Transform Flow: A Mobile Augmented Reality Visualisation and Evaluation Toolkit

Samuel Williams
Computer Science
University of Canterbury
Christchurch, New Zealand

Richard Green
Computer Science
University of Canterbury
Christchurch, New Zealand

Mark Billinghurst
The HIT Lab NZ
University of Canterbury
Christchurch, New Zealand

***Abstract*—It is common in mobile augmented reality (AR) research to examine results that were attained with unpublished tools and data sets, which makes it difficult to compare and improve existing work without significant effort. We discuss the development of an open source toolkit called Transform Flow, which includes a data capture application for iOS, a desktop application to replay and analyse captured data sets with different algorithms, and a mobile application that can run these algorithms in real-time. Our results suggest that our toolkit can be a centre for collaborative research, as it provides a common platform on which tracking algorithms for mobile AR can be developed, studied and eventually deployed.**

I. Introduction

Evaluating and comparing algorithms is an important part of quality research. The ability to compare different approaches with the same data sets is critical to the development of improved methods. This is also true for mobile augmented reality (AR), where many tracking algorithms are developed specifically within the constraints of a particular hardware configuration and testing methodology.

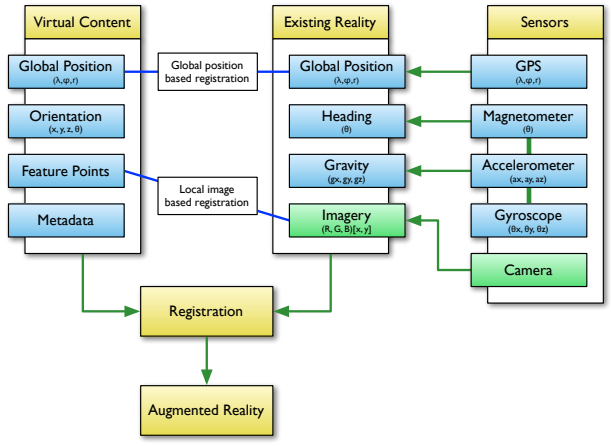

Fig. 1. An overview of typical data flow in mobile outdoor AR.

Ideally, sensor and visual data, as outlined in Figure 1, is captured from a variety of mobile devices in such a way that it can be repeatedly replayed. Algorithms can be critically analysed, frame by frame, to isolate edge cases and bugs in a controlled environment; test cases can be developed and run automatically to check for regressions. In addition, with the right design, both offline and online processing of input data should be possible, so that algorithms can be seamlessly deployed on real hardware for user evaluation and application development.

An open source platform that facilitates the analysis of new and existing algorithms would allow researchers to develop and test new ideas easily. Existing code can be reused, which typically reduces the challenges and risks associated with software development. Such a platform could also serve as a useful educational tool for students wanting to learn about different tracking algorithms and how they are implemented.

We describe the ongoing development of Transform Flow, which has recently been released under the MIT license. It was originally developed as part of a single research project, but is being released to the community to encourage collaborative research, development and education in the area of mobile AR.

II. Background

Testing and evaluating tracking algorithms designed for mobile devices is currently unsystematic[1], [2], [3]. This is a big problem for new research which seeks to improve on existing approaches, because it makes it hard to compare results objectively.

Even as recently as 2009, researchers were developing custom devices for outdoor AR[4] research. Tracking algorithms are often structured around poorly specified platforms and obscure hardware[5], [6], [4]; missing details or obsolete hardware make identical reconstruction, and thus comparisons based on published results, impossible.

Data sets and testing tools are often not publicly published[7] which makes it difficult to check whether a new approach is a significant improvement over existing methods. In particular, algorithms that fail on specific edge cases[3] warrant further analysis and study; but without the specific data sets and systematic evaluation tools this is not possible.

