicated a significant effect of the number of markers used on the total cost achieved by the optimization algorithm (Wilks' $\Lambda = .248$, $F[2, 8] = 12.135$, $p = .004$, $\eta_p^2 = .752$). The post hoc comparison showed that the performance of the 10 markers ($M = .09$, $SD = .03$) was

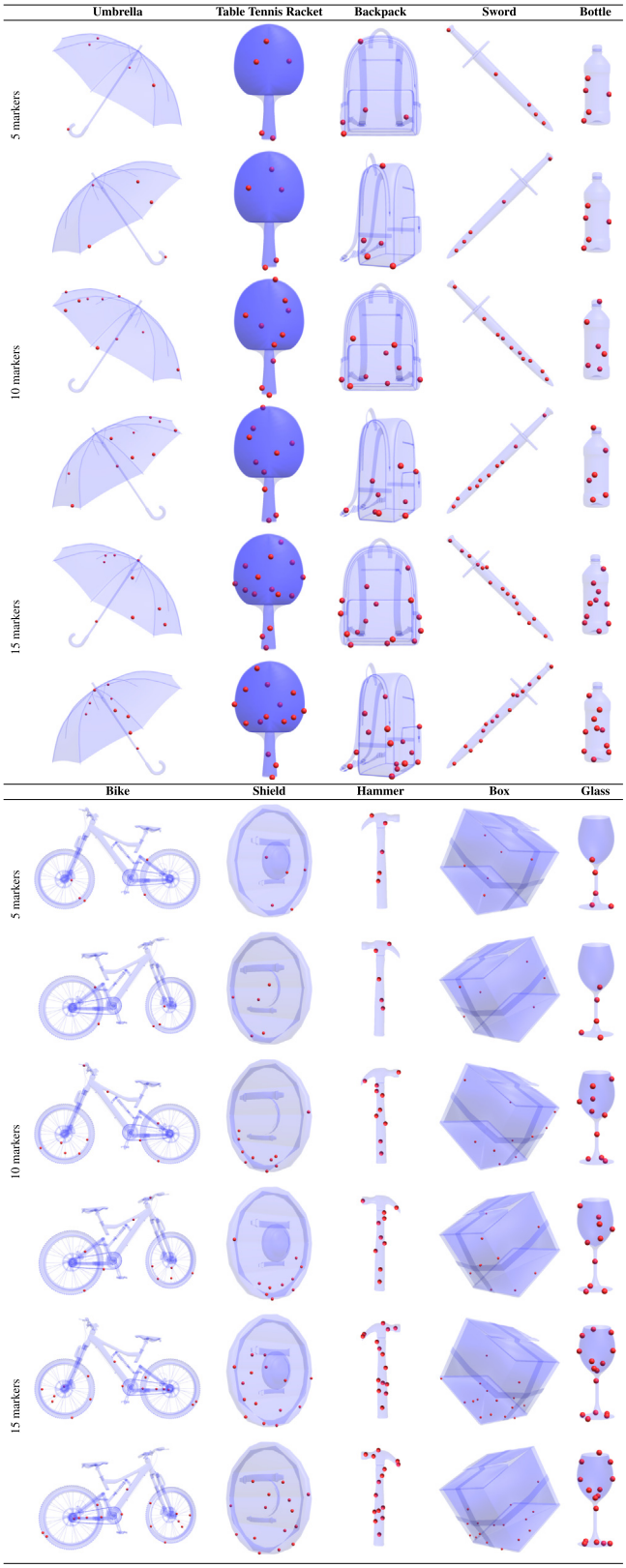


**Fig. 5.** Output props with the optimal marker placement by a different number of targeted markers.

statistically significantly lower than the 15 markers ($M = .03$, $SD = .22$) at $p = .003$. There was no statistically significant

