OpenGV: A Unified and Generalized Approach to Real-Time Calibrated Geometric Vision

Laurent Kneip and Paul Furgale
CECS, Australian National University and ASL, ETH Zurich

***Abstract*— OpenGV is a new C++ library for calibrated real-time 3D geometric vision. It unifies both central and non-central absolute and relative camera pose computation algorithms within a single library. Each problem type comes with minimal and non-minimal closed-form solvers, as well as non-linear iterative optimization and robust sample consensus methods. OpenGV therefore contains an unprecedented level of completeness with regard to calibrated geometric vision algorithms, and it is the first library with a dedicated focus on a unified real-time usage of non-central multi-camera systems, which are increasingly popular in robotics and in the automotive industry. This paper introduces OpenGV's flexible interface and abstraction for multi-camera systems, and outlines the performance of all contained algorithms. It is our hope that the introduction of this open-source platform will motivate people to use it and potentially also include more algorithms, which would further contribute to the general accessibility of geometric vision algorithms, and build a common playground for the fair comparison of different solutions.**

I. Introduction

Even in times of powerful and freely available localization frameworks that rely on temporal model predictions, the purely geometric computation of a camera pose remains a fundamental problem that requires robust and accurate real-time solutions. The most prominent class of applications is formed by large-scale 3D reconstruction frameworks [1], which are capable of transforming sets of unordered images into 3D models of entire cities. The absence of temporal or topological information linking the images demands a purely geometric solution of camera poses prior to photometric error minimization. However, even classical visual-SLAM algorithms [2], [3] or modern visual-inertial solutions [4] that typically rely on temporal predictions given by a motion model require geometric vision algorithms for any of the "not so normal" tasks, such as bootstrapping, loop-closure, pose recovery, or relocalization. Further potential applications exist for multi-robot localization and mapping. In summary, we can easily identify a persistent need for geometric pose computation algorithms in all domains where structure-from-motion plays a role, such as localization and mapping, object tracking, object modelling, geolocalization, robot extrinsic calibration, augmented reality, and photogrammetry, to name just a few.

Although the need for geometric vision algorithms is ubiquitous within robotics and computer vision, no existing software library for camera pose computation covers all interesting cases. There are many generic structure-from-motion libraries available, such as OpenCV [5], VXL (derived from TargetJR and IUE), Gandalf (mainly homography estimation), LibMV, Bundler, CMVS (using bundler), and OpenMVG. These mostly aim at offering complete pipelines for 3D reconstruction or dense model generation, covering all processing steps from interest point extraction down to large-scale non-linear optimization. However, the geometric camera pose computation algorithms provided by these packages never go beyond the standard single-camera problems of space resectioning, homography estimation, or the computation of the fundamental or essential matrix[1].

The goal of the OpenGV project is to introduce a novel common open-source platform for accessing, comparing, and collecting all types of efficient camera pose computation algorithms. In contrast to many other libraries, OpenGV is a pure 3D library that aims at highly efficient solutions to calibrated problems, thus mainly targeting applications in the robotics, automotive, and consumer electronics industries that require real-time solutions to absolute and relative camera pose computation [2]. Besides a more focussed scope, the main difference to existing frameworks is that OpenGV is not limited to single camera applications. The palette of algorithms is completed by including solutions for both *central* and *non-central* camera systems. Central camera systems are characterized by a single viewing origin—all rays from the camera originate at a single point—wheras non-central

[1] VXL additionally contains methods for solving the orthogonal procrustes problem, and Libmv also contains methods for pose estimation with unknown focal length.

[2] By "real-time", we understand here the capability of an algorithm to—even if embedded into an iterative RANSAC scheme—support processing at common camera frame-rates (>10 Hz).

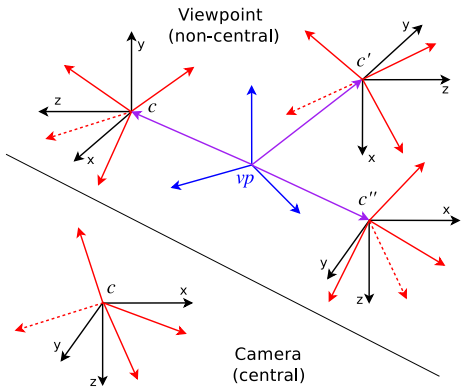

Fig. 1. Illustration of a *camera* c and a *viewpoint* vp (in blue). The camera is a single reference frame with bearing vector measurements (in red). The viewpoint has multiple cameras, each one containing its own bearing vector measurements and an individual transformation to the viewpoint frame.

camera systems have rays originating from multiple points. Figure 1 illustrates the difference between these two cases. Non-central algorithms are more robust than central algorithms in the case of multi-camera systems [6], and thus are of utmost importance for real-time 3D vision tasks in modern robotics and automotive applications where multi-sensor or multi-camera systems are ubiquitous. OpenGV contains a flexible interface designed for painless interoperability with existing libraries, and therefore is preferably understood as a complementary extension to existing frameworks rather than a competition.

