Evaluating RGB+D Hand Posture Detection Methods for Mobile 3D Interaction

Daniel Fritz
Vienna University of Technology
Favoritenstr. 9-11/188/2
1040 Wien, Austria
+43-1-58801-18802
mail@danielfritz.at

Annette Mossel
Vienna University of Technology
Favoritenstr. 9-11/188/2
1040 Wien, Austria
+43-1-58801-18893
mossel@ims.tuwien.ac.at

Hannes Kaufmann
Vienna University of Technology
Favoritenstr. 9-11/188/2
1040 Wien, Austria
+43-1-58801-18860
kaufmann@ims.tuwien.ac.at

ABSTRACT
In mobile applications it is crucial to provide intuitive means for 2D and 3D interaction. A large number of techniques exist to support a natural user interface (NUI) by detecting the user's hand posture in RGB+D (depth) data. Depending on a given interaction scenario, each technique has its advantages and disadvantages. To evaluate the performance of the various techniques on a mobile device, we conducted a systematic study by comparing the accuracy of five common posture recognition approaches with varying illumination and background. To be able to perform this study, we developed a powerful software framework that is capable of processing and fusing RGB and depth data directly on a handheld device. Overall results reveal best recognition rate of posture detection for combined RGB+D data at the expense of update rate. Finally, to support users in choosing the appropriate technique for their specific mobile interaction task, we derived guidelines based on our study.

Categories and Subject Descriptors
H.5.2 [Information Interfaces and Presentation]: User Interfaces| Interaction styles, I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism | Virtual Reality

General Terms
Algorithms, Performance, Reliability

Keywords
Natural User Interface, RGB+D Hand Posture Detection, Handheld Augmented Reality, Comparative Study

1. INTRODUCTION
Over the last years, progress in the field of natural user interaction (NUI) as a new paradigm of human computer interaction (HCI) was huge. From multi-touch displays to finger and gesture recognition with RGB images as well as depth data, a large number of 2D and 3D interaction methods have been developed to provide intuitive interfaces. While multi-touch approaches require complex gestures to enable 3D interaction, detecting the user's hand posture provides a more straightforward and thus natural 2D/3D user interface. Such an interface allows users to freely move their hand when interacting with a stationary or mobile device. Natural 3D interaction is especially of interest in virtual and augmented reality (VR, AR) environments. While in desktop VR/AR scenarios, the interaction environment can be tailored to meet certain requirements of the applied method - such as constant illumination and non-cluttered background - to ensure workstation robust hand recognition, interaction with a handheld device might be performed elsewhere. Thus, a mobile NUI must be able to cope with less constrained settings. Previous work demonstrates that depth data, processed on a stationary, can be combined with RGB data to enable more reliable hand segmentation in cluttered or poorly illuminated environments. However, to provide a truly mobile NUI, RGB and depth data must be processed and fused directly on the handheld device. Based on RGB+D data, a developer can choose from a large number of existing hand detection approaches to build a powerful mobile NUI. However, a study comparing and elaborating the properties of each technique to provide guidelines for mobile NUIs is missing yet. This paper presents the following three main contributions:

Figure 1: RGB+D NUI for mobile AR/VR

- A systematic study to evaluate existing RGB and RGB+D approaches for markerless hand posture recognition in a mobile interaction task.
- Guidelines to help developers choose the most suitable markerless posture recognition technique for a certain application scenario.
- A powerful software framework to process and fuse RGB and depth data on a mobile device. Thereby, we aim for a markerless NUI for handheld VR/AR to provide intuitive and robust hand posture recognition (Figure 1).

2. RELATED WORK
Over the last years, 2D multi-touch has become the de-facto standard on handheld devices for interaction. However, due to the implicit characteristics of 2D touch input, only 2D gestures can be provided. Various multi-touch approaches have been designed to

Laval Virtual VRIC '14, April 9-11, 2014, Laval, France.

manipulate 3D objects but are not usable for natural and intuitive 3D interaction due to the missing 3D input. In order to enable natural 3D interaction, the third dimension needs to be taken into account and integrated to provide 3D gestures. On mobile devices, the built-in camera is usually used for vision-based methods (marker-based or markerless) to determine the hand and its 3D position for interaction. Marker-based approaches use color markers or gloves in order to perform hand segmentation. However, they cannot serve for natural interaction since pre-conditioning of the hand with additional equipment is required. Thus, we focus on markerless-based methods for hand segmentation and 3D interaction.