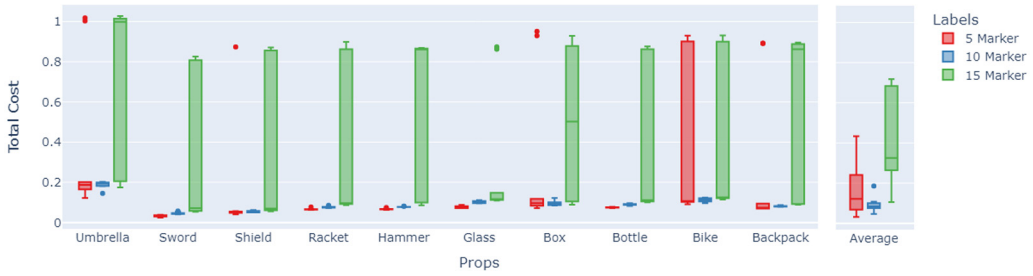


**Fig. 6.** Average cost per prop for a different target number of markers and the overall average of all props.

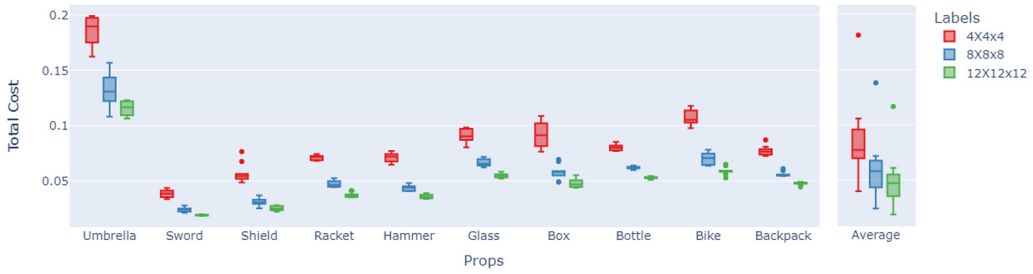


**Fig. 7.** Average cost per prop for different volume areas and the overall average of all props.

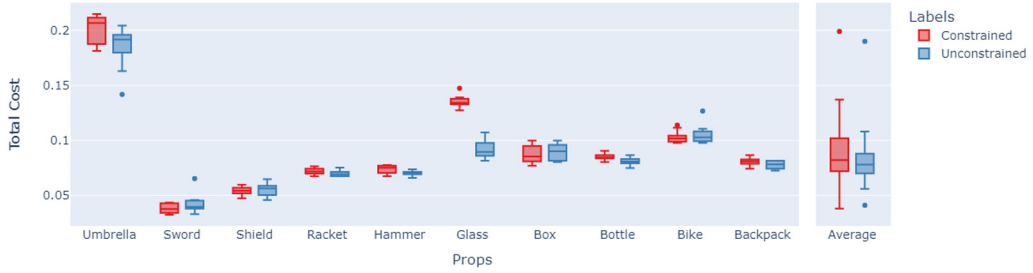


**Fig. 8.** Average cost per both constrained and unconstrained prop, and the overall average of all props.

difference between the performance of 5 markers ($M = .18, SD = .14$) and 10 markers ($p = .255$) or between the performance of the 5 markers and 15 markers ($p = .067$).

### 5.2. Captured area resolution

In this evaluation, we considered the resolution of the captured volume (see Fig. 3) used to calculate the visibility cost. The selected targeted resolutions of the area are $4 \times 4 \times 4$, $8 \times 8 \times 8$, and $12 \times 12 \times 12$ sampled resolutions. In Fig. 7, we provide the results of this evaluation for each object.

We ran a one-way repeated-measures analysis of variance (ANOVA) to explore potential differences in the optimal total cost across the three examined volume areas. The analysis revealed statistically significant results across the three examined volumes (Wilks' $\Lambda = .097$, $F[3, 7] = 37.362$, $p < .001$, $\eta_p^2 = .903$). The post hoc comparison showed that the average cost for $12 \times 12 \times 12$ ($M = .05$, $SD = .02$) was significantly lower than that for $8 \times 8 \times 8$ ($M = .06$, $SD = .03$) at $p = .036$ and $4 \times 4 \times 4$ ($M = .08$, $SD = .03$) at $p = .000$. Moreover, the average cost for $8 \times 8 \times 8$ was statistically significantly lower than that for $4 \times 4 \times 4$ at $p = .000$.

