Calibrating an Optical See-Through Rig with Two Non-Overlapping Cameras: the Virtual Camera Framework

Jim Braux-Zin[1], Adrien Bartoli[2], Romain Dupont[1], Régis Vinciguerra[1]

[1]CEA, LIST, 91191 Gif-sur-Yvette, France, {jim.braux-zin, romain.dupont, regis.vinciguerra}@cea.fr
[2]ISIT, Université d'Auvergne, 63000 Clermont-Ferrand, France, adrien.bartoli@gmail.com

Abstract

We present a novel extrinsic calibration method for optical see-through systems. It is primarily aimed at tablet-like systems with a semi transparent screen, a camera tracking the user position and another camera analyzing the scene but easily generalizable to any optical see-through setup. Relative poses of the cameras and the screen are all needed for proper alignment. The proposed algorithm is based on the user indicating the projections onto the screen of several reference points chosen on a known object. A convex estimation is computed through the resectioning of virtual cameras and used to initialize a global bundle adjustment. Both synthetic and real experiments show the viability of our approach.

1. Introduction

Video see-through augmented reality systems – where virtual objects are added on a real scene video stream – are more and more widespread due to low cost and broad availability of compatible hardware. However, critical applications such as surgery or driving assistance cannot allow any level of indirection between the reality and the user. Optical see-through systems where the augmentations are directly superimposed on reality would be better suited to those cases, and would greatly improve the immersion feeling in more standard applications such as games. Until recently those systems were limited and reserved to niche usages. For example, airplanes Head Up Displays use a heavy and pricey combination of projector, lenses and mirrors and are mostly monochrome. An interest is growing in the industry toward transparent head mounted displays (glasses) but, despite recent improvements in weight and size, such systems are still invasive and subject to rapid moves of the head that makes scene analysis difficult.

In this paper, we focus on a tablet-like system composed

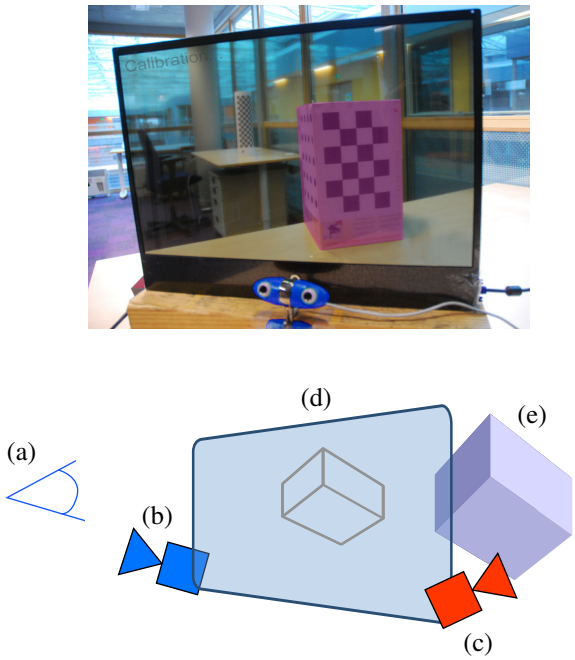

Figure 1: System setup. The components are: (a) user, (b) face tracking camera, (c) scene analyzing camera, (d) transparent screen, (e) scene

of a transparent LCD screen (Figure 1) with two mounted cameras facing opposite sides. This kind of setup alleviates most of the drawbacks of head-mounted displays while still being mobile and versatile. The front camera tracks the user position and the back camera analyzes the scene, providing in particular localization information. Their poses relative to the screen are needed in order to align the display with the reality but the cameras fields of view do not overlap and none of them can see the screen which makes classical calibration methods impractical.

The problem of extrinsic calibration of two non-overlapping cameras has already been investigated. Sev-

eral tracking-based methods have been proposed for mobile [3, 5, 9] or static [11, 2] cameras but none of these methods would allow to compute the pose of the screen as needed. To our knowledge, the only existing works that can give the required poses use a moving mirror to estimate the pose of an object located outside the camera field of view [14, 8, 12, 15]. Applying such methods on each camera with the screen as the target object would result in a fully calibrated system.

However these methods minimize the reprojection error of the target reflections (here the screen) into the camera which is not relevant in our configuration. What matters to the user is the *misalignment* error, i.e. the 2d distance on the screen between the displayed virtual scene and the real one. This is the idea behind most calibration methods of head-mounted displays [16]. By considering the whole system this approach makes no assumption on the sensors and tracking methods. It can be applied without changes to 3d sensors or electromagnetic head tracking for example. Moreover, errors in the trackers or in the model can be compensated. In this paper we present a new method minimizing the misalignment error in the context of the following calibration scenario.

A known object is put behind the screen such as being visible by the scene camera. The position in the camera coordinates of several reference points is assumed to be known. The user indicates from the other side the apparent projections of the object points on the screen, while the front camera tracks his position. The input is currently done by clicking on the projections in a predefined order and repeating the process from different points of view.