OpenGV is the first library to present a unified approach to geometric vision not depending on the specifics of the employed imaging system. The library operates directly in 3D and abstracts the camera projection system by using transformed image information only. It employs a generalized description of multi-camera systems, and is therefore applicable to almost any—either central or non-central—camera. It is the first library to provide implementations that solve the highly complicated problems of computing the absolute or relative pose of a generalized camera. Although the benefits of the latter in multi-camera structure from motion are regularly praised in the literature, those algorithms have so far only been presented without accompanying implementations. OpenGV now makes it possible to put the credo of *using many cameras as one* [7] into practice by providing the required building blocks to migrate well-known single camera structure-from-motion concepts to multi-camera scenarios. OpenGV is written in C++, and provides multiple efficient solutions to identical calibrated absolute and relative camera pose computation problems as well as a Matlab interface. The framework makes it easy to compare algorithms against each other and provides a suitable benchmark-tool for future algorithm development. Our goal is for OpenGV to become a commonly accepted geometric vision library for camera pose computation and automatic benchmarking, thus making it easy for future algorithms to be included and avoiding the problem that algorithms posted as independent code fragments on laboratory web-pages may have different interfaces and therefore be difficult to compare.

The remainder of this paper is structured as follows. Section II highlights OpenGV's generalized approach towards multi-camera systems as well as all contained algorithms. In Section III, we highlight some important implementation details. In Section IV, we present the comparative evaluation of all contained algorithms obtained by our automatic Matlab-based benchmarking tool.

Access: The entire library is hosted as an open-source project on github and can be forked under

https://github.com/laurentkneip/opengv

The library also contains its own webpage with an exhaustive documentation of the interface and numerous examples. It can be accessed under

http://laurentkneip.github.io/opengv

All results in this paper can be easily reproduced by installing the library and executing the respective benchmark files.

II. OpenGV

This section introduces OpenGV. It starts by outlining the employed generic description of multi-camera systems used by all algorithms within the library. It then gives a brief introduction to all the problems that can be solved, and finally introduces some important details and novelties about OpenGV's sample-consensus functionalities.

A. Unified description of input data

OpenGV employs a generalized description of camera systems allowing an application to nearly any optical system. The description is centered around an elegant representation of a multi-camera system, without losing the ability to represent fully generalized camera models. An example for the latter is for instance given by a catadioptric camera looking at an arbitrarily shaped mirror.

The specifics of particular camera models are hidden from the interface by representing each image measurement as a 3D bearing vector: a unit vector originating at the camera center and pointing toward the landmark. This can be accomplished easily for most practically relevant cameras—including perspective, dioptric, and catadioptric imaging devices—given that the calibration parameters of a suitable central camera model are known. Each bearing vector has only two degrees of freedom, which are the azimuth and elevation inside the camera reference frame. 3D bearing vectors are—next to normalized image coordinates—a standard choice in computer vision. They provide the advantage of being able to represent omnidirectional landmark observations.

As OpenGV assumes calibrated cameras, landmark observations are always given in form of bearing vectors expressed inside a camera frame. In the present context, a *camera* therefore denotes a camera frame with a set of bearing vectors, all pointing from the origin to landmarks. The possibility of also describing multi-camera systems that cannot be represented by a single camera center is then given by the introduction of *viewpoints*, which allow OpenGV to transparently handle both central and non-central cameras. A viewpoint can contain an arbitrary number of central cameras each one having its own landmark observations (e.g. bearing vectors). Figure 1 illustrates a viewpoint with 3 cameras. A practical example of a viewpoint would be the set of images and related measurements captured by a fully-calibrated, rigid multi-camera rig with synchronized cameras. The viewpoint represents a single (multi-image) snapshot captured by multiple rigidly connected cameras at the same time and hence it can be parametrized by only one single pose (e.g., the viewpoint). Each camera has a known transformation to the viewpoint frame. In the central case the viewpoint simply contains a single camera with an identity transformation. The most general case—the generalized camera—can also be described by the viewpoint; each bearing vector would then have its own camera and related transformation. In fact, the bearing vectors along with the individual camera-to-viewpoint transformations are nothing but alternative representations of Plücker line vectors.

