Project acronym: POPART Project number: 644874 Work package: Real-Time Camera Tracking Deliverable number and name: D2.3: Camera tracking library	Title: Camera tracking library Work Package: WP2 Version: 1 Date: 01/12/2015 Author: Lilian Calvet
Type: [ ] Report [ ] Demonstrator, pilot, prototype [ ] Website, patent filings, videos, etc. [X] Other	Co-Author(s): Carsten Griwodz Simone Gasparini To: Albert Gauthier, Project Officer
Status: [ ] Draft [ ] To be reviewed [ ] Proposal [X] Final / Released to EC	Confidentiality: [X] PU – Public [ ] CO – Confidential [ ] CL - Classified
Revision: Final	
Contents: Deliverable 2.3: Camera tracking library.	

Deliverable description

The goal of this deliverable is to provide all the code modules required in order to perform the localization of a multi-camera system in a prereconstructed environment, ranging from the feature extraction over all the synchronised video streams to the final pose estimation of the system.
The code is delivered along with the following report, divided in the two following parts:

- the artificial feature extraction (additional report related to the cctag library - D2.1),
- the camera localization modules' (including the system calibration) description.

I. Artificial feature detection: The CCTag library

1. CCTag system description

(c.f. deliverable description D-2.1)

2. Additional experiments (report)

We conducted a large number of experiments with synthetic images to quantify the performance of the CCTag system. We use a combination of several metrics.These are the (1) false positive rate, (2) misidentification rate and (3) false negative rate. The false positive rate captures the rate at which a marker is reported where none is present. The misidentification rate expresses the rate at which a marker is successfully detected, but misidentified as another marker with another ID. The false negative rate measures the rate at which no marker is reported although one is present. Reducing the false negative rate increases the risk of false positives and misidentifications. Metrics(1) and (2) related to the proposed fiducial system are reported in Table 1 whereas the metric (3) is reported in Figure 1, from (a) to (c). All three metrics are influenced by the conditions under which the fiducial systems achieves a reported performance, and here in particular, the systems (i)scaling tolerance, (ii) occlusion tolerance, (iii) motion blur tolerance and (iv) depth of field blur. Scaling tolerance is the ability to detect and identify markers at a very small or very large size in pixels. A wide-scaling tolerance allows a larger usable range of cameras as well as a more opportunities for placing markers in a scene. Figure 1 reports results for the tolerance from (i) to (iii) (Figure 1 from (a) to (c)) while Figure 1.(d) represents the obtained imaged center accuracy. The distance from the camera to the marker is used to evaluate the scaling tolerance (i) (Figure 1(a)). The percentage of occlusion represents the percentage of area of the fiducial which is not visible (i.e. obscured) over the total area of the fiducial (reported in Figure 1.(b)). Finally, the amount of blur can be expressed by two different quantities representing respectively the amount of blur due to depth of field and the amount of motion blur (supposed unidirectional in accordance with our assumptions). The results delivered by the proposed solution has been compared to the ones delivered by the following fiducial marker systems: i) ARTKPlus[1]: this system remains, yet not recent, a reference in the publicly available solution. ii) PRASAD [2]: this system is the closest to our proposal as using concentric rings in order the propose a fiducial detection resilient to motion blur. iii) RuneTag[3]: as we want to demonstrate the robustness of our system to severe occlusion, we also decided to conduct a comparison against this recent solution.

Scene. The following evaluation is conducted on a scene consisting of a single marker. The size of each “category” of marker in the scene is defined so that they all cover the same area of the supporting plane and that a circular marker is of unit radius. The supporting plane belongs to the (x, y) plane while the marker center corresponds to the scene origin. The marker center is located at (x m , y m , D) cam in the camera coordinate system, where (x m , y m) is randomly distributed in [−0.5, 0.5] 2 and D is the distance from the marker center to the camera optical center whose unit is the circular marker radius. If not specified, the angle between the optical axis and the supporting plane normal is randomly distributed in [0, 75] degrees.

Camera. The synthesized cameras follow the pinhole camera model whose pixels are square, the principal point is located at the center of the pixel plane and the focal length is of 800 pixels while the image resolution is 640 × 360 pixels.