Similarly, public data sets commonly used for computer vision evaluation don't include inertial sensor

978-1-4799-0883-7/13/$31.00 ©2013 IEEE

Available online: https://github.com/HITLabNZ/transform-flow

measurements[8], [9], [10], which makes them inappropriate for modern mobile AR research[11]. Inertial gravity measurements have been synthesised from ground truth camera poses, which allows some types of hybrid tracking algorithms to be benchmarked on existing data sets[12], however other sensors including gyroscope, magnetometer and compass were not considered. Modern hybrid algorithms may depend on a full range of inertial sensor measurements to operate correctly and efficiently, and thus prior work evaluated with these data sets would not be easily comparable.

Modern consumer mobile phones and tablets provide an excellent variety of sensors in low cost, readily available off-the-shelf packages. This is an ideal platform for many types of AR research, because it is representative of the type of environment algorithms would be expected to work in if deployed to a wider audience. However, despite this level of standardisation, we lack modern tools and data sets for testing and evaluating tracking algorithms on these platforms.

Published research in mobile AR often lacks source code[3], [11]. There are many reasons for this: copyright/licensing/intellectual property restrictions, poor quality code (suitable for evaluation but not for actual use), lack of time to properly release code (documentation, ongoing maintenance, compatibility with multiple platforms); but practically speaking, it makes the development and testing of new approaches difficult. Implementing a tracking algorithm from scratch is a large undertaking and requires a significant investment of time.

In addition, published data sets and evaluations generated using a specific hardware device quickly become obsolete, as the hardware and sensors are constantly improving and changing. To accurately compare algorithms on consumer level hardware, a working implementation is required, otherwise performance and accuracy cannot be accurately compared. Therefore it is necessary to have access to the source code to perform a realistic comparative analysis of any kind.

The pace of rapid innovation in consumer mobile devices drives changes in the supporting mobile operating systems (OS), including the software frameworks and libraries on which our tracking and registration algorithms are built. Major platforms (e.g. Android, iOS) are completely different in their underlying implementations, such that compiling code for multiple platforms can be a huge burden. There is a culture in commercial AR of only providing the compiled static library, but as hardware changes, these libraries may stop working, and fixing these bugs is practically impossible. If the source code is available and of a reasonable quality, many of these issues at least become addressable.

There are several open source projects which implement inertial sensor based mobile AR tracking algorithms. However, many of the most promising ones seem unmaintained[13], [14], [15]. Other libraries which are maintained, are device specific[16] and application specific[17]. None of the evaluated libraries provide specific tools for the research and development of mobile outdoor AR tracking algorithms.

A. Motivations

The software described in this paper was developed to support a research project investigating improved methods for outdoor AR tracking. It represents a significant investment of time. We have decided to publish the code, documentation, and data sets as completely unencumbered open source, in the hopes that its availability will reduce the barriers we faced when first entering this field.

In addition, we believe that the structure of our contribution can encourage a systematic approach to the development of new algorithms. The existing code base includes many practical examples, and over time we hope that our tooling can serve as a useful platform for a wide variety of tracking implementations.

This is an ongoing project and we realise that it may not fit everyone's requirements. We have specifically chosen online systems that support collaboration, so as to maximise the value of the toolkit - not just for our needs - but for the needs of this field in general.

III. Motion Models

A motion model is an abstract interface that wraps the implementation of a specific tracking algorithm. The set of inputs and outputs are well defined, and relate specifically to the tracking task being performed. Multiple algorithms can be implemented with the same programming interface, which allows the visualisation and browser applications to interact with different implementations without significant changes.

We specifically designed our motion model abstraction around the hardware capabilities of modern mobile devices and the types of data required for accurate global outdoor AR tracking. Specific types of motion models may require additional inputs or outputs. For local image based tracking, we may output a camera pose relative to a local frame of reference. Other motion models might require direct access to the magnetometer. This is supported by subclassing and modifying the source code appropriately.

To serve as an example, we include two motion models in the current distribution. These motion models can be compiled and used both in the visualisation tool and in the browser application:

A. Basic Sensor Motion Model

The included basic sensor motion model implements a sensor fusion based tracking algorithm. It combines the compass and gyroscope using a low-pass filter for improved bearing calculations. It tracks relative changes in rotation around the gravity axis, and exposes these as an output.