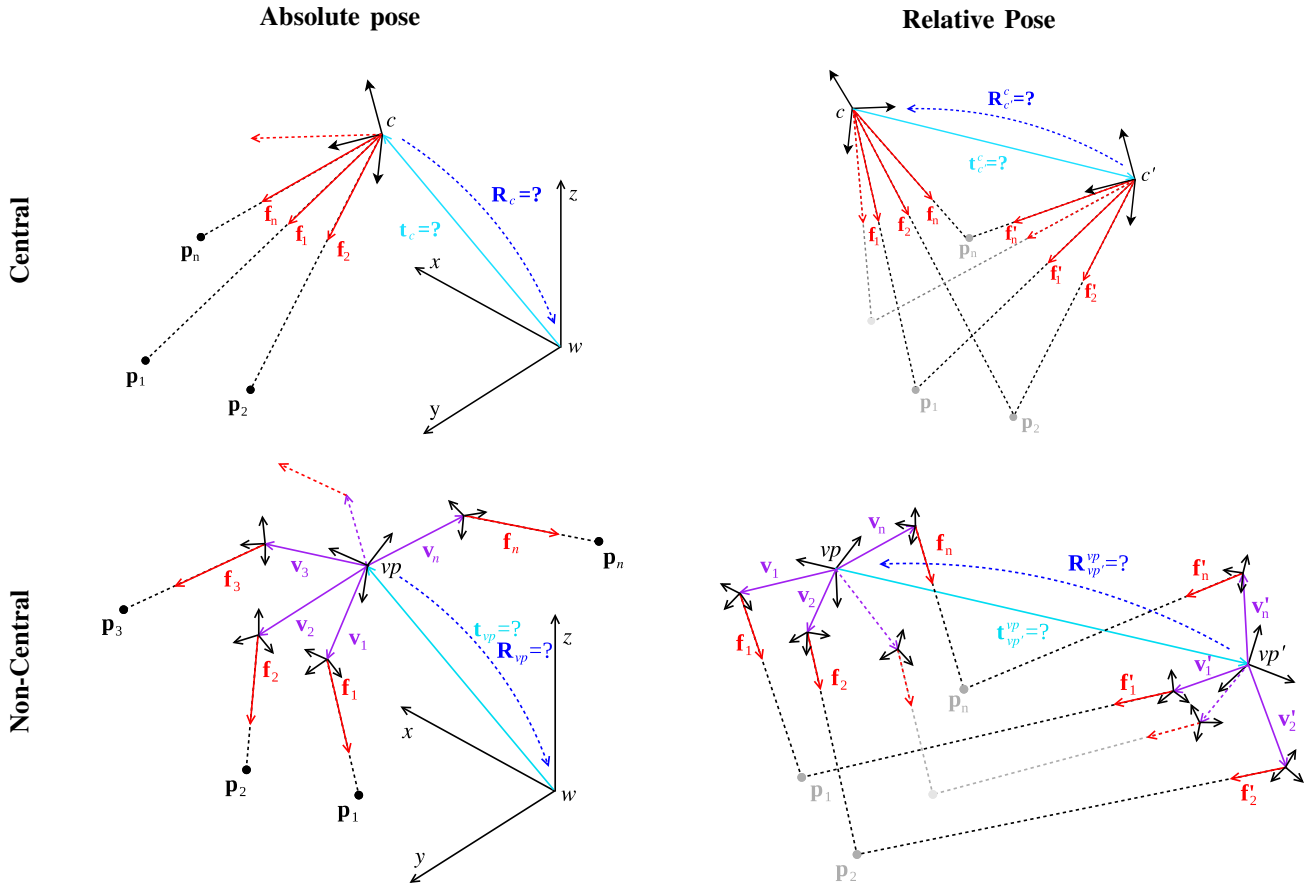

Fig. 2. An overview of the absolute and relative pose computation problems included in OpenGV. All problems can be tackled with either minimal or non-minimal closed-form solvers, as well as within non-linear optimization and robust estimation frameworks. Known variables are bearing vectors $\mathbf{f}_i$ in red, world points $\mathbf{p}_i$ in black, or transformations between the cameras in multi-camera scenarios (the position of cameras inside the viewpoint are given by $\mathbf{v}_i$). Each algorithm solves for either absolute or relative transformation parameters ($\mathbf{t}$ and $\mathbf{R}$). w represents the world frame, c a camera frame, and vp a viewpoint frame containing multiple cameras, as outlined in Figure 1.

As further outlined in Section III, OpenGV implements the "adapter pattern" [8] to define a single interface for accessing viewpoints and bearing vectors according to the above description. Users can leave their internal datastructures unchanged, and interact with the OpenGV algorithms through the implementation of a single adapter class.

B. Overview of the included problem solvers

The main purpose of OpenGV is to provide a set of algorithms to compute the pose of a viewpoint, meaning its position and orientation. OpenGV is able to handle both the absolute and the relative case. In the absolute pose situtation, OpenGV computes the pose of a viewpoint in the world reference frame from a number of correspondences between bearing vectors and 3D points expressed in a world frame (2D-3D correspondences). In the relative pose situation, OpenGV computes the pose of a viewpoint with respect to another viewpoint given a number of correspondences between bearing vectors in two different viewpoints (2D-2D correspondences). As summarized in Figure 2, the library provides central as well as non-central solutions to each of these problems, as well as minimal and non-minimal variants. The library therefore provides an unprecedented completeness and unification of calibrated geometric camera pose computation algorithms.

The central absolute pose problem consists of finding the pose of a viewpoint with a single camera given a number of 2D-3D correspondences between bearing vectors in the camera frame and points in the world frame. The problem seeks the transformation encoded by the position $\mathbf{t}_c$ of the camera seen from the world frame and the rotation $\mathbf{R}_c$ from the camera to the world frame. OpenGV currently hosts a minimal variant using only two points in case the rotation is known (P2P [9]), the minimal P3P solvers presented in [10] and [11], and the n-point solver presented in [12].

The non-central absolute pose problem consists of finding the pose of a viewpoint given a number of 2D-3D correspondences between bearing vectors in multiple camera frames and points in the world frame. The problem seeks the transformation encoded by the position $\mathbf{t}_{vp}$ of the viewpoint seen from the world frame, and the rotation $\mathbf{R}_{vp}$ from the viewpoint to the world frame. OpenGV includes the state-of-the-art minimal and non-minimal algorithms presented in [6], denoted gP3P and gPnP.