Problem and outline As formalized in Section 2, the calibration process should minimize the distance between the display and the user input. This error function is highly non convex due to the involvement of two rotations. In our goal to build a generic and versatile system[1], we do not use any prior information on the poses of the cameras and initialize the estimation with a robust convex scheme based on virtual cameras. This process is explained in Section 3 along with the bundle adjustment step that follows. At last we evaluate in Section 4 the accuracy of the method on both synthetic and real world data.

2. Problem formulation

We adopt the following notations: 3d points and vectors are represented with capital letters $X = (X_x, X_y, X_z)^T$, 2d points and vectors with lower-case letters x, matrices with bold capital letters $\mathbf{M}$ and cameras with calligraphic letters $\mathcal{C}$. The same notation is used to describe the camera itself and its associated coordinates system. The World coordinate system is noted $\mathcal{W}$. We will use the Mahalanobis norm written $||x||_\Sigma = \sqrt{x^T \Sigma^{-1} x}$.

There are three rigidly fixed components in the considered setup: the semi-transparent screen, the camera $\mathcal{C}_u$ looking at the user and the camera $\mathcal{C}_s$ looking at the scene. The transparent screen is used as the reference. It defines the world coordinate system $\mathcal{W}$, with the axis as in Figure 2a and the origin anywhere on the screen. The pose of $\mathcal{C}_u$ in $\mathcal{W}$ is defined by the transformation $\mathbf{M}_u = \begin{bmatrix}\mathbf{R}_u & T_u\end{bmatrix}$ with $\mathbf{R}_u$ the 3×3 rotation matrix and T_u the 3×1 translation vector. Similarly, $\mathbf{M}_s$, $\mathbf{R}_s$, and T_s define the pose of $\mathcal{C}_s$.

The m user positions in $\mathcal{C}_u$ are noted U_i, $i = 1 \dots m$. The n object reference points in $\mathcal{C}_s$ are noted O_j, $j = 1 \dots n$. In the world coordinates system, they are respectively $(\mathbf{M}_u U_i)_{i=1\dots m}$ and $(\mathbf{M}_s O_j)_{j=1\dots n}$.

For given user position U_i and object reference point O_j, c_{ij} is the intersection of the ray $(\mathbf{M}_u U_i, \mathbf{M}_s O_j)$ with the plane $z = 0$ (the screen) in $\mathcal{W}$. We define f_{int} the intersection function:

$$f_{\text{int}}\left(\begin{bmatrix}x_1\\y_1\\z_1\end{bmatrix}, \begin{bmatrix}x_2\\y_2\\z_2\end{bmatrix}\right) = \begin{pmatrix}x_1 - z_1 \times \frac{x_2 - x_1}{z_2 - z_1}\\ y_1 - z_1 \times \frac{y_2 - y_1}{z_2 - z_1}\end{pmatrix} \quad (1)$$

for $z_1 > 0$ and $z_2 < 0$ so we have:

$$c_{ij} = f_{\text{int}}(\mathbf{M}_u U_i, \mathbf{M}_s O_j) \quad (2)$$

The c_{ij} are expressed in world units: conversion from pixels coordinates in the screen to real world coordinates is straightforward given the dimensions of the screen.

Noise model The positions of the user and of the object points are subject to noise in the pose estimation. The user input is also subject to noise, due to human factors. The noises are assumed to be Gaussian and the covariance matrices diagonal (independent noise on each coordinate). We write with a tilde the noisy versions of the system inputs:

$$\widetilde{U}_i = U_i + n(0, \mathbf{\Sigma}_U) \quad (3)$$
$$\widetilde{O}_j = O_j + n(0, \mathbf{\Sigma}_O) \quad (4)$$
$$\widetilde{c}_{ij} = c_{ij} + n(0, \mathbf{\Sigma}_c) \quad (5)$$

for all i in $1 \dots m$ and j in $1 \dots n$. The noises are mostly constant for a given setup, and there are ways to estimate their covariances (see Appendix B for an example). This knowledge must be used to improve the calibration accuracy. We need to distinguish the estimated parameters from their true values. We note the estimates with a hat: $\widehat{\mathbf{M}}_u$, $\widehat{\mathbf{M}}_s$, $\widehat{U}_{i=1\dots m}$, $\widehat{O}_{j=1\dots n}$.