Image signal corruption. As we are dealing with binary patterns, pixels take their values in {0, 255}, resp. black or white. In order to vary both the contrast and the signal to noise ratio, once the original image is rendered, all its pixel values are divided by the scalar value c randomly distributed in [1, 6], thus simulating different lighting conditions. A first blur component related to the depth of field is simulated by applying a gaussian filtering of standard deviation σ randomly distributed in [0, 2]. Then a unidirectional motion blur of length l (whose magnitude is 0 pixel if not specified) is applied in a random direction. Finally, a noise of n gray pixels randomly distributed in [0, 5] is added on every pixel of the obtained image.

Results. All the results in Figure 1 are expressed in terms of detection rate, i.e. $\tau_d = 1 - \tau_n$ where τ_n is the false negative rate. We can see that even without any occlusion and motion blur, the proposed system already outperforms the other ones. This can be mainly justified by the following: extensive experiments have shown that our system performs quite well even in the case where the marker is strongly inclined relatively to the pixel plane. This comes mainly from the fact that there are always image cuts along which the concentric circles are “minimally distorted”, and identical information content is equally readable along any single direction. Markers that require the extraction of data from one or more dedicated directions to extract encoded information (including a classical barcode as well as 2D patterns) pay their much higher information-coding capacity with an increase in misidentifications with distortion increases. Tolerance to distortion must then be countered by mechanisms that avoid ID collisions, but these increase the Hamming distance and reduce their information-coding capacity. It is also important to mention that the RuneTag solution presents very poor performance as the solution relies on the performance of an ellipse detection algorithm used to detect the imaged circular dots whose performance drastically drop down while the size of the imaged marker is decreasing.

Accuracy under very challenging shooting conditions. In the last experiment, whose results are illustrated in Figure 1(d), the tests have been run in very challenging conditions in terms of lighting, noise, distance and motion blur. In such conditions, no fiducial system used for comparison has been able to detect any marker out of 800 images apart from the proposed one, which has been able to detect 506 of them, (i.e. 63 %). This experiment has been conducted with c = 5, i.e. pixels values ranking between 0 and 51 pixels, a noise randomly distributed in [0, 10] pixels and with a camera to marker distance D = 30. We want to emphasize two important points with these experiments. The first one is that, even in such very challenging conditions, we obtain a high detection rate, i.e. 94%, 80%, 57% and 22% for a magnitude of motion blur of 0, 5, 10 and 15 pixels respectively. The second point is to

emphasize that, even under these challenging conditions, with an increasing motion blur magnitude, the overall accuracy of the imaged center estimation is less than one pixel, while its median is less than 0.4 pixels for a motion blur magnitude less than l = 10 pixels.

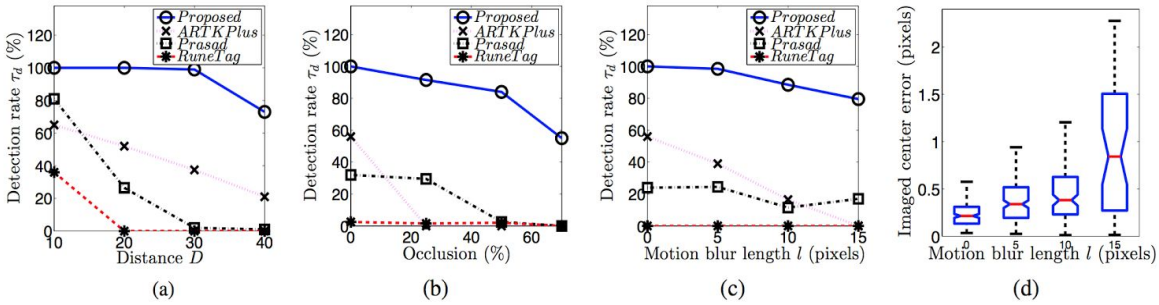


Figure 1. Detection rate (a) vs. the distance D (b) vs. the percentage of occlusion (c) vs. motion blur magnitude (d) Accuracy of the imaged center estimation vs. the motion blur magnitude. The red line represents the median of the error while a blue box represents the first quartile. See text for details.

