Direct 3D Manipulation Using Vision-Based Recognition of Uninstrumented Hands

Siniša Kolarić, Alberto Raposo, Marcelo Gattass

Tecgraf - Computer Graphics Technology Group - Department of Informatics
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro - RJ, Brazil
{skolaric}, {abraposo}, {mgattass}@tecgraf.puc-rio.br

Abstract

We describe a prototype of an interactive Mixed Reality application for direct spatial manipulation, based on the vision-based recognition of uninstrumented hands. In our application, we dynamically integrate both of the user's hands into the virtual environment, effectively creating a logical 2 × 3 DoF manipulation device. Using this logical device, we were able to implement, in conjuction with hand gestures, a number of fundamental 3D manipulation operations like select, deselect, translate, rotate, and scale.

1. Introduction

Most current 3D editors and viewers still operate within the WIMP (Windows, Icons, Menus and Pointing device) paradigm, thus utilizing the mouse and related devices, as well as various graphical metaphors, to manipulate 3D geometric objects. For example, to rotate an object, the user usually has to select that object with the mouse, then select the ROTATE tool from the menu or toolbar, and finally move (drag) the mouse with a mouse button pressed and the corresponding graphical manipulator widget activated. The planar (2 DoF) movement of the mouse thus gets translated into a 3D angular range constrained to a rotation plane, causing the object to rotate in the associated global coordinate system.

While the WIMP approach is a tried and familiar one, it coerces the user into performing intrinsically threedimensional operations like TRANSLATE, ROTATE or SCALE by means of a proxy planar mouse operation, which is dissimilar to the way humans actually perform these operations in the physical world, doing free-form hand movements. Given the current state-of-art in computing (especially computer vision), would it be possible to push the technological envelope just a little bit, and build a practical computer-vision based system that is capable of performing 3D manipulation operations in the most natural way, that is, using a user's own hands directly?

In this paper we describe how we built a prototype of such system, capable of manipulating 3D virtual objects by means of a small set of manipulating operations, utilizing a standard PC, a stereo pair of low-resolution cameras and a software application based on computer vision techniques. Using this technology, we integrated our hands into the virtual workspace, thus in effect creating a logical 2 × 3 = 6 DoF input device, capable to manipulate 3D objects in our virtual environment.

Our exposition is structured as follows: Section 1 (current) explains the motivation for this work. Section 2 gives an overview of the related work. Section 3 describes the user's workplace, and explains how we defined the operations for manipulating 3D geometric objects using uninstrumented hands. Section 4 describes technical details of how we utilized computer vision (CV) techniques to determine positions of certain hand features, and to recognize hand gestures, as well as describes experimental results. Section 5 discusses experimental results and future work.

2. Related work

We split our survey of related work into two principal parts: Section 2.1 gives a survey of the work done in the field of direct 3D manipulation, and Section 2.2 gives an overview of the relevant computer-vision techniques for hand recognition.

2.1. Direct 3D manipulation

The expression "direct manipulation", coined by Shneiderman [43], describes an interaction modality where the user can modify computational objects using actions that correspond at least loosely to the physical world. For the purposes of this exposition, we define "direct 3D manipulation" as a special direct manipulation type which:

• deals with manipulation of virtual 3D geometric objects,

• uses free-form hand movements for spatial input [16], and

• has a minimal (or equal to zero) spatial displacement between the user's physical hand (and of its virtual representation) and the manipulated virtual 3D object.

Systems capable to manipulate geometry include Sutherland's Sketchpad [45], and Clark's "Designing surfaces in 3-D" [8]. Bolt's "Put-that-there" system [3] uses a 6-DoF tracker and hand gestures, together with speech input, to manipulate simple shapes on a large, wall-sized screen.

