Project acronym: POPART Project number: 644874 Work package: Real-time camera tracking Deliverable number and name: D2.1: Open source artificial feature detector on GPU	Title: Open source artificial feature detector on GPU Work Package: WP2 Version: 1 Date: 01/07/2015 Author: Carsten Griwodz
Type: [ ] Report [ ] Demonstrator, pilot, prototype [ ] Website, patent filings, videos, etc. [X] Other: source code	Co-Author(s): Lilian Calvet 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: D2.1: Open source artificial feature detector on GPU.	

CCTag description

Artificial features, also referred as fiducial markers, consist of artificial objects designed so that their image can be measured in an image acquired by a digital camera, and deliver both geometrical (e.g. camera-marker pose) and semantic (e.g. identifying) information. They have been widely used for 3D tracking of objects, which consists of continuously determining the object's location in a video sequence when either the object or the camera are moving [1]. They are especially useful in applications such as pedestrian or robot navigation, Augmented Reality and photo-modeling.

In the POPART project, markers are used to provide a reliable and precise camera tracking solution even in the presence of rapid camera motion. Camera tracking is the task of recovering the camera position and orientation within a video sequence based on natural features, whereas markers are usually required in texture-less scenes. Common state-of-the-art methods for camera tracking are based on SLAM (Simultaneous Localization and Mapping), which is a closely related technique for robotics navigation [2]. Basically, SLAM methods use the live video stream to track visual features across the frames and estimate the movement of the camera. These methods perform well for general camera motion, but they suffer from drift due to the accumulation of errors and the difficulty of handling cycle closures of the camera trajectory.

The camera tracking method proposed in POPART adopts a different tracking paradigm, which is based on the continuous (i.e. for each frame) localization of the camera with respect to a priori 3D visual mapping of the scene that associates reconstructed 3D features to their multiple visual appearances, e.g. through the identifying information delivered by the imaged markers or, for natural features, by means of descriptors within a set of key views.
In our context, such a 3D visual mapping is performed based on a set of high-quality images (e.g. static shooting in good lighting conditions) of the scene via a Structure-from-Motion algorithm such as those proposed in [3,4,5]. The Structure-from-Motion solution used in POPART builds on and improves the open source library OpenMVG.
The camera tracking problem is thus stated in terms of image-based localization, a problem which consists of accurately determining the position and orientation from which a novel view was taken relative to a known 3D representation of the scene [6].

Problem statement and contributions

A critical problem for camera tracking algorithms, both for SLAM and image-based localization approaches, is rapid camera motion (e.g. rapid rotation, camera shake, etc.), which may drastically reduce the number and the precision of tracked features due to motion blur. Some methods exists in SLAM to deal with blur (blur estimation [7] or the use of "edglets" [8]). For image-based localization methods, natural features such as SIFT [9], AKAZE [10], BRISK [11], are to some extent robust to motion blur since they are scale invariant and may thus be matched when local features are large in a blurred image. A more general solution may consist of selecting a very short exposure time that minimises blur in captured images. Unfortunately, this approach is not universally applicable: The use of low exposure times incurs noise penalties in low-light situations.

In this work package, our specific problem consists of providing a fiducial *marker system*, i.e. a set of planar patterns and the associated computer vision algorithms to localize and identify them in real-time, while also achieving robustness to motion blur.

Conventional planar markers are usually made up of both low and high frequency components. In general, low frequency components are used for the pattern localization step, whereas high frequency components, e.g. bitonal squares or dot, are used to encode the identifying information. Although low frequency components still remain to some extent localizable under motion blur conditions, a critical problem is that high frequency components are not preserved, thus making the identifying information unreadable in the image.
To prevent this problem, many solutions such as [12,13,14] first detect imaged markers in a sharp view and then track them across the frames. However, such tracking algorithms require a slow camera motion and they require reinitialization once the tracked features are totally lost (e.g. due to occlusion combined with fast camera motion). Another solution works by increasing the size of the markers introduced in the scene but may be too invasive and so not acceptable.

The proposed marker system relies on already existing planar markers made up of a set of concentric circles (see Figure 1.(a)). They have been first introduced by participants of the POPART project in [15,16], and are referred to as CCTag. In the papers, the authors present the advantages of CCTags in solving the Structure-from-Motion problem based on the circular points of the supporting plane delivered by such circular primitives.

Motivations

In this work, using planar markers made up of concentric circles are motivated by the two following insights:

1. By using a sufficiently large number of circles, it is possible to build in a CCTag view, parametric curves that converge onto the imaged center. This property leads to a localization algorithm robust to challenging images (e.g. noise, blur, strong occlusions, etc.).
2. Both its geometric and photometric characteristics make a CCTag identifiable even in the presence of motion blur. An intuitive formulation of this result can be expressed as follows: in the presence of sufficient motion blur, the only sharp edges present in the view are those parallel to the direction of blur [17]. Thus, using concentric circles allows to preserve, to some extent, edges located along the virtual line passing through their imaged center and whose the direction is orthogonal to the (supposed unidirectional) motion blur direction. This property is illustrated in Figure 2. Experiments have shown that this property is still true even in presence of perspective distortions.