False negative and confusion. Lastly, Table. 1 shows the results of the evaluation of the false positive and the inter-confusion marker rate. The false positive rate evaluation has been performed on five video sequences whose the image resolution is 640 × 360 pixels. All the systems but PRASAD present a false positive rate of 0%. However, our system presents a misidentification rate of 2.8% which is more than the ones delivered by ARTKPlus but still negligible for our camera tracking application for which constraints brought by the geometry can be used for ID disambiguation (e.g. through guided matching techniques) as long as a subset of imaged fiducial are correctly detected and identified in the image.

	Prop.	PRASAD	RuneTag	ARTKPlus	Nb. frames
Garden	1	51	0	0	1268
False pos.	0%	4%	0%	0%	
Indoor	0	21	0	0	1685
False pos.	0%	1.3%	0%	0%	
Street	0	80	0	0	1153
False pos.	0%	6.9%	0%	0%	
Stripes	0	172	0	0	762
False pos.	0%	22.6	0%	0%	
Text	0	23	0	0	525
False pos.	0%	4.4%	0%	0%	
Total	1	347	0	0	5393
False pos.	0%	6.4%	0%	0%	
Nb. detect.	744	221	72	351	800
Fig. 8 (a)	93%	27.6%	9%	44%	
Nb. conf.	21	71	0	0	800
Fig. 8 (a)	2.8%	32%	0%	0%	

Table 1. False positive and misidentification evaluation. See text for details.

Time performance.
Our CPU implementation is able to deliver a frame rate of 4 fps on one CPU (i5-4590, 3Ghz) core on a 1280 × 720 image, which has been raised to 12.4 fps using the GPU (NVidia GTX

980 Ti, CUDA 7.0) implementation. As mentioned in the deliverable report D-2.1, the CCTag detection is performed on a pyramid of images, which provides a trade-off between the search for distances between the edge of a ring and the ability to detect markers of various sizes and at a wide range of distances from the camera. In our experiments, we used pyramids of N = 4 levels, where the image resolution is halved both in X and Y direction between levels. Once candidate markers are selected over all the pyramid levels, their imaged center optimization including followed by the identification step are performed on the original image to ensure a maximum accuracy.

The GPU version replaces the implementation of the pyramid building phase completely, where the possible parallelization still allows a speedup from 720p to 1080p videos that is superlinear. In the detection stage, voting is handled on the GPU, but the incremental formation of convex arc segments remains on the CPU. An experiment implementation of the arc-formation on the GPU exists, but as a stand-alone module, it is clearly slower due to the linear nature of the arc-formation. Although only a few marker candidates remain after the detection stage, there lies a decent parallelization potential in the validation stage, which is due to a brute-force search step for the accurate localization of the imaged center. The inner loop of this search has also been moved to the GPU, for a speed gain of approximately 30%. Since this is applied to a very limited number of markers, as well as a number of center candidates that is small in terms of GPU capacity, the limited speedup was to be expected.

<table>
<tr><th>resolution</th><th>stages of detection</th><th>CPU</th><th></th><th>GPU</th><th></th></tr>
<tr><td>720p</td><td>loading, pyramid building</td><td>48.6 ms</td><td rowspan="3">4 fps</td><td>12.5 ms</td><td rowspan="3">12.4 fps</td></tr>
<tr><td>720p</td><td>candidate detection</td><td>152 ms</td><td>32 ms</td></tr>
<tr><td>720p</td><td>candidate validation/identification</td><td>58 ms</td><td>36 ms</td></tr>
<tr><td>1080p</td><td>loading, pyramid building</td><td>105 ms</td><td rowspan="3">2.7 fps</td><td>21 ms</td><td rowspan="3">8.8 fps</td></tr>
<tr><td>1080p</td><td>candidate detection</td><td>209 ms</td><td>52 ms</td></tr>
<tr><td>1080p</td><td>candidate validation/identification</td><td>61 ms</td><td>40 ms</td></tr>
</table>

Table 2. Comparison of CPU and GPU performance by stage, on a video showing up to 5 markers.

