Reliable and Unobtrusive Inter-Device Collaboration by Continuous Interaction

Rudolf Kajan, István Szentandrási, Adam Herout, Alena Pavelková
Faculty of Information Technology
Brno University of Technology
Božetěchova 1/2
612 66, Brno, Czech Republic
{ikajanr, iszent, herout, ipavelkova}@fit.vutbr.cz

ABSTRACT

Currently, there is a lack of support for seamless task migration among devices - starting a task on one device and continuing it on another, without the need of manual application state inspection and data transfer. We are solving this problem by employing our framework for application state acquisition coupled with user interface based on an intuitive metaphor: video recording. Our solution utilizes a combination of natural features based detection and marker tracking in order to reliably establish the homography between the screen and the observation of the mobile device's camera. This allows us to employ a fast and precise *continuous* interaction even on low-end mobile devices. In every moment, user is given relevant task and content-migration options for selected application. The experimental results show that our solution provides reliable task migration at interactive frame rates.

Keywords
Task Migration; Augmented Reality; Multi-device environment; Mobile interaction; Situated public displays

1 INTRODUCTION

An important role of intelligent user interfaces is the support of intuitive information sharing and task migration among users' devices, whether they are desktops, mobile or ultra-mobile devices. On a daily basis users want to continue with a previously started task on another device.

Whether users want to take their planned trip from desktop to a mobile device, continue a game, share commented photos on a public display or seamlessly continue watching a video stream on another device, task migration is very desired, yet at the same time also very challenging. For web applications this is true mainly because their state is no longer represented by the URL, as asynchronous operations are continuously changing their internal state in the background. Desktop applications are primarily document-centric and content migration involves manual file localizations, transfer to another device and file processing with a selected mobile application. If the user decides to simply photograph an interesting on-screen content, the resulting picture quality is very low when compared to the original on-screen quality and no additional content-related metadata are transferred.

In this paper, we present a way to make this task migration or content sharing process instant and intuitive. In particular, the method explored here is completely mobile-centric, i.e. no user interaction needs to be done on the side of the content provider screen. That makes our approach especially feasible for acquiring information from public displays.

We make the following contributions in our paper: a) We present a novel interaction technique for *continuous* interaction across mobile and desktop platforms; b) We present a framework for real-time application state acquisition and reconstruction on target platform; c) We report on a user study focused on the usability of our prototype and a system performance evaluation.

2 RELATED WORK

Chang and Li proposed DeepShot [6] – a framework for capturing work state which uses natural visual features (SURF, [4]) and tracks them. Their approach is a simplified sibling of PTAM (Parallel Tracking and Mapping, [11]).

Feature based techniques in general try to find a correspondence between extracted 2D feature points (SIFT [13], SURF [4], FAST [16], etc.), and their positions in real world. To achieve this, image feature point descriptors (SIFT [13], SURF [4], BRIEF [5], BRISK [12],

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

etc.) are matched with feature points in video frame capturing the real world. Camera pose can then be established by projecting the 3D points into the image and minimizing the distance to their corresponding 2D positions. For discarding outliers, algorithms like RANSAC or TUKEY-Estimator are used [19].

A notable natural feature point based method establishing camera pose in an unknown scene is the above-mentioned PTAM [11]. This method uses parallel tracking and mapping to create a map of the scene. The tracking and mapping are separate task, running simultaneously. The tracking part tracks the motion of a hand-held camera while the mapping part creates a map of the 3D space.

If the tracked scene is composed of a single plane, camera pose relative to the plane can be computed from several frames using image rectification algorithm as described by Pirchheim et al. [15]. This system computes 6DOF camera poses relative to the plane. When the relation between the camera pose and plane is known, other camera poses can be computed using inter-frame homographies.

However, there is one major drawback of feature based techniques: despite various techniques to balance the features' density in the camera view, it is impossible to ensure the presence of enough visual features in the whole camera view. In the case of observing a computer screen, the problem is even more difficult because unlike the real world, the monitor screen tends to contain surfaces of exactly constant color, backlit by the monitor lamp and thus avoiding any external lighting which would help distinguish unique locations.

Instead of natural features, artificial markers can be used to establish camera pose relative to the marker. Widely used markers, such as ARToolkit [10] markers or ARTag [7][1], are both suitable. Some more recent markers, like Uniform Marker Fields [17] or ReacTIVision [9], are good options as well. Uniform Marker Fields has already been used in previous research by Kajan et al. for content migration [8]. Our proposed system features a more robust and faster way to migrate content thanks to the combination of natural feature points based method and Vuforia [2] marker and is much less obtrusive thanks to the marker's visual integration into our system.

