Pmomo: Projection Mapping on Movable 3D Object

Yi Zhou *
Shanghai Jiao Tong University
Shanghai, China

Shuangjiu Xiao †
Shanghai Jiao Tong University
Shanghai, China

Ning Tang
Shanghai Jiao Tong University
Shanghai, China

Zhiyong Wei
Shanghai Jiao Tong University
Shanghai, China

Xu Chen
Shanghai Jiao Tong University
Shanghai, China

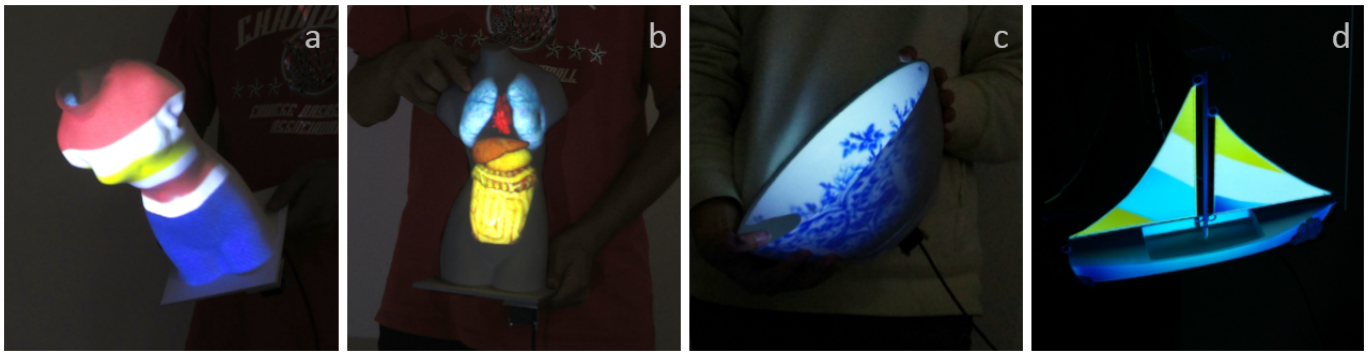

Figure 1. Augmented Reality applications of Pmomo: virtual clothes (a); animated 3D organs with beating heart and breathing lungs inside the bust model (b); virtual porcelain texture on a bowl model (c); virtual canvas on a boat model (d).

ABSTRACT

We introduce Pmomo (acronym of projection mapping on movable object), a dynamic projection mapping system that tracks the 6-DOF position of real-world object, and shades it with virtual 3D contents by projection. The system can precisely lock the projection on the moving object in real-time, even the one with complex geometry. Based on depth camera, we developed a novel and robust tracking method that samples the structure of the object into low-density point cloud, then performs an adaptive searching scheme for the registration procedure. As a fully interactive system, our method can handle both internal and external complex occlusions, and can quickly track back the object even when losing track. In order to further improve the realism of the projected virtual textures, our system innovatively culls occlusions away from projection, which is achieved by a facet-covering method. As a result, the Pmomo system enables the possibility of new interactive Augmented Reality applications that require high-quality dynamic projection effect.

Author Keywords

Dynamic Projection Mapping; Tracking; Real-Time; Augmented Reality

ACM Classification Keywords

H.5.2 [Information Interfaces and Presentation]: Graphical user interfaces(GUI), Input devices and strategies.

INTRODUCTION

Projection Mapping, also known as Spatial Augmented Reality (SAR), is a frequently used method to create phantasms in real world. With this method, visual phenomena of real-world objects could be changed or enhanced — colour, texture and even geometry. Static projection mapping is common in multi-media shows, applied on static objects such as sculptures or buildings.

While there have been many researches on Projection Mapping on static sculptures [1][20][30][31], Dynamic Projection Mapping that can keep tracking of a moving target and aligning the projection at interactive level is still a challenge. To achieve high-quality results, the accuracy and the robustness of the system must achieve high requirements as to make light bundles be projected exactly onto the target object in real-time. Also, cumulated errors and delays in tracking can cause visible deviation between the projected texture and the

*papagina@sjtu.edu.cn
†xsjiu99@cs.sjtu.edu.cn

CHI 2016, May 7–12, 2016, San Jose, California, USA.

http://dx.doi.org/10.1145/2858036.2858329

real object, leading to visible artifacts. Last but not least, the users' spontaneous and rapid interactions with the object may lead to occlusions, which makes the tracking and texture projection even more challenging.

There are some works related to dynamic projection mapping. In most of the cases, there is either strong assumption on object's geometry (e.g. quasi-planar), or object's motion (e.g. preset or limited movement). Some works([4][5][11][13][17][18][19][21][32][33][34]) perform adaptive projection on deformable and movable surfaces, e.g. transforming deformable paper into virtual displays; some others perform projection on 3D objects with simple structures [27][23], or on 3D objects with very limited movement range [3].

Our system, in contrast, could handle complex geometries with 6-DOF free motion at the interactive level. In order to achieve a robust real-time alignment between the projection and the object, we developed a novel depth-image based tracking algorithm that can handle rigid objects with arbitrary structures and manage both internal and external occlusions. In addition, we use an AHRS (Altitude Heading Reference System) sensor to improve the real-time alignment performance, also solving the problem that depth-image can't perceive the rotation of symmetrical shapes. As to further improving the realism of projection, our system novelly culls off the occlusions from the projection (Shown in Figure 2). As a result, our system is unique with the combination of these features, and is robust for real interaction. With our system, real-world objects can be projected with realistic virtual exteriors, and transformed into movable 3D displays, bringing the possibility of a variety of interactive AR applications in the field of art, education and entertainment. Figure 1 illustrates some applications of our system.

In summary, we contribute a novel dynamic projection mapping system, Pmomo, that can simultaneously:

1. Align the projection with the object in 6-DOF motion environment in real-time.
2. Support object with complex geometry.
3. Cull occlusions from projection .

We organize our paper as follows: In Section 2, we review background and related techniques in this field. In Section 3, we present the pipeline and algorithm of our system. In Section 4, we report our experiment results and in Section 5 we conclude with future works.