Sachs et al present "3-Draw" [37], a 3D computer-aided design tool capable of drawing complex free-form shapes, using two 6-DoF sensors (one sensor to control 3D drawing and editing tools, and the other sensor to control an object's position and orientation). Krueger's VIDEODESK [22] uses overhead video cameras to track 2D hand positions, and is capable of manipulating splines by modifying their control points using index fingers and thumbs of both hands. Butterworth et al present 3DM [7], an interactive surface-modeling system that uses one single bat (3D mouse) with 6 DoF. Shaw and Green present THRED (Two-Handed Refining EDitor) [40], a free-form sketching system that uses two three-button bats, one for each hand (the dominant hand picks and manipulates 3D objects, and the less-dominant hand sets context). The JDCAD system [24] by Liang and Green uses a 6-DoF bat for spatial input. Deering's HoloSketch [12] is a VR system for 3D geometry creation and manipulation that uses head-tracked stereo glasses and a 3D mouse/wand ("one-fingered data glove") augmented by an offset digitizer rod, effectively making it a six-axis wand. Mapes and Moshell present PolyShop [26], which uses two ChordGloves (datagloves which have electric contacts on fingertips and on the palm) for bimanual spatial input. Mine's CHIMP (Chapel Hill Immersive Modeling Program) [28] uses two separate bats, one for each hand. User can perform a unimanual operation for translations and rotations, and a bimanual symmetric movement for scaling. Cutler et al present [9] two-handed direct manipulation on a "Responsive Workbench" [23], using pinch gloves, stylus providing single dinstinguished point of action, and a 6 DoF tracker attached to stereo shutter glasses for head tracking. Nishino et al [30] present a system using one- and two-handed gestures (deform, grasp, point, scale, rotation) to model and manipulate 3D objects, using data gloves. Schkolne, Pruett and Schr¨oder present "Surface Drawing" [39], [38] where shapes, drawn using wired datagloves, "float" in the space above Responsive Workbench. The FingARtips technique [6] by Buchmann et al tracks hand gestures by using image processing software and finger- and hand-based fiducial markers, and allows users to interact with virtual content using natural hand gestures. Bettio et al [2] present a system where the user stands in front of a large stereo display, and manipulates the model using optically tracked unmarked hands.

In the field of 3D interaction techniques, Strauss and Carey [44] describe graphical manipulators, like trackball, one-axis scale, jack, handle box, and one-axis translate. Mine [27] discusses virtual environment interaction including movement (specifiying direction and speed), selection (local and at-a-distance), manipulation (change in position, orientation and center of rotation) and scaling (center of scaling and scaling center; uniform and non-uniform scaling). Poupyrev et al showcase the Go-Go technique [33], for non-linear mapping for direct manipulation in Ves. Bowman and Hodges [4] evaluate techniques for grabbing and manipulating remote objects in virtual environments. Mine et al [29] address the lack of haptic feedback in VEs and propose to use the sense of proprioception. Zhai [48] reviews the usability of various 6-DoF input devices. Dachselt, Hinz and Huebner give a classification of 3D widgets [10] [11].

For hand postures and hand gestures, Quek [35] [34] and Pavlovic et al [32] give reviews in the context of HCI. For two-handed gesture, Guiard [14] gives a theoretical framework for the study of asymmetry in the context of human bimanual action.

2.2. Computer vision for hand recognition

In our work, we are interested in non-contact tracking of *uninstrumented* (i.e. unmarked, bared) human hands, and for this we use passive computer vision techniques. Therefore, we aren't considering more traditional methods like using colored gloves or attaching rectangular patterns onto hands, or using cybergloves. Also, we aren't considering active computer vision techniques, which use light projectors.

2.2.1. Vision-based hand detection.

Vision-based hand detection is a process that returns a negative answer if no hand has been detected in an image, and a positive answer otherwise. If detected, most hand detection methods will also return the location (for example, a bounding rectangle) of the hand in the picture. In our work, we use hand detection to initialize hand tracking (see Section 2.2.2 for an overview of hand tracking literature, and Section 4.4 for our hand tracking implementation). Detecting human hand in an arbitrary image is a difficult problem. Because of this, many past approaches adopted various constraints on experimental setups which made the tracking problem

easier, like constant backgrounds, markers, non-existence of other skin-colored objects in view, and colored gloves.

The object detection method that has taken the CV community by storm, and which we adopt in our work, was due to Viola and Jones [47]. This method is robust, fast and reliable, and can be trained to recognize any kind of object. It uses AdaBoost [13] learning algorithm to combine weak classifiers (those with at least (50 +)% of guessing probability, where is arbitrarily small) into strong, arbitrarily accurate classifiers. The application of Viola-Jones method to detecting specifically hands has been researched by K¨olsch and Turk in [21] and Ong and Bowden in [31].

2.2.2. Vision-based hand tracking. *Tracking* is the process of estimating the position of a tracked object, taking its previous position into consideration. Since the hand is an articulate 3D structure which gets projected onto the 2D image plane, we can choose to track the hand either in its original 3D space (in this case we say that we employ model-based hand tracking), or in the 2D projection image plane (appearance-based tracking). We can also talk about hybrid tracking, a recent mode of tracking which combines elements of model- and appearancebased tracking.

• Appearance-based hand tracking — various tracking methods in this class differ in what cues they use for tracking—some, for example, use just one cue like skin color or hand motion, and other use a combination of cues (for example, skin color and motion). For example, in Camshift [5] just the hand's color is being used as a cue; in CONDENSATION [17], hand contours + hand motion, and in ICONDENSATION [18] hand contours + hand motion + skin color; in "Flock of Features" [20] the combination skin color + KLT features is being used [41], [46] (see Section 4.4 for more on KLT features).