A step towards direct information transfer from a desktop screen to an ultramobile device are the VR Codes by Woo et al. [18]. In this solution, a digital payload is encoded into solid gray surfaces on the screen by a time multiplex. The encoding requires a significant computational effort on the desktop monitor side, and assumes a particular design of the camera (rolling shutter) on the mobile side. This method could be more promising for the desired purpose of task migration if it allowed encoding the information into arbitrary images and into dynamic content.

In order to allow mobile use, the requester device continuously tracks itself with respect to interactive displays in its surrounding using its built-in camera and computes its spatial relationship between itself and each identified display.

Our inspiration and the leading metaphor was video recording on mobile platforms and augmented reality applications in general. In these scenarios users usually just point their device's camera at the object of interest and immediately begin to record (capture) it or interact with it. Our interaction technique provides the user *at every moment* with relevant task and content-migration options for the selected application and its content. Our approach thus emphasizes spontaneous and unplanned content access with minimal user input, while being very responsive.

3 PROPOSED SYSTEM ARCHITECTURE

With the *continuous interaction* in mind, we have designed a highly responsive system which allows for intuitive task migration without the need of manual application state inspection or copying of "raw" pixels without any additional semantic information. The task migration process from the system architecture's point of view is a two-way communication between a *content provider* and a *content requester* device (See Fig. 1).

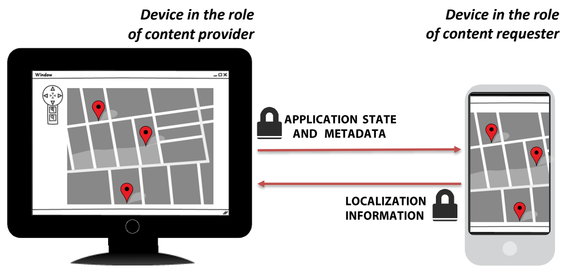
Device in the role of content provider
Device in the role of content requester
APPLICATION STATE AND METADATA
LOCALIZATION INFORMATION

Figure 1: The task migration process between a content requester and a content provider device.

The *content provider* device is the device with the application state that needs to be transferred to the other device or platform based on the user's current context. A device in this role is able to share the state of its applications with authenticated clients – *content requesters*. The *content provider* device provides the state of its applications by either querying individual applications for their current work state (URL and internal settings for web applications, the document along with current page number for document viewers, streamed multimedia content, and other metadata) or provides general services, e.g. providing high quality screenshots of a selected screen area or text from a selected area via optical character recognition.

Content requesters are responsible for communication initiation with the target provider device, for selection of the screen region or application of interest and selection of requested/offered content based on the user's intent. In a typical scenario, *content requesters* are mobile or ultra-mobile devices (smart phone, tablet, PDA).

3.1 System Components and Their Functionality

The communication between requester and provider devices during the task migration process is shown in Figure 2. Prior to the task migration, a network connection between the content requester and the provider must be established. The *network session manager* module, which is present on both devices, is responsible for network connection initiation to a remote *content provider* (e.g. public display, laptop). At the moment, the communication is implemented through a WiFi connection, due to its availability on a broad range of devices.

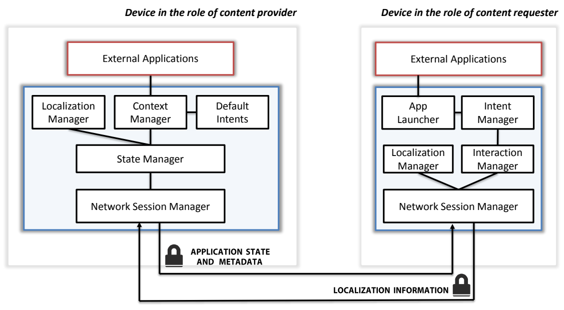

Figure 2: System overview. Content requester communicates wirelessly with a content provider. The system on both sides consists of a stack of functional blocks (blue rectangle) whose purpose is to ensure information sharing between the built-in or third-party applications (red rectangles).