In order to initialise the basic sensor motion model, at least one motion update is required to establish a gravity vector, and one heading update to establish a global bearing. After that point, gyroscope updates are combined with heading, and along with the gravity vector and GPS, this provides a fairly robust global frame of reference.

B. Hybrid Motion Model

The hybrid motion model derives from the basic sensor motion model and incorporates image processing into the bearing calculation[18]. It uses the rotation about gravity as the estimate for image processing which computes an accurate global bearing. In the case of a visual tracking failure, sensor fusion provides continuity but typically at a decreased level of accuracy. This ensures a robust output even in challenging scenarios.

IV. Data Capture

We created a mobile data acquisition system[19] to capture sensor data and video frames. This application was developed for iOS using Objective-C. The rate at which sensor data and video frames are captured is independent, and can be customised at compile time. All data is saved in a CSV (comma separated values) formatted log file, including video frames that reference external PNG (portable network graphics) image files in the same directory.

Sensor data is captured using Apple's CoreLocation and CoreMotion frameworks. CoreLocation provides WGS84 latitude, longitude, altitude and bearing, while CoreMotion provides gravity, linear acceleration rotation rate and magnetic flux. The position and bearing typically have quite a high latency, while the motion data is usually captured at 30Hz.

Video frames are captured using AVFoundation that provides access to the camera at a variety of resolutions. We record the captured image as interpolated RGB, which is universally supported across all devices and a convenient format for further processing. We typically capture at 480×360 at 10Hz - higher frame rates generate huge amounts of data and can be harder to analyse.

Phone specific data including the device name and hardware identification are recorded as one of the first entries in the log file. This can be used for hardware or device specific calibration files.

In order to support global tracking algorithms, when recording is triggered, the current position and bearing are written to the log file if possible. This allows at least one global position and bearing entry to exist before frame data and motion data, which is typically required for initialisation.

Fig. 2. The data capture application running on an iPhone 5.

The tool includes a real-time visualisation of sensor measurements. This allows for a greater understanding of the effects that the physical device motion is having on the sensor data. It can be instructive to check the gyroscope, accelerometer or other sensors while manipulating the device. Per-device issues including calibration problems and drift can be assessed quickly and isolated.

A. Android Support

An existing Android data capture tool, developed by another researcher, Alexander Pacha, has been modified and contributed to the Transform Flow project[20]. It can output Transform Flow style data sets from a wide range of Android devices. It has not yet been extensively tested, but supporting a wide range of hardware can be challenging. The coordinate systems for inertial sensors may not be consistent, the field of view for the camera may not be correct, the sensor output may be more or less accurate than generally expected, and the latency in sensor measurements might be significantly different from the norm. Manufacturer and platform specific implementations of the sensor fusion algorithms may also produce different results.

Dealing with these issues is critical for wide-spread deployment on consumer grade hardware. The Android data capture tool is the first step to understanding and solving these problems, and this process will present many good opportunity to further refine and collaborate on the Transform Flow library.

B. Data Set Format

Data sets are recorded using the on-screen switch. This information is saved into the phone's memory and can be downloaded to a computer using the Xcode organiser. The data sets themselves were designed to be simple to work with and flexible enough to support future requirements.

We use CSV as it is a simple format for structured data and can be logged easily. We uniquely identify each row in the log using an index starting from 1, a function name (e.g. "Gyroscope") and a timestamp. The function name relates to the structure of the remaining arguments in the row and is used for processing rows into useful data structures.

It is easy to modify the data capture tool to add additional data structure records, e.g. adding battery status, GPS course, current waypoint, or other parameters that may be of interest. This allows for the data capture application to be extended with new capabilities as required for specific research areas, while retaining core functionality and compatibility with existing processing/visualisation tools.