• Model-based hand tracking — here the backprojection of a predefined 3D parametric hand model is being matched against the input video frame. At each frame, extracted features are being compared with the current 3D model, and the matching error computed; if the error is too large, the 3D model is adjusted in the attempt to decrease the error—if the error is still too big, we repeat the model adjustment, otherwise we found a good matching and the tracking was successful. Examples include the classic DigitEyes system [36], where a 27-DoF hand model is being tracked.

• Hybrid hand tracking — here elements of both the model-based and appearance-based tracking are combined in an effort to get the best of two worlds. Shimada et al in [42] and Athitsos and Sclaroff in [1] synthesize a large number of 2D views of a software 3D hand model, and tag each of these views by the corresponding, exact hand pose vector. After this preprocessing step, appearance-based matching methods are used to process real images.

2.2.3. Human skin color modeling and detection. Human skin color is an important cue that can be useful for both hand detection and hand tracking; it can be used either standalone (i.e. on its own) for both detection and tracking, or as a helper method, i.e. as a means to increase the robustness of other detection and tracking methods. In their recent work [19], Kakumanu et al give a detailed and comprehensive survey of human skin color modeling approaches, as well as of human skin detection methods.

2.2.4. Stereo 3D reconstruction. By expression stereo 3D reconstruction we refer to the process and techniques for determining 3D position and 3D structure from pairs of correspondences in the left and the right image of the input stereo stream. For an overview of triangulation methods, see Hartley and Sturm [15].

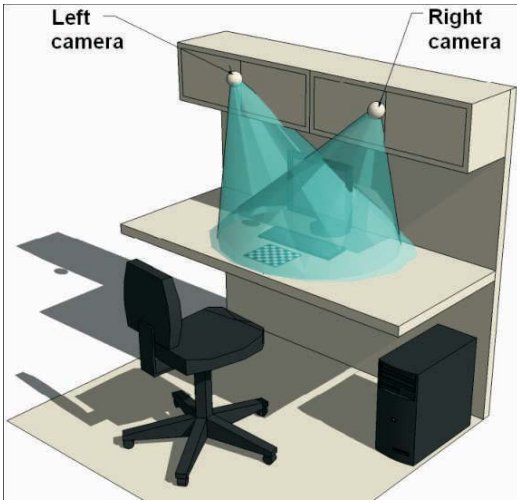

Figure 1. The user's workplace.

3. Direct 3D manipulation using uninstrumented hands

3.1. User's workplace

The workplace (Figure 1), as we define it, consists of a standard office cubicle equipped with a personal computer with two cameras (i.e. a stereo pair)

connected. Cameras are fixed at the top of the cubicle and are directed down, at a certain angle, relative to the surface of the desktop. A stereo pair of cameras enables us, due to the phenomenon called stereo disparity, to estimate 3D positions of various hand features, thus offering us a way to integrate our hands into the VE.

Accordingly, we define the workspace as the intersection of the two visual cones defined by the respective cameras' fields of view—the user must move his hands in this working space, in order for the system to register hand movements and gestures. If a hand exits the workspace, the system stops tracking the hand.

3.2. Defining hand postures

Figure 2 depicts the three hand postures we use in our application. Note that the image depicts the right hand only, but both the left and the right hand can assume these postures. (The left hand assumes postures which are simply mirrored relative to the vertical axis.) Also please note that the hand postures shown in the image are inclined at an angle, which approximates the natural hand inclination when it is being tracked in the workspace. We now define the following hand postures (also called hand states), from which all the manipulation operations are being defined (see Section 3.3), as follows:

1. `HAND_POSTURE_OPEN` — flat palm with all fingers spread apart

2. `HAND_POSTURE_POINTING` — all fingers closed, except the index finger

3. `HAND_POSTURE_FIST` — all fingers closed

As a matter of convenience, we defined one more posture, `HAND_POSTURE_UNKNOWN` , which designates any hand posture that is not recognized by the system.

3.3. Defining direct 3D manipulation operations

Using the hand postures defined above, we now define direct 3D manipulation operations. Manipulation operations can be either one-handed or two-handed:

1. `OP_SELECT` (one-handed) — selects an object. Based on the posture `HAND_POSTURE_POINTING`.

2. `OP_DESELECT` (one-handed) — deselects an object. Based on the posture `HAND_POSTURE_POINTING`.