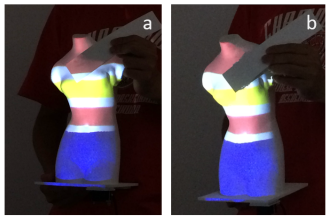

Figure 2. Occlusion culling contrast. (a) shows projection without occlusion culling; (b) shows projection with occlusion culling.

BACKGROUND

Since tracking is the main technical challenge, in this section, we will review the previous work related to dynamic projection systems, with an emphasis on the tracking method.

To achieve projection on dynamic objects, many sorts of tracking sensors have been tested. Bandyopadhyay et. al. [3] suggested to employ the magnetic tracker to obtain the position of the object. But, this type of sensor suffers from a very short working distance and the cost can be impractical. The authors of "Lumipen" [27] used a high-speed vision sensor and a projector with a high-speed optical gaze controller to track objects by keeping the target in the center of the image. This method shows good performance on high-speed balls, e.g. pingpong ball and yo-yo, but it is hard to handle free rotations. Wang et. al. [35] presented an augmented reality surgical navigation system with customized stereo cameras designed to track both the patient and instrument with tracking algorithm based on assumption of the 3-D contour of teeth. In the show of "Face Hacking" [2], optical markers were utilized to track human face and perform projection on the face. However, optical makers can visually interfere with the projection effect, and demand complex setup with multiple infrared cameras setting around the scene.

Some researchers employed the method of explicit tracking by projection features. This kind of method uses the camera to observe the projection patterns on the object to find the correct alignment. Cotting et. al. [8] injected imperceptible codes in a projector. Zheng et. al. [39] aligned the projection and the object by iteratively minimizing the visual distortion. Based on the work of Yang and Welch [37] and the work by Johnson and Fuchs [15], Resch et. al. [28] developed a shader lamp system which used a frame by frame tracking method of comparing discrete features found in the camera and projector image to achieve interactivity close to real-time. A concern of using the projection as an active light sensor is that the object can not be projected with fancy texture or rapidly changing animations since it will interfere with the alignment. Also the occlusion handling can be a problem.

Another kind of tracking method is based on depth camera. The advantage is that the depth camera is not sensitive to light condition, and does not interfere with the projection. Besides, the commodity depth camera like Kinect 2.0 provides a high resolution, a high precision and a wide field of view, making it an accessible and practical technological solution to the issue.

For state of art tracking technology on depth data, KinectFusion [14][26] presented impressive real-time 3D reconstruction capability for static scene. And in recent, DynamicFusion [25] extended the reconstruction to dynamic scene, bringing the possibility of real-time tracking and segmenting 3D objects. But for the scenario of dynamic projection alignment at interactive level, there are still many concerns. For example, it is hard to segment the target object in real interaction when the object is holding by users' hands and is often close to and overlapped with the users' body. Also there are concerns on the accumulated error, the delay for complex computing, and the failures under sharp and continuous motion.

As to registration method for 3D model, ICP(Iterative Closest Point) [6] [29] is a commonly used concept, e.g. KinectFusion employed it to track the motion of the Kinect camera. ICP registers the source and the target by a closest point scheme. It starts with an initial guess of transformation, and in each iteration, it searches for the pairs of closest points on the source and the target and refines the transform vector that minimizes the matching errors. The main worry in using ICP is that the algorithm can easily be trapped in local minima. Thus, it usually depends on the assumptions of simple scene and good initialization when using ICP for real-time tracking. E.g. [12] assumes that the scene for tracking contains only the target, the user's hands and a clear table top so that the target can be easily segmented out of the depth and color images of Kinect. Then it utilizes SIFT (Scale-invariant Feature Transform)[22] with RGB image for getting the rough pose as the initialization for ICP.

A more promising searching approach for registration is CMA-ES (Covariance Matrix Adaptation Evolution Strategy), for it can efficiently provide semi-global optimization. The optimization procedure is initiated with an ancestor, and in each iteration, offspring of the best fitter from the last generation are generated as new samples with Gaussian distribution. It stops when the fitness of the best fitter converges to a threshold, or the number of iterations reaches a threshold. Jordt et. al. [16] provided a tracking method that employs a grid to represent the objects and uses CMA-ES for searching the optimized position of each point in the grid. The number of points determines the dimension of the search space. As a result, this method can support deformable object but can not work for complex structures with large quantity of grid points. The authors of "Flexpad" [33] proposed another method for tracking the deformation of paper for projection alignment. They constructed 16 deformation models of paper and adopted CMA-ES to search the approximate optimum of weighted factors of the deformation.

Occlusion is also a big problem for tracking. Some previous works, e.g. [10] , tackle the occlusion problem by a probabilistic external occlusion map obtained by using the intensity discrepancy. KinectFusion [14] handles occlusion by means of raycasting which is quite expensive. Flexpad [33] detects occlusions created by the user's hands by distinguishing the optical feature of skin on infrared image. And [12] detects hands by building the color model for each user's skin. There are also works using multiple cameras to solve occlusions, e.g., using 16 cameras [9].

In conclusion, the previous tracking methods have the weakness of only handling limited motion range or speed, limited geometry, or limited occlusion, and in some cases, the hardware is too complicated but producing low tracking capability.

METHOD OF PMOMO

The Pmomo system tracks the 6-DOF position of real-world object, and shades it with virtual 3D textures by projection. The work flow of our system is shown in Figure 3.

In preparation phase, a 3D mesh model of the target object needs to be acquired. This can be achieved by either scanning the real object by reconstruction technologies or directly designing and 3D-printing the digital models. Based on the raw 3D mesh model, four models need to be generated for tracking and projection. The first one is a low-density point-cloud model for tracking. The second one is a high-density point-cloud model for occlusion culling. the two point-cloud models are constructed to have even point distribution. The third one is a mesh model that is remeshed from the second one. A point-facet list records the matches of the points in the second model and their adjacent facets on the third one. The fourth model is a textured model designed for projection.