Nested data can be supported using sequential relationships. For ease of readability, gyroscope, accelerometer and gravity vectors are recorded separately, but are immediately followed by a motion entry with the same timestamp, which ties them together into one logical unit for processing. In the future, additional motion specific

A more detailed specification can be found in the online source code documentation

fields could be added, e.g. magnetometer measurements, without modifying any other structures.

V. Visualisation

We have developed a desktop application[21], shown in Figure 3, for viewing data sets captured using the mobile data acquisition tool. This application is written in C++11 and uses OpenGL for rendering. It currently compiles for Mac OS X and Linux support is almost complete.

Our current implementation assumes that the user is interested in visualising data within a global frame of reference, with position defined by (latitude, longitude, altitude) and rotation defined by (bearing, gravity). To systematically capture this data, we define an abstract motion model that is used to process incoming sensor data and image frames.

The camera pose is computed in two steps, a quaternion rotation based on the bearing and gravity information, and a displacement based on the latitude and longitude. We use East, North, Up (ENU, maps to XYZ) Cartesian coordinates which are intuitive and practical for the small data sets we are usually evaluating.

The input data set, combined with a motion model, produces an immutable per-frame globally registered camera pose. This sequence of frames can then be visualised and analysed without further motion processing. In the case that a motion model requires several sensor updates before it can be initialised reliably, the visualisation will start at the first frame after the localisation becomes valid.

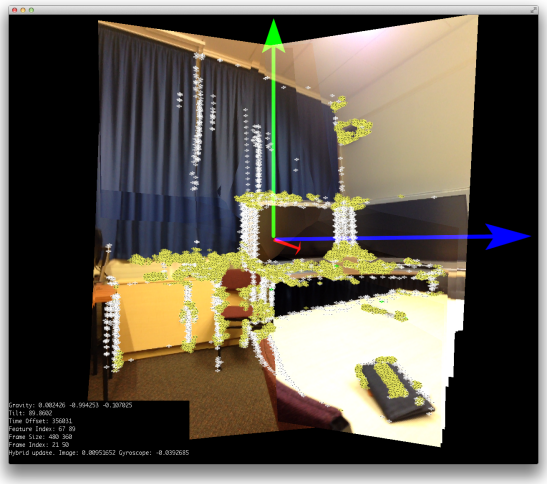

Fig. 3. The Transform Flow Visualisation application rendering 50 combined frames in a 3D environment. The makers have been added by different feature detection algorithms.

Displaying frames in a 3D environment is challenging because a camera frame is a planar projection of a typically non-planar environment. Without depth information it is impossible to accurately reproduce the actual 3D structure. To work around this problem, the visualisation tool uses planar projections based on the camera's field of view and an arbitrary scale factor. Frames can then be rendered in a 3D environment and explored using an arbitrary view position. Pure rotations result in the easiest to interpret visualisation as there is no change in planar alignment.

A. Analysis

Visual inspection of data sets in a 3D frame of reference (see Figure 4) provides a good high level overview of algorithm behaviour and exposes tracking errors including incorrect coordinate frames (e.g. rotation on the wrong axis), sensor integration issues (e.g. large jumps between frames) or bad feature point detection.

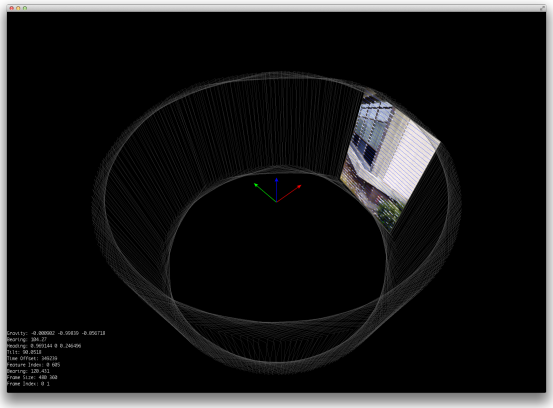

Fig. 4. A 360° panoramic data set, with the first frame visible, in the Transform Flow Visualisation tool. The user can interactively navigate through individual frames and view their associated metadata.