The target device is located either via network discovery or by scanning a specific code associated with target device (e.g. on-screen or printed visual marker / matrix code). Sudden changes in network connectivity are addressed by usage of MobileIP[1] in the existing infrastructure. MobileIP is able to seamlessly handle connection hand-overs during migration in a way that is transparent to the *content provider*. The communication channel is secured by IPSec - suite of protocols which provide data source authentication, data integrity, confidentiality and protection against attacks[2].

The *localization manager* on the *requester* device is used for fast and accurate client-side within-screen localization. The localization is based either on marker tracking or natural image keypoint detection, features extraction and matching. Natural features based detection is employed during the initial position estimation and when the devices cannot successfully track the marker. This approach minimizes the amount of transferred data between the devices, because only the detected 2D position coordinates are sent back to the *content provider*.

The transferred and processed content is much smaller compared to the purely feature-based solutions where either the feature vectors or the whole camera stream are sent to the target device or to an intermediate server for processing and camera localization. This allows for fast and reliable real-time interaction with immediate feedback even on low-end devices.

Based on the obtained camera-localization information, the *provider's context manager* queries individual applications and gathers their internal state. In order to obtain the full application state from web applications, we have implemented an extension for Google Chrome browser and plugins for applications from Microsoft Office suite.

If the selected application is unable to provide its state and metadata, only general intents are available. General intents include high-quality screenshots, and text and phone numbers recognition for the selected part of the screen (the OCR functionality is implemented via Microsoft MODI library[3]).

The acquired application state is sent to the *intent manager* on the *requester device* which translates these JSON-encoded messages to intents directly usable on the requester platform.

Afterwards, the *state manager* provides the user with a visual feedback and updates the GUI, based on the available actions for the selected content. The options include resuming work on a *requester device* – continuing work with a reconstructed web application state on a current device, editing text in an available text editor, manipulation and viewing of images, audio/video playback, etc.

3.2 Marker Tracking and Natural Keypoints based Detection

Our solution utilizes a combination of natural features based detection and marker tracking in order to reliably establish the homography between the screen and the observation of the mobile device's camera. This allows us to employ a fast and precise *continuous* interaction even on low-end mobile devices.

During the initialization phase and in case of fast camera movement, we employ natural features based detection. Detecting keypoints and extracting features on the

[1] http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=656077
[2] http://tools.ietf.org/html/rfc3776
[3] http://support.microsoft.com/kb/982760

mobile phone would be too costly on some low-end devices. Instead, the features are computed and matched on the content provider. Similar approach was taken in [6] and [3]. The difference is, that our solution does not stream the video, as it would generate high network traffic (see experiments). Instead, we use natural features detection as a fallback method, and send frames only in large intervals.

A major disadvantage of pure natural features based methods is that they rely on rich features being present on the target display. This assumption is rarely met in the highly manhattanic world of desktop and web applications. As a solution, we utilize a *virtual cursor* using the Vuforia library on the content requester side combined with a small natural image target on the content provider. The natural image target is used to compute the required offset on the content provider caused by the camera movement. The computed relative correction is sent to the content provider. This is our primary method of camera movement tracking.

The image target also serves a secondary objective as a reference position to draw the augmented UI elements of the application. These elements give visual feedback to the users, so they move within acceptable distance from the content provider. The augmented layer also hides the obtrusive marker on the client side.

If multiple users are simultaneously interacting with a single provider, their primary mean of localization is natural features based detection of a target. In the case that multiple targets overlap, clients automatically fall back to natural keypoints tracking on the target display. This approach ensures that the interaction will not be interrupted even if multiple users are migrating the same elements at the same time.

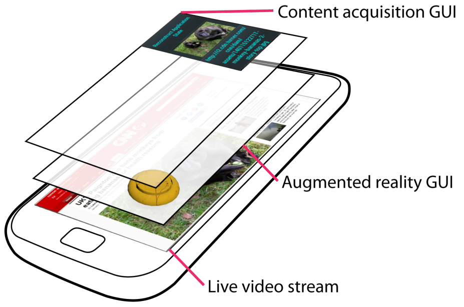

Figure 3: The user interface of the requester device consists of three layers - live video stream from the device's camera, 3D GUI based on augmented reality and content acquisition GUI dynamically updated with respect to the current content selection.

4 IMPLEMENTED SOLUTION – CHROME AND ANDROID

As a proof of concept and as the prototype for user testing and exact experimental evaluation, we created a pilot version of the whole system.

4.1 Content Provider Chrome Plugin

In order to be able to access and examine the full application state of an online application, the *content provider* needs to access the loaded content in a web browser. We have developed an extension for the Chrome browser. This extension acts as a communication bridge between our application on the *content provider* device and web applications running inside the browser.