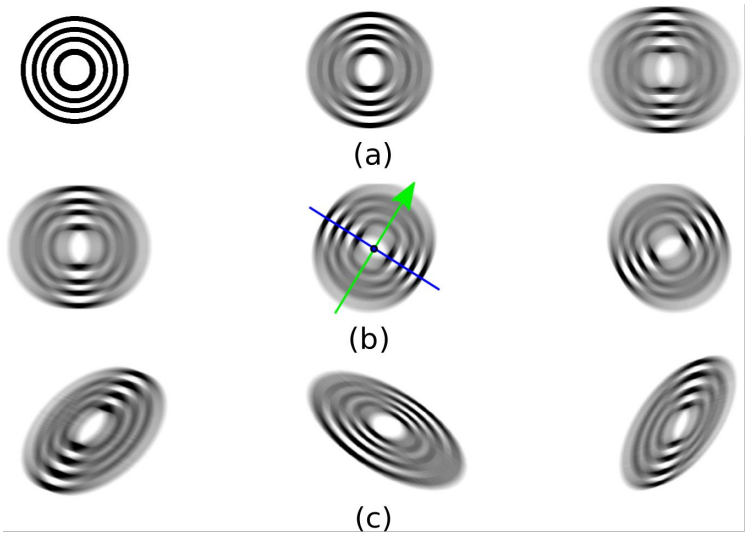


Figure 1. Three CCTags to which unidirectional motion blur has been applied (a) with increasing magnitude from left (null) to right; (b) in three different directions; (c) on their images from three different points of view. This figure illustrates the property that, under motion blur conditions, edges located along the line passing through the center of the circles (blue line) whose direction is orthogonal to the motion blur one (green line) are preserved.

System overview

Image pyramid. Although the detection algorithm is theoretically invariant to scale change, a multiresolution analysis of the input image is performed in order to decrease the time complexity of its underlying algorithms. As a result, all the steps dedicated to the detection of the imaged CCTag pattern are performed on every level of the image pyramid. In order to reach optimal accuracy in the imaged center localization and to use all the information delivered by the input image, the imaged center optimization and the identification step (described in the following) are performed on the base image of the pyramid.

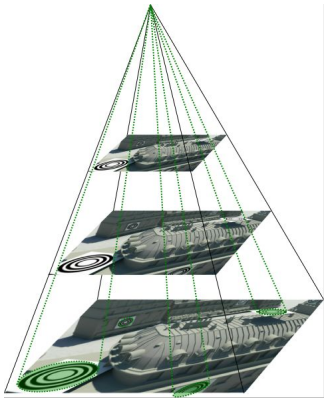


Figure 2. Image pyramid

For each level of the pyramid, a binary image of edge points is obtained via the Canny edge detection algorithm. It is important to mention that this step involves various image convolutions, including a Gaussian

filtering (used in order to reduce the noise) followed by two Gaussian derivatives in order to compute both horizontal and vertical derivative of the image. These convolutions are highly suitable operations for parallelization on GPU architectures. In the binary image of edge points, the approach is subsequently searching for point-pairs of edge points, where one point of the pair lies on the inner ellipse and the other on the outer ellipse of the imaged CCTag.

Vote. All the edge points are subject to a voting procedure in order to elect edge points located on the inner ellipse, through a procedure that ensures that an elected edge point is necessarily received from an edge point lying on the outer ellipse. The CCTag pattern consists of N circular circles, each bounded by two concentric circles. The imaged marker is thus consisting of 2N elliptical edge segments. We aim to detect edge points-pair (u,v) such that u is a point lying on the outer ellipse and v is a point lying on the inner ellipse. The detection problem is formulated as a problem of vote for v, i.e. the edge point of the inner ellipse.
From the binary image of edges, an edge point is first (arbitrarily) selected. It issues a voting intention. This vote consists is valid only if it is possible to build from it a polygonal line whose vertices are edge points that satisfy some approximation conditions of a parametric curve (not described here) defined by the projective transformation relating the supporting plane of the marker to the pixel plane. If it is the case, the vote is issued. Its purpose is to reach the only point v that is assumed to be located on the inner ellipse. The voting procedure applied on real images is illustrated Figures 3 and 4.