3. `OP_TRANSLATE` (one-handed) — translates (moves) selected objects. Based on the posture

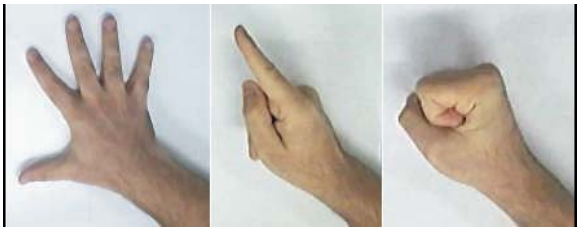

Figure 2. Three hand postures utilized by the system: `HAND_POSTURE_OPEN` (left), `HAND_POSTURE_POINTING` (middle) and `HAND_POSTURE_FIST` (right).

5. `OP_SCALE` (two-handed) — scales objects. Based on two `HAND_POSTURE_FIST` hand postures.

Therefore, we have two two-handed spatial operations: `OP_ROTATE` and `OP_SCALE` — these use both hands at the same time. The remaining three spatial operations (`OP_SELECT, OP_DESELECT` and `OP_TRANSLATE`) are one-handed — must be performed by just one hand, either just by the left or just by the right hand. We will now describe each of these operations in detail. 3.3.1. Selecting and deselecting objects.

Operation `OP_SELECT` selects an object in the scene, while operation `OP_DESELECT` deselects an (already selected) 3D object. In order to (de)select a 3D object, user extends the index finger of one and exactly one hand (thus changing that hand's state into `HAND_POSTURE_POINTING`), and moves the hand into the object. (We emphasized "one and exactly one" because two tracked hands in state `HAND_POSTURE_POINTING` perform the operation `OP_ROTATE` , see Section 3.3.3.)

As soon as the application detects that the tracked hand's centroid entered the interior of the object, while the hand is in state `HAND_POSTURE_POINTING` , the objects gets (de)selected. The user can now move her hand out of the object; the object stays (de)selected.

In WIMP terms, operation `OP_SELECT` is equivalent to moving the mouse pointer onto a screen object (for example, onto an icon) and then pressing the mouse button, thus selecting the icon. Similarly, operation `OP_DESELECT` is equivalent to moving the mouse pointer onto an already selected screen object (for example, onto an already selected icon) and then pressing the mouse button, which deselects the icon.

3.3.2. Translating objects. Operation `OP_TRANSLATE` translates (moves) all the currently selected objects (Figure 3). For this to work, one and exactly one hand must be in the

HAND_POSTURE_FIST state (We say "exactly one" because if both tracked hands are in the HAND_POSTURE_FIST state, we will be performing the OP_SCALE operation, see Section 3.3.4.).

The operation initiates at the moment the user changes the state of one (and exactly one) of her hands into HAND_POSTURE_FIST. The position of that hand's centroid is then the starting point of the translation vector. User moves the hand about, and the selected objects move along too. When the user decides the translation is just right, she changes the hand's state into HAND_POSTURE_OPEN thus terminating the OP_TRANSLATE operation.

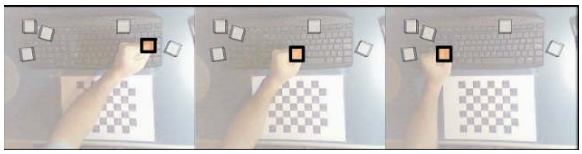

Figure 3. OP_TRANSLATE operation, based on one HAND_POSTURE_FIST posture.

3.3.3. Rotating objects. Operation OP_ROTATE rotates all the currently selected objects (Figure 4). At the moment when the application detects that both hands have their index finger extended (thus entering into the HAND_POSTURE_POINTING state), the hands' respective positions get memorized and defined as two points *A, B* of the initial rotation axis *AB*, with rotation center *C* at the middle point between those two points:

$$C = \frac{AB}{2}.$$

The user can now move her hands about (all the while both hands stay in the HAND_POSTURE_POINTING state), and the selected objects rotate too around C. When the user decides to stop the rotation, she changes both hands' state into HAND_POSTURE_OPEN thus terminating the OP_ROTATE operation.

3.3.4. Scaling objects. Operation OP_SCALE scales all the currently selected objects (Figure 5). At the moment both hands enter into the HAND_POSTURE_FIST state, the centroids of both hands get memorized and defined as two points A,B of the initial scaling axis AB, with center C at the middle point between those two points:

$$C = \frac{AB}{2}$$

Now the user can move her hands (all the while both hands stay in the HAND_POSTURE_FIST state), and the selected objects scale in real-time around *C*, in the directions defined by three axes inferred from *AB*.

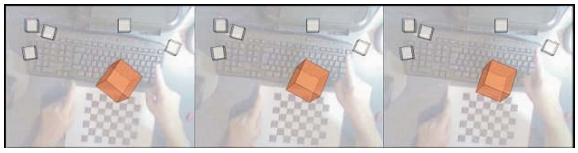

Figure 4. The two-handed OP_TRANSLATE operation is based on two HAND_POSTURE_POINTING postures. An example of CCW rotation is shown.