Calibration problem The goal of the proposed method is to find the best $\widehat{\mathbf{M}}_u$ and $\widehat{\mathbf{M}}_s$ given $\widetilde{U}_{i=1\dots m}$, $\widetilde{O}_{j=1\dots n}$

[1] Useful generalizations of the method are presented in Section 3.6

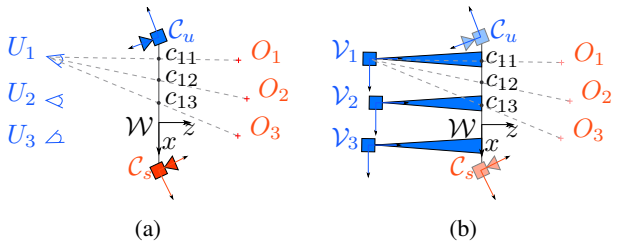

Figure 2: Top view of the system with (a) notations (same color meaning same coordinates system), (b) user-centered virtual cameras (see text).

and the corresponding $\widetilde{c}_{ij}$ (user inputs), using equation (2). With Gaussian noise, the optimal solution is given by minimizing the covariance-weighted bundle adjustment cost function:

$$f(\mathbf{M}_u, \mathbf{M}_s, U_{i=1\dots m}, O_{j=1\dots n}) = \sum_{\substack{i=1\dots m \\ j=1\dots n}} \sum \|\widetilde{c}_{ij} - f_{\text{int}}(\mathbf{M}_u U_i, \mathbf{M}_s O_j)\|^2_{\mathbf{\Sigma}_c} + \sum_{i=1\dots m} \left\|\widetilde{U}_i - U_i\right\|^2_{\mathbf{\Sigma}_U} + \sum_{j=1\dots n} \left\|\widetilde{O}_j - O_j\right\|^2_{\mathbf{\Sigma}_O} \tag{6}$$

The first term is the 2d alignment error and is highly non-convex making initialization critical. Next section presents a novel direct convex initialization.

3. Calibration with Virtual Cameras

3.1. Virtual cameras definition

The introduction of virtual cameras allows to see the problem as several classical camera resectioning problems. We define m virtual cameras $\mathcal{V}_i$ whose optical centers are the user positions, whose common focal plane is the screen plane ($z = 0$) and whose principal points lie at the origin of $\mathcal{W}$. Due to their definitions, the virtual cameras share some interesting properties. Their optical axis are all parallel and pointing toward the $+z$ axis in $\mathcal{W}$. Thus the poses of the cameras are defined by, for all i in $1 \dots m$:

$$\mathbf{M}_i = \begin{bmatrix} \mathbf{I} & T_i \end{bmatrix} \tag{7}$$

where $\mathbf{I}$ is the 3×3 identity matrix and $T_i = \mathbf{M}_u U_i$ is the position of U_i in $\mathcal{W}$. Since the c_{ij} are expressed in $\mathcal{W}$ units, the "pixels" of the virtual cameras are perfectly square: the focal lengths in x and y are equal and the skew is 0. Moreover, since the cameras are defined by their focal plane and principal point in $\mathcal{W}$, their intrinsic parameters are linked to the position of their optical center:

$$\mathbf{K}_i = \begin{bmatrix} f_i & 0 & u_i \\ 0 & f_i & v_i \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -T_{iz} & 0 & T_{ix} \\ 0 & -T_{iz} & T_{iy} \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

The c_{ij} are the projections of the object points in $\mathcal{V}_i$, so in homogeneous coordinates:

$$c_{ij} \propto \mathbf{K}_i \mathbf{M}_i^{-1} \mathbf{M}_s O_j \tag{9}$$

$$c_{ij} = \lambda_{ij} \mathbf{H}_i O_j \quad \lambda_{ij} \in \mathbb{R} \quad \forall j \in 1 \dots n \tag{10}$$

where

$$\mathbf{H}_i = \mathbf{K}_i \mathbf{M}'_i \text{ and } \mathbf{M}'_i = \mathbf{M}_i^{-1} \mathbf{M}_s = [\mathbf{R}_s | T_s - T_i] \tag{11}$$

3.2. Virtual cameras resectioning

We will see that computing the intrinsic and extrinsic parameters of a virtual camera $\mathcal{V}_i$ allows to estimate the pose of the real scene camera $\mathcal{C}_s$. This process is called camera resectioning [6]. We use the Direct Linear Transform (DLT) approach presented in [1] to estimate $\mathbf{H}_i$ from equation (10). The scalar factor is eliminated with a cross product:

$$c_{ij} \wedge \mathbf{H}_i O_j = 0 \quad \forall j \in 1 \dots n \tag{12}$$

These vector equations give $2 \times n$ independent linear equations in the coefficients of $\mathbf{H}_i$. With $n \geq 6$ object points, it is possible to compute a least square solution for the 12 coefficients using a SVD decomposition. This solution minimizes the algebraic distance as opposed to the geometric distance but is a suitable initialization for further refinement.