A great number of approaches for markerless hand detection based on RGB data use skin color as the main feature for segmentation [2, 5]. For these methods no special training is necessary; however, color as a feature is sensitive to variations in lighting condition. It is also impossible to distinguish hands from other skin-colored objects. Baldauf et al. [2] use the RGB camera of a Smartphone and skin-specific values as threshold to segment skin-colored pixels. Morphological operations are applied to remove noisy pixels and fill holes in potential skin areas. After calculating the contours, the largest connected area is presumed to be the hand. Fingertips are determined by evaluating the distance between hand contour points and the inner and outer hand circle as well as using the hand's orientation. Other approaches use Haar-like features [6] for hand detection [3, 11]. Although classifiers based on Haar-like features are computationally expensive and require an offline training phase, they can differentiate hands from a face or other skin-colored objects. The work of Rodriguez et al. [11] uses a Haar-like feature classifier to detect a hand in a RGB webcam image. Then, two filters are applied to remove repetitions and false positives. The hand detection provides size and 2D position; the missing hand's depth value to allow for 3D interaction is estimated by evaluating the hand size (relative depth estimation).

Combining a RGB- and a depth-camera results in RGB+D data, which is used by a number of approaches [14, 15] to improve RGB-based hand segmentation and to provide absolute depth values for more precise 3D interaction. Van den Bergh and Van Gool [14] use a RGB- and a Time-of-Flight-camera (ToF) to generate RGB+D data. The low resolution of the depth camera is sufficient to detect foreground and remove background objects and to determine the hand's distance from the camera. The high resolution of the RGB camera allows for accurate and robust hand detection. Using a Haar-like feature-based classifier, the user's face is detected and the absolute distance of the face is determined using the ToF depth data. Based on this distance, a threshold is introduced to remove the background in the RGB data. Within the remaining RGB pixels, the hands are detected by applying skin color segmentation.

3. DETECTION OF 3D HAND POSTURES

Based on the related work, we chose five different markerless approaches that have been proven to be reliable, fast and/or robust. Therefore, we combined and extended existing techniques to be executed on a handheld device. The aim was to detect two different hand postures to provide natural 3D interaction with a mobile virtual scene. We did not apply color techniques for hand detection in RGB data due to its limitations to distinguish between hands and similar colored objects.

Our selected approaches use either RGB data for hand segmentation, posture detection and relative depth estimation or RGB+D data for improved hand segmentation and absolute depth calculation. Figure 2 shows the general setup of the prototype and interaction space. The user can interact using two different postures: *Posture 1* (five fingers outstretched) controls the virtual hand to select a virtual object. With *Posture 2* (pointer finger outstretched), the user can translate a selected object.

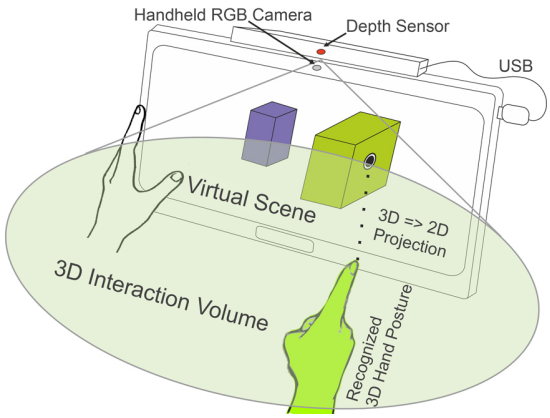

Figure 2: Mobile hardware setup and 3D hand interaction

3.1 Pure Cascaded Classifier

The *First Approach* (**A1**) is based on RGB data and two Haar-like feature-based cascaded classifiers [6], C1 and C2, to detect *Posture 1* and *Posture 2*. The classifiers deliver the hand's 2D position and its size, which is subsequently used for relative depth estimation to obtain the hand's 3D position for interaction.

3.2 Cascaded Classifier + Image Processing

The *Second Approach* (**A2**) uses a Haar-like feature-based cascaded classifier C3 to detect the palm only. This classifier is trained with the outstretched thumb and the bottom half of the hand's palm as features and does not depend on the other fingers for detection. Within the detected hand region, different image processing operations are then applied to determine the hand's contour, resulting in the following sub-types of A2:

A2-C: Canny Edge Detection [8]
A2-H: Threshold with automatic adaption, based on the hand's minimum/maximum values of hue, saturation and value (HSV), minus 10% (top and bottom) to remove outlier
A2-T: Fixed threshold with optional user adaption
A2-D: Using RGB+D data to remove fore- and background for robust hand segmentation

In A2-C, A2-H and A2-T, the convex hull and convexity defects are computed for fingertip detection. Convexity defects describe the deviations of the contour to the convex hull. The convexity defects are fingertip candidates and have to meet certain criteria to be classified as fingertips, i.e. a minimum distance to the hand's midpoint. Depending on the amount of detected fingertips, the hand's postures are determined as 4-5 fingertips define *Posture 1* and 1-2 fingertips define *Posture 2*. The hand's 3D position is calculated with relative depth estimation, as described in A1.