In real-time phase, the first step is to obtain the transform matrix of the target object. This tracking procedure is achieved by registering the low-density point cloud model to the current depth image by a semi-global optimization algorithm. The next step is to generate a layer of facets for occlusion by using the high-density point cloud model and the point-facet list. In the third step, the textured model is covered with the occlusion facets rendered by the background color, and then transformed by the transform matrix obtained from the first step. Finally, the final rendered result is projected onto the real object.

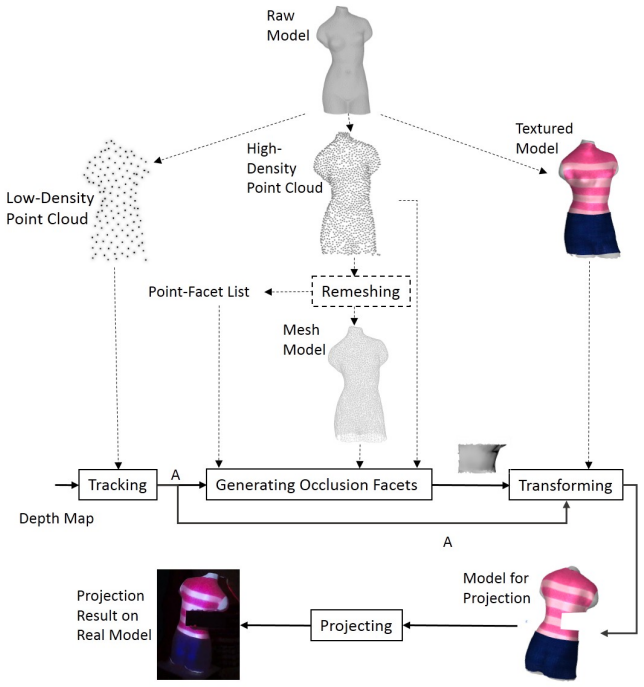

Figure 3. Overall work flow of the Pmomo system. Procedures with solid lines are run in real-time. Procedures with dash lines are done in preparation phase. *A* refers to the transform matrix of the target object.

Setup and Calibration

To achieve projection mapping, we need to build a virtual scene precisely calibrated to the real scene. Within the virtual scene, there is a 3D model that shares the same pose and position of the real object and a camera that shares the same view port of the real projector. The image rendered from the

viewport of the virtual camera determines the projection content.

At each frame of Pmomo, 6-DOF position of the real object is tracked, and its correspondent virtual 3D model is then transformed and updated to that new position. Kinect 2.0 depth camera is used to capture the real-time scene and calculate the translation vector. And AHRS sensor is used to obtain the rotation vector.

The model of the AHRS sensor we use is 3DM-S10A/B from Shanghai Siyue Electronics Co., Ltd, which provides stable orientation angles ($\pm 2^\circ$ for dynamic error and $\pm 0.5^\circ$ for static error) with high update rate (100fps) and no restriction in 6-DOF range and costs about twice the price of a Kinect 2.0 sensor.

Figure 4 shows the physical setup of Pmomo, consisting of a standard desktop PC (Intel i7 processor, 8 GB RAM, and an NVIDIA GeForce GTX 770 graphics card), an AHRS sensor mounted on the object, a Kinect 2.0 and a projector (Panasonic PT-BX40) placed on a shelf close to each other. The setup creates an interaction volume with best performance of about $1.5 \times 1.0 \times 1.0\ m$.

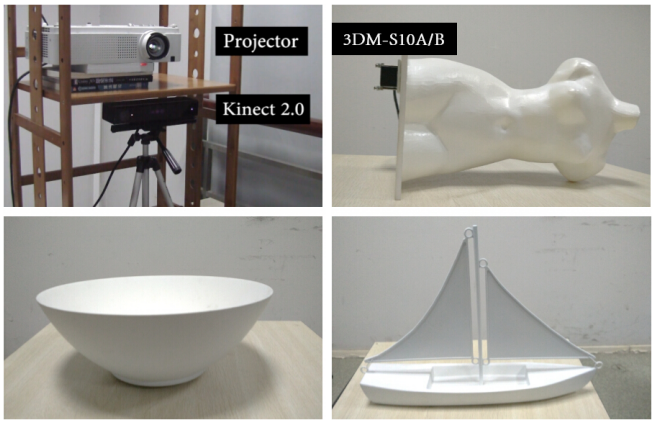

Figure 4. Physical setup and models for testing.

In the calibration work, the AHRS sensor needs to be calibrated to the coordinate of Kinect depth camera. It is mounted on the object in the way that its three axes are aligned to the object's coordinate. Let the rotation of the object in the depth sensor's coordinate be called R^k and the rotation obtained from the sensor be R^s, then,

$$R^k = R^t R^s \quad (1)$$

R^t is the rotation from depth camera's coordinate to the sensor's coordinate. R^t is obtained by the same registration algorithm described in the Tracking Section. As to get the best registration result, the object should be put in the middle of the depth camera's view field without external occlusions.

We use Kinect RGB camera to indirectly calibrate the projector to the infrared camera (the infrared image shares the same view of the depth image). First, we obtain the transform matrix between the RGB camera and the infrared camera by Zhang's method [38]. Then, we calibrate the RGB camera and the projector by [24] and obtain the transform matrix between them. Using the two calibration data, the transform matrix from Kinect infrared camera to the projector can be calculated.

Tracking

In order to achieve robust tracking and texture projection performance for randomly moving rigid object with complex geometry under arbitrary occlusions, we employs the CMA-ES based tracking method with three major modifications:

1. We reduce the complexity of tracking by homogenizing the input geometry to evenly distributed low-density point cloud;
2. We embed an occlusion detection method with self-adaptive threshold in the iterations of CMA-ES;
3. The step size of each iteration is also self-adaptive in order to handle bad tracking conditions.

Our pipeline for object tracking is shown in Figure 5. At each frame n in the running stage, we store a transform matrix A_n and an occlusion list L_n. We use the result from the last frame as the initial guess and search for a transform matrix A_n that best registers the low-density point cloud to the current depth image. Then the occlusion list is updated to indicate which points in the point cloud are occluded in the current frame. The final output of the tracking process is the predicted transform matrix as to compensate the computation and device input delay.