The code is written in C++ (and CUDA's C++ dialect for the GPU implementation). The CUDA code makes use of Dynamic Parallelism, a feature that was introduced with CUDA Compute Capability 3.5, and that allows kernels running on the GPU to instantiate further kernels itself. This feature allows various blocking copy-operations between host and graphics card, and helps to achieve minor speed gains. Although most code has been written from scratch to remove library dependencies, it makes use of the the radix sort implementation in the template library CUB. Unfortunately, this limits us to the CUDA version 7.0, since CUB is broken in the more recent version 7.5.

The GPU implementation works on the float datatype, in contrast to the CPU, which makes use of doubles. Although the difference in precision is noticeable already in the 4 decimal position, it is sufficient to locate identical tag positions. The accuracy requirement is highest in the identification step that has only recently been moved to the GPU, but results remained accurate.

Summary. To summarize, the capabilities of the proposed system are listed below.

	CCTag	ARTKPlus	RuneTag	Prasad	MonoSpectrum[8]
Free of camera calibration					
Allows camera calibration					
Robust to occlusion					
Robust to motion blur					
Open source release CPU					
Open source release GPU					
# reference points per fiducial [min-max]	[1 to 3]	4	[5 to 129]	0	0
Library size	32 (3 rings) 128 (4 rings)	512	762 (Rune-43)	4	825

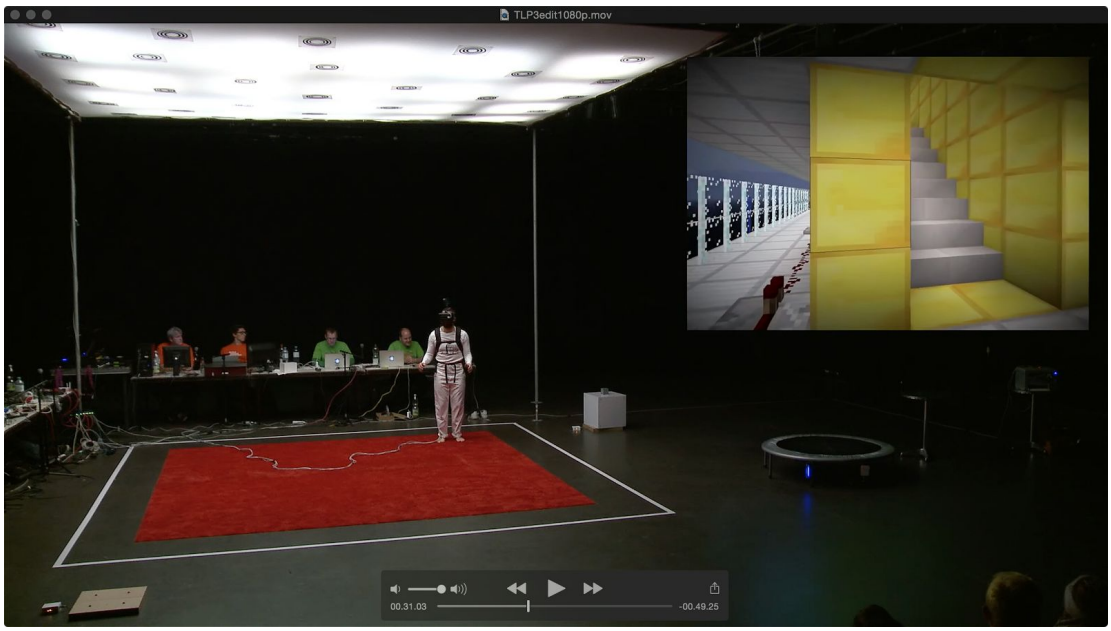
Table 2. Summary of capabilities of existing fiducial systems. Green cells represent the available features whereas the red cells represent the ones not available. See text for details.

• Free of camera calibration: the system does not require the intrinsic parameters of the camera to be known.
• Allows camera calibration: the geometry of an imaged fiducial provides constraints for the camera's intrinsics estimation and, under "normal" shooting conditions, it can further provide the image of the circular points to constrain the camera calibration process (e.g., the well-known plane-based calibration method).
• Robust to occlusion: the fiducial can be detected even if it is occluded in the views (e.g., up to 60%).
• Robust to motion blur: in presence of a linear motion blur (i.e., motion blur due to a fast camera translation), the proposed fiducial is orthogonal to the blur direction and can be still detected and identified.
• Open source release: the code will be soon released as open source with both a CPU and a fast GPU implementation.