In A2-D, after detecting the hand's palm with the cascaded classifier, the absolute depth of the hand is determined using the depth map, resulting in the 3D position of the hand. In the next step, all pixels except those of the hand depth layer are removed from the RGB data for robust hand segmentation and determination of the fingertips, as described above. Thereby, we overcome the limitation of [15] that requires the hand to be the closest object to the camera for correct hand segmentation.

4. FRAMEWORK

In our developed framework, the handheld device and the depth sensor are rigidly coupled and calibrated to provide correctly registered RGB+D data. For intrinsic and extrinsic camera calibration, pictures of a 7×4 checkerboard pattern were simultaneously taken with the built-in mobile RGB camera and the externally connected depth imaging device. All data was then processed with the *MIP* tool [7] to estimate both internal and external camera parameters. Based on the calibration, depth data is mapped onto the RGB image, resulting in RGB+D data [4].

All computations – capturing and fusing of RGB and depth data, detection of hand and posture – are performed on an *Asus Eee Pad Transformer Prime (TF201)* tablet. It runs Android 4.1.1 and is equipped with a quad-core 1.4GHz processor, 1GB main memory, 10.1 inch display with 1280 x 800 pixel, USB 2.0 port and a 1.2-megapixel front-facing camera that provides the RGB data. A *Kinect for Windows* is used as depth sensor.

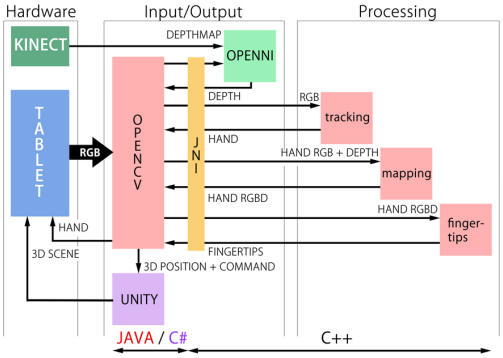

Figure 3: Software Processing Pipeline

The proposed setup uses RGB+D data with a resolution of 320x240 pixels. Via USB, the Kinect is directly connected to the tablet [9] and is interfaced with the hand posture detection application using *OpenNI* [10]. The drivers with OpenNI support, provided by *Avin2's SensorKinect* [12], had to be re-compiled with *Android NDK* [1] in order to use the Kinect with Android. Unfortunately, this driver does not support the *Near Mode* feature of the Kinect for Windows. Thus, the proposed setup requires a minimum distance of 48cm for reliable and robust hand depth data. In Figure 3, the software processing pipeline is depicted. We use *OpenCV* [8] for hand and fingertip detection in RGB data. To minimize tracking latency, Kinect integration as well as image and depth data processing is entirely implemented in C++, using *Android NDK* [1] and the *Java Native Interface (JNI)*. Rendering of the virtual scene as well as 3D interaction is handled by the game engine *Unity3D* [13], which is integrated into the Android application

5. EVALUATION

With our framework, we tested the proposed RGB classifier and RGB(+D) segmentation techniques for posture recognition by evaluating five different test sets with varying hand positions, background and illumination conditions.

5.1 Test Sets & Ground Truth

We established five test sets to simulate various realistic VR/AR 3D interaction scenarios. Each test set consists of equally weighted (50:50 ratio) positive (with hand posture) and negative (without posture) images.

Set1: *Posture 1*; constant light, white background (104 images)
Set2: *Posture 1*; varying light, white background (110 images)
Set3: *Posture 1*; constant light, cluttered background (148 images)
Set4: *Posture 2*; varying light, white background (138 images)
Set5: *Posture 1&2*; varying light, white background (220 images)

All images were captured with our application using the front-facing camera (RGB) and Kinect (depth map). Next, we annotated all images (center point of the hand) to formulate a well-defined and robust ground truth.