The central relative pose problem then consists of finding the pose of a viewpoint with a single camera with respect

to a different single camera viewpoint given a number of 2D-2D correspondences between bearing vectors expressed in the respective camera frames. The problem seeks the transformation encoded by the position $\mathbf{t}_{c'}^{c}$ of the second camera seen from the first one and the rotation $\mathbf{R}_{c'}^{c}$ from the second camera back to the first camera frame. OpenGV currently has its greatest variety of solutions in the central relative pose computation domain. The extent reaches from a minimal variant using only two points in case the rotation is known (2pt [9]), a two-point solver for the case of a pure rotation change, an n-point solver for relative rotation, the five-point algorithms presented in [13], [14], and [15]—the latter one solves for rotation immediately—the seven-point algorithm [16], and the eight-point algorithm [17], to the very recent n-point solution presented in [18], which shows for the first time the ability to solve relative pose as an eigenvalue-minimization problem. Note that [13], [14], the seven-point algorithm, as well as [17] can be applied for more than 5,7, or 8 points as well.

Finally, the non-central relative pose problem consists of finding the pose of a viewpoint with respect to a different viewpoint given a number of 2D-2D correspondences between bearing vectors in multiple camera frames. The problem seeks the transformation encoded by the position $\mathbf{t}_{vp'}^{vp}$ of the second viewpoint seen from the first one and the rotation $\mathbf{R}_{vp'}^{vp}$ from the second viewpoint back to the first viewpoint frame. There is currently only one method for solving this complicated problem in the library, which is the 17-point algorithm presented in [19]. It can be used with an arbitrary number of points, and has the advantage of remaining robust in certain degenerate situations that previous linear generalized relative pose algorithms ignored.

The set of available methods is completed by n-point non-linear optimization methods for both central and non-central absolute and relative pose computation, as well as a few triangulation and point-cloud alignment methods (e.g. [20]). All algorithms—except the triangulation methods—are also made available inside a generic sample consensus framework, which currently includes the most basic and popular RANSAC approach presented in [21].

Since the entire library operates in 3D, we also need a way to compute and threshold reprojection errors in 3D within RANSAC. OpenGV looks at the angle θ between the original bearing-vector $\mathbf{f}_{meas}$ and the reprojected one $\mathbf{f}_{repr}$. By adopting a certain threshold angle $\theta_{threshold}$, we constrain $\mathbf{f}_{repr}$ to lie within a cone of axis $\mathbf{f}_{meas}$ and of opening angle $\theta_{threshold}$, as illustrated in Figure 3. The threshold-angle $\theta_{threshold}$ can be easily approximated from classical reprojection error-thresholds ψ expressed in pixels by using the focal length l, and $\theta_{threshold} = \arctan\frac{\psi}{l}$. OpenGV uses an efficient implementation of this threshold, which we will describe here. The most efficient way to compute the angle between bearing vectors is given by taking the scalar product of $\mathbf{f}_{meas}$ and $\mathbf{f}_{repr}$, which equals to $\cos\theta$. Since this value is between -1 and 1, and we actually want an error that minimizes to 0, we use $\epsilon = 1 - \mathbf{f}_{meas}^{T}\mathbf{f}_{repr} = 1 - \cos\theta$ to express a reprojection error. The threshold error is therefore given by

$$\epsilon_{threshold} = 1 - \cos\theta_{threshold} = 1 - \cos(\arctan\frac{\psi}{l}). \quad (1)$$

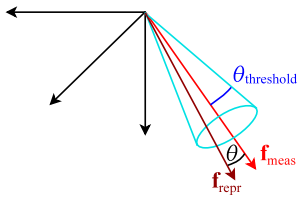

Fig. 3. OpenGV computes reprojection errors in 3D by considering the angle θ between the measured bearing vector $\mathbf{f}_{meas}$, and the reprojected one $\mathbf{f}_{repr}$.

C. RANSAC with multi-camera systems

OpenGV contains a special adapter that is able to hold multiple sets of bearing-vector correspondences originating from pairs of cameras originating from *different* viewpoints. The correspondences in this adapter are accessed via a composite-index built from a pair-index (referring to a specific pair of cameras), and a correspondence-index (referring to the correspondence within that camera-pair). This formulation of pairwise correspondences arises in many practical problems such as:

- non-central relative pose problems that involve two viewpoints originating from motion-estimation with a multi-camera rig. In the situation where the cameras are pointing in different directions, and where the motion between the viewpoints is not too big (a practically relevant case), the correspondences typically originate from the same camera in both viewpoints. We therefore can do a camera-wise grouping of the correspondences in the multi-camera system. Figure 4 illustrates this situation with four camera-pairs.
- central multi-viewpoint problems. By multi-viewpoint we understand here problems that involve more than two viewpoints. As indicated in Figure 5, a problem involving three viewpoints for instance allows to identify three camera-pairs as well. The number of camera-pairs in an

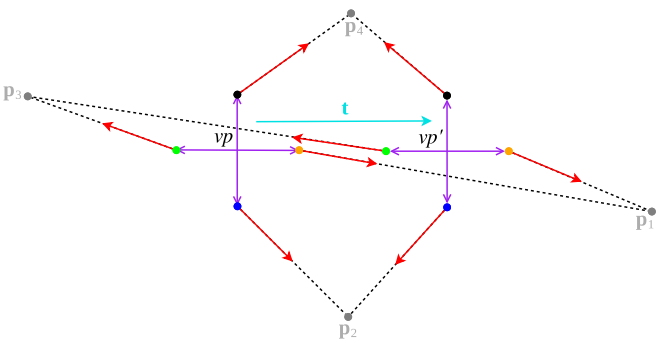

Fig. 4. Correspondences originating from 4 camera-pairs in two viewpoints (black, green, blue, and orange camera).