Code download
The code can be downloaded here
https://github.com/poparteu/cctag

For the GPU version, use branch gort_integration. For the CPU version, use the branch develop.

Evaluation
An intermediate version of the CCTag tracker was tested and used successfully in a cooperation with Austrian artists and other international researchers, notably in collaboration with the FP7 project FIPS (no. 609757) and with the University of Stellenbosch. The cooperation culminated in 3 public performances at the WUK in Vienna (http://www.wuk.at/event/id/17912) on October 8-10, 2015. In this artistic project, the Third Life Project, CCTag tracking was performed with a single camera mounted on backpack-like contraption worn by an actor who wore an Oculus Rift and navigated a large Minecraft world by translation of his real-world movements, tracked exclusively with the CCTag-based tracker, into the Minecraft client.



Screenshot from a recording of a public performance of the Third Life Project

Although the speed of the tracker was rather limited at the time of the performance (4 fps), it delivered accurate results on both position and orientation, and allowed the actor to move naturally at normal walking speed in the virtual environment.

Besides integrating the CCTag into a tracker for the first time, we gained important understanding of lighting conditions and camera limitations during these performances. More about the Third Life Project can be found at http://www.loviska.com/thirdlifeproject.html.

A video that summarizes the performances will be produced, but the recordings of all three public performances can be made available on request already.

II. Camera tracking: POPART contributions to OpenMVG

We moved the code of cameraLocalization (https://github.com/poparteu/cameraLocalization) directly inside OpenMVG. The objectives of this move is to simplify testing, ease long term

integration and maintenance by avoiding code redundancy and different data formats. While the former is still mantained in the original repository, the new code is under development and publicly available at https://github.com/poparteu/openMVG, mostly in the “dev_mikros” branch.

NOTE: All the following modules can indifferently use natural (SIFT) or artificial (CCTag) features points.

Calibration of the multi-camera system. The multi-camera system calibration has been implemented in the OpenMVG library. Such a calibration is performed in order to introduce the rigidity constraints related to the positions of the witness cameras, rigidly fixed on the camera rig, during the camera tracking. These constraints consist in constant relative poses between the witness cameras and the main camera, a data then used during the camera tracking in order to perform the camera system resectioning.
The calibration procedure has been implemented as follows:

1. Robust estimation of the relative pose of every camera-pair “witness/main” cameras,
2. Bundle adjustment over all the N-tuples of images (extracted from the synchronised video streams) where N representants the total number of cameras, i.e. in configuration, a maximum of 2 witness cameras plus the main camera.

The first step (1) is decomposed as follows:

- For a given witness/main camera-pair, compute independently the pose of the witness camera and the main camera with respect to coordinate system of the 3D visual database for all the synchronised image-pairs over the time t=1..T.
- For a given witness/main camera-pair, compute all relative pose between the witness camera and the main camera over the time t=1..T.
- For a given witness/main camera-pair, over all the obtained relative poses over t, compute the one minimizing the reprojection error in its associated image-pair.

The step (2) is stated as a nonlinear least squares problem whose function is to minimize the sum of the reprojection errors over all the N-tuples of images. The optimization is performed through the Google Ceres Solver (via the levenberg-marquardt algorithm) while the initial solution consists of the set of initial solutions delivered by the step (1).
The algorithm has currently been tested on a real dataset [4] as well as on synthetic data [5] and has performed well (in terms of reprojection errors).

Frame query from the database. We have been focusing on the off-line camera tracker that could be used in post-production to track a sequence of images with respect to an existing 3D reconstruction of the scene. The localization algorithm can work with natural features alone (SIFT), fiducial markers (CCTags) or both of them at the same time.
We present here some preliminary results on simple datasets and we are working to make it functional on real-world use cases.

POPART Dataset Levallois - Synthetic data [4]
video available here: https://youtu.be/S1ywiDbb3oA

Using the dataset created by MIK we tested the localization algorithm based on SIFT. We rendered a camera movement in Maya in order to have a sequence of images and we localize each image separately.