Each frame can have a set of debug notes and visual markers associated with it. These are essentially debug messages from the motion model. Per-frame logging is a useful debugging tool and provides specific feedback about how the motion model was working internally. Per-frame feature points show a structured analysis of the image data and allow a user analyse the details of individual points by selecting them visually.

B. Evaluation

Systematic evaluation is an important process for ascribing a metric to the quality of a tracking algorithm without manual intervention. It allows for the critical and consistent comparison of different algorithms over a wide variety of data sets and for the comparison of the same algorithm as it is being developed and tweaked.

Transform Flow includes support for tracking points, which are spatial markers registered against pixel coordinates for a specific image frame in a specific data set. Tracking points are stored in a separate CSV file inside the data set, and the visualisation tool provides some features to assist with the generation of this data. Tracking points which represent the same visual feature are grouped by a tracking index number, and these form the basic data structure required for further processing.

We have implemented a method to evaluate the accuracy of per-frame bearing. We use groups of static tracking points (such as in data set 2013D0 shown in Figure 5) which are generally fixed physical features, and compute

the 3D position of these points using the camera pose computed by the tracking algorithm. We project these points on the plane $Z = 0$ and compute the bearing. With purely rotational motion, the relative bearing should not change, which is our ground truth. We compute the bearing of each tracking point and measure the average, standard deviation and standard error. The stability of this metric reflects the quality of the tracking algorithm.

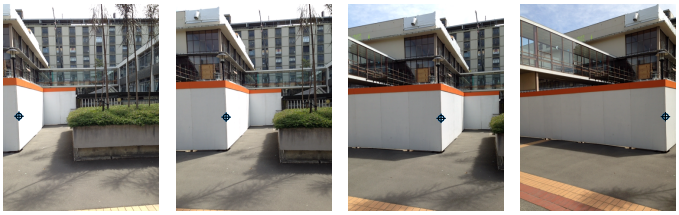

Fig. 5. The 2013D0 tracking point shown in four frames.

In the future, we would like to add a projective based evaluation technique[22]. Given the same feature point in multiple frames, for all frames which contain that feature point, the projection of feature point into a 3D environment should converge to a single 3D position. The distribution of this convergence is directly related to the quality of the tracking algorithm, and similarly to the bearing evaluation, provides us with a metric to compare different approaches systematically.

C. Sample Data Sets

In order to facilitate consistent testing and evaluation, and additionally provide sample data for the visualisation tool, 14 data sets have been published and documented[23]. These data sets are captured using the published iOS capture tool, and include a full range of sensor measurements. We hope that the publication of these data sets will stimulate others to do the same and that over time a large corpus of sample data can be built up to support a wide range of evaluation techniques.

VI. Deployment

We have developed an iOS application, the Transform Flow Browser[24] that can be used to run algorithms developed using the Transform Flow motion model abstraction. It uses a similar setup to the capture tool, but rather than logging the events, it applies them directly to a motion model. The AR visualisation is then rendered using the calculated frame of reference.

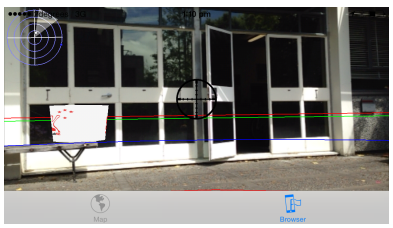

Fig. 6. The Transform Flow Browser, showing a 3D model on top of The HIT Lab NZ.

The browser can display points of interest using 3D content (via Wavefront OBJ files) and 2D billboards (constructed directly from UIView instances), which are rendered on top of a real-time video stream using OpenGLES. The visualisation includes a planar grid which is useful for understanding the practical tracking quality, e.g. jitter, rotations. The implementation itself is multi-threaded, and uses Grand Central Dispatch to offload the rendering, camera frame capture and motion model computations to separate CPU cores so that the main user interface remains responsive.