### 5.3. Constrained meshes

We also evaluated the performance of our algorithm to find an optimal marker set when we used a constrained and an unconstrained prop as input (see Fig. 4 for the constrained and unconstrained props we used in this experiment). We present the results for this comparison in Fig. 8.

We used a paired sample t-test to explore a potential difference in optimizing constraints and unconstrained props. We did not find a significant difference ($t[9] = 1.33$, $p = .216$) between the constrained ($M = .09$, $SD = .04$) and the unconstrained ($M = .08$, $SD = .04$) props.

### 5.4. Ablation test

We conducted an ablation test to evaluate and understand how the proposed cost terms contribute to the final result. The test consists of removing cost terms from Eq. (1) and running simulations with the remaining ones. Fig. 9 shows the method's performance without the visibility, overlapping, and symmetry cost terms. The figure shows an optimal marker set in each of the props, although some issues are highlighted in the optimized results. Specifically, inconsistencies in the marker set include visibility problems, such as the position of the marker inside the umbrella or on the bottom of the box, overlapping markers on the backpack and sword, and symmetry problems on the table tennis racket and sword. We argue that such issues could cause errors during the motion capture process.

### 5.5. Expert evaluation

We evaluated our method against a MoCap expert to show how our optimization-based method could outperform an empirical method of placing markers on the props. We considered all

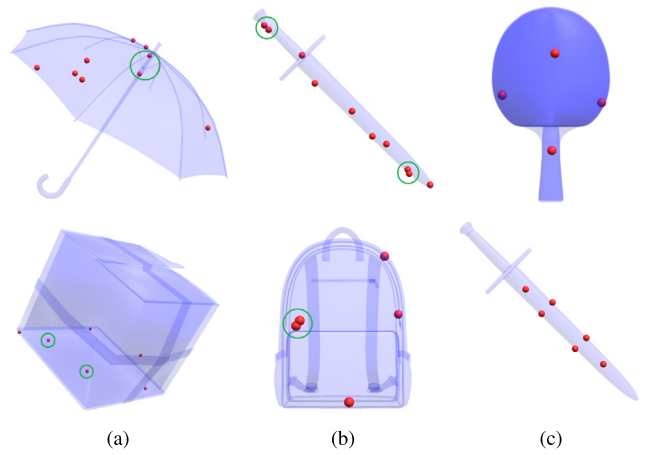


**Fig. 9.** Marker set configuration for different ablation tests (a) without visibility term, (b) without overlapping term, and (c) without symmetry term.