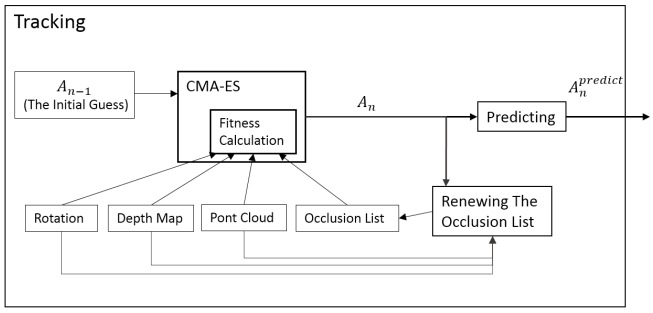

Figure 5. Work flow of Tracking. A_n is the transform matrix of the current depth image frame, $A_n^{predict}$ is the predicted transform matrix, A_{n-1} is the transform matrix of the last depth image frame.

In registration, we use the CMA-ES scheme to obtain the translation vector T_n, while the rotation vector R_n is from the AHRS sensor. First our algorithm initializes with the translation vector T from the last frame, then randomly generates some variants of T using covariance matrix. For each variant of T, the system uses it together with R_n to transform the point cloud $\{p\}$, and then selects the visible points in the transformed point cloud as point set $\{P\}$. Next, the system projects $\{P\}$ to the depth map, getting their 2-D (u, v) position. For each point P_i in $\{P\}$, the system gets its counterpart P'_i by back-projecting the pixel on (u, v) with its depth value on the depth map to the 3-D space. The RMS distance of all the pairs $\{(P'_i, P_i)\}$ is calculated as the fitness error. After that, the system picks up the vector T' that has the lowest fitness error, and updates T by T'. The above steps are iterated until the fitness error reaches a threshold.

We therefore define the following fitness term for each transformation guess.

$$F(A) = \sqrt{\frac{1}{n}\sum_{p_i \in S} \|\phi(\pi(Ap_i)) - Ap_i\|^2} \quad (2)$$

where S is a set of sample points, n is the size of S, $P_i = Ap_i$ is the transformed position of p_i in the depth camera's coordinate. $P'_i = \phi(\pi(Ap_i))$ is the counterpart of P_i by depth map. The function $q = \pi(p)$ performs perspective projection of $p \in \mathbb{R}^3 = (x, y, z)^{\mathrm{T}}$ to obtain $q \in \mathbb{R}^2 = (x/z, y/z)^{\mathrm{T}}$ on depth image. The function $p = \phi(q)$ back-projects the q on depth image to world coordination by the depth value.

Transformed sample points invisible to the depth camera are excluded. The visibility of a transformed point P_i is determined by the satisfaction of the following two requirements:

1. If

$$Angle(\vec{n}, \vec{pk}) \leq 90^\circ \quad (3)$$

   where $\vec{n}$ refers to the normal of point P_i, $\vec{pk}$ refers to the vector from point P_i to the center of depth camera, and $Angle(\vec{n}, \vec{pk})$ refers to the angle between $\vec{n}$ and $\vec{pk}$.

2. If p_i is not marked as occluded in the occlusion list.

Self-adaptive Occlusion Detection

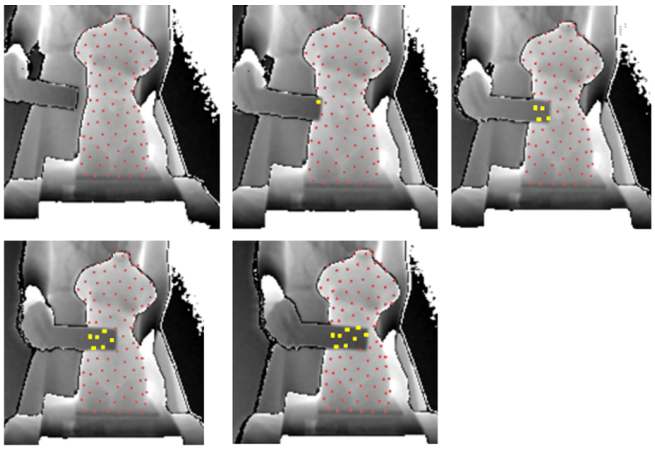

Figure 6. Detection of occluded sample points in real-time. Red points are visible points on the point cloud model, yellow ones are occluded ones at each frame. The background image is a visualized depth map.

For real-time occlusion, we observe that the changes in the set of occluded sample points are very small between two consecutive frames, as shown in Figure 6. Thus, the last frame's occlusion list can be used for the tracking of the current frame, and after obtaining the current A_n, we update the occlusion list for next frame. The occlusion list is updated by checking every transformed sample point with the following two conditions:

1. If the Equation 3 holds.

2. If

$$(depth_{real}(P) - depth_{virtual}(P)) > thr \quad (4)$$

where

$$thr = max(F(A_n), minThr) \quad (5)$$

where $minThr$ is a constant value determined by the precision of tracking. The benefit of the adaptive occlusion threshold is that it can reduce the chance of losing track due to massive misdiagnosis of occlusion points.

Step-size Control

In some cases, the CMA-ES method will fall into incorrect local optimal. For example, when a person holds a object, the algorithm might track onto the person incorrectly instead of the object. To handle this, we use an adaptive step-size for CMA-ES to generate new samples. The step-size σ affects the deviation between the ancestor and the offspring. For each frame, a default value is initially set to σ. In the iterations of CMA-ES, if the Euclidean distance between the fittest translation vector of an iteration and the translation vector from the previous frame exceeds a maximal value, our algorithm will restart the iterations with the step size σ reduced to $k\sigma$. k is the reducing scale, and we empirically set it to be 0.25. With the step-size control scheme, the system also gains the feature that when the tracking suffers from severe errors, the system will hold on around the previous position. This feature, combined with the semi-global searching scheme, provides two advantages for real interaction. First, when the system loses track of the object due to big but instant occlusions (e.g. people walking through the area between the object and Kinect camera), the tracking will hold on to the previous position for a while and when the occlusions are removed, our system can quickly track back the object in seconds. Second, if the system tracks incorrectly to a wrong place, the user can help the system track back the object by moving the object to that place with the indication of the light bundles of projection.