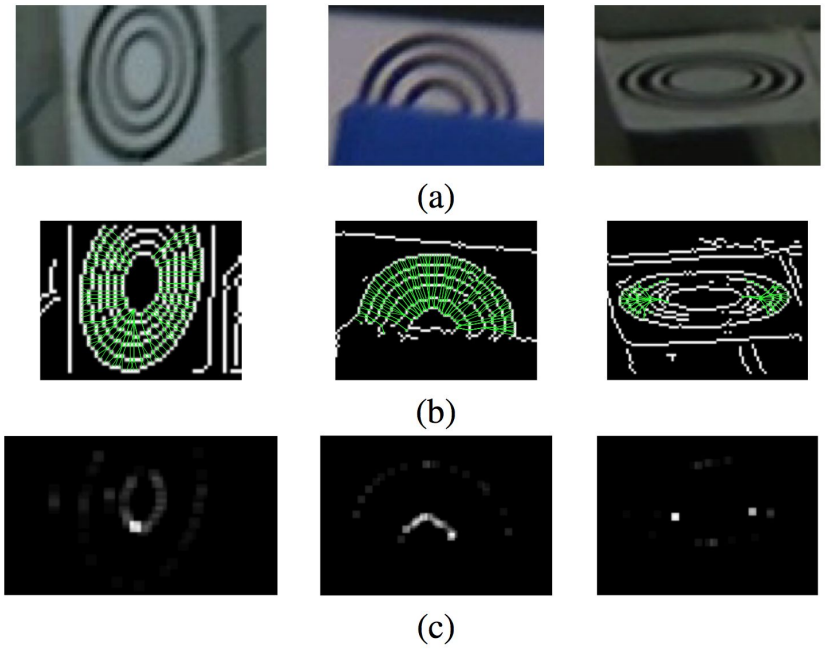


Figure 3. (a) Three images of CCTags. (b) Binary images returned by the edge detection with superimposed pixels for all edge points that have voting intentions. (c) Images of the result of the voting procedure, with an Gaussian filter applied to the result to increase visibility for this illustration.

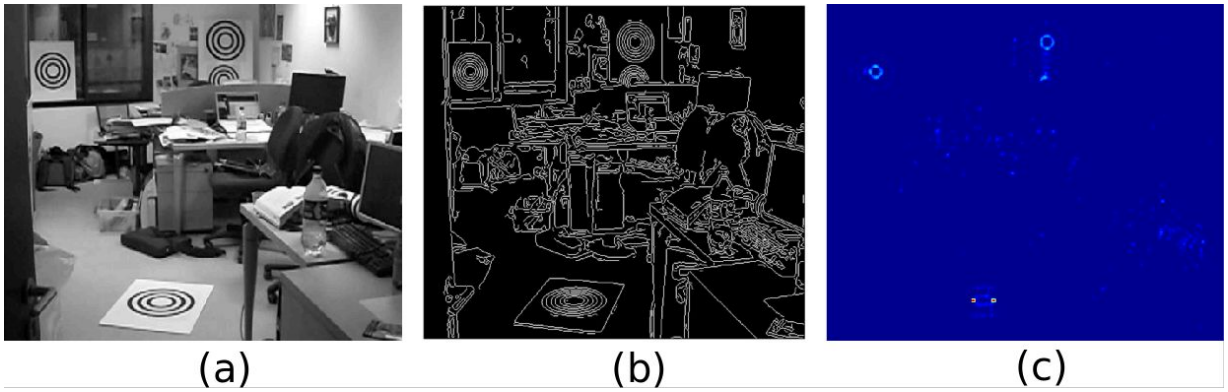


Figure 4. Example of a voting result based on a real image. (a) Original image in grayscale. (b) Binary image from the canny edge extraction. (c) Vote result.

Edge linking. Each edge point that received a “sufficiently” large number of votes is a candidate for being an edge point of the inner ellipse. Candidates are grouped by a linking procedure in order to form uninterrupted edge segments of the inner ellipse, approaching arcs of this ellipse. This linking step is performed based on the constraint that the edge segment must always be convex.

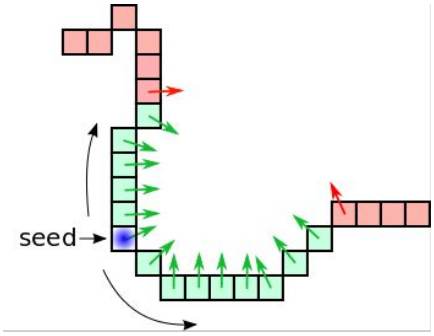


Figure 5. Creation of convex edge segments. The blue dot represents the position of the seed. The black arrows illustrate the two directions followed by the edge linking algorithm into both directions. The set of green pixels represents the convex edge segment obtained. The linking is stopped when the gradient direction are “diverging” (red arrows).

Ellipse growing. A robust estimation of the outer ellipse Q is then performed from the set of points E that voted for the inner ellipse. This step consists in the initialization of the ellipse growing algorithm [kanatani] which aims to collect all the edge points located on the outer ellipse.
The principle of the ellipse growing algorithm is the following:

1. an elliptical hull around Q is computed ; it is defined by two ellipses, an expansion and a contraction of Q, (referred as Q+ and Q-) such that their semi-axis are respectively increased and decreased of δ pixels
2. all the edge points connected to the initial set of edge points E and located in the elliptical hull are collected and a new estimate of Q is computed
3. E and Q are then updated

This procedure is repeated until no new edge point are added to E (see Figure 6).