The next step is to extract the intrinsic $\mathbf{K}_i$ and extrinsic $\mathbf{M}'_i$ parameters, such as $\mathbf{H}_i = \mathbf{K}_i \mathbf{M}'_i$. We start from an initial direct decomposition using the equation

$$\overline{\mathbf{H}}_i \overline{\mathbf{H}}_i^T = \mathbf{K}_i \mathbf{R}_s \mathbf{R}_s^T \mathbf{K}_i^T = \mathbf{K}_i \mathbf{K}_i^T \tag{13}$$

where $\overline{\mathbf{H}}_i$ is the left 3×3 submatrix of $\mathbf{H}_i$. With noisy input, this gives a general intrinsic matrix unlikely to respect the properties of eq. (8):

$$\mathbf{K}_i^{\text{init}} = \begin{bmatrix} f_{xi} & s_i \cdot f_{xi} & u_i \\ 0 & f_{yi} & v_i \\ 0 & 0 & 1 \end{bmatrix} \tag{14}$$

This initial decomposition is iteratively refined using soft constraints [6] to force the intrinsic matrix into the proper form. The Levenberg-Marquardt algorithm is used to minimize:

$$f(\mathbf{K}_i, \mathbf{M}'_i, O_{j=1\dots n}) = \sum_{j=1\dots n} \|\widetilde{c}_{ij} - \lambda_{ij} \mathbf{K}_i \mathbf{M}'_i O_j\|^2_{\mathbf{\Sigma}_c} + \sum_{j=1\dots n} \left\|\widetilde{O}_j - O_j\right\|^2_{\mathbf{\Sigma}_O} + w\left(|f_{xi} - f_{yi}| + |s|\right) \tag{15}$$

where w is a weight variable which should be gently increased during iterations. In practice we observed that it is sufficient to do a full Levenberg-Marquardt optimization until convergence with $w = \frac{1}{\sigma_{\min}}$, where $\sigma_{\min}$ is the smallest diagonal coefficient of $\mathbf{\Sigma}_c$ and $\mathbf{\Sigma}_O$. This ensures the soft-constraint has an impact, regardless of the noise levels.

3.3. Extracting the pose of the real camera $\mathcal{C}_s$

It is now possible to extract T_i from $\mathbf{K}_i$ using equation (8). Then we build $\mathbf{M}_i$ from T_i using equation (7). At last we extract $\widehat{\mathbf{M}}_s^{(i)}$ from $\mathbf{M}'_i$ using equation (11).

We need a way to aggregate the estimates $\widehat{\mathbf{M}}_s^{(i)}$ into an optimal $\widehat{\mathbf{M}}_s$ to initialize a bundle adjustment. We start by eliminating obvious failures: virtual cameras whose non-linear refinement step failed or whose camera matrix has an improper shape like f_x and f_y focal lengths of opposite sign. Then we choose to compare them using the cost (16) after the refinement step and pick the best candidate. It appeared to be a good heuristic in experiments with more accurate results than averaging estimates.

3.4. Estimation of the user camera $\mathcal{C}_u$

The virtual cameras are centered on the user positions ; the resectioning of each $\mathcal{V}_i$ gives an estimate T_i of $\mathbf{M}_u U_i$. The $\widehat{\mathbf{M}}_u$ estimate can then be obtained by solving a 3d-3d alignment problem [7]. Another approach is to do the same camera resectioning process with virtual cameras centered on the object points to compute $\widehat{\mathbf{M}}_u$ as presented in Appendix A. Thus, there are three possible approaches that will be discussed:

Symmetrical: estimate $\widehat{\mathbf{M}}_s$ with *user*-centered virtual cameras and $\widehat{\mathbf{M}}_u$ with *object*-centered virtual cameras.

Only user-centered cameras: estimate $\widehat{\mathbf{M}}_s$ with *user*-centered virtual cameras and $\widehat{\mathbf{M}}_u$ with 3d-3d alignment.

Only object-centered cameras: estimate $\widehat{\mathbf{M}}_u$ with *object*-centered virtual cameras and $\widehat{\mathbf{M}}_s$ with 3d-3d alignment.

The most critical step of the calibration is the DLT. It is sensitive to noise on observations and not on the camera centers. As a result, it is better to "put" the camera centers on the noisier inputs (user positions or object reference points). In practice, it is more difficult to track the user than to locate a known object. The well-conditioning of the DLT is also tied to the non-planar constraint of the calibration points (see Section 3.6), hard to verify for user positions which are limited by the size and shape of the screen. Finally, in order to take the optimal estimate regardless of the configuration we estimate the reprojection error with every approach and pick the best candidate.

3.5. Bundle Adjustment

Once the initialization parameters $\widehat{\mathbf{M}}_u$ and $\widehat{\mathbf{M}}_s$ are estimated, a global covariance-weighted Bundle Adjustment step minimizes the function (6) using the Levenberg-Marquardt algorithm.

3.6. Discussions

Constraints and degenerate cases Constraints on the problem geometry arise from the DLT step. We already wrote that the constraint $m \geq 6$ is necessary for the solution to be unique. Moreover, some configurations of the object points could lead to degenerate cases and multiple solutions. Those cases are discussed in [6], the most notable ones being when the camera and points all lie on a twisted cubic or on the union of a plane and a single straight line containing the camera center. In particular, the reference object cannot be flat or placed too far from $\mathcal{C}_s$ so that it appears almost flat.