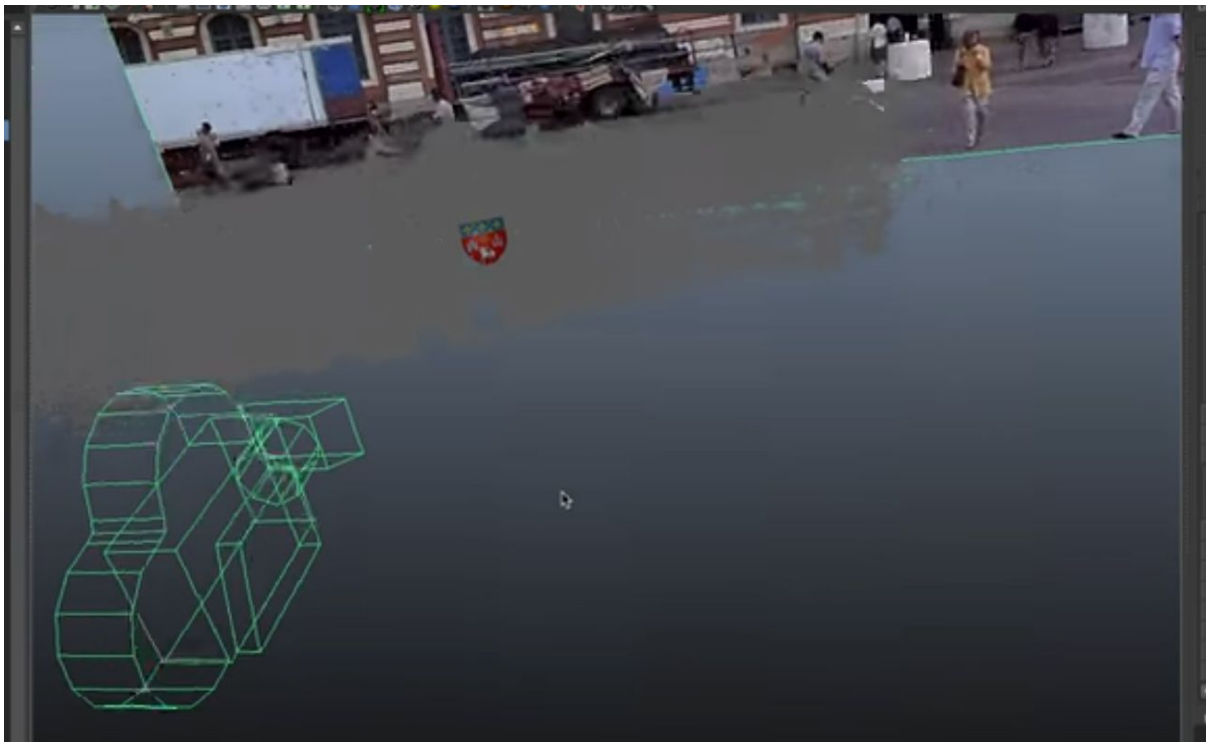
The CG model of the Levallois City Hall with some compositing (the Coats of arms): a camera movement has been rendered and used to test the off-line localization algorithm.