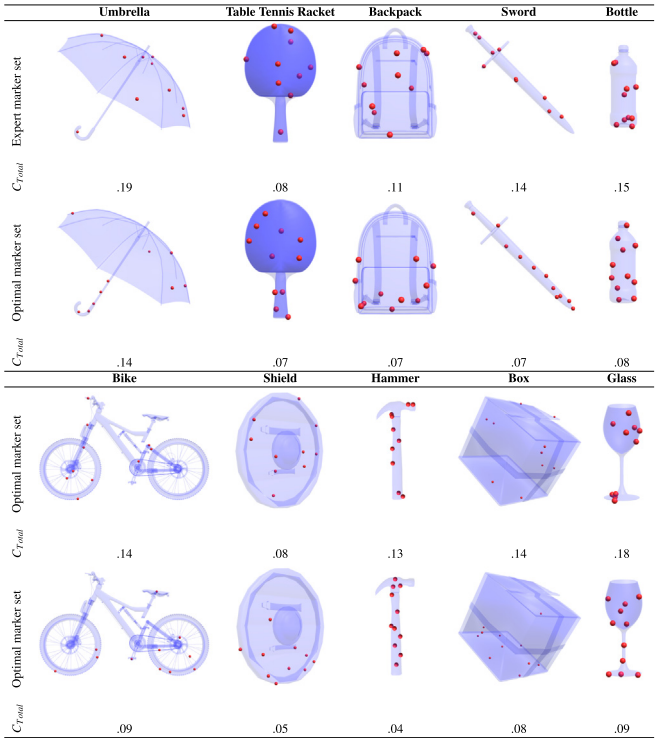


**Fig. 10.** Total cost comparison between expert marker set layout and optimal marker set.

10 meshes and asked a MoCap expert (38 years old; 12 years of MoCap experience) to place markers on each prop empirically. For each prop, we asked the expert to place 10 markers and then we evaluated the suggested layout with a layout resulted from our optimization method. To do so, the expert's marker set was scored based on our cost terms. Fig. 10 shows the results of this test. Our method outperforms the expert's marker set in all cases. In some cases, the expert's configuration is reasonably close to the optimal solution (e.g., the table tennis rack and shield), implying that an expert can find a good enough, but not optimal, configuration, as our algorithm does.

## 6. Discussion

Our method provides optimal marker set configurations for a given prop and user-specified constraints. We validated the proposed method through multiple comparisons, such as the number of markers, different capture area resolutions, and props with and without constraints. According to the results obtained from our first evaluation, in most cases, better results were provided when we requested that our optimizer find optimal configurations for 10 markers. The results showed low performance (higher total costs) when a user requested a 15-marker set. When requesting a lower number of markers (5 markers in our case), we found that the total cost values were generally the lowest, but for some props, the total cost was higher than the other cases (e.g., the average for the bike or box props). These results suggest that larger meshes require more markers to obtain an optimal marker set with lower costs. Our results also confirmed that our method can determine which marker set could work better for a given prop.

The results also suggest that a higher resolution of the captured area provides better performance on the algorithm. However, we should note that a higher resolution of the captured volume increases the total time our system requires to find the optimal marker set configuration. On average, our system requires 1.5 h to provide the optimal solution for the $8 \times 8 \times 8$ resolution. However, keep in mind that the method of Rahimian and Kearney [26], who also considered the visibility of the markers, requires nearly the same time (approx. 2 h) to provide an optimal camera placement based on the possible points around the CAVE.

Moreover, according to our results, our method can efficiently compute optimal marker sets for both unconstrained and constrained meshes. Furthermore, the results of a MoCap session depend on the camera placement. Thus, an optimal camera configuration must also be considered as an initial step to ensure the scene is covered appropriately.

Lastly, we would like to mention a couple of limitations of our method. First, our method requires a lot of time to find the optimal solutions, which mainly depends on the initial parameters (e.g., the resolution of the captured volume area). Second, the procedure of marker placement only considers the props' self-occlusion; therefore, in more complex scenarios (e.g., considering occupied spaces in the room, problems during the triangulation process), the accuracy of our method in obtaining a reliable marker set might be decreased. In addition, due to limited access to MoCap equipment, we did not conduct an experiment to compare our results in real-world MoCap sessions to support the validity of our proposed marker set layout. Thus, we argue that we need more experimentation to show and support the proposed method's reliability in real-world MoCap testing rather than in a simulation environment.

## 7. Conclusions

In this paper, we proposed a novel method to find an optimal marker set configuration for a given prop (3D object) through the reversible-jump Markov chain Monte Carlo optimization method. We obtained multiple marker set layouts by testing different props. We tested our approach by comparing the performance through several simulations with variations in the number of markers used per prop, the resolution of the captured volume, and constrained areas over the props. The results suggest the feasible average performance of our method between the tested variations, especially when examining the number of markers. In this case, our method recommended that the optimal marker set of 10 markers should be placed on the examined objects. Our method could provide an optimal marker set configuration even when we used meshes with constrained areas. Moreover, our method performs better than a MoCap expert; however, we note that a MoCap expert could also provide good enough results. Our future work will address the limitations of this study and expand

our method to include captured areas with occluders, which can be considered more realistic MoCap setups. We will also explore how to optimize marker sets for non-rigid props such as garments and flexible props.