When the user selects contents of a web page (blocks of text, images, videos or links to other web-based resources) or a complete online application state (e.g., a trip planned in Google Maps) for a migration, the extension finds the active web application and through the code injection inserts a script into it[4]. This script is able to directly manipulate with the page's Document Object Model (DOM) and extract the required parts of the online application and send them back to the *content provider*. In the case of full-state migration request, the plugin extracts all files with internal variables (e.g., referenced javascripts) and sends them to the *content provider*. This script executes in the context of a page that's been loaded into the browser, making it a part of a web application, not a part of the extension. The script is also able to inject information into web application, thus allowing for continuation of task. We have chosen this approach as all widely-used browsers support code injection into loaded pages. After the extraction, the *content provider* forwards content data to the *requester device*, where the application state is reconstructed, thus allowing the user to continue with a task on the other device.

4.2 Content Requester Android Application

For the implementation of a mobile *content requester* prototype we chose the Android platform because of its availability on a broad range of mobile and ultramobile devices. For the initial design of our application, the main goal was a clean and minimalistic design of control elements, which will support the video recording metaphor.

After starting the application, the user is welcomed with the option to choose between connecting to a previously used *provider* device, using network discovery to discover available *provider* devices or by scanning a specific code associated with target device.

After connection, the detection algorithm computes the position of the *requester* device with respect to the current *provider* and optionally position of a virtual cursor

[4] http://developer.chrome.com/extensions/content_scripts.html

which helps the user to identify the exact spot the user is pointing at.

The main user interface can be seen in Figure 3. The interface consists of three layers. Live video stream from device's camera is on the lowest layer. On top of the video stream is an augmented reality 3D GUI. The main advantage of the augmented reality interface element is that it naturally guides the user to the appropriate distance and angle with respect to the content provider by the means of size and rotation change. This interface is used to mark the content for migration – similarly to a video camera's *record* button. It is used by user's non-dominant hand. The topmost layer consists of 2D elemets which are dynamically changed based on the content which the user is currently selecting for migration and display preview of recorded items – blocks of text, images, hyperlinks. If any of this content is touched by the user's (dominant) hand, options for further processing are displayed. These options are dependent on the selected content and include editing, sharing on social networks, saving to the device or launching an appropriate Intent.

During the interaction between the *content provider* and the *requester* device, the application stores the history of the accessed content and allows the user to access it later.

5 EXPERIMENTS

We tested both the reliability of our feature detection-based algorithm and the tracking performance of the Vuforia library as a part of the user tests. We created a setup consisting of one device in the role of content provider (17 inch laptop computer with 1920 × 1080 resolution display and an Intel®Core^{TM}i7 processor running at 2.2 GHz) and one device in the role of content requester (Samsung Galaxy S2 smartphone).

The study we conducted involved 25 participants. In the beginning, the participants were asked to fill in a questionnaire. This questionnaire asked them about their technical expertise, their knowledge regarding mobile phones, as well as some demographic statements. The average age of attendees was 28 years, the youngest participant was 21 and the oldest was 42. Ten participants were from non-technical professions.

5.1 Tracking Reliability

In order to evaluate the reliability of the feature detection based algorithm used during initiation, the requester device was attached to a base perpendicular to the floor, in a fixed height, focused at a chosen part of the screen (see Fig. 4). The experiment was conducted in a room with artificial (fluorescent) lighting. The devices were connected through a WiFi connection.

The requester device was held at a distance of 10, 20 and 30 cm and a pitch angle of 75°, 90°, 105°, and

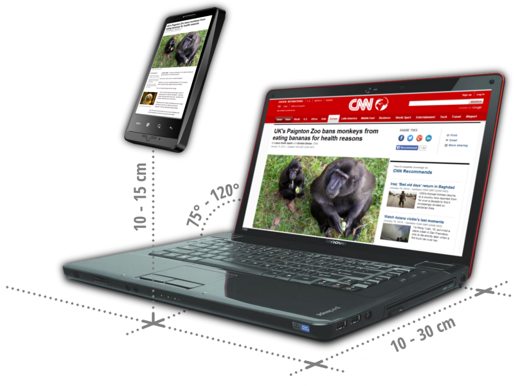

Figure 4: Parameters of reliability setup for initialization – distance between mobile device and laptop, range of screen angles used during tests and height range of a mobile device in order to be focused on the same point on the screen.