Dataset Toulouse Capitole

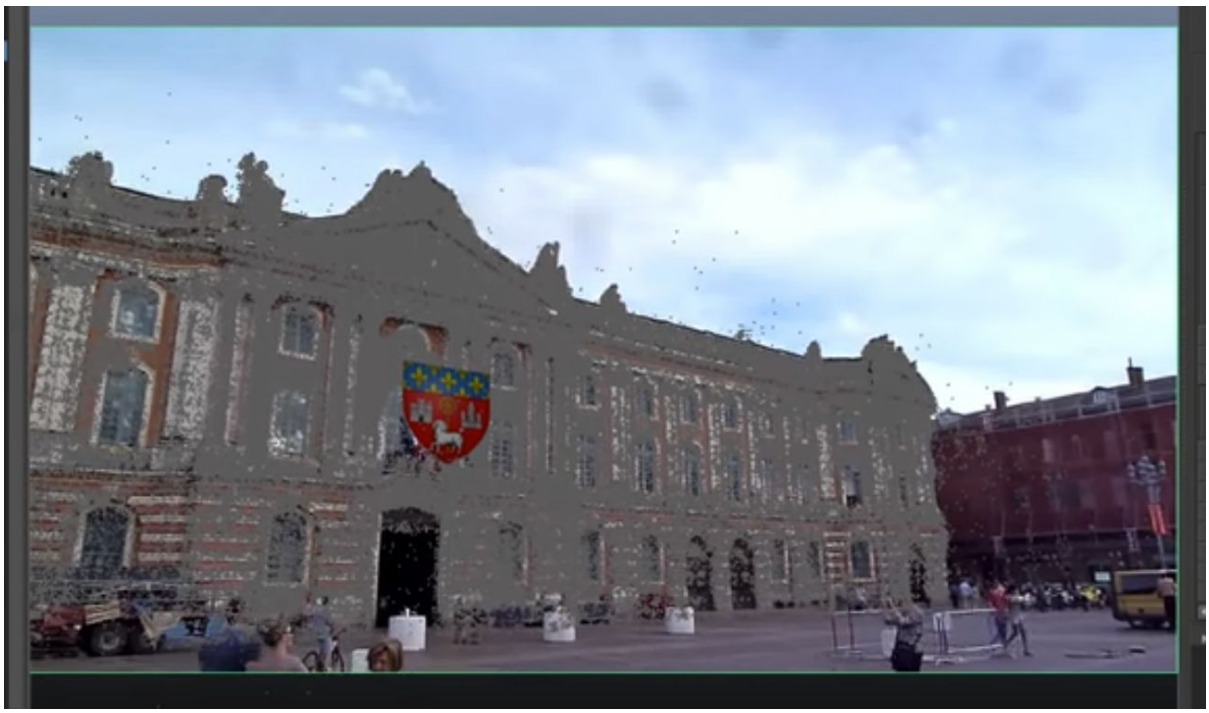
Maya screencast available here:

- https://youtu.be/X7zKK-W-qRQ
- https://youtu.be/I6oltk9iVUU

We tested the off-line camera localization algorithm based on SIFT on a dataset taken in Toulouse. The City Hall (Capitole) has been reconstructed in 3D from a set of images. Then some videos have been taken in the main square and processed off-line with our algorithm. The videos are HD 720, and the camera has been calibrated off-line with an OpenCV-based application. There are around 2000 frames for each video and all of them have been correctly localized with an RMSE error for both sequences around 1.5pixel. If a further global bundle adjustment is run on the entire sequence refining the camera intrinsics as well, the final result can be further improved and RMSE of 0.5pixel is obtained.



A screenshot of the scene taken from Maya, the camera has been localized on all the images of the sequence, the Toulouse's coat of arms has been manually added to the 3D point cloud of the Capitole (in grey) in order to visually assess the quality of the localization during the rendering.



A screenshot of the scene taken from Maya showing that the 3D Point cloud of the building match the image.

SIFT features extraction

Due to its stability, the SIFT feature extraction was chosen for the tracking of natural features. A survey of open-source implementations found that the performance of most of these libraries was far from real-time, and the difference of code-bases was limited, ie. the family tree of version was easily traceable, with the exception of the implementation published on Github by user “Celebrandil” (https://github.com/Celebrandil/CudaSift), which relies on CUDA and achieves real-time speed on a pair of 1080p videos, albeit with hard-coded SIFT parameters that are “legal” but find only a quite small number of features. The implementation shows a deep understanding of the SIFT idea, in particular where accuracy is traded for speed, and where the implementation differs from all other by not following order of Pyramid building described in the original SIFT paper by Lowe (DOI 10.1023/B:VISI.0000029664.99615.94). The speed would certainly have made it worthwhile a closer investigation of the version’s performance in spite of its unconventional decision decision. However, it’s license is an exclusive research license without any opening for commercial use, and it can therefore not serve the commercial goals of POPART. Since the other implementations failed to achieve real-time speed, we decided to allocate some time on an own implementation.