## CRediT authorship contribution statement

**Pedro Acevedo:** Software, Investigation, Visualization, Validation, Data curation, Writing – original draft. **Banafsheh Rekabdar:** Methodology, Conceptualization, Writing – original draft, Writing – review & editing. **Christos Mousas:** Methodology, Conceptualization, Supervision, Writing – original draft, Writing – review & editing.

## Declaration of competing interest

The authors declare no competing interest.

## Data availability

Data will be made available on request.

## References


[1] Menache A. Understanding motion capture for computer animation. Elsevier; 2011.
[2] Parent R. Motion capture. In: Computer animation. Elsevier; 2012, p. 187–98. http://dx.doi.org/10.1016/b978-0-12-415842-9.00006-x.
[3] Charbonnier C, Assassi L, Volino P, Magnenat-Thalmann N. Motion study of the hip joint in extreme postures. Vis Comput 2009;25(9):873–82. http://dx.doi.org/10.1007/s00371-009-0317-5.
[4] Han S, Liu B, Wang R, Ye Y, Twigg CD, Kin K. Online optical marker-based hand tracking with deep labels. ACM Trans Graph 2018;37(4). http://dx.doi.org/10.1145/3197517.3201399.
[5] Moeslund TB, Granum E. A survey of computer vision-based human motion capture. Comput Vis Image Underst 2001;81(3):231–68. http://dx.doi.org/10.1006/cviu.2000.0897.
[6] Cruz-Neira C, Sandin DJ, DeFanti TA. Surround-screen projection-based virtual reality: the design and implementation of the CAVE. In: Proceedings of the 20th annual conference on computer graphics and interactive techniques. 1993, p. 135–42.
[7] Aristidou A, Cohen-Or D, Hodgins JK, Shamir A. Self-similarity analysis for motion capture cleaning. Comput Graph Forum 2018;37(2):297–309. http://dx.doi.org/10.1111/cgf.13362.
[8] Aristidou A, Lasenby J. Real-time marker prediction and CoR estimation in optical motion capture. Vis Comput 2012;29(1):7–26. http://dx.doi.org/10.1007/s00371-011-0671-y.
[9] Chatzitofis A, Zarpalas D, Daras P, Kollias S. DeMoCap: Low-cost marker-based motion capture. Int J Comput Vis 2021;129(12):3338–66. http://dx.doi.org/10.1007/s11263-021-01526-z.
[10] Ghorbani N, Black MJ. SOMA: Solving optical marker-based MoCap automatically. In: Proceedings of the IEEE/CVF international conference on computer vision. ICCV, 2021, p. 11117–26.
[11] Kitagawa M, Windsor B. MoCap for artists: workflow and techniques for motion capture. Routledge; 2020.
[12] Tobon R. The mocap book: a practical guide to the art of motion capture. Foris Force; 2010.
[13] Liu G, McMillan L. Estimation of missing markers in human motion capture. Vis Comput 2006;22(9–11):721–8. http://dx.doi.org/10.1007/s00371-006-0080-9.
[14] Green PJ. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. Biometrika 1995;82(4):711–32. http://dx.doi.org/10.1093/biomet/82.4.711.
[15] Holden D. Robust solving of optical motion capture data by denoising. ACM Trans Graph 2018;37(4). http://dx.doi.org/10.1145/3197517.3201302.
[16] Vicon-Shogun. VFX motion capture. 2023, URL: https://www.vicon.com/software/shogun, Accessed on 12 21, 2022.
[17] Müller M, Röder T, Clausen M, Eberhardt B, Krüger B, Weber A. Mocap database hdm05. Inst Inform II Univ Bonn 2007;2(7).
[18] Loper M, Mahmood N, Black MJ. MoSh. ACM Trans Graph 2014;33(6):1–13. http://dx.doi.org/10.1145/2661229.2661273.