Dealing with a wide variety of camera and screen configurations is not trivial. Specifically, different cameras have different intrinsic properties, the most important being the field of view. We estimate this parameter as 55° which is, in practice, ±2° for commonly used iOS devices. Device specific calibrations might be useful in a research setting but would be cumbersome for end user applications. Ideally, a per-device database of camera intrinsic properties would allow for wide spread deployment with acceptable accuracy, or for unknown devices, some kind of online calibration procedure.

A. Android Support

The browser application is very much platform specific code as it involves custom UI, rendering and other functionality. In addition, custom code is required for interacting with the hardware (e.g. cameras, sensors). This means that there is a moderate amount of per-platform work required. However, the Transform Flow library and it's supporting libraries all support cross-platform compilation. Because of that, the core tracking functionality need not be reimplemented for different platforms.

Several researchers are presently working on adapting the existing AndroidAR browser to support Transform Flow via the Android NDK. We hope this work progresses and allows us to support a wide range of devices.

VII. Source Code

One of the important goals of this project was to create something that would facilitate collaboration and further research. As such, the data capture, analysis and browser tools have been released under the MIT license on GitHub. The MIT license allows developers to use the source code free of charge with very few limitations (e.g. commercial use is acceptable).

GitHub provides a fantastic environment based on Git[25] where other researchers can easily fork the source code and contribute back their modifications. We hope to integrate new motion models and evaluation methodologies as they are developed, so that our project can serve as a useful tool for comparative analysis for future work.

A. Unit Testing

Ensuring that changes and modifications don't introduce problems with existing code is a useful requirement for collaborative projects. GitHub, when combined with Travis-CI[26], provides immediate feedback when users

make source code submissions (pull requests) on public projects.

Many component parts of the Transform Flow toolkit have unit tests, which means that any time the code is changed, either directly in the repository on GitHub, or as a pull request from another user, the unit tests will be run and the results reported appropriately. This helps to ensure that new users making contributions can feel confident that they are not breaking existing code and supports existing maintainers who might be refactoring or adding new features.

In addition, unit tests also serve to document various parts of the system and how to use them. Functional tests provide useful examples of specific functionality and integration tests show how to combine different parts of a system. This allows new users to become familiar with the code quickly and easily, and serves as a working example of the available functionality and how it should be used.

VIII. Conclusion

The data capture and visualisation tools are useful for developing, analysing and evaluating mobile outdoor AR tracking algorithms. We believe that they can reduce the time it takes to investigate new ideas and will ultimately improve the collaboration between existing researchers.

The AR browser client application can be used to validate tracking algorithms on a variety of real world devices and provides a starting point for usability testing and application development.

We have confidence in the long term viability of the project and will continue to maintain it for the foreseeable future. To support this, we implemented an open source development methodology which encourages collaboration and shared responsibility. We look forward to seeing how it develops over the next few years, and hope to see it used in exciting new research.

References

[1] S. Lee, Y. Seo, and H. S. Yang, "Scalable building facade recognition and tracking for outdoor augmented reality," in *Information Technology Convergence*. Springer, 2013, pp. 923–931.

[2] G. Klein and D. Murray, "Parallel tracking and mapping on a camera phone," in *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*, ser. ISMAR '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 83–86.

[3] G. Reitmayr, T. Langlotz, D. Wagner, A. Mulloni, G. Schall, D. Schmalstieg, and Q. Pan, "Simultaneous localization and mapping for augmented reality," in *Proceedings of the 2010 International Symposium on Ubiquitous Virtual Reality*, ser. ISUVR '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 5–8.

[4] W. T. Fong, S. K. Ong, and A. Y. C. Nee, "Computer vision centric hybrid tracking for augmented reality in outdoor urban environments," in *Proceedings of the 8th International Conference on Virtual Reality Continuum and its Applications in Industry*, ser. VRCAI '09. New York, NY, USA: ACM, 2009, pp. 185–190.