Delay Management

While the AHRS sensor has an insignificant input delay that is negligible, Kinect has a more apparent input delay D_k, with about 90 milliseconds for Kinect 1.0 and about 60 milliseconds for Kinect 2.0. As a result, for the process of real-time depth image registration, the rotation parameter from the AHRS sensor needs to be the one obtained D_k milliseconds before the current frame.

For the real-time projection, we use the latest rotation from the AHRS sensor, and use the predicted translation by a linear model:

$$T_n^{predict} = T_n + f * (D_c + D_k) * (T_n - T_{n-1}) \quad (6)$$

Where D_c is the computation delay, D_k is the Kinect input delay. f is the frame rate, T_n is the estimated translation vector of the nth frame, $T_n^{predict}$ is the predicted one.

Due to the noises in depth image, the tracked position will randomly fluctuate even if the model remains static, and the motion prediction term will amplify the noise. But the fluctuation is not visually evident in fast motion. Thus, Like Wang et. al. [35], we perform Extended Kalman filtering (EKF) to estimate the quasi-static pose to reduce the fluctuation when the model is static or moving slowly.

Model Constructing

In order to achieve best performance, an evenly-distributed point cloud is needed for our system. With unevenly-distributed sample points, if a dense part is occluded, the remaining sample points may not be sufficient for tracking.

The process of generating an evenly-distributed point-cloud $\{p\}$ from a mesh model has two steps. First, we generate Poisson Disk sample points on every facet with a certain density [7]. Second, we initialize $\{p\}$ as an empty point set and traverse all the Poisson-Disk-sampled points on the mesh model. If a point is close to (below a certain threshold) any points in $\{p\}$, we drop it, else add it to $\{p\}$.

Our method also generates a point-facet list while remeshing the point-cloud model for occlusion culling. A point-facet list is a list that tells which facets are adjacent to a certain point. The approach for finding the adjacent facets of point p is to traverse every facet in the remeshed model and choose the facet that contains p as one end point.

Occlusion Culling

In Pmomo, a facet-covering method is used for occlusion culling. The pre-generated high-density point cloud model and its corresponding mesh model are utilized in the method. Using the current transform matrix as input, the high-density point cloud model is aligned to the depth image, and occluded sample points are picked up by the same occlusion detection function in the tracking process. Since we obtained point-facet list for point cloud from mesh, we could easily find facets that are corresponding to the point and add them into the occlusion facets union. Then the occlusion facets are covered on the textured 3D model and are rendered by the background color which is black in the projection scene.

By overlapping black facets on the occluded part of the model, the occluders will be projected in black so that they can be culled from final projection image. The advantage of this method is that for inner occlusion condition, the occlusion facets will be covered on the invisible inner side so that not interfere with the front side.

Computation Complexity

The computation complexity of the tracking process is: $O(NMS)$. N refers to the number of points in the point cloud, M refers to the number of searching iterations, S refers to the number of offspring (translation vector variants) generated in each CMA-ES iteration. Thus, the computation complexity is independent with the size and the shape of the object.

Unlike in the tracking phase where the fitness function is iteratively computed for at most NMS times to find the optimal transform matrix, in the step of generating occlusion facets, the occlusion detection function is calculated only once. Thus, the calculation with high-density point cloud model is affordable. Moreover, the higher the density, the higher precision will the system acquire for occlusion culling.

Initialization

The initial guess of the position of the object is manually set to be around the real position. Then our algorithm will semi-globally search the right position, and within several frames, the system will automatically align to the object. Figure 7 shows the process of the automatic alignment by snapshots extracted from a real-time record.

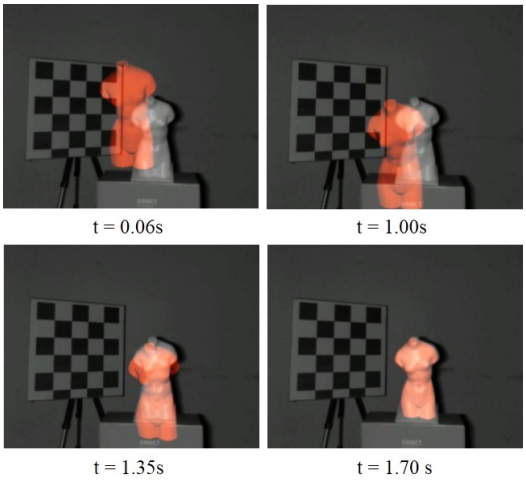

Figure 7. The initialization of tracking. The pictures are snapshots of the virtual scene with the Kinect infrared image rendered as the background.

EVALUATION

For performance estimation in previous works on dynamic projection , The authors of "Flexpad" [33] calculated the RMS depth distance between the paper's surface and the corresponding depth image, showing an average error of 6.47 mm out of over 20,217 frames of data. But it lacked the specification that under what speed of object motion the precision had been measured. The system of [28] presented the evaluation result that its translation error along each x,y,z axis varied from 0.9*mm* to 4.5*mm* for static condition and 2.1*mm* to 31.6*mm* for moving condition, which was hugely impacted by the projection texture according to the authors. Also, they lacked the speed and occlusion information for their test, and the estimation was conducted only on one car model. In our experiment, we estimate the registration error under different speed and occlusion proportion, also estimate the real-time projection alignment error caused by the motion prediction.

We demonstrate our system on three challenging models: a bust, a bowl and a sailing boat. The bust has diverse density of meshes (e.g. denser at navel and sparser on thighs); the bowl has big proportion of internal occlusion (e.g. as shown in the picture c1 in Figure 10, when the inner side is completely occluded by the outer side, the inner occlusion rate reaches 50%); the sailing boat has cut-out and sunken structure, and the area of its frontage changes sharply when it is rotating.

The three models are all 3D-printed, so we have their ground truth 3D models for testing the tracking performance of our system. We also perform test with their comparatively low-quality 3D models reconstructed by KinectFusion with Kinect 2.0 sensor.

In the experiment, ten volunteers are asked to freely move the models in the view field of Kinect camera and occasionally occlude the model by hands or stuff like cards or tapes.