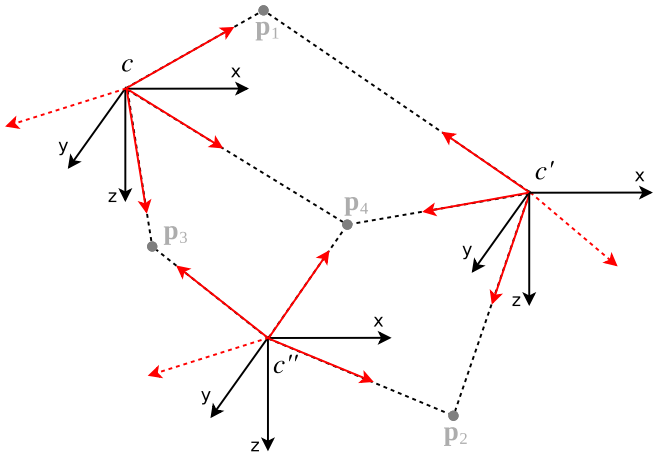

Fig. 5. Multi-viewpoint example with three central viewpoints. It gives rise to the three camera pairs (c, c'), (c', c''), and (c'', c). The first pair has a set of correspondences originating from points p_1 and p_4, the second one from p_2 and p_4, and the third one from p_3 and p_4.

n-view problem amounts at most to the combination of 2 out of n, meaning $n(n-1)/2$.

The benefit of keeping track of the pair-wise correspondences between cameras becomes clear when considering a random sample-consensus scheme. When computing the relative pose of a non-overlapping multi-camera system with small viewpoint-change, the most straightforward solution consists of simply applying RANSAC and drawing random samples from a single set containing all feature correspondences. The downside of this approach is that we might not necessarily sample a balanced number of points from each camera pair. This is especially bad in situations where the cameras are pointing into different directions, because we potentially lose the benefit of computing the pose with omni-directional landmark observations and—in the worst case—the solution collapses back to the single camera case. It is well known that the latter is actually an ill-conditioned problem, where the disambiguation between rotation and translation induced disparity in the image plane is far from trivial. It moreover renders the scale of the problem unobservable, and thus ultimately renders multi-camera solutions degenerate. By keeping track of the groups, we can easily ensure homogenous sampling of correspondences over the different camera-pairs. Concerning the multi viewpoint case, one could of course solve a central relative pose problem for each camera-pair individually. However, the simultaneous access of correspondence groups becomes useful here when attempting a joint solution of multiple relative poses by exploiting constraints over loops of Euclidean transformations, which should have identity rotation and zero translation.

OpenGV therefore includes a custom RANSAC mechanism and dedicated adapters that are able to operate on multiple sets of grouped correspondences. It ensures an evenly distributed sampling over all cameras. While this might at first appear as a minor implementational technicality, a deeper reflection quickly shows that there is also a potential trap behind this approach, namely in the situation where one of the images is actually more affected by outliers than another one. The multi-sample consensus mechanism is therefore preferably extended by an adaptive part that modifies the number of samples drawn from each set towards best increment in performance. This sampling strategy certainly shares similarities with the GroupSac approach [22], however differs in the fact that it maintains sampling within all groups rather than identifying dominant groups and restricting the sampling to them.

From a theoretical point of view, the approach is best explained by first recalling the original functionality of the RANSAC algorithm. Let p denote the desired probability of the algorithm chosing at least one set containing only inliers during one of all the iterations. Let n denote the number of samples needed for instantiating a hypothesis. Now let w denote the probability of one sample being an inlier. We have $w = \frac{\text{number of inlier points}}{\text{total number of points}} = \frac{y}{z}$. When drawing n samples, the probability of all points being inliers is w^n, and the probability that at least one of the points is an outlier is therefore given by $1 - w^n$. After k iterations, the probility that in each iteration at least one outlier has been sampled goes down to $(1 - w^n)^k$, and by definition this needs to be equal to $1 - p$. This finally leads to the well-known formula introduced by Fischler and Bolles [21] for computing k, the number of iterations required to satisfy a given p:

$$k = \frac{\log(1-p)}{\log(1-w^n)}. \tag{2}$$

The situation changes when considering samples drawn from multiple sets of correspondences. In this case, we define a per-set probability of drawing an inlier, which is $w_i = \frac{y_i}{z_i}$, with $i \in \{1, ..., m\}$ and m being the number of sets (e.g. camera-pairs). If we draw n_i samples from correspondence set i, the final law for computing the remaining number of iterations changes to

$$k = \frac{\log(1-p)}{\log(1-\prod_{i=1..m} w_i^{n_i})}. \tag{3}$$

The question now is, which strategy—depending on the n_i—should we choose? The equation arising from (3) considering pairwise correspondence sets and including the term

$$\prod_{i=1..m} w_i^{n_i} = \prod_{i=1..m} \left(\frac{y_i}{z_i}\right)^{n_i}, \tag{4}$$

or the vanilla version in (2) that includes the term

$$w^n = \left(\frac{\sum_{i=1..m} y_i}{\sum_{i=1..m} z_i}\right)^{\sum_{i=1..m} n_i}. \tag{5}$$

In general, we want this term to be maximal, which causes the required number of iterations to go down. Some simple testing reveals that—in case of homogeneous sampling—(5) is bigger or equal to (4), which suggests that the best way of sampling the points is completely random. It is also easy to see that (4) becomes bigger than (5) when pushing the sampling towards the set with highest w_i, in case the latter are not equal.