120°. On the content provider device a fullscreen web application containing text, several smaller images, and a map was displayed during the experiment. The resolution of the images sent to the content provider to compute the initial homography was 320 × 240.

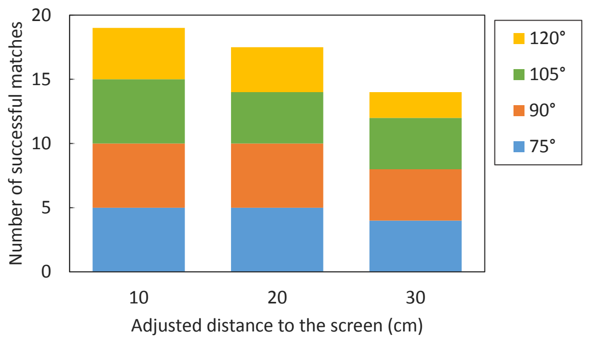

Figure 5: The results of natural feature detection reliability for different pitch angles. For each distance settings 20 images were taken - 5 per each angle.

Figure 5 contains the evaluation of the reliability of our natural feature based detection for different pitch angles for the content provider device. For each distance settings 20 images were taken – 5 per each angle. The results show that the natural features based detection was highly reliable. For angles 120° and 75° the colors shown on the content provider were highly shifted changing the visible key-point features causing slightly worse results. This issue is caused mostly by the display used in testing and would harm any computer vision based technique.

During user testing, the participants were given several tasks to migrate data. During the tasks we recorded the tracking status of our system. The detection algorithm ran at 20 frames per second. Figure 6 shows the probability of successfully localizing the content requester relative to the content provider. The blue line shows the probability of successfully tracking for a given amount of time given that we successfully tracked the previous frame. After the tracking was lost, a full frame was transferred to the content provider in order to be used for natural features based detection (hence the step around the $1s$ mark). The delay interval of 1 second for full frame sending was chosen not to overload the network connection. The results show that after $4s$ the cursor tracking algorithm was able to restore tracking with 99 % probability. This causes a short but noticeable delay for the user after the tracking is lost and needs to be restored. Despite this delay, the overall performance of the natural feature detection is good enough to provide the users with *continuous interaction*, and is an area which we are planning to improve.

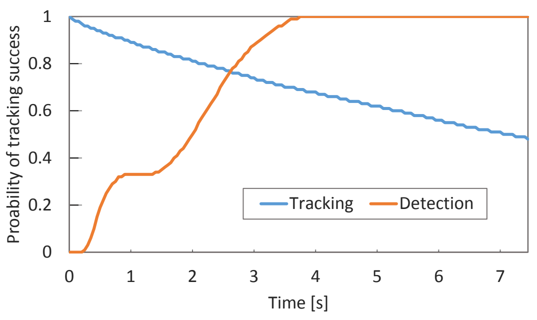

Figure 6: Breakdown of the probability distribution for the tracking phase (blue) – the probability of tracking continuously for a given time; and the detection phase (brown) – the probability distribution in time of successfully restoring cursor tracking.

5.2 Speed Performance

In order to be able to compare our solution with alternative frameworks (e.g. Touch Projector or DeepShot), we tested four target applications: maps from Google Maps, photos from Picasa, long articles with images from CNN.com and short textual information from Twitter. For each application, 10 information requests were sent and processed. The setup for this experiment was identical to the previous test. All tests were done using the hardware from the setup for the reliability testing (See section Reliability above).

Table 1 summarizes the breakdown of the time consumed by the initiation phase of the interaction for a single frame. The majority of the time (59.2 %) was consumed by network transfer of the reference image. This gives 1.3 FPS for the natural features based position estimation part. In the setup experiment we sent reference images in $1s$ intervals to avoid flooding the network.

Once the tracker was initialized, it was able to track the cursor with full 20 FPS speed provided by the camera on the tested smartphone. After the user decided to migrate content from the content provider, the required time to transfer information was $19ms$ on average including network communication (approximately 73 %).

Activity	Time spent
Client side processing	11 ms
Network transfers (WiFi)	442 ms
Provider-side processing	289 ms
State acquisition via plugin	4 ms
Sum	746 ms

Table 1: Timing breakdown of the initialization phase. Client side processing covers camera image retrieval and resizing operation. Provider-side processing includes image reconstruction, acquiring screenshot and homography calculations.