For testing the tracking performance with the ground truth 3D model data, we record around 100,000 frames (30 fps, around 56 minutes in total) of tracking data for the test, with each model at least 18 minutes of testing time. So does the test with 3D model data from KinectFusion.

Tracking Precision

Occlusion Proportion: 0% - 15% Acceleration: 0 - 30 cm/s^2				
Velocity (cm/s)	0 - 10	10 - 20	20 – 30	30-50
Reg-Error (mm) Ground Truth	**5.1**	**6.1**	6.7	7.2
Reg-Error (mm) KinectFusion	**5.7**	**7.5**	8.8	9.3
Occlusion Proportion: 15% - 25% Acceleration: 0 - 30 cm/s^2				
Velocity (cm/s)	0 - 10	10 - 20	20 – 30	30-50
Reg-Error (mm) Ground Truth	9.7	11.2	12.3	18.8
Reg-Error (mm) KinectFusion	10.7	13.2	15.6	29.7
Occlusion Proportion: 0% - 15% Velocity: 0 - 20 cm/s				
Acceleration (cm/s^2)	0 - 20	20 - 40	40 - 60	
Reg-Error (mm) Ground Truth	**5.0**	5.6	9.0	
Pred-Error (mm) Ground Truth	**3.4**	4.0	11.3	
Reg-Error (mm) KinectFusion	**6.3**	6.8	12.3	
Pred-Error (mm) KinectFusion	**4.8**	6.5	15.0	

Figure 8. Average tracking error. Reg-Error refers to the registration error on depth frame; Pred-Error refers to the prediction error.

As to estimate the performance in detail and independently to models, we mix all the testing data and classify them under moving speed, acceleration and occlusion proportion. The tracking precision is estimated by the average RMS Euclidean distance error between the model and the corresponding depth image per frame as defined in equation 2.

The velocity V_n is calculated by

$$V_n = f * \|T_n - T_{n-1}\| \quad (7)$$

T_n is the translation at the frame nth, f is the update rate of the depth image frame.

The acceleration is obtained from the AHRS sensor. We use the function below to subtract the gravitational component from the raw acceleration data.

$$a_n = a_n^s - R_n^s g \quad (8)$$

a_n^s is the raw acceleration vector obtained from AHRS sensor. R_n^s is the raw rotation matrix. g is the acceleration of gravity.

The occlusion proportion includes both the external and internal occlusions. It is calculated by the ratio of sample points satisfying Equation 3 and Equation 4 to sample points satisfying Equation 3.

We set the maximal number of CMA-ES iterations for each frame as 18, number of variants per iteration as 15, with at most 270 times of fitness computation in total. And the sample point clouds we use for tracking have 270 to 280 points. In our test, the overall average time for tracking computation in each frame is 16 milliseconds.

The first two tables in Figure 8 shows the average registration error on depth frame under different categories. It shows that with the ground truth 3D model data, our system has very good real time alignment when the moving speed is under 20 cm/s and the occlusion proportion under 15%. When the velocity exceeds 30 cm/s, and the occlusion proportion exceeds 15%, the average error can exceed 18 mm, but the system is still able to track the object. And for the case of using 3D models reconstructed by KinectFusion, as expected, the tracking error is overall a bit higher, but still feasible.

As to the performance in real-time projection alignment, we demonstrate the prediction error in the third table. It is calculated by the average Euclidean distance between $T_n^{predict}$ (the prediction at the nth frame) and $T_{n+delay}$ (the registration at the future frame after the delay time). According to the result, the prediction error is not evident when the acceleration is not very sharp. But it raises with the increase of acceleration value which conforms to the fact that when the user is sharply changing the motion, e.g. shaking the object, the projection can't be aligned perfectly on the object.

Visual Effect

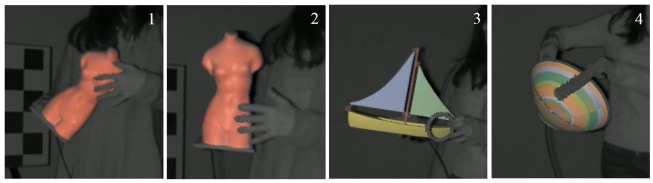

Figure 9. Tracking and occlusion culling performance observed in virtual scene. The pictures are snapshots of real-timely updated virtual scene. Picture 1 and 2 shows occlusion culling of user's fingers. Picture 3 shows occlusion culling of a roll of tape. Picture 4 shows occlusion culling of a piece of paper.

We set a virtual scene to visually monitor the tracking and occlusion culling effect on the screen. As shown in Figure 9, in the scene, we set the virtual camera at the view of the depth camera, render the infrared image as background and update the 3D model's position in real time. The numbers of points in the point-cloud model we use for generating occlusion facets are 3153, 5964, 5800 for the bust, the boat and the bowl respectively. It can be visually perceived that the occlusions are culled in high detail while providing a good real-time tracking. For example, the contour of the user's fingers can be clearly seen in the picture 1 and 2 of Figure 9. The

silhouettes are not very smooth yet, but it can be improved by increasing the density of the point-cloud model.

Figure 10 shows the dynamic projection effect in the real scene. The three models in the pictures are originally white, but are shaded with artistic colorful textures. Users are asked to hold the models in hand and freely translate and rotate them. The result shows that the projection can stick on to the real model under arbitrary pose and position. Small visual artifacts of the alignment may occur when the object is in sudden acceleration (e.g. Picture a4 in Figure 10).