Multi-view optimization The aggregation issue (Section 3.3) arises from the fact that no multi-view constraint is enforced. The problem formulation with an observation matrix is

$$\begin{bmatrix} \lambda_{11}c_{11} & \cdots & \lambda_{1n}c_{1n} \\ \vdots & & \vdots \\ \lambda_{m1}c_{m1} & \cdots & \lambda_{mn}c_{mn} \end{bmatrix} = \begin{bmatrix} \mathbf{K}_1[\mathbf{R}_s|T_s - T_1] \\ \vdots \\ \mathbf{K}_m[\mathbf{R}_s|T_s - T_m] \end{bmatrix} \begin{bmatrix} O_1 \cdots O_n \end{bmatrix} \tag{16}$$

where the λ_{ij} are the projective depth which can be computed using fundamental matrices or recursive algorithms [17, 10]. One can see that the DLT approach ignores three constraints: the fact that $\mathbf{R}_s$ and T_s are shared by all views and that the T_i are linked to the $\mathbf{K}_i$ (8). To our knowledge, there is no way to compute a direct estimate taking advantage of these constraints. Moreover, the DLT approach allows to easily parallelize the algorithm.

Generalization of the method The method has been introduced as a way to calibrate a particular system, but great care has been taken so as to keep it versatile by not using any *a priori* knowledge on the camera poses. It ensues several possible generalizations. Head-mounted displays can be calibrated using only one user-centered virtual camera. At the opposite, by using only one object-centered virtual camera, it is possible to calibrate a head tracking system with a unique reference point. Another interesting configuration is when the object is static relative to the system. Then there is no need for a physical scene camera $\mathcal{C}_s$ which can be replaced by the object local coordinates system: $\mathbf{M}_s$ is then the pose of the object in $\mathcal{W}$. This allows an easy way to calibrate an interactive showcase setup. This is actually the configuration chosen for the test case (Appendix B) as it removes the burden of object pose estimation.

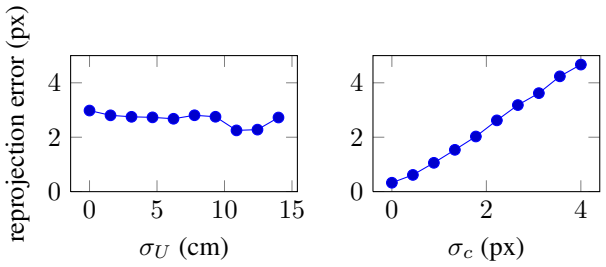

Figure 4: Bundle Adjustment evaluation on synthetic data

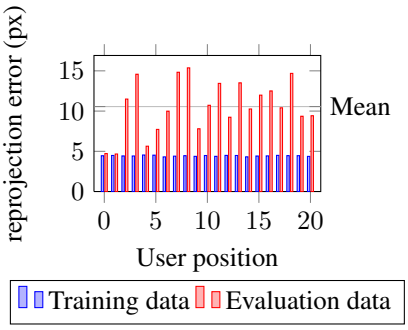

Figure 5: Leave-one-out cross validation

4. Evaluation

This section aims to demonstrate the robustness of our method and its accuracy. The algorithm has been implemented in the Python interpreted programming language with processing times of 1-3 seconds per calibration.

Evaluation on synthetic data To study the behaviour of the convex initialization, a synthetic scene is generated to mimic the geometry and noise of the real world test case presented in B. Starting from this realistic basis, the influence of several parameters is measured and averaged over 50 samples. The error in rotation and translation with regard to the ground truth is measured rather than the reprojection error in the virtual cameras which is not a good indicator of the quality of the initialization. In Figure 3, rows 1 and 2, it can be seen that noise on user positions does not affect user-centered virtual cameras estimation, and noise on object points does not affect object-centered virtual cameras. The most interesting outcome of those tests is the crucial importance of the problem geometry (see Section 3.6) as can be seen on the object scale plot (row 3).

The bundle adjustment accuracy is showed Figure 4. The reprojection error is almost constant at 3 pixels which is the value of the two dimensional noise. This proves the optimality of the solution.

Real world test The calibration process was put to the test in a real world setup as explained in B. A wireframe mesh of the box was aligned and displayed on the transparent screen. The alignment error was almost imperceptible, supporting the viability of our approach. In order to produce some quantitative results, we acquired data for 20 different user positions and realized a leave-one-out cross validation. The real camera and object poses are estimated with the data from 19 user positions and used to compute the alignment error from the left-out one. Results can be seen on Figure 5. The reprojection error average is about 10 pixels which corresponds to less than 3 mm on the screen. Examples of reprojections can be seen Figure 6.

Comparison with related work It is interesting to compare this result to the state of the art mirror-based methods [12, 15]. These two papers present real experiments of comparable scale and ideal conditions that show an angular error of $7^\circ = 0.122\,\mathrm{rad}$ in [12] and $0.057\,\mathrm{rad}$ in [15]. For small values of the angular error α of the camera orientation, the resulting position error of the user or object is $\alpha \cdot d$ where d is the distance to the camera.