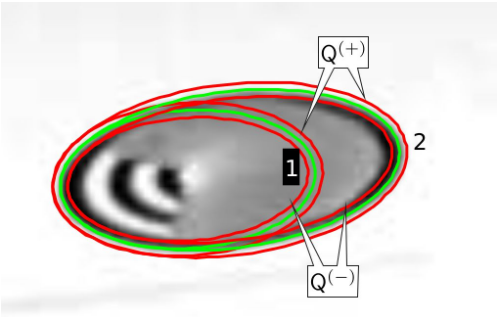


Figure 6. Ellipse growing illustration. This configuration leads to a convergence of the algorithm in two iterations. The two green ellipses represents ellipses Q computed at each iteration. Red ellipses represent Q+ and Q- defining the elliptical hulls.

All the detected outer ellipses are then back-projected on the base layer of the image pyramid (as illustrated Figure 2).

Imaged center optimization. It is important to mention that, under any general (projective) 2D transformation, the imaged center is not located on the ellipse(s) center(s), image(s) of the concentric circles. In fact, this property is only verified under affine and euclidean 2D transformation [18]. Another important property is that the outer ellipse, imaged of the outer circle, along with its imaged center, deliver the euclidean structure of the supporting plane. In other words, once the outer ellipse and the imaged center are retrieved, it is possible to rectify the image signal in order to obtained its euclidean representation thus readable as a simple 1D barcode.
For each candidate, it's imaged center is then estimated through an optimization based on the pixel level.

Identification. The identification step is then performed by “reading” the rectified 1D signals located between the imaged center and a certain number of edge point located on the outer ellipse via the method presented in [19]

An overview of the entire system is illustrated in Figure 7.

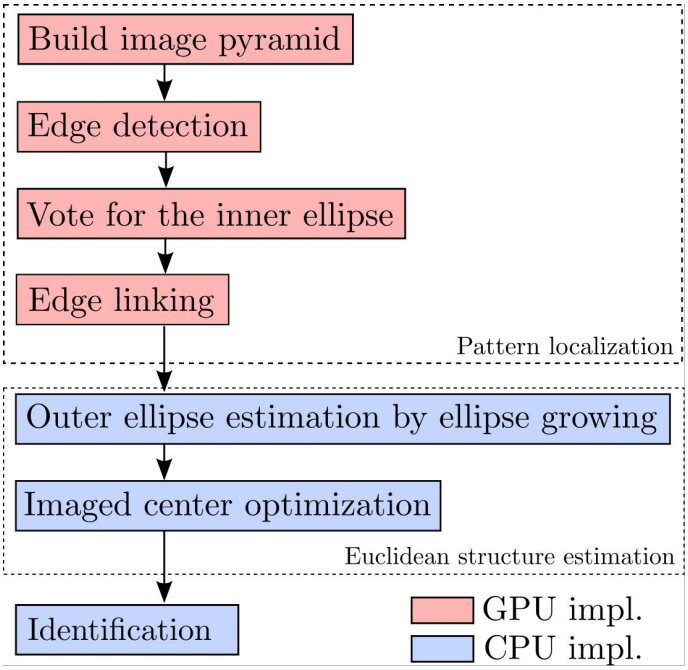


Figure 7. System overview. Steps corresponding to GPU implementations are marked in red and CPU implementations are marked in blue.

Time performance

The design of CCTag as shown in Figure 7 allows partial parallelization, due to several pixel-centric steps at the beginning of the process, followed by a sequence of point selection, which requires knowledge about all candidates for CCTag identification, optimization of the center selection, and finally, an identification of the particular CCTag ID from an estimation of ring size ratios. The optimal split between GPU-side and CPU-side code is not certain yet, for two reasons:

- The optimization of both CPU and GPU code is still progressing with the intent of supporting an even larger number of cameras and higher resolutions at real-time;
- The initial assumption that the edge image, which must certainly be generated on the GPU side, would only be required by the GPU, was found to be untrue. Since it must be transferred from the GPU to the CPU anyway and optimized CPU code exists for all stages, there is room for experimentation.

The overview of steps in Figure 7 shows the sequence that has to be performed for every single camera image. Identification in this step provides a list of IDs for all CCTags that have been identified from an image, where an increase of markers in a image affects all steps from edge linking to identification.

Building the image pyramid is a means of reducing the effort of estimating circle radii and circle widths by creating lower resolution images. In the GPU implementation, this makes firstly use of the GPU's texture engine that can use normalized coordinates to create enlarged as well as reduced versions of the input

image through hardware bilinear filtering. This is followed by Gaussian filtering to reduce noise for each resolution. Gaussian filtering is well-suited for GPU implementation because it can be implemented by separated convolution filters, which reduces the number of required memory accesses and floating point operations.

Edge detection for CCTags relies on the Canny edge detector, which is mostly comprised of localized decisions. The most time-consuming step of the Canny edge detector is the edge hysteresis step, which requires multiple sweeps of the entire image plane to upgrade possible edge points to probably edge points based on neighbourhoods. The small step consumes currently 80% of the GPU computation time for the Canny algorithm. The thinning step, on the other hands, could be optimized by improved lookup tables.