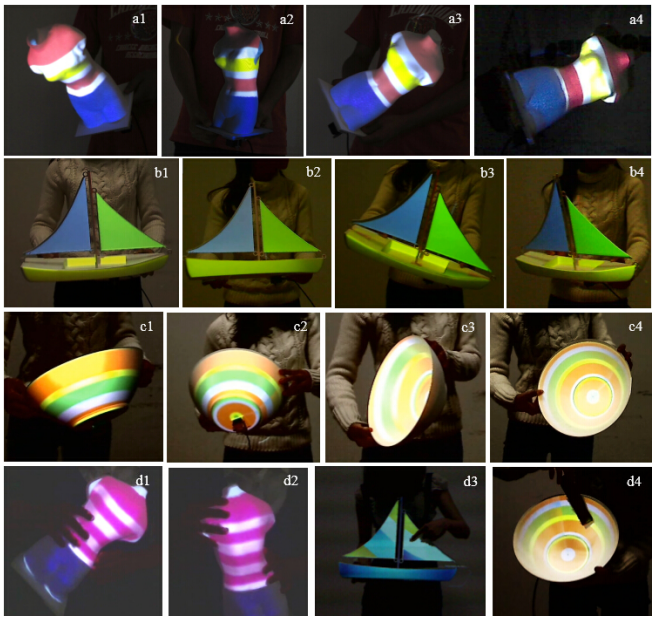

Figure 10. Real projection mapping effect of Pmomo. The white plastic models in the pictures are shaded with colorful textures by projection. All the pictures are video snapshots of projection effect on moving objects.

Failure Handling

The system loses track when the object is occluded with very big proportion or it is moved at extremely high speed. However, since our system will hold on to the previous "right" position when losing track, it can keep tracking when instant big occlusion is passing by (as shown in 11). Also since the system performs semi-global tracking, when the system fails tracking the object and projects the light bundles to an incorrect place, our volunteers can help the system track back the object in no longer than 15 seconds by moving it to the lighted place.

TRACKING WITHOUT AHRS SENSOR

Generally, our tracking algorithm could work without AHRS sensor. However, in such case more iterations would be required in CMA-ES algorithm since estimating both translation and rotation requires larger searching range. For extreme cases (e.g. when the full 6-DOF transformation between frames is very large), the algorithm needs 0.5-1.5 seconds to converge (the converged RMS error is around 10mm). That severely interferes the real-time alignment of our system

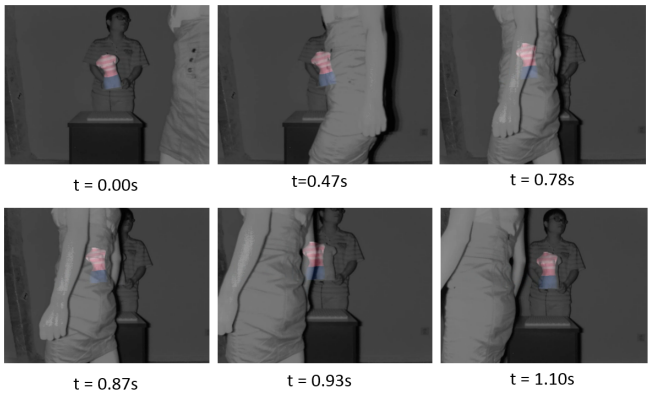

Figure 11. Case of total occlusion. At the first frame the object is tracked. Then someone walks through between the depth camera and the target object, totally occluding the object. At the last frame, the object is tracked again.

and influences its application. By adding an AHRS sensor into the system to provide the rotation data, we can achieve a much better performance at an economical price.

CONCLUSION

As a dynamic projection mapping system, Pmomo system has four notable advantages: real-time alignment for random motion, arbitrary complex geometries, robustness to occlusions and automatical occlusion culling from projection.

To achieve a robust tracking for complex shapes under occlusions, we develop a depth image based tracking algorithm with the novel features of complexity reducing, self-adaptive occlusion handling and self-adaptive step-size control. And, we also develop a facet-covering method to cull occlusions from projection.

The experimental results show that Pmomo system can perform dynamic projection mapping at a very high interactive level. And we believe the technique of Pmomo will lead to a wide range of novel augmented reality applications in art, education and commercials.

Our approach do have some limitations. First, occluders should not be too close to the target object, otherwise it will be unidentifiable on the depth image. Second, large external occlusions might lead to the fail of tracking. When the occlusion proportion exceeds a certain high level, the remaining visible part will not be sufficient for tracking. This limitation can be overcome by employing additional depth cameras with other perspectives. Third, for deformable object, our system can only tolerate small local deformation, as long as such deformation does not impact the overall fitness error of the visible sample points, but can't handle big deformations. Another limitation lies in that the camera and the projector need to be settled close to each other and of similar view direction so as to ensure they can view the same occlusions.

We are planning to develop more advanced features and capability for the system in the future: First, we want to further optimize the system by parallelizing the tracking algorithm on GPU so that the system can perform high-speed 6-DOF

tracking only with depth camera. Second, we want to extend our system to support multiple objects simultaneously. Third, we want to enable the possibility of a bigger scene and a wider 360 degree projection by using multiple cameras and projectors or steerable camera-projection systems [36].

ACKNOWLEDGEMENTS

We would like to thank Dong for helping to make the calibration board and all the little gadgets that we used during the research, Wang for helping us settling the equipment, Kan for helping design the 3D models, Xiao and Yue for the revision of the paper, our dear friend Xiong and Shuai for the technical help in coding, Stephane for his emotional support, and all the volunteers, for performing the tests and giving us valuable suggestions.