For a user at 70 centimeters from the camera (center of the user position area in our setup), an error of $0.06\,\mathrm{rad}$ translates to a user position error of $0.06 \times 0.7 \approx 0.04$ meters. The object is placed at one meter from the screen so the object position error is $0.06 \times 1 = 0.06$ meters. The resulting alignment error is approximately $\frac{0.04+0.06}{2} = 0.05$ meters, without considering the translation errors and the trackers defects. This is not suitable for Augmented Reality applications and is more than 15 times higher than the 3 mm we obtained.

5. Conclusion

This article presents a solution to the calibration of an optical see-through system consisting of a transparent screen and two cameras. Our original scheme consisting of a convex initialization with virtual cameras followed by a bundle adjustment has proven to be robust and accurate – even with a simplistic user tracker – and generalizable to most common see-through configurations. Future works will involve the integration of a better user tracker and an augmented reality stack. We also plan to adapt the method to non-planar transparent surfaces such as car windshields.

Appendix

A. Object-centered Virtual Cameras

It is possible to do the camera resectioning process (see Section 3.2) using virtual cameras centered on object points. The problem is mostly symmetric with slight changes. Because their optical axis point toward the $-z$ axis, the poses

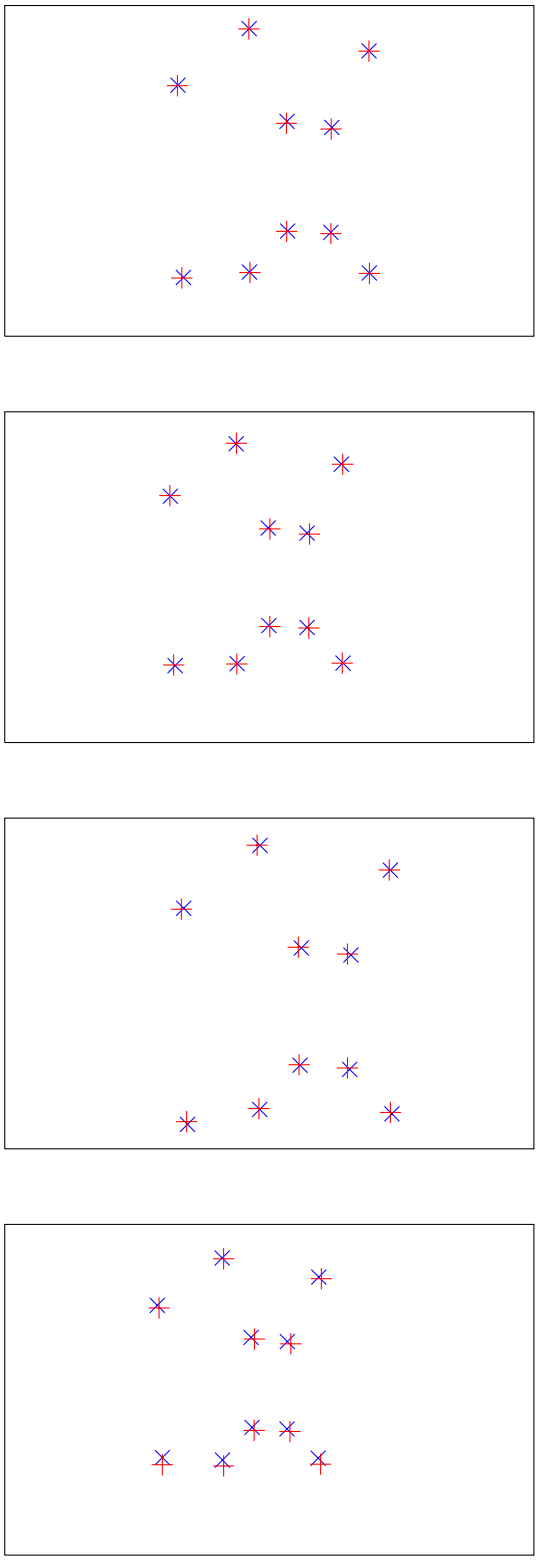

Figure 6: Examples of alignment quality on cross-validation data. Best viewed in color.

of the virtual cameras are defined by $\mathbf{M}_j^O = [\mathbf{R}^O | T_j^O]$ where $\mathbf{R}^O = \mathrm{diag}(-1, 1, -1)$ and $T_j^O = \mathbf{M}_s O_j$. The intrinsic equation (8) becomes:

$$\mathbf{K}_j^O = \begin{bmatrix} (T_j^O)_z & 0 & -(T_j^O)_x \\ 0 & (T_j^O)_z & (T_j^O)_y \\ 0 & 0 & 1 \end{bmatrix} \tag{17}$$

and the projections of the points $U_{i=1\ldots m}$ in the virtual camera j are $c_{ij}^O = (-(c_{ij})_x, (c_{ij})_y)^T$. Other than that, the resolution process is similar and leads to an estimate $\widehat{\mathbf{M}}_u$ of the pose of $\mathcal{C}_u$.