[19] CMU. Carnegie-mellon mocap database. 2006, URL: http://mocap.cs.cmu.edu/, Accessed on 1 30, 2023.
[20] Mousas C, Newbury P, Anagnostopoulos C-N. Efficient hand-over motion reconstruction. In: International conferences in central europe on computer graphics, visualization and computer vision. 2014, p. 111–20.
[21] Schröder M, Waltemate T, Maycock J, Röhlig T, Ritter H, Botsch M. Design and evaluation of reduced marker layouts for hand motion capture. Comput Anim Virtual Worlds 2017;29(6):e1751. http://dx.doi.org/10.1002/cav.1751.
[22] Le BH, Zhu M, Deng Z. Marker optimization for facial motion acquisition and deformation. IEEE Trans Vis Comput Graphics 2013;19(11):1859–71.
[23] Zell E, McDonnell R. Compact facial landmark layouts for performance capture. Comput Graph Forum 2022;41(2):121–33. http://dx.doi.org/10.1111/cgf.14463.
[24] Chvátal V. A combinatorial theorem in plane geometry. J Combin Theory Ser B 1975;18(1):39–41.
[25] Aissaoui A, Ouafi A, Pudlo P, Gillet C, Baarir Z-E, Taleb-Ahmed A. Designing a camera placement assistance system for human motion capture based on a guided genetic algorithm. Virtual Real 2017;22(1):13–23. http://dx.doi.org/10.1007/s10055-017-0310-7.
[26] Rahimian P, Kearney JK. Optimal camera placement for motion capture systems. IEEE Trans Vis Comput Graphics 2017;23(3):1209–21. http://dx.doi.org/10.1109/TVCG.2016.2637334.
[27] Ercan AO, Gamal AE, Guibas LJ. Object tracking in the presence of occlusions using multiple cameras: A sensor network approach. ACM Trans Sens Netw 2013;9(2). http://dx.doi.org/10.1145/2422966.2422973.
[28] Calli B, Walsman A, Singh A, Srinivasa S, Abbeel P, Dollar AM. Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols. 2015, http://dx.doi.org/10.48550/ARXIV.1502.03143.
[29] Brahmbhatt S, Ham C, Kemp CC, Hays J. ContactDB: Analyzing and predicting grasp contact via thermal imaging. In: The IEEE Conference on Computer Vision and Pattern Recognition. CVPR, 2019.
[30] Taheri O, Ghorbani N, Black MJ, Tzionas D. GRAB: A dataset of whole-body human grasping of objects. In: Computer vision - ECCV 2020. Springer International Publishing; 2020, p. 581–600. http://dx.doi.org/10.1007/978-3-030-58548-8_34.
[31] Corsini M, Cignoni P, Scopigno R. Efficient and flexible sampling with blue noise properties of triangular meshes. IEEE Trans Vis Comput Graphics 2012;18(6):914–24. http://dx.doi.org/10.1109/TVCG.2012.34.
[32] Olague G, Mohr R. Optimal camera placement for accurate reconstruction. Pattern Recognit 2002;35(4):927–44. http://dx.doi.org/10.1016/S0031-3203(01)00076-0.
[33] Mandery C, Terlemez Ö, Do M, Vahrenkamp N, Asfour T. The KIT whole-body human motion database. In: 2015 international conference on advanced robotics. ICAR, 2015, p. 329–36. http://dx.doi.org/10.1109/ICAR.2015.7251476.
[34] Eades P. Symmetry finding algorithms. In: Computational morphology - a computational geometric approach to the analysis of form. Elsevier; 1988, p. 41–51. http://dx.doi.org/10.1016/b978-0-444-70467-2.50009-6.
[35] Wolter JD, Woo TC, Volz RA. Optimal algorithms for symmetry detection in two and three dimensions. Vis Comput 1985;1(1):37–48. http://dx.doi.org/10.1007/bf01901268.
[36] Knuth DE, Morris Jr JH, Pratt VR. Fast pattern matching in strings. SIAM J Comput 1977;6(2):323–50. http://dx.doi.org/10.1137/0206024.