When the user decides to stop the scaling, she changes one (or both) hand's state into HAND_POSTURE_OPEN thus terminating the OP_SCALE operation.

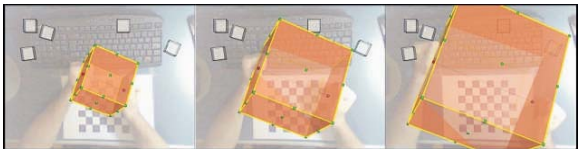

Figure 5. The two-handed OP_SCALE operation is based on two HAND_POSTURE_FIST postures.

4. Vision-based hand recognition

In this section we describe how we used computer vision techniques in order to detect, track and recognize hands and their gestures in the workspace defined in Section 3.1.

4.1. Calibrating the stereo rig

Before anything, the stereo rig must be calibrated i.e. its parameters (intrinsic and extrinsic) determined. By knowing these camera parameters, CV techniques we adopted can reconstruct 3D position of our hands in the workspace, using any triangulation method, for example the Hartley- Sturm method [15].

The set K of intrinsic parameters for a single camera includes two focal lengths (f_x, f_y), the principal point (o_x, o_y) and four distortion parameters (k_1, k_2, k_3, k_4):

$$K = \{(f_x, f_y), (o_x, o_y), (k_1, k_2, k_3, k_4)\}$$

We determined these intrinsic parameters using the Zhang's method [49]. For the calibration pattern we used a 8 × 7 checkerboard pattern.

Having determined two sets K_L, K_R of intrinsic parameters (for the left and right camera), we can proceed to determining extrinsic parameters (orientation and distance of a camera relative to the pattern). For this, we now fix the 8 × 7 checkerboard on the desk surface, and orient cameras so that they both have the pattern in their field of view. Since we already know both cameras' intrinsic parameters, we can now use a function from the OpenCV1 library to compute just the extrinsic parameters (rotation matrices

R_L and R_R, and translation vectors $\vec{T_L}$ and $\vec{T_R}$ of both cameras, relative to the pattern we've just fixed on the desk's surface. The extrinsic parameters R and $\vec{T}$ of the stereo rig are then simply

$$R = R_R R_L^T \qquad \vec{T} = \vec{T_L} - R^T \vec{T_R}$$

These parameters R and $\vec{T}$ now completely determine the geometry of our stereo rig and allow us to perform absolute, Euclidean 3D reconstruction of the hand's 3D position in the workspace shown in Figure 1.

4.2. Hand detection

With the stereo rig calibrated, we can now proceed to detecting hands in the stereo input video stream. Here detection serves for the purpose of initiating the process of tracking, described in Section 4.4 below.

For this end, we define two "detection areas" (left and right), one for each of both hands in the application client area. By definition, if a hand is not being tracked, its "detection area" gets shown on the application screen as a red rectangle, at the predetermined location and with a predetermined size. If the user now moves her hand into the corresponding detection area, and puts her hand into the predefined posture (we chose HAND POSTURE OPEN as the tracking initialization posture), the system will detect the hand, output the corresponding bounding rectangle and start tracking the hand within this bounding rectangle.

As we've already mentioned, we chose the Viola-Jones method as our detection method, which requires training and validation using four sets of samples:

• Two positive sample sets:

  – **Positive training set A** - for this set we moved our right hand in posture `HAND_POSTURE_OPEN` randomly in the workspace, approximately under the natural inclination (see Figure 2 left), under our lab's standard lighting conditions, and took 1000 photos containing the hand. Note that "approximately natural inclination" indicates that we included a number of shots of hands rotated to a degree relative to all three axes, in order to increase the robustness of the classifier.

  – **Positive validation set B** - under the same conditions as above, we took additional 100 photos to be used as validation images after the training is complete.

• Two negative sample sets:

  – **Negative training set C -** for the negatives, we created a set of 1000 images that do not contain hands in posture `HAND_POSTURE_OPEN` , using a public image database (CMU VASC Image Database).

  – **Negative validation set D -** we created an additional negative validation set containing 100 images, using the same image database.

We then used OpenCV facilities to train boosted cascades of weak classifiers in the following way:

1. we manually marked bounding rectangles for hands in the positive training samples, and saved the list of bounding rectangles in a file F.

2. before training, we set the required false positives threshold to be 10- 6.

3. we ran the training tool on file F and on the two training sets, the positive A and negative C. After the training has completed, we obtained a 15-stage classifier for posture `HAND_POSTURE_OPEN` with detection rate of approximately 98%.