B. Setup details and noise estimation

The non-linear refinement steps and the bundle adjustment need an estimate of the variance of the noise on the user positions $\mathbf{\Sigma}_U$, object positions $\mathbf{\Sigma}_O$ and user inputs $\mathbf{\Sigma}_c$. We define here a real world test case used for evaluation and as a reference for synthetic data generation.

Object points Since the object is static in this real world test, the camera $\mathcal{C}_s$ is defined as the local coordinates system of the known object (interactive showcase configuration from Section 3.6). The only noise on the object points is due to errors in the model. In practice those are very small, we set

$$\mathbf{\Sigma}_O = \mathrm{diag}(1^2, 1^2, 1^2) \text{ in millimeters} \tag{18}$$

The object is a $372 \times 305 \times 229$ mm rectangular box put at approximately 1 meter from the screen. 10 reference points are chosen: the 6 visible corners and 4 arbitrary additional points on the visible sides.

User positions Any tracker can be used to track the user position, such as commercially available state of the art monocular head tracking softwares [13]. We use a Minoru USB stereo webcam[2] ($2 \times 640 \times 480$) detecting the user left pupil in each camera. The eye position is first estimated using the OpenCV eye detector [18], then a Mean-Shift [4] algorithm is used to find the pupil center. This method has the advantage of being absolute and more robust than tracking-based methods. However accurate pupil center detection with a low resolution camera is difficult without a dedicated setup such as a frontal infrared projector. Thus the estimated position is quite noisy with some jitter. To model the noise, we recorded the estimated positions of a user while he stood still for several hundreds of frames and computed the variance of this sequence. We got:

$$\mathbf{\Sigma}_U = \mathrm{diag}(5^2, 5^2, 20^2) \text{ in millimeters} \tag{19}$$

The position of the user is measured at all clicks and averaged over each sequence. As a result, for sequences of n

[2] http://www.minoru3d.com/

clicks, the variance of the measured user positions is divided by n i.e. $\mathbf{\Sigma}_{U_{\text{average}}} = \frac{1}{n}\mathbf{\Sigma}_U$. We use 10 clicks sequences so we have:

$$\mathbf{\Sigma}_{U_{\text{average}}}^{(n=10)} = \text{diag}(1.6^2, 1.6^2, 6.3^2) \text{ in millimeters} \quad (20)$$

The camera has a narrow field of view which restricts the user position to the area where

$$U_i \in [-300, 300] \times [-150, 150] \times [400, 1000] \text{ in mm} \quad (21)$$

This area is further shrinked by the constraint that all object points must be visible through the screen. The reference number of user positions has been fixed to 20.

User inputs The transparent screen is a 22" SAMSUNG LTI220MT02. The active area is 473.6×296.1 mm with a resolution of 1680×1050 pixels. The pixel size is then $s_{\text{px}} = 0.282$ mm/px.

The "noise" in the user inputs has several causes: slight blur introduced by the transparent screen, precision of the mouse (one pixel), shape of the mouse cursor, hand trembling, bad vision, inability to stand perfectly still during a clicks sequence and other human factors... In the considered setup, a user study shows the average error to be 3 pixels:

$$\begin{aligned} \mathbf{\Sigma}_c &= \text{diag}(3^2, 3^2) && \text{in pixels} \\ &= \text{diag}(0.846^2, 0.846^2) && \text{in millimeters} \end{aligned} \quad (22)$$

References

[1] Y. Abdel-Aziz and H. Karara. Direct linear transformation from comparator to object space coordinates in close-range photogrammetry. In *ASP Symposium on Close-Range Photogrammetry*, pages 1–18, 1971. 3

[2] N. Anjum, M. Taj, and A. Cavallaro. Relative Position Estimation of Non-Overlapping Cameras. In *ICASSP*, volume 2, pages II–281 –II–284, 2007. 2

[3] Y. Caspi. Alignment of non-overlapping sequences. In *ICCV*, 2001. 2

[4] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR*, 2000. 6

[5] S. Esquivel, F. Woelk, and R. Koch. Calibration of a Multi-camera Rig from Non-overlapping Views. In *Pattern Recognition*, volume 4713 of *Lecture Notes in Computer Science*, pages 82–91. Springer Berlin / Heidelberg, 2007. 2

[6] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004. 3, 4

[7] B. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(April), 1987. 4

[8] R. Kumar, A. Ilie, J. Frahm, and M. Pollefeys. Simple calibration of non-overlapping cameras with a mirror. In *CVPR*, 2008. 2

[9] P. Lebraly, E. Royer, O. Ait-Aider, C. Deymier, and M. Dhome. Fast calibration of embedded non-overlapping cameras. In *ICRA*, 2011. 2

[10] J. Oliensis and R. Hartley. Iterative extensions of the Sturm/Triggs algorithm: Convergence and nonconvergence. *PAMI*, 29(12):2217–2233, Dec. 2007. 4

[11] A. Rahimi, B. Dunagan, and T. Darrell. Simultaneous calibration and tracking with a network of non-overlapping sensors. In *CVPR*, 2004. 2