POPSift extracts SIFT features, but it is not complete yet. For the time being, it takes 25ms until Gaussian and DoG are computed for a 1440x1080 frame on an NVidia GTX 780 Ti card (1280x960: 16ms, 640x360: 4.4ms, 320x180: 1.5ms). Although visual inspection is promising, we have not confirmed yet that it can be used as a drop-in replacement for another SIFT implementation. One reason is the use of normalized vs. non-normalized feature vectors, and there is also a very wide range of parameters that can be chosen, and have a major influence on success and speed.

Beyond compatibility tests, we are convinced that the sequence of Pyramid building can theoretically be changed; whether this is also valid with the inaccuracies introduced by a real implementation remains to be seen. Furthermore, existing implementations seem to have a bug in common where subsequent octaves are built from the previous octaves’ most strongly blurred frame, which contradicts the theory. The effect of this must be understood. Finally, the last stage in POPSift, feature extraction, is currently still a port from the bemap project (BEnchMarks for Automatic Parallelizer), and although this is acceptable under its BSD license, it should be replaced to gain speed.

Robust pose estimation of the multi-camera system. The current implementation of the pose estimation of the multi-camera system is performed as follows:

- For each image of a tuple of images associated to a timestamp t, the camera resectioning is performed and must succeed for, at least, one image in order to deliver the pose of the system.
- Once the N (>= 1) camera poses are obtained, then providing a pose guess, all the reliable 2D-3D correspondences (so called *inliers*) are used simultaneously in order to refine the pose of the multi-camera system while integrating the rigidity constraints related to the relative pose between the witness camera and the main camera (data available from the multi-camera system calibration stage).

We are currently evaluating a more robust approach in which the camera rig is treated as a non central camera and use a so-called “generalized PnP” algorithm [6] to compute the rig pose using all the features detected by all the camera at the same time, as it has been implemented in OpenGV [7].

Code download
The code can be downloaded here
https://github.com/poparteu/openmvg
Reference branch: **dev_mikros**

Executables: (launch with -h (--help) for the list of arguments)

- Localization:
  - `openMVG_main_cctagLocalizer` allows to localize a sequence using only CCTags.
  - `openMVG_main_voctreeLocalizer` allows to localize a sequence using SIFT features and eventually CCTags.
  - `openMVG_main_rigLocalizer` is a more general script that allows to localize a rig using different type of features

- Calibration
  - `openMVG_main_rigCalibration` allows to calibrate a rig using different (mixture of) type of features

References

[1] H. Kato and M. Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR’99), pages 85–94. IEEE Comput. Soc, 1999.

[2] M. G. Prasad, S. Chandran, and M. S. Brown. A motion blur resilient fiducial for quadcopter imaging. In Proceedings of the 2015 IEEE Winter Conference on Applications of Computer Vision (WACV15), WACV ’15, pages 254–261, Washington, DC, USA, 2015. IEEE Computer Society.

[3] M. G. Prasad, S. Chandran, and M. S. Brown. A motion blur resilient fiducial for quadcopter imaging. In Proceedings of the 2015 IEEE Winter Conference on Applications of Computer Vision (WACV15), WACV ’15, pages 254–261, Washington, DC, USA, 2015. IEEE Computer Society.

[4] POPART 2015. POPART Mixed Realities Dataset https://www.openaire.eu/search/dataset?datasetId=r37b0ad08687::ed1d7d023e53a230fdce3058299ad7d5 (Grimstad Studio)

[5] POPART 2015. POPART Virtual Dataset - Levallois Town Hall. Zenodo.

[6] Kneip, L., Furgale, P., & Siegwart, R. (2013). Using multi-camera systems in robotics: Efficient solutions to the NPnP problem. *Proceedings - IEEE International Conference on Robotics and Automation*, (2004), 3770–3776. doi:10.1109/ICRA.2013.6631107

[7] L. Kneip, P. Furgale, "OpenGV: A unified and generalized approach to real-time calibrated geometric vision", Proc. of The IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China. May 2014.

[8] M. Toyoura, H. Aruga, M. Turk, and X. Mao. Mono-spectrum marker: an AR marker robust to image blur and defocus. The Visual Computer, 30(9):1035–1044, Dec. 2013.