5.2 Results

The hand posture detection was evaluated with *f-score,* as the combination of precision and recall as their weighted average, and *accuracy*, as the proportion of true results. For performance evaluation, we measured the achieved application frame rate.

Table 1. Performance evaluation

Mean (σ)	A1	A2-C	A2-H	A2-T	A2-D
Framerate (fps)	13.86 (1.91)	16.6 (1.76)	16.62 (1.88)	16.79 (1.79)	8.13 (1.51)

The mean frame rate of each approach with standard deviation σ are listed in Table 1. Since A1 as well as the A2-D require complex computations, their processing is slower, resulting in significantly less frame rates than A2-C, A2-H, A2-T. Before evaluating f-score and accuracy for each (compound) approach, we first tested separately the performance of the three Haar-like feature classifiers C1, C2, C3. Therefore, we calculated the mean of f-score and accuracy for each classifier over all test sets. As listed in Table 2, the results indicate similar performances of classifier C1 to detect *Posture 1* in A1 and classifier C3 to detect the hand's palm in A2. The results of classifier C2 to detect *Posture 2* in A1 were found less accurate than C1 and C3.

Table 2. Hand detection - Classifier evaluation

Mean (σ)	C1	C2	C3
F-Score (%)	86.03 (12.12)	60.61 (5.79)	82.22 (9.89)
Accuracy (%)	90.13 (7.06)	72.65 (13.22)	81.26 (8.89)

Next, all test sets were independently processed with each hand posture detection approach. Then, the obtained results were compared with the ground truth to determine true and false detections. We defined detection as true positive if the distance between the center point of the detected hand and the center point of the ground truth is less than 15 pixels.

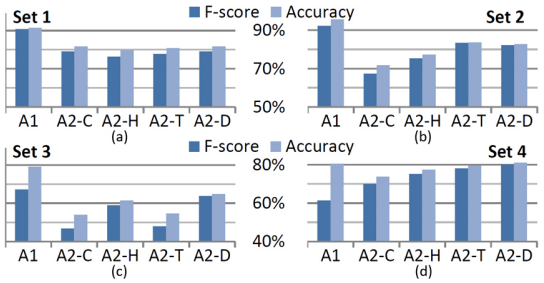

Figure 4. Evaluation of Set 1, Set 2, Set 3 and Set 4

As illustrated in Figure 4, the detection of *Posture 1* in front of white backgrounds shows good results for all approaches with constant illumination (Set1, Figure 4a) but poorer results for A2-C and A2-H for varying lighting conditions (Set2, Figure 4b).

When evaluating *Posture 1* in front of cluttered background (Set3), considerable differences between A2-C and A2-T could be found, as illustrated in Figure 4(c). Compared to Figure 4b, the results for *Posture 2* (Set4) exposed almost similar performance

of A2-C, A2-H, A2-T, A2-D and a large deviation for A1, as depicted in Figure 4(d). After separately evaluating the detection of *Posture 1* and *2*, we tested their detection in the combined Set5. As depicted in Figure 5(a), A1, A2-T and A2-D perform considerably better than A2-H and especially A2-C.

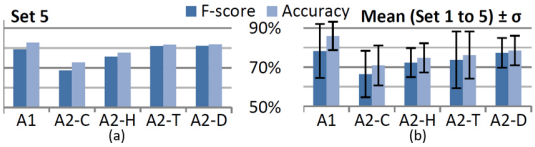

Figure 5. Evaluation of Set 5 and Mean of Set 1 to 5

Finally, we calculated the mean f-score and mean accuracy over all test sets for each approach. As illustrated in Figure 5(b), the best results are achieved with A1 and A2-D, while A2-C performed worst.

5.3 Discussion

The results of the performance evaluation indicate that all approaches achieve interactive frame rates to provide intuitive 3D interaction. However, A1 and A2-D perform slower caused by the use of two computationally expensive classifiers in A1 and the gathering of depth data and mapping in A2-D. The Haar-like feature classifiers C1 and C3 delivered good result even under varying lighting conditions and with cluttered backgrounds. Results for classifier C2 have found lower f-score and accuracy. That might be caused by the shape of an outstretched single finger, which is small and therefore provides fewer features for detection and differentiation from the background. This problem can be mitigated with more intensive training of the classifier.