Voting requires, first, a sweep of the edge image, as well as gradient images, to identify stretches that may be spanning the thickness of a black or white circle of a CCTag in an image. These candidates are stored in a list (an array on the GPU), and are subsequently chained to determine whether they may comprise a chain of stretches pointing towards an image center. While the creation is localized and well-suited for GPU operation, chaining is not, because global pointers should be avoided on GPUs. Instead, lookup tables are required, which are rather cache-unfriendly because of the disconnection of candidates in the list and the coordinates of potentially neighbouring pixels in the edge image. Furthermore, chain length is defined by a run-time configuration parameter, which reduces optimization options further. In spite of this, there is such a large number of steps that consist mainly of 1D and 2D index lookup operations that this step is still much better suited for the GPU than for the CPU. The output of this step is a very small number of CCTag centers and points located on outer circles of those tags.

Edge linking has not been moved to the GPU yet. Because of the small number of CCTags that are expected in an image, the small number of points that are evaluated in every step of the edge linking loop, and the unknown duration of the loop, it is highly uncertain whether moving it to the GPU is a good idea. However, all decisions that are taken are strictly local to a known edge point, and there are supposedly many points of the same outer ellipse of a CCTag that are growing together to form an arc segment. It is thus still a matter of experimentation whether this can lead to a further performance improvement.

Computation time of the different implementations have been measured based on 5 images of reference:

- ‘24cc.png’: an image containing 24 CCTags,
- ‘5cc.png’ and ‘6cc.png’: images containing 5 and 6 CCTags resp., i.e. close to the expected use case,
- ‘0cc-1.png’ and ’0cc-2.png’: images without imaged CCTags with cluttered background

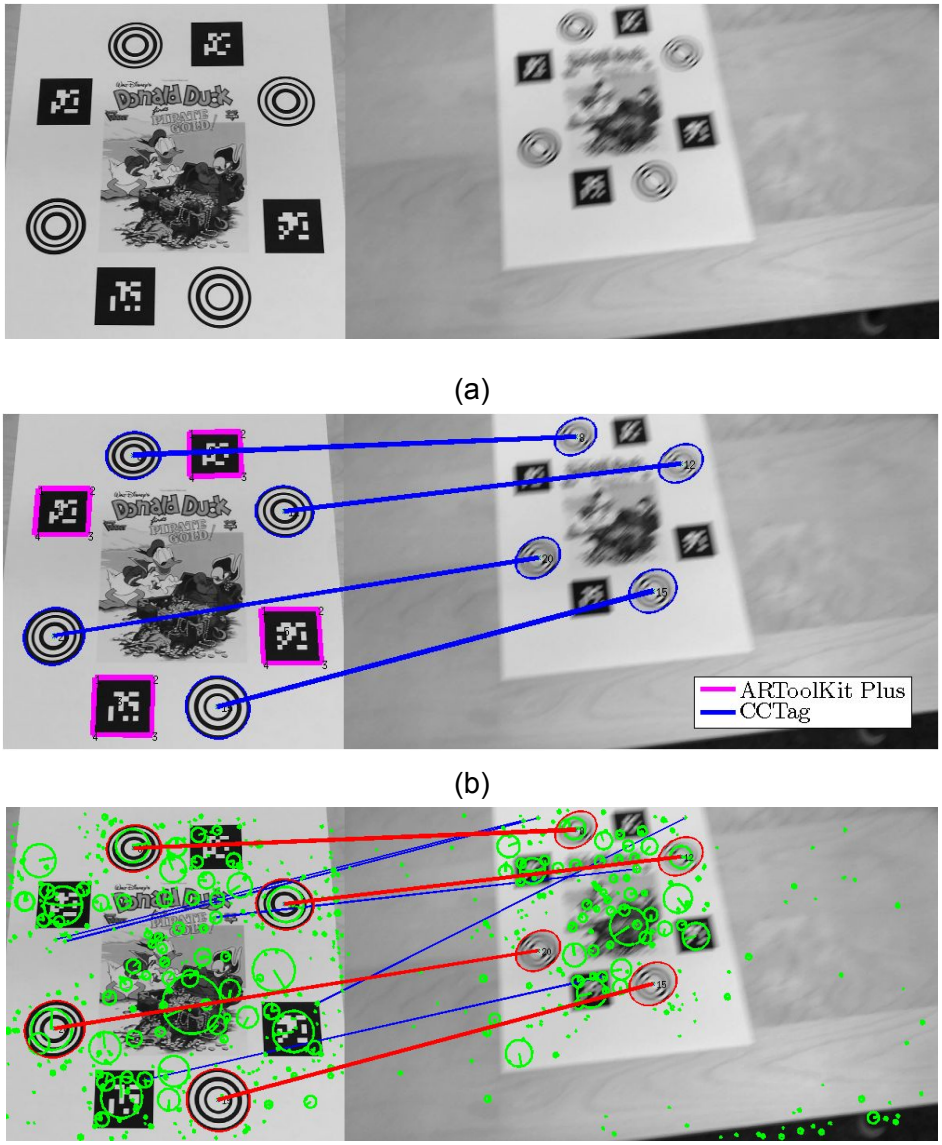
All these images have a resolution of **3264x2448**. CPU implementations are run on one CPU core (2,3 GHz Intel Core i7). The GPU implementation uses an NVidia GeForce GTX 980 with CUDA 7.0.