The results show a significant speed increase when compared to task migration solutions based on visual features – the authors of the DeepShot [6] task migration framework report 7.7 seconds (SD 0.3 seconds) for processing the request, and allows for real-time information feedback for a selected screen area. A big advantage of our system is the utilization of video stream, which enables continuous interaction instead of discreet selection.

5.3 Bandwidth usage

Table 2 summarizes the required bandwidth of our system measured during the user test described in section Reliability. The last row contains the theoretical minimum required bandwidth if we used natural features only. In this scenario we assume that the content requester detects and extracts at least 20 binary feature vectors of 512 bits (common for state-of-the-art binary feature descriptors). These feature vectors are then sent to the content provider for homography computations. We also assume a speed of at least 15 FPS for continuous detection.

Mean	Peak	Average	Natural features
546.5 B/s	82.4 kB/s	7.8 kB/s	19.2 kB/s

Table 2: Bandwidth usage of our system used for interaction between content provider and content requester.

The results show that our system requires on average 2.5× less bandwidth than the theoretical minimum bandwidth used up by a pure natural features-based approach. However, 88.4 % of the time during interactions (cursor tracking) our system requires just $0.5\,kB/s$ bandwidth, which is approximately 35× less than a natural features based approach. Our system needs more bandwidth only in the initialization phase and in the case when the cursor tracking is lost during the interaction. In the future, this part could be replaced by computing features on the content requester side.

5.4 Content Selection Accuracy

In order to measure accuracy of content selection with our system, we have used targeting tasks based on ISO 9241-9 standard [14]. However, we have used a rectangular target instead of a distinct target point. We asked participants to try to navigate pointer into the rectangular area, while being as fast as possible.

The task started after the connection between requester and provider devices was established and the tracking subsystem was fully initialized. Afterwards users were notified about trial's start and moved the virtual cursor inside the area filled with text or images. The task ended once the cursor was inside the area and user touched the content acquisition button with non-dominant hand. We measured time and virtual cursor's coordinates throughout trials.

The trials were performed for three different sizes of the target area, corresponding to the sizes of standardized web elements. During the testing trials, timestamp and virtual cursor position was recorded for every received position information. From these recorded data, Throughput was computed. Throughput, in bits per second, is a composite measure derived from both the speed and accuracy in responses [14]. The results are shown in Figure 7.

Another information obtained from these data is average target re-entry. This information estimates how many times has the virtual cursor left and re-entered the target area after it entered it the first time. As can be seen in Figure 7 right, target re-entry strongly depends on the size of the target area. The reason for majority of target re-entries in small target areas was the natural hand tremble.

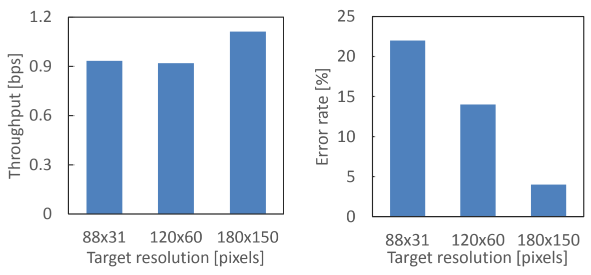

Figure 7: Left: Average throughput acquired during experiments where the user was moving the virtual cursor from starting point to the target area. In the graph is shown connection between the throughput and the sizes of the target area, which were chosen according to the sizes of standard web elements. Right: Average error rate depending on the size of the target area.

When compared to commonly used pointing devices, our system had a lower Throughput (TP), but also lower Error Rate (ER) for primary migration targets - images, text paragraphs, links (in [14] the reported values were: joystick TP 1.8 bps ER 9%, touchpad TP 2.9 bps ER 7%, trackball TP 3.0 bps ER 8.6%, mouse TP 4.9 bps ER 9.4%).

These results show that our system is comparable to commonly used pointing devices and usable even by inexperienced users. In the near future, we will improve both Throughput and Error Rate. We will compensate for natural hand tremble (which is the main source of lower TP and higher ER) by employing a smooth estimate of cursor's position and add the option to (semi-)automatically zoom for a better selection of content from remote providers.

6 CONCLUSION

We presented a solution for seamless task migration among a broad range of devices. Our approach emphasizes spontaneous and unplanned content access with minimal user input, while being very responsive. This interaction is based on an intuitive metaphor of video recording.