4. we successfully tested the classifier using sets B and D.

5. we built the trained classifier into our prototype application. An OpenCV function loads the classifier; other function detects hands in posture `HAND_POSTURE_OPEN` in the current video frame. Note that we trained the classifier with our right hand; for the left hand, before the recognition stage we mirror the left side of the application area in order to be able to use the same classifier to recognize the left hand.

4.3. Hand segmentation based on human skin color

The hand detection method described above not only gives an answer whether there is or isn't a hand in an image, but also provides the bounding rectangle of the image region containing the hand. Considering this region of interest (ROI) only, we now make use of the characteristic hue of human skin to determine the pixels belonging to a hand. The reason we perform this segmentation is to increase the hand tracking robustness— see Section 4.4.

To this end, we used color histograms — both in the detection stage (using HSV color space), and for the learning (using normalized RGB histogram) of the color of the hand that has just been detected, i.e. we perform color learning immediately after the hand has been detected (pre-tracking stage).

4.4. Hand tracking

After a hand has been detected in an image, and hand pixels color-segmented using the properties of the

* www.intel.com/technology/computing/opencv/

human skin, we start tracking it. For tracking we chose the method proposed by K¨olsch in [20], which in turn is based on Kanade-Lucas-Tomasi (KLT) features, also called "good features to track". We chose this method because it is robust and can track hands in complex, cluttered environments.

KLT features are based on the early work done in [25], and then developed further in [46] and [41]. To increase robustness, the "Flocks of Features" approach to tracking by K¨olsch adds two additional properties to simple KLT tracking:

• tracked KLT features never exceed a predetermined maximum distance from the median of all tracked KLT features, and

• tracked KLT features can never be closer to each other than a predetermined minimum distance.

Differently from the application showcased by Kölsch in [20], which is able to track only one hand using just one (monocular) camera, our application: 1) implements four fully independent object trackers (four due to each camera tracking up to two hands), and 2) uses stereo disparity for 3D reconstruction of the hand's position in 3D workspace.

We now clarify what is meant by "tracking a hand". After a hand has been detected as explained in Section 4.2 in both cameras' views, and hand pixels color-segmented, up to *N* (for example, 100) KLT features are being collocated on the hand (i.e. on the blob defined by the detected hand's pixels). By averaging in each frame the 2D positions of all of these *N* features, we obtain a mean (average) position *P* of the hand being tracked. Therefore the 2D position *P* is the output of the tracking routine. Since we can have up to two active (tracked) hands, and each hand gives rise to one triangulated 3D position, we can have up to 2×3 = 6 DoF at our disposal to implement spatial manipulation operations.

3D reconstruction (triangulation). Finally, with the two corresponding 2D points u, u0 tracked (point u in the left camera view, point u0 it the right camera view, we can compute, in real time, the global 3D position x of a hand (either the left or the right hand) in the workspace. For this we use the triangulation method by Hartley and Sturm [15], a fast, non-iterative method that always finds the global optimum under the assumption of Gaussian noise present in the input images.

4.5. Hand posture recognition

The last step in the CV pipeline is the hand posture recognition, which enables us to implement a simple static gesture recognition. For hand posture recognition, we again use the Viola-Jones method. For this we repeated the training process explained in Section 4.2, only with positive samples containing other postures besides `HAND_POSTURE_OPEN`.

4.6. Tests

The software application was developed utilizing the C++ language, OpenCV computer vision library and OpenGL graphics library. All experiments were done on a personal computer equipped with an 2.66 GHz dual core processor, 2 GB RAM, and two web cameras connected to USB 2.0 ports grabbing 30 color frames per second at the resolution of 320 × 240 pixels.

Using the system described, we achieved trackingrelated latencies from 7 to 30ms with just one hand tracked (i.e. with two trackers active), and up to 60ms with both hands tracked (i.e. with all four trackers active). Taking the application as a whole, i.e. taking all the other system processes into consideration, we achived frame rates from 8 to 15 fps.

We'll now assess qualitatively estimation accuracy for a hand's position. Since the difference between hand's estimated position and ground truth is difficult to measure for an uninstrumented hand, we give here the figures demonstrating the hand's trajectory in space, from which we can deduce visually the amount of noise present in estimated positions. We trace three simple figures in space with the right hand: a line, a circle and an "eight" figure (Figure 6).