	Old CPU	New CPU	GPU	Time per/
Total time	5.75” 2.82” 2.95” 2.23” 2.78”	3.41” 1.7” 1.5” 0.99” 1.5”	Not connected to CPU part yet	/image 24cc.png 5cc.png 6cc.png 0cc-1.png 0cc-2.png
Cut collection (before center optimization)	~4ms	~0ms	CPU only	/marker
Center optimization	~20ms	~1ms	CPU only	/marker
Build image pyramid	(function not isolated)	516ms 519ms 521ms 511ms 526ms	includes transfer 8.76ms 9.25ms 9.52ms 8.77ms 9.57ms	/image 24cc.png 5cc.png 6cc.png 0cc-1.png 0cc-2.png
Vote	80ms 274ms 271ms 233ms 247ms	80ms 274ms 271ms 233ms 247ms	10.52ms 9.31ms 9.35ms 9.51ms 10.52ms	/image 24cc.png 5cc.png 6cc.png 0cc-1.png 0cc-2.png
Cut selection (on going code optimization)	~100ms	Target: 0ms (will be removed)	CPU only	/marker

Table 1. Time performance of the previous CPU implementation, the current CPU one and the current GPU implementation. Further details about all the substeps of the algorithm are provided in **Annex 1**.

Results

Real data. Some results on real data are presented in Figure 8 and 9 where the robustness of the CCTag system under challenging shooting conditions such as fast camera motion but also challenging lighting conditions is illustrated.



(a)

(b)

(c)

Figure 8. (a) A two-view matching of both natural and artificial features under the following challenging shooting condition: the left view is correctly focused contrary to the right view which is acquired by a fast moving camera which generates blur distortion and feature occlusions. (b) The ARToolkitPlus system correctly detects and identifies the artificial markers (surrounded in magenta) only in the sharp (left) view whereas no one is detected in the blurred view; matching hence clearly failed. (c) The natural feature matching algorithms SIFT is applied on this pair provide aberrant matching (represented by the red lines). The proposed system correctly detect and identify the imaged CCTags (concentric circle sets) in the two views even in presence of both motion blur and occlusion: matching succeeds (blue lines).