Analyzing the results for posture recognition, A1 was found to work well with cluttered background and showed overall good results for *Posture 1*. The differences for *Posture 2* reflect the results of the classifier evaluation of C2. A2-C performed poorly with cluttered backgrounds and was overall ranked last in this evaluation. Since the Canny Edge detection cannot differentiate between hand contour and other edges, this approach is not recommended in a minor to non-constrained handheld NUI environment. A2-H achieved good overall results and outperformed A2-C as well as A2-T, particularly with cluttered backgrounds. To be able to react to varying lighting conditions, user can manually adapt the threshold in A2-T. For those situations, it shows good results but performs worse in situations with cluttered background. The depth data in A2-D is used to remove the background; this results in good performance with cluttered backgrounds, varying lighting conditions and the best overall performance amongst the A2 approaches. The standard deviations in Table 2 and Figure 5c are mostly influenced by cluttered backgrounds and C2, as discussed above.

Based on the given results and the test sets, we derived guidelines to support users to choose the most suitable hand recognition technique for a certain handheld VR/AR 3D interaction scenario:

- **A1** shows very good results if the underlying classifier is well trained, but provides slower frame rate and requires one classifier for each posture.
- **A2-H** is overall the best choice if only RGB data is available and fast tracking is required.
- **A2-T** provides high frame rates, can be quickly implemented and is suitable if 3D interaction is performed in front of a white background. For different illumination situations, manual user adaption is required to react to light changes.
- **A2-D** is the most robust approach for all 3D interaction situations, but provides smaller update rates.

6. CONCLUSION & FUTURE WORK

We have presented a software framework to provide natural 3D interaction for handheld VR/AR environments using RGB and RGB+D data. Based on a systematic study, we could provide useful guidelines for users to choose amongst the large variety of approaches. Future work will focus on dynamic gesture recognition as well as integration of smaller depth sensing devices to provide a handheld NUI with improved usability.

7. REFERENCES

[1] Android NDK, [Software] Revision 8 http://developer.android.com/tools/sdk/ndk/index.html. Accessed: 2013-11-30.

[2] Baldauf, M., Zambanini, S., Fröhlich, P. and Reichl, P. 2011. Markerless visual fingertip detection for natural mobile device interaction. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services* (2011), 539–544.

[3] Bilal, S., Akmeliawati, R., Salami, M.J. El, Shafie, A. a. and Bouhabba, E.M. 2010. A hybrid method using haar-like and skin-color algorithm for hand posture detection, recognition and tracking. In *Proceedings of International Conference on Mechatronics and Automation (ICMA)* (2010), 934–939.

[4] Kinect Calibration, [Online] http://nicolas.burrus.name/index.php/Research/KinectCalibration. Accessed: 2013-08-21.

[5] Lee, T. and Höllerer, T. 2007. Handy AR: Markerless inspection of augmented reality objects using fingertip tracking. In *Proceedings of the 11th IEEE International Symposium on Wearable Computers* (2007), 83–90.

[6] Lienhart, R. and Maydt, J. 2002. An extended set of haar-like features for rapid object detection. In *Proceedings of the International Conference on Image Processing (ICIP)* (2002), I–900 – I–903.

[7] MIP - MultiCameraCalibration, [Software] Version 1.0.0 http://www.mip.informatik.uni-kiel.de/tiki-index.php?page=Calibration. Accessed: 2013-08-21.

[8] OpenCV library, [Software] Version 2.4.6 http://opencv.org/. Accessed: 2013-11-25.

[9] OpenNI and Kinect for Android Tutorial, [Online] http://pointclouds.org/blog/nvcs/raymondlo84/index.php. Accessed: 2013-08-21.

[10] OpenNI, [Software] Version 1.5.4.0 unstable https://github.com/OpenNI/OpenNI/tree/unstable. Accessed: 2013-11-25.

[11] Rodriguez, S., Picon, A. and Villodas, A. 2010. Robust vision-based hand tracking using single camera for ubiquitous 3D gesture interaction. In *Proceedings of the 2010 IEEE Symposium on 3D User Interfaces* (2010), 135–136.

[12] SensorKinect, [Software] Version 0.93 https://github.com/avin2/SensorKinect. Accessed: 2013-08-21.

[13] Unity3D, [Software] Version 3.5.1 http://unity3d.com/. Accessed: 2013-11-25.

[14] Van den Bergh, M. and Van Gool, L. 2011. Combining RGB and ToF cameras for real-time 3D hand gesture interaction. *2011 IEEE Workshop on Application of Computer Vision (WACV)*. (2011), 66–72.

[15] Yeo, H., Lee, B. and Lim, H. 2013. Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware. *Multimedia Tools and Applications*. (2013), 1–29.