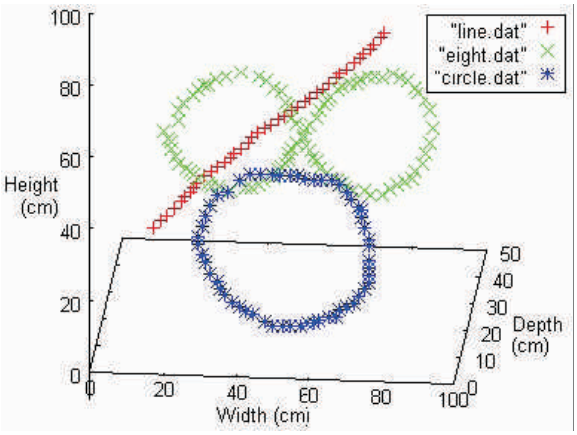

Figure 6. 3D plot of estimated hand positions, obtained by tracing a line, a circle and an "eight" in the workspace.

5. Discussion and further work

We presented an application where the user can employ intuitive manipulative hand gestures in order to manipulate 3D virtual objects. Considering the good

latencies and fps rates we achieved, our system fulfills the requirements of an interactive system.

Since we chose a robust and fast method for hand detection, we are not limited to physical setups as shown in Figure 1 — the system can perform the detection, for example, in outdoors environment too. The same holds for the tracking method we chose — due to its robustness, we can track hands in much more complex and more cluttered environments.

Due to these properties, future work includes expanding the system to work in diverse environments, and not just in the highly controlled environment shown in Figure 1. Further, we would like to increase the expressiveness of direct manipulation by including fingers into the manipulation operations. This entails tracking not only the 2-DoF point (global hand position) in the image plane, but some type of model-based tracking capable to track fingers as well. Finally, we would like to augment the set of manipulation operations by adding more complex, physically based manipulation and deformation operations, like "attract", "repel", "cut", "shear" and similar.

Acknowledgments

This work was performed at Tecgraf/PUC-Rio, a laboratory mainly funded by PETROBRAS.

6. References

[1] V. Athitsos and S. Sclaroff. 3d hand pose estimation by finding appearance-based matches in a large database of training views. *Technical Report BU-CS-TR-2001-021*, Computer Science Department, Boston University, Boston, USA, 2001.

[2] F. Bettio, A. Giachetti, E. Gobbetti, F. Marton, and G. Pintore. A practical vision based approach to unencumbered direct spatial manipulation in virtual worlds. In *Eurographics Italian Chapter Conference*, Conference held in Trento, Italy, February 2007. Eurographics Association.

[3] R. A. Bolt. put-that-there: Voice and gesture at the graphics interface. In SIGGRAPH '80: *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, pages 262–270, New York, NY, USA, 1980. ACM.

[4] D. A. Bowman and L. F. Hodges. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *Symposium on Interactive 3D Graphics*, pages 35–38, 182, 1997.

[5] G. R. Bradski. Computer vision face tracking for use in a perceptual user interface. Intel Technology Journal, (Q2):15, 1998.

[6] V. Buchmann, S. Violich, M. Billinghurst, and A. Cockburn. Fingartips: gesture based direct manipulation in augmented reality. In *GRAPHITE*, pages 212–221, 2004.

[7] J. Butterworth, A. Davidson, S. Hench, and M. T. Olano. 3dm: a three dimensional modeler using a head-mounted display. In *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 135–138, New York, NY, USA, 1992. ACM.

[8] J. H. Clark. Designing surfaces in 3-d. Commun. ACM, 19(8):454–460, 1976.

[9] L. D. Cutler, B. Froehlich, and P. Hanrahan. Two-handed direct manipulation on the responsive workbench. In *Symposium on Interactive 3D Graphics*, pages 107–114, 191, 1997.

[10] R. Dachselt and M. Hinz. Three-dimensional widgets revisited - towards future standardization. In *Proceedings of the Workshop 'New Directions in 3D User interfaces'*, 2005.

[11] R. Dachselt and A. Huebner. Virtual environments: Three dimensional menus: A survey and taxonomy. *Comput. Graph.,* 31(1) :53–65, 2007.

[12] M. F. Deering. Holosketch: a virtual reality sketching/ animation tool. *ACM Trans. Comput.-Hum. Interact.,* 2(3): 220–238, 1995.

[13] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[14] Y. Guiard. Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model, 1987.

[15] R. Hartley and P. Sturm. *Triangulation. 68(2)*: 146–157, November 1997.

[16] K. Hinckley, R. Pausch, J. C. Goble, and N. F. Kassell. A survey of design issues in spatial input. In *UIST '94: Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 213–222, New York, NY, USA, 1994. ACM.

[17] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision,* 29(1):5–28, 1998.

[18] M. Isard and A. Blake. ICONDENSATION: Unifying lowlevel and high-level tracking in a stochastic framework. *Lecture Notes in Computer Science*, 1406:893–908, 1998.