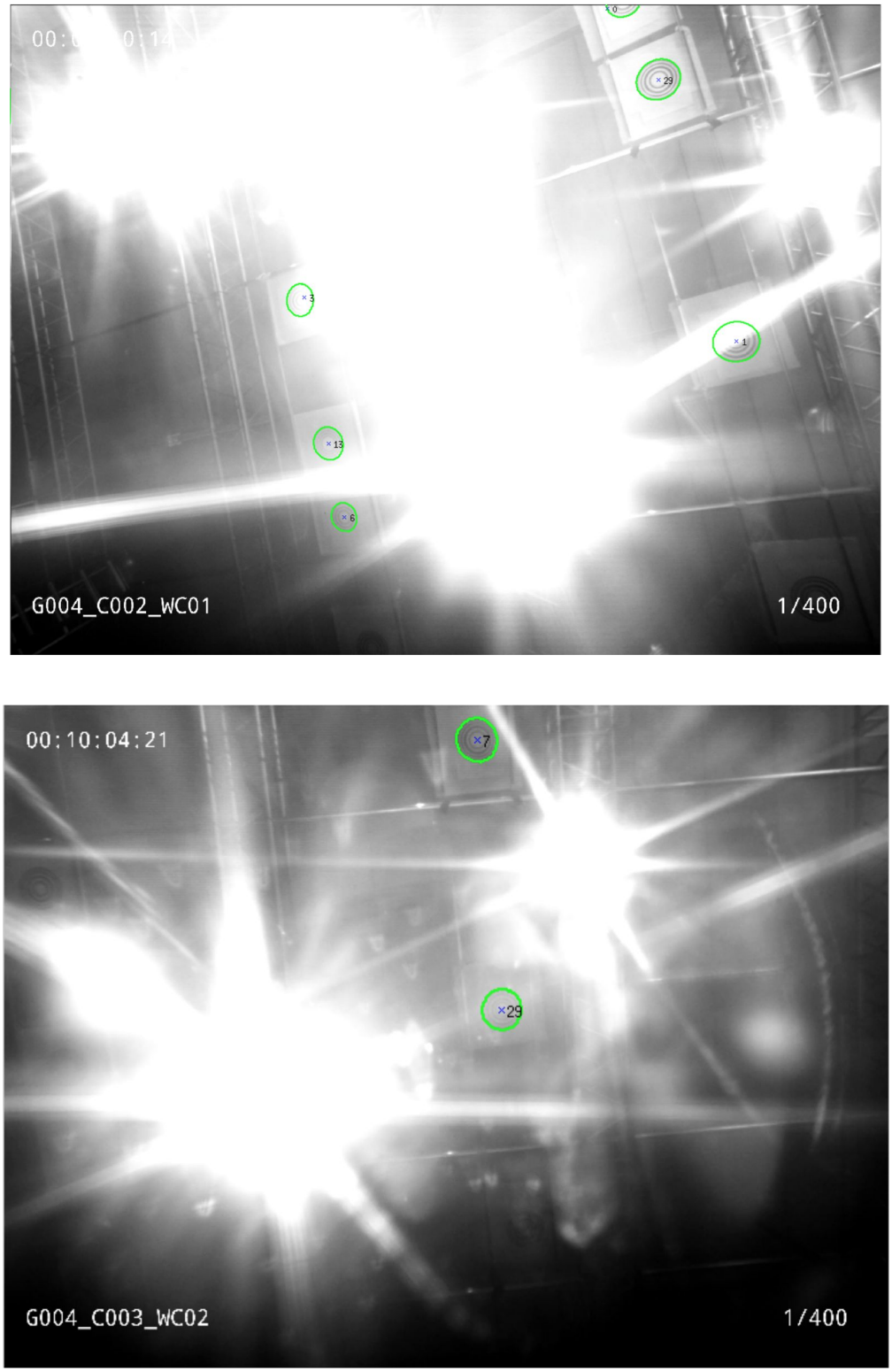
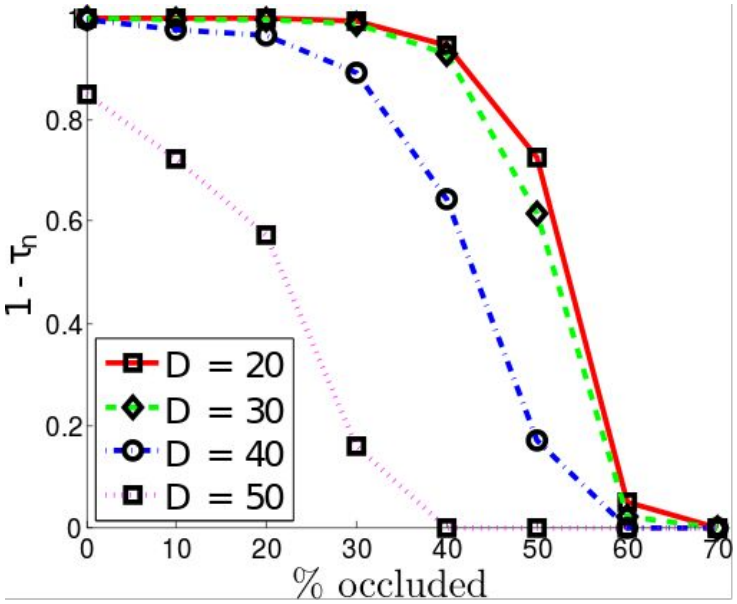
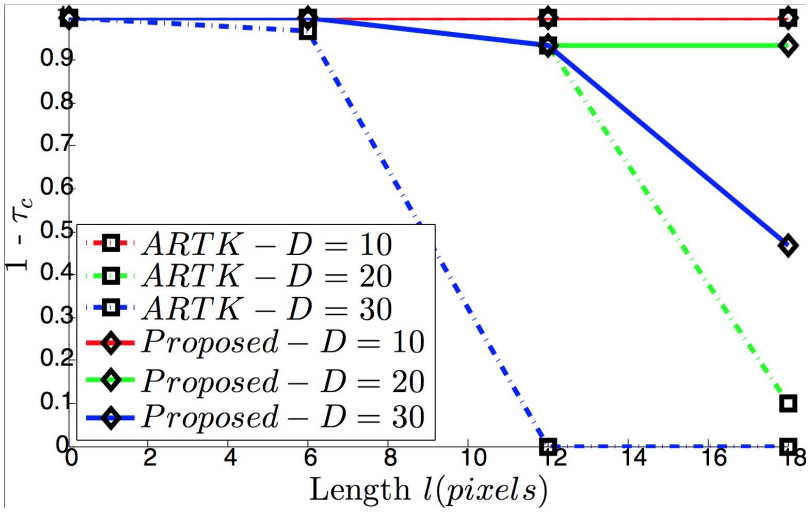


Figure 9. Robustness of the CCTag system under challenging lighting conditions (real data). The CCTag localized by the detection algorithm are superimposed in green.

Synthetic data. Simulation have also been performed. Results are presented in Figure 10.



(a)



(b)

Figure 10. Results delivered by the proposed method based on synthetic data. The presented values correspond to the true positive rate over 100 images for each configuration. A configuration consists in a CCTag of unit radius at a distance D to the camera (of focal length 1000 pixels). The angle formed between the optical axis of the camera and the normal of the supporting plane is randomly taken between 0 and 60 degrees. In (a), results are expressed in function of the percentage of occlusion. In (b), they are expressed while varying the magnitude of the unidirectional motion blur applied in a random direction. The proposed method is compared to the results delivered by the open source solution ARToolkitPlus [12] (only in (b) as [12] is not meant to be robust to occlusion).