The problem is that this statistical analysis does not consider the fact that the accuracy of computing a pose with homogeneously distributed points is in general higher, which in turn increases the probability of finding the correct inliers, and thus also reduces the number of iterations.

While a further investigation of the sampling strategy would certainly go beyond the scope of this paper, we restrict ourselves here to an experimental proof of the benefit of homogeneous sampling in balanced situations. The results are provided in Section IV.

III. Implementation

OpenGV depends on the open-source linear algebra toolbox Eigen, as well as the standard library extension boost. Both are header-only libraries, and are readily available for most architectures. All methods in OpenGV accept a variable called an *adapter* as a function-call parameter. Algorithm calls in OpenGV are based on the adapter pattern [8]. Adapters hold the algorithm input (e.g. landmarks, bearing-vectors, poses, multi-camera configuration, and correspondences) in any user specific format (or references to the alike) and they are responsible for converting the data from this format into the format required by OpenGV. This happens either on demand, or—better—when constructing the adapter. The unified access outlined in Section II is then enforced by deriving all adapters from a base class that defines the methods used by the algorithms to access the data in OpenGV format. There are three adapter base-classes:

- `AbsoluteAdapterBase`, the base-class for adapters holding 2D-3D correspondences for absolute-pose methods,
- `RelativeAdapterBase`, the base-class for adapters holding 2D-2D correspondences for relative-pose methods, and
- `PointCloudAdapterBase`, the base-class for adapters holding 3D-3D correspondences for point-cloud alignment methods.

The adapter pattern gives the library great flexibility. Users only have to implement the above adapters for the specific data-format they are using, and can then access the full functionality of the library. Adapters for Matlab mex-arrays are already included, and further adapters such as for instance an adapter for OpenCV keypoint and match-types and a camera model are already planned. A mex-function that exposes all algorithms to Matlab is provided as well. The mentioned benchmark-tool is implemented in Matlab, and uses this interface in order to apply the compiled C++-functions to simulation data created in Matlab. The results are provided in the following section.

IV. Benchmark

The algorithms in OpenGV have been used in several projects and perform all reasonably well. Since multiple solutions to identical problems are comfortably made available, OpenGV provides an unprecendented playground for geometric vision algorithms to be compared against each other. In the following, we present the results generated by our automated benchmark tool. Anyone can reproduce and modify them by just installing the library and using the corresponding Matlab-files, which eases the choice of algorithm for future users. Note that the results generated by OpenGV algorithms are not guaranteed to be totally equivalent to the results presented in the original papers as code for many published algorithms is not available. The present results are based on our best-effort reimplementations of the algorithms included. As this is an open-source project, we invite others to contribute by improving a given algorithm in the library or extending the library with new algorithms.

Figure 6 shows the accuracy of most algorithms. It indicates the mean and median error of the computed rotation (norm of difference between the axis-angle representation of the ground truth and the recomputed rotation) as a function of measurement noise. The value for each noise-level and algorithm is averaged over 5000 random problems. Each problem consists of randomly chosen landmarks having a distance between 4 and 8 to the world frame origin, and a randomly chosen viewpoint with a maximum distance of 2 to the origin, as well as a random orientation. In the relative case this will define the second viewpoint, with the first one being kept at the world frame origin. In the non-central case, we additionally create a random multi-camera system with 4 cameras each one having a maximum distance of 0.5 to the viewpoint origin. Noise is added to the measurements by extracting the orthogonal plane of each bearing vector, normalizing the current noise-level using a focal length of 800, adding random noise in the orthogonal plane, and renormalizing the resulting perturbed bearing vector. The analyzed noise-levels reach from 0 to 5 pixels. All random numbers are drawn from uniform distributions. The analyzed solutions are restricted to those solving for the entire pose, because the accuracy of other algorithms (e.g. P2P, 2pt) depends additionally on the accuracy of the "known" part. If an algorithm has a solution multiplicity, the benchmarks currently do an automatic selection of the best solution based on comparison to groundtruth. As expected, non-linear iterative minimization of the reprojection error outperforms all other solutions in each type of problem.

Regarding central absolute pose, the direct approach P3P (Kneip) presented in [10] shows a minor improvement in robustness compared to traditional two-step approaches such as P3P (Gao) [11]. The EPnP algorithm [12] has superior robustness too, and returns valuable results even in case of using only 6 correspondences (minimum). In the non-central case, the GP3P algorithm performs very well, however can easily face a situation with almost parallel bearing vectors, which renders the position unobservable. The GPnP algorithm has poor robustness, especially when using only a few points. The behavior in the position estimate is a bit better (not shown), but it can still be concluded that—despite of the important recent progress in computational efficiency presented in [6]—the GPnP problem still leaves room for further investigations.