[12] R. Rodrigues, J. Barreto, and U. Nunes. Camera pose estimation using images of planar mirror reflections. In *ECCV*, 2010. 2, 5

[13] Seeing Machines. Faceapi. http://www.seeingmachines.com/product/faceapi/. 6

[14] P. Sturm and T. Bonfort. How to Compute the Pose of an Object without a Direct View? In *ACCV*, 2006. 2

[15] K. Takahashi, S. Nobuhara, and T. Matsuyama. A new mirror-based extrinsic camera calibration using an orthogonality constraint. In *CVPR*, 2012. 2, 5

[16] A. Tang, J. Zhou, and C. Owen. Evaluation of calibration procedures for optical see-through head-mounted displays. In *ISMAR*, 2003. 2

[17] B. Triggs. Factorization Methods for Projective Structure and Motion. In *CVPR*, 1996. 4

[18] P. Viola and M. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. In *CVPR*, 2001. 6

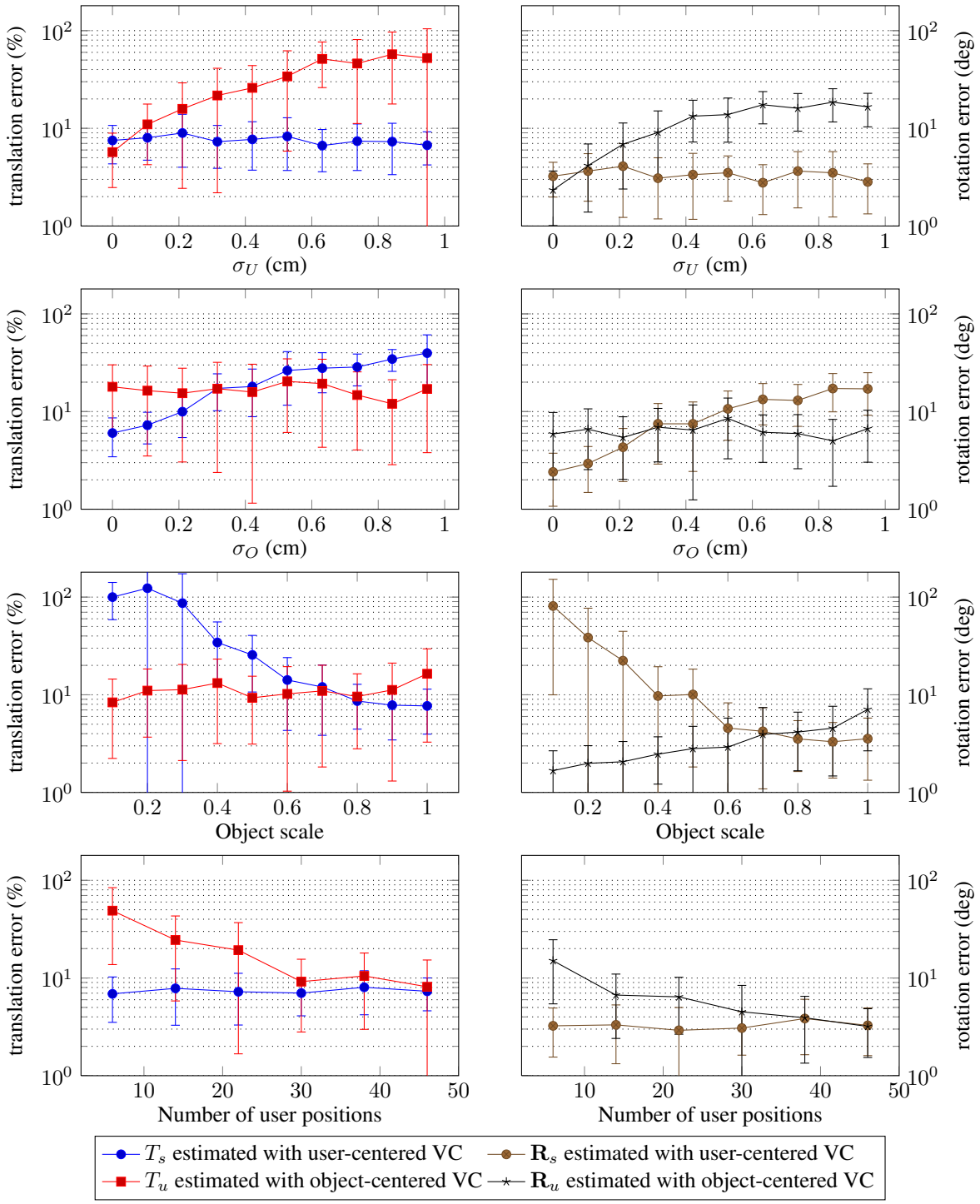

Figure 3: Comparison of the virtual cameras methods robustness to user positions noise, object positions noise, object scale and number of user positions. Dots correspond to the average value over 50 samples and bars to the standard deviation among those samples (best viewed in color). All plots use a logarithmic scale on the y-axis.