Our interface allows for *continuous interaction* - mobile device's display is updated in real-time and receives continuous feedback based on the content user is currently looking at. In every moment user is given relevant task and content-migration options for selected application and its content.

In order to reliably establish the homography between the screen and the observation of the mobile device's camera our system utilizes a combination of natural features based camera pose detection and virtual cursor tracking. This allows us to employ a fast and precise interaction even on low-end mobile devices, support migration of static and dynamic screen content and allow for simultaneous interaction of multiple users.

We created a prototype implementation of the whole solution which allows for task and content migration from web applications to a mobile client.

This prototype was examined within a user study and by a set of performance evaluation experiments. The results indicate that it substantially outperforms the existing solutions: the localization and task migration is done in real time on a mid-level cellphone; the localization is reliable even for different observation angles and for cluttered screen content. Our solution operates on a video stream with all the benefits: if one camera frame fails for a reason, the mobile client program determines the location from a subsequent valid one.

In near future, we are going to focus on increasing interaction distance, thus allowing for continuous interaction with remote and unreachable displays. This will be made possible by employing automatic and semi-automatic zoom functionality for selected content. We will also explore the possibilities of making the tracked cursor as unobtrusive as possible and minimize the required time for the fiducial to be present on the screen.

7 ACKNOWLEDGMENTS

This research was supported by the research CEZMSMT project IT4I - CZ 1.05/1.1.00/02.0070, by project V3C, TE01020415, and by project D-NOTAM TA02030835.

8 REFERENCES

[1] ALVAR tracking subroutines library web page, 2012. http://www.vtt.fi/multimedia/alvar.html.

[2] Vuforia - enable your apps to see, 2014. http://www.vuforia.com.

[3] D. Baur, S. Boring, and S. Feiner. Virtual projection: exploring optical projection as a metaphor for multi-device interaction. In *Annual Conference on Human Factors in Computing Systems*, pages 5–10, 2012.

[4] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[5] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: binary robust independent elementary features. In *Proceedings of the 11th European conference on Computer vision: Part IV*, ECCV'10, pages 778–792, Berlin, Heidelberg, 2010. Springer-Verlag.

[6] T.-H. Chang and Y. Li. Deep shot: a framework for migrating tasks across devices using mobile phone cameras. In *Proc. SIGCHI*, pages 2163–2172, 2011.

[7] M. Fiala. ARTag, a fiducial marker system using digital techniques. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, CVPR '05, pages 590–596, Washington, DC, USA, 2005. IEEE Computer Society.

[8] R. Kajan, I. Szentandrasi, A. Herout, and M. Zacharias. On-screen marker fields for reliable screen-to-screen task migration. In *SouthCHI*, pages 692–710, 2013.

[9] M. Kaltenbrunner and R. Bencina. reactivision: a computer-vision framework for table-based tangible interaction. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, TEI '07, pages 69–74, New York, NY, USA, 2007. ACM.

[10] H. Kato and M. Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *IWAR'99*, pages 85–94, 1999.

[11] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '07, pages 1–10, Washington, DC, USA, 2007. IEEE Computer Society.

[12] S. Leutenegger, M. Chli, and R. Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2548–2555, Washington, DC, USA, 2011. IEEE Computer Society.

[13] D. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150 – 1157 vol.2, 1999.

[14] I. S. MacKenzie, T. Kauppinen, and M. Silfverberg. Accuracy measures for evaluating computer pointing devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, pages 9–16, New York, NY, USA, 2001. ACM.

[15] C. Pirchheim and G. Reitmayr. Homography-based planar mapping and tracking for mobile phones. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '11, pages 27–36, Washington, DC, USA, 2011. IEEE Computer Society.

[16] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2*, ICCV '05, pages 1508–1515, Washington, DC, USA, 2005. IEEE Computer Society.

[17] I. Szentandrasi, M. Zacharias, J. Havel, A. Herout, M. Dubska, and R. Kajan. Uniform marker fields: Camera localization by orientable de bruijn tori. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 319–320, Nov 2012.

[18] G. Woo, A. Lippman, and R. Raskar. Vrcodes: Unobtrusive and active visual codes for interaction by exploiting rolling shutter. In *Proceedings of the 2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, ISMAR '12, pages 59–64, Washington, DC, USA, 2012. IEEE Computer Society.

[19] H. Wuest, F. Vial, and D. Stricker. Adaptive line tracking with multiple hypotheses for augmented reality. In *Proceedings of the 4th IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '05, pages 62–69, Washington,