In the central relative pose situation, it can be observed that the most efficient algorithms—the 7pt [16] and the 8pt [17]

TABLE I

EXECUTION TIME OF ALL CAMERA POSE COMPUTATION METHODS

Algorithm	Type	Ref.	Execution time [ms]
P2P	Abs. central	[9]	0.0003
P3P (Kneip)	Abs. central	[10]	0.0033
P3P (Gao)	Abs. central	[11]	0.0115
EPnP (50 points)	Abs. central	[12]	0.1015
2-point	Rel. central	[9]	0.0002
5-point (Stewenius)	Rel. central	[13]	0.1045
5-point (Nister)	Rel. central	[14]	0.2479
5-point (Kneip)	Rel. central	[15]	0.9432
7-point	Rel. central	[16]	0.0261
8-point	Rel. central	[17]	0.0264
Eigensolver	Rel. central	[18]	0.0831
GP3P	Abs. noncentral	[6]	0.0837
GPnP (50 points)	Abs. noncentral	[6]	0.5270
17-point	Rel. noncentral	[19]	0.0912

solvers—are less resilient to noise than the minimal solvers, even if using more points. Since working on calibrated cameras—and thus "normalized" data—the algorithms currently dispose of any explicit data normalization techniques. 5pt (Stewenius) [13] is the most accurate non-iterative solver, and returns almost identical accuracy than the iterative eigensolver [18] when using the same number of points (not shown). The latter, however, returns a unique solution, and potentially outperforms the polynomial solvers in terms of computational efficiency. Note that not all iterations were taken into account. The sanity checks performed by the 5pt (Kneip) algorithm [15] are quite strict, and therefore–in some rare ill-posed experiments—might reject all found solutions. In the non-central relative pose case, the linear 17pt algorithm [19] again performs far from optimal under noise. It is also potentially affected by a singularity, namely when the rotation between the viewpoints approaches identity, which renders the scale unobservable. One of the next goals in OpenGV therefore is an implementation of the polynomial solver presented in [23].

Table I summarizes the execution times of all non-iterative algorithms. All experimental results in this paper have been generated on an Intel Core 2 Duo with 2.8 GHz. P3P (Kneip) out-performs P3P (Gao) because the latter is a two-step approach with an iterative SVD of a 3x3 matrix in the second step. This could be done in closed-form as well, however only at the cost of reduced accuracy. EPnP and GPnP as well as all other n-point methods in OpenGV have at most linear complexity in the number of points. The 7pt and the 8pt solvers are far more efficient than the polynomial solvers. The relatively high execution time of 5pt (Nister) [14] is mainly related to OpenGV's implementation of the Sturm root bracketing approach, which can certainly be improved. The only iterative algorithm in the list is the eigensolver, which is included because—compared to plain non-linear optimization—it converges comparably easy to the global minimum. The indicated number, however, is for a low number of iterations only. It is only a qualitative statement of the efficiency of the eigensolver, which has constant iteration time independently of the number of features, and therefore drastically outperforms nonlinear optimization.

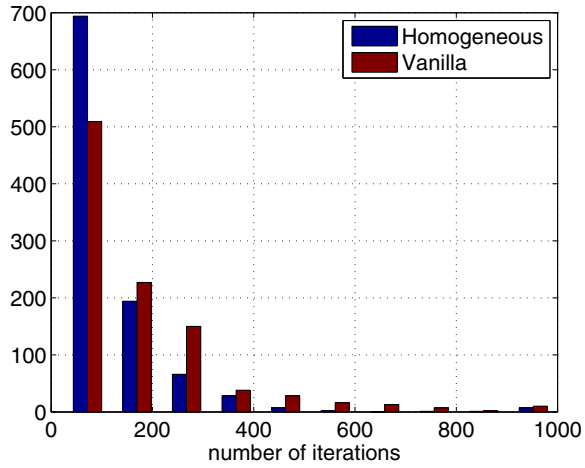

Fig. 7. Histogram of RANSAC-iterations in non-central relative pose computation depending on the sampling strategy. The experiment considers the performance of different sampling strategies in the presence of camera-wise grouping of correspondences. The *homogeneous* sampling strategy samples a balanced number of correspondences arising from each camera, whereas the *vanilla* strategy does completely random sampling.

Figure 7 finally shows the performance improvement resulting from homogeneous sampling in non-central relative pose computation. The experiment analyzes a balanced situation with similar outlier fractions in all images. 1000 random experiments are effectuated for each sampling strategy with 0.5 pix noise and an outlier fraction chosen uniformly between 10% and 20%. Although probabilitistics say that completely random sampling is at least as good if not better than homogeneous sampling, the results prove that the gain in accuracy by drawing a balanced number of correspondences from each camera practically leads to higher accuracy, higher probability of finding the correct inliers, and thus a reduced number of iterations. The mean number of RANSAC iterations is 113 for homogeneous sampling, and 164 for completely random sampling.

V. OUTLOOK

The present paper introduced OpenGV, a novel platform for calibrated real-time geometric vision algorithms. We provided a comprehensive overview of the goals behind the OpenGV-project, including use-cases, basic properties, as well as a comparative evaluation of the contained algorithms. OpenGV is the first library to unify state-of-the-art real-time solutions to the outlined types of problems in a single place, but there obviously remains space for extensions. Our future efforts consist of adding more algorithms to the library, thus extending the comparative benchmarks for the different problem classes. We also hope that we can convince people to add their algorithms to the library—no matter if they are old or new—which could ultimately make OpenGV *the* place where algorithms can be exposed to identical conditions, and thus easily and fairly compared against each other.