[5] G. Reitmayr and T. W. Drummond, "Going out: Robust tracking for outdoor augmented reality," in *Proc. ISMAR 2006*, IEEE and ACM. Santa Barbara, CA, USA: IEEE CS, October 22–25 2006, pp. 109–118.

[6] G. Reitmayr and T. Drummond, "Initialisation for visual tracking in urban environments," in *Proc. ISMAR 2007*, Nara, Japan, Nov. 13–16 2007, pp. 161–160.

[7] D. Wagner, A. Mulloni, T. Langlotz, and D. Schmalstieg, "Real-time panoramic mapping and tracking on mobile phones," in *Proceedings of the 2010 IEEE Virtual Reality Conference*, ser. VR '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 211–218.

[8] S. Gauglitz, T. Höllerer, and M. Turk, "Evaluation of interest point detectors and feature descriptors for visual tracking," *Int. J. Comput. Vision*, vol. 94, no. 3, pp. 335–360, Sep. 2011.

[9] S. Lieberknecht, S. Benhimane, P. Meier, and N. Navab, "A dataset and evaluation methodology for template-based tracking algorithms," in *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*, ser. ISMAR '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 145–151.

[10] M. Hayashi, I. Kitahara, Y. Kameda, and Y. Ojta, "A study of camera tracking evaluation on TrakMark data-set," in *Proceedings of "The 2nd International Workshop on AR/MR Registration, Tracking and Benchmarking" Workshop in conjunction with ISMAR11*, 2011, p. 5.

[11] D. Kurz and S. Benhimane, "Gravity-aware handheld augmented reality," in *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ser. ISMAR '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 111–120.

[12] D. Kurz and S. Benhimane, "Handheld augmented reality involving gravity measurements," *Computers & Graphics*, vol. 36, no. 7, pp. 866–883, 2012.

[13] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavári, L. M. Encarnaçao, M. Gervautz, and W. Purgathofer, "The studierstube augmented reality project," *Presence: Teleoperators and Virtual Environments*, vol. 11, no. 1, pp. 33–54, 2002.

[14] G. Reitmayr and D. Schmalstieg, "Opentracker: A flexible software design for three-dimensional interaction," *Virtual reality*, vol. 9, no. 1, pp. 79–92, 2005.

[15] "LibreGeoSocial," 2013. [Online]. Available: http://www.libregeosocial.org/

[16] "DroidAR," 2013. [Online]. Available: https://github.com/bitstars/droidar

[17] "mixare," 2013. [Online]. Available: http://www.mixare.org

[18] S. Williams, R. Green, and M. Billinghurst, "Hybrid tracking using gravity aligned edges," in *Proceedings of the 14th Annual Conference of the New Zealand Chapter of the ACM Special Interest Group on Computer-Human Interaction*, ser. CHINZ 2013. ACM, 2013.

[19] S. Williams, "Tranform Flow capture application for iOS," 2013. [Online]. Available: https://github.com/HITLabNZ/transform-flow-capture-ios

[20] A. Pacha and S. Williams, "Tranform Flow capture application for Android," 2013. [Online]. Available: https://github.com/HITLabNZ/transform-flow-capture-android

[21] S. Williams, "Tranform Flow visualisation application," 2013. [Online]. Available: https://github.com/HITLabNZ/transform-flow-visualisation

[22] M. Hayashi, K. Makita, T. Kurata, I. Kitahara, Y. Kameda, and Y. Ohta, "Projective indices for AR/MR benchmarking in TrakMark," 2013.

[23] S. Williams, "Tranform Flow data sets," 2013. [Online]. Available: https://github.com/HITLabNZ/transform-flow-data

[24] S. Williams, "Tranform Flow browser application for iOS," 2013. [Online]. Available: https://github.com/HITLabNZ/transform-flow-browser-ios

[25] L. Torvalds and J. Hamano, "Git: Fast version control system," 2010. [Online]. Available: http://git-scm.com

[26] "Travis CI - free hosted continuous integration platform," 2013. [Online]. Available: https://travis-ci.org