REFERENCES

1. Daniel G Aliaga, Yu Hong Yeung, Alvin Law, Behzad Sajadi, and Aditi Majumder. 2012. Fast high-resolution appearance editing using superimposed projections. *ACM Transactions on Graphics (TOG)* 31, 2 (2012), 13.
2. Kuwahara Asai, Sakata Lacroix, Koyaku Kobayashi, Yamato Noda, Hiraoka Nakamura, and Doi Nagashio. 2014. "Face Hacking. (2014). http://www.face-hacking.com
3. Deepak Bandyopadhyay, Ramesh Raskar, and Henry Fuchs. 2001. Dynamic shader lamps: Painting on movable objects. In *Augmented Reality, 2001. Proceedings. IEEE and ACM International Symposium on*. IEEE, 207–216.
4. Hrvoje Benko, Ricardo Jota, and Andrew Wilson. 2012. MirageTable: freehand interaction on a projected augmented reality tabletop. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 199–208.
5. Hrvoje Benko, Andrew D Wilson, Federico Zannier, and Hrvoje Benko. 2014. Dyadic projected spatial augmented reality. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 645–655.
6. Paul J Besl and Neil D McKay. 1992. Method for registration of 3-D shapes. In *Robotics-DL tentative*. International Society for Optics and Photonics, 586–606.
7. Robert Bridson. 2007. Fast Poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH*, Vol. 2007. ACM, 5.
8. Daniel Cotting, Martin Naef, Markus Gross, and Henry Fuchs. 2004. Embedding imperceptible patterns into projected images for simultaneous acquisition and display. In *Mixed and Augmented Reality, 2004. ISMAR 2004. Third IEEE and ACM International Symposium on*. IEEE, 100–109.
9. Ali O Ercan, Abbas El Gamal, and Leonidas J Guibas. 2013. Object tracking in the presence of occlusions using multiple cameras: A sensor network approach. *ACM Transactions on Sensor Networks (TOSN)* 9, 2 (2013), 16.
10. Vincent Gay-Bellile, Adrien Bartoli, and Patrick Sayd. 2010. Direct estimation of nonrigid registrations with image-based self-occlusion reasoning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 32, 1 (2010), 87–104.
11. Chris Harrison, Hrvoje Benko, and Andrew D Wilson. 2011. OmniTouch: wearable multitouch interaction everywhere. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 441–450.
12. Robert Held, Ankit Gupta, Brian Curless, and Maneesh Agrawala. 2012. 3D puppetry: a kinect-based interface for 3D animation.. In *UIST*. Citeseer, 423–434.
13. David Holman, Roel Vertegaal, Mark Altosaar, Nikolaus Troje, and Derek Johns. 2005. Paper windows: interaction techniques for digital paper. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 591–599.
14. Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and others. 2011. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 559–568.
15. Tyler Johnson and Henry Fuchs. 2007. Real-time projector tracking on complex geometry using ordinary imagery. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 1–8.
16. Andreas Jordt and Reinhard Koch. 2013. Direct model-based tracking of 3d object deformations in depth and color video. *International journal of computer vision* 102, 1-3 (2013), 239–255.
17. Shingo Kagami and Koichi Hashimoto. 2015. Sticky projection mapping: 450-fps tracking projection onto a moving planar surface. In *SIGGRAPH Asia 2015 Emerging Technologies*. ACM, 23.
18. Mohammadreza Khalilbeigi, Roman Lissermann, Wolfgang Kleine, and Jürgen Steimle. 2012. FoldMe: interacting with double-sided foldable displays. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*. ACM, 33–40.
19. Mohammadreza Khalilbeigi, Roman Lissermann, Max Mühlhäuser, and Jürgen Steimle. 2011. Xpaaand: interaction techniques for rollable displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2729–2732.
20. Alvin J Law, Daniel G Aliaga, Behzad Sajadi, Aditi Majumder, and Zygmunt Pizlo. 2011. Perceptually based appearance modification for compliant appearance editing. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 2288–2300.

21. Johnny C Lee, Scott E Hudson, and Edward Tse. 2008. Foldable interactive displays. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*. ACM, 287–290.
22. David G Lowe. 1999. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, Vol. 2. Ieee, 1150–1157.
23. David Molyneaux, Hans Gellersen, and Joe Finney. 2013. Cooperative augmentation of mobile smart objects with projected displays. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 3, 2 (2013), 7.
24. Daniel Moreno and Gabriel Taubin. 2012. Simple, accurate, and robust projector-camera calibration. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*. IEEE, 464–471.
25. Richard A Newcombe, Dieter Fox, and Steven M Seitz. 2015. DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 343–352.
26. Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. 2011. KinectFusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE, 127–136.
27. Kohei Okumura, Hiromasa Oku, and Masatoshi Ishikawa. 2012. Lumipen: Projection-based mixed reality for dynamic objects. In *Multimedia and Expo (ICME), 2012 IEEE International Conference on*. IEEE, 699–704.
28. Christoph Resch, Peter Keitler, and Gudrun Klinker. 2014. Sticky projections: a new approach to interactive shader lamp tracking. In *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*. IEEE, 151–156.
29. Szymon Rusinkiewicz and Marc Levoy. 2001. Efficient variants of the ICP algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*. IEEE, 145–152.
30. Behzad Sajadi, Mahdi Abbaspour Tehrani, Mehdi Rahimzadeh, and Aditi Majumder. 2015. High-resolution lighting of 3D reliefs using a network of projectors and cameras. In *3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2015*. IEEE, 1–4.
31. Christian Siegl, Matteo Colaianni, Lucas Thies, Justus Thies, Michael Zollhöfer, Shahram Izadi, Marc Stamminger, and Frank Bauer. 2015. Real-time pixel luminance optimization for dynamic multi-projection mapping. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 237.
32. Martin Spindler and Raimund Dachselt. 2009. PaperLens: advanced magic lens interaction above the tabletop. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. ACM, 7.
33. Jürgen Steimle, Andreas Jordt, and Pattie Maes. 2013. Flexpad: highly flexible bending interactions for projected handheld displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 237–246.
34. Jay Summet and Rahul Sukthankar. 2005. Tracking locations of moving hand-held displays using projected light. In *Pervasive Computing*. Springer, 37–46.
35. Junchen Wang, Hiromichi Suenaga, Keika Hoshi, Liangjing Yang, Etsuko Kobayashi, Ichiro Sakuma, and Hongen Liao. 2014. Augmented reality navigation with automatic marker-free image registration using 3-D image overlay for dental surgery. *Biomedical Engineering, IEEE Transactions on* 61, 4 (2014), 1295–1304.
36. Andrew Wilson, Hrvoje Benko, Shahram Izadi, and Otmar Hilliges. 2012. Steerable augmented reality with the beamatron. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 413–422.
37. Ruigang Yang and Greg Welch. 2001. Automatic and continuous projector display surface calibration using every-day imagery. In *Proc. of 9th International Conf. in Central Europe in Computer Graphics, Visualization, and Computer Vision*. Citeseer.
38. Zhengyou Zhang. 2000. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22, 11 (2000), 1330–1334.
39. Feng Zheng, Ryan Schubert, and Greg Weich. 2013. A general approach for closed-loop registration in AR. In *Virtual Reality (VR), 2013 IEEE*. IEEE, 47–50.