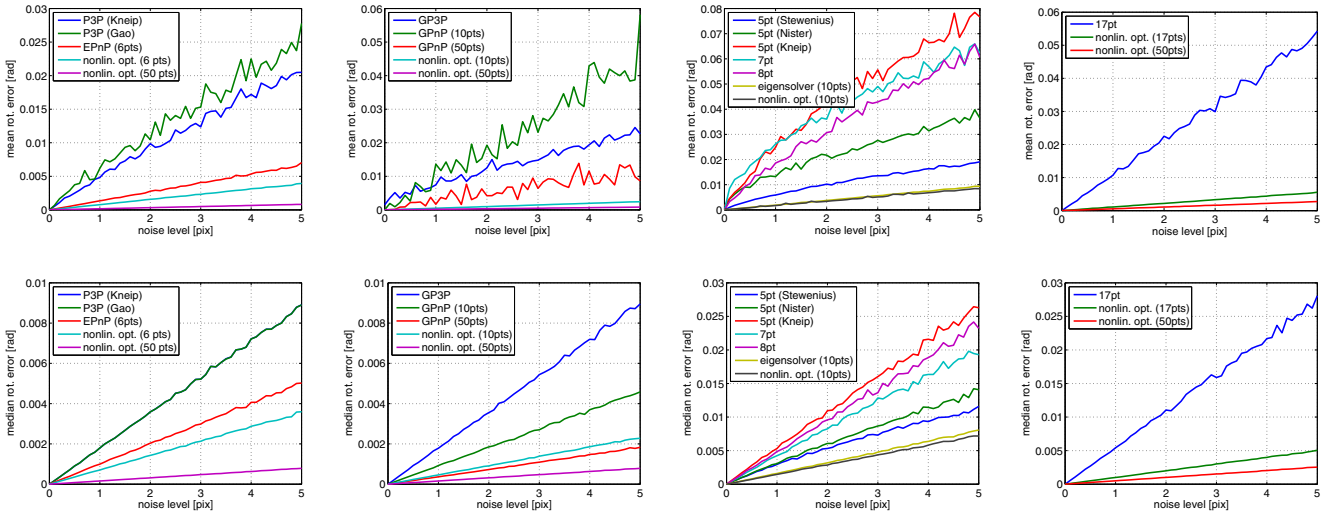

Fig. 6. Benchmark of multiple algorithms showing rotation error in function of measurement noise. The plot reflects the current status of OpenGV by including the main closed-form and iterative optimization methods. The value for each algorithm/noise level is averaged over 5000 random experiments. Noise levels are expressed in pixels, but internally applied on the bearing vectors' orthogonal plane using a focal length of 800 pixels.

ACKNOWLEDGMENT

The research leading to these results has received funding from ARC grants DP120103896 and DP130104567, and the EU project FP7-269916 (*Vcharge*).

REFERENCES

[1] S. Agarwal, N. Snavely, I. Simon, S.M. Seitz, and R. Szeliski. Building rome in a day. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 72 –79, 2009.

[2] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, Nara, Japan, 2007.

[3] A. Davison, D. Reid, D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(6):1052–1067, 2007.

[4] J. Kelly and G.S. Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *International Journal of Robotics Research (IJRR)*, 30(1):56–79, 2011.

[5] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[6] L. Kneip, P. Furgale, and R. Siegwart. Using Multi-Camera Systems in Robotics: Efficient Solutions to the NPnP Problem. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013.

[7] R. Pless. Using many cameras as one. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 587–593, Madison, WI, USA, 2003.

[8] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[9] L. Kneip, M. Chli, and R. Siegwart. Robust real-time visual odometry with a single camera and an IMU. In *Proceedings of the British Machine Vision Conference (BMVC)*, Dundee, Scotland, 2011.

[10] L. Kneip, D. Scaramuzza, and R. Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, USA, 2011.

[11] X.S. Gao, X.R. Hou, J. Tang, and H.F. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25(8):930–943, 2003.

[12] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate O(n) solution to the pnp problem. *International Journal of Computer Vision (IJCV)*, 81(2):578–589, 2009.

[13] H. Stewénius, C. Engels, and D. Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(4):284–294, 2006.

[14] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(6):756–777, 2004.

[15] L. Kneip, R. Siegwart, and M. Pollefeys. Finding the Exact Rotation Between Two Images Independently of the Translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Firenze, Italy, 2012.

[16] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, second edition, 2004.

[17] H.C. Longuet-Higgins. *Readings in computer vision: issues, problems, principles, and paradigms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[18] L. Kneip and S. Lynen. Direct Optimization of Frame-to-Frame Rotation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, Sydney, Australia, 2013.

[19] H. Li, R. Hartley, and J.-H. Kim. A Linear Approach to Motion Estimation using Generalized Camera Models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Anchorage, Alaska, USA, 2008.

[20] K.S. Arun, T.S. Huang, and S.D. Blostein. Least-Squares Fitting of Two 3-D Point Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 9(5):698–700, 1987.

[21] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[22] K. Ni, H. Jin, and F. Dellaert. GroupSAC: Efficient consensus in the presence of groupings. In *Proceedings of the International Conference on Computer Vision (ICCV)*, Kyoto, Japan, 2009.

[23] H. Stewénius and D. Nistér. Solutions to Minimal Generalized Relative Pose Problems. In *Workshop on Omnidirectional Vision (ICCV)*, Beijing, China, 2005.