Library size. Two library sizes, i.e. the number of unique CCTag ids, are currently available. One is composed of CCTags made up of three-circle markers and is composed 32 unique markers as each circle can have two possible width. The second one is composed of CCTags made up of four circles and its size is 128. The current results, presented above, correspond to experiments performed with the three-circle CCTags.

Code

The code is available at:

https://github.com/poparteu/CCTag

References

[1] V. Lepetit and P. Fua. Monocular Model-Based 3D Tracking of Rigid Objects. Foundations and Trends in Computer Graphics and Vision, 1(1):1–89, 2005

[2] A. J. Davison. Real-Time Simultaneous Localisation and Mapping with a Single Camera. Proceedings of the 2003 IEEE International Conference on Computer Vision (ICCV2003), 2:1403—-1410, 2003

[3] P. Moulon, P. Monasse, and R. Marlet. Adaptive structure from motion with a contrario model estimation. In K. M. Lee, Y. Matsushita, J. M. Rehg, and Z. Hu, editors, Proceedings of the 11th Asian conference on Computer Vision (ACCV 2013)
Berlin Heidelberg

[4] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3D. ACM Transactions on Graphics, 25(3):835–846, July 2006.

[5] C. Wu. Towards Linear-Time Incremental Structure from Motion. In Proceedings of the International Conference on 3D Vision, pages 127–134. IEEE, June 2013.

[6] T. Sattler. Efficient & Effective Image-Based Localization. PhD thesis, Aachen University, 2013.

[7] H. S. Lee, J. Kwon, and K. M. Lee. Simultaneous localization, mapping and deblurring. In Proceedings of the 2011 IEEE International Conference on Computer Vision (ICCV2011), pages 1203–1210. IEEE, Nov. 2011.

[8] E. Eade and T. Drummond. Edge landmarks in monocular SLAM. Image and Vision Computing, 27(5):588–596, Apr. 2009.

[9] D. Lowe. Distinctive image features from scale-invariant keypoints. International journal of computer vision, pages
1–29, 2004.

[10] P. F. Alcantarilla, J. Nuevo, and A. Bartoli. Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces. In Proceedings of the 2013 British Machine Vision Conference (BMVC2013), 2013.

[11] S. Leutenegger, M. Chli, and R. Y. Siegwart. BRISK: Binary Robust invariant scalable keypoints. In Proceedings of the 2011 IEEE International Conference on Computer Vision (ICCV2011), pages 2548–2555. IEEE, Nov. 2011.
[12] 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
[13] M. Fiala. ARTag, a Fiducial Marker System Using Digital Techniques. In Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR2005), volume 2, pages 590–596. IEEE, 2005
[14] D. Wagner and D. Schmalstieg. ARToolKitPlus for Pose Tracking on Mobile Devices. In Proceedings of 12th Computer Vision Winter Workshop CVWW07, pages 139–146, 2007.
[15] L. Calvet and P. Gurdjos. An Structure-from-Motion Paradigm Based on the Absolute Dual Quadric and Images of Circular Points. In Proceedings of the 2013 IEEE International Conference on Computer Vision (ICCV2013), pages 985–992. IEEE Computer Society, 2013.
[16] L. Calvet, P. Gurdjos, and V. Charvillat. Camera tracking based on circular point factorization. In Proceedings of the 2012 International Conference on Pattern Recognition (ICPR2012), pages 2128—-2131, 2012.
[17] Georg S. W. Klein, Tom Drummond: A Single-frame Visual Gyroscope. BMVC 2005
[18] Jun-Sik Kim, Pierre Gurdjos, In-So Kweon: Geometric and Algebraic Constraints of Projected Concentric Circles and Their Applications to Camera Calibration. IEEE Trans. Pattern Anal. Mach. Intell. 27(4): 637-642 (2005)
[19] Orazio Gallo, Roberto Manduchi: Reading 1D Barcodes with Mobile Phones Using Deformable Templates. IEEE Trans. Pattern Anal. Mach. Intell. 33(9): 1834-1843 (2011)

Annex 1

1. Build a pyramid:
    a. canny:
        i. sobel (compute dx, dy)
        ii. edge extraction
        iii. thinning
2. For each pyramid level:
    a. edge point collection
    b. voting procedure
    c. thresholding on number of received vote
    d. sort elected edge points, refered as *seeds*

    e. create flow candidate, i.e. the couple (inner elliptical segment,outer ellipse). segment) from all seeds
        i. edge linking
        ii. Transfer GPU => CPU
        iii. Outlier removal through robust estimation of the outer ellipse: ellipse growing initialization
        iv. ellipse growing
        v. search for assembling flow components together (Least Median of Squares)
        vi. create the final CCTag instance from one or two assembled flow components
3. Imaged center optimization:
    a. back projection of detected CCTag on the original image
    b. Sharp cut selection
    c. Non linear least squares optimization: minimize the rectified signals difference of the signal located between the imaged center and edge points on the outer ellipse
4. Identification: reading of the rectified signal (implies a signal interpolation) around the optimized imaged center.