[19] P. Kakumanu, S. Makrogiannis, and N. Bourbakis. A survey of skin-color modeling and detection methods. *Pattern Recogn.*, 40(3):1106–1122, 2007.

[20] M. Kolsch and M. Turk. Fast 2d hand tracking with flocks of features and multi-cue integration. In *CVPRW'04: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04)* Volume 10, page 158, Washington, DC, USA, 2004. IEEE Computer Society.

[21] M. Kölsch and M. Turk. Robust hand detection. In *FGR*, pages 614–619, 2004.

[22] M. Krueger. *Artificial Reality II*. Addison-Wesley: Reading, MA, 1991., second edition, 1991.

[23] W. Krueger and B. Froehlich. The responsive workbench [virtual work environment]. *Computer Graphics and Applications, IEEE,* 14(3):12–15, May 1994.

[24] J. Liang and M. Green. Jdcad: a highly interactive 3d modeling system. *Computers & Graphics*, 18(4):499–506, 1994.

[25] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision, 1981.

[26] D. P. Mapes and J. M. Moshell. A two-handed interface for object manipulation in virtual environments. *Presence*, pages 403–416, 1995.

[27] M. Mine. Virtual environment interaction techniques. Technical Report TR93-005, UNC Chapel Hill, Dept of Computer Science, North Carolina, USA, 1995.

[28] M. R. Mine. Working in a virtual world: Interaction techniques used in the chapel hill immersive modeling program. *Technical Report TR96-029*, 12, 1996.

[29] M. R. Mine, F. P. Brooks, Jr., and C. H. Sequin. Moving objects in space: Exploiting proprioception in virtualenvironment interaction. *Computer Graphics*, 31(Annual Conference Series):19–26, 1997.

[30] H. Nishino, K. Utsumiya, and K. Korida. 3d object modeling using spatial and pictographic gestures. In VRST, pages 51– 58, 1998.

[31] E. Ong and R. Bowden. A boosted classifier tree for hand shape detection, 2004.

[32] V. Pavlovic, R. Sharma, and T. Huang. Visual interpretation of hand gestures for human-computer interaction: A review, 1997.

[[33] I. Poupyrev, M. Billinghurst, S. Weghorst, and T. Ichikawa. The go-go interaction technique: non-linear mapping for direct manipulation in vr. In *UIST '96: Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 79–80, New York, NY, USA, 1996. ACM. [34] F. Quek. Eyes in the interface. IVC, 13(6):511–525, August 1995.

[35] F. K. H. Quek. Toward a vision-based hand gesture interface. In *VRST '94: Proceedings of the conference on Virtual reality software and technology*, pages 17–31, River Edge, NJ, USA, 1994. World Scientific Publishing Co., Inc.

[36] J. M. Rehg and T. Kanade. Visual tracking of high DOF articulated structures: an application to human hand tracking. In *ECCV (2)*, pages 35–46, 1994.

[37] E. Sachs, D. Stoops, and A. Roberts. 3-draw: a three dimensional computer aided design tool. *Systems, Man and Cybernetics, 1989. Conference Proceedings., IEEE International Conference on*, pages 1194–1196 vol.3, 14-17 Nov 1989.

[38] S. Schkolne, M. Pruett, and P. Schr¨oder. Surface drawing: creating organic 3d shapes with the hand and tangible tools. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 261–268, New York, NY, USA, 2001. ACM.

[39] S. Schkolne and P. Schroder. Surface drawing. *Technical Report CS-TR-99-03*, Pasadena, CA, USA, 1999.

[40] C. Shaw and M. Green. THRED: A two-handed design system. Multimedia Systems, 5(2):126–139, 1997.

[41] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994.

[42] N. Shimada, K. Kimura, and Y. Shirai. Real-time 3-d hand posture estimation based on 2-d appearance retrieval using monocular camera. ratfg-rts, 00:0023, 2001.

[43] B. Shneiderman. Direct manipulation. a step beyond programming languages. *IEEE Transactions on Computers*, 16(8):57–69, August 1983.

[44] P. S. Strauss and R. Carey. An object-oriented 3d graphics toolkit. *SIGGRAPH Comput. Graph.*, 26(2):341–349, 1992.

[45] I. Sutherland. Sketchpad: A man-machine graphical communication system. *PhD thesis, Massachusetts Institute of Technology*, January 1963.

[46] C. Tomasi and T. Kanade. Detection and tracking of point features. *Technical Report CMU-CS-90-166*, Carnegie Mellon University, USA, Apr. 1991.

[47] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2002.

[48] S. Zhai. User performance in relation to 3d input device design. *SIGGRAPH Comput. Graph.*, 32(4):50–54, 1998.

[49] Z. Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1330–1334, 2000.