Fast feature matching for detailed point cloud generation

Daniel Berjón, Rafael Pagés and Francisco Morán

Abstract—**Structure from motion is a very popular technique for obtaining three-dimensional point cloud-based reconstructions of objects from unorganised sets of images by analysing the correspondences between feature points detected in those images. However, the point clouds stemming from usual feature point extractors such as SIFT are frequently too sparse for reliable surface recovery. In this paper we show that alternate feature descriptors such as A-KAZE, which provide denser coverage of images, yield better results and more detailed point clouds. Unfortunately, the use of a dramatically increased number of points per image poses a computational challenge. We propose a technique based on epipolar geometry restrictions to significantly cut down on processing time and an efficient implementation thereof on a GPU.**

Index Terms—**Feature matching, GPU, 3D reconstruction, point cloud, implementation**

I. Introduction

The usual workflow for obtaining a point cloud reconstruction from an unstructured set of images using Structure-from-Motion (SfM) techniques begins with the extraction of feature/key points in all input images. Then, feature matches across all [relevant] pairs of images must be determined (i.e., which feature points in several different images depict the same 3D point on the original object), so that when all the geometric constraints of later stages of the reconstruction process are applied, the location of the original 3D points relative to the cameras that saw them can be inferred, as well as the poses of the cameras themselves. There are numerous algorithms for feature detection in the literature, each adapted to different needs; in the case of 3D reconstruction, it is desirable that the detected features be reasonably invariant to changes of scale, rotation and perspective distortion (none actually succeeds at the latter, hence the need for a dense image coverage of the subject to be reconstructed). It is also necessary that the key point associated with the detected feature is precisely located at the projection of the point that originated the image feature; in other words, feature detectors for this task are necessarily very local, whereas in other higher-level tasks other feature detectors may be more desirable. Arguably the most widely used feature detector for many tasks, including 3D reconstruction, is Scale-Invariant Feature Transform (SIFT) [1]. Indeed, despite being a patent-encumbered algorithm, it is the built-in option for the well-known state-of-the-art SfM frameworks Bundler [2] and VisualSfM [3]. SIFT features are robust, leading to reliable camera pose estimations, but not very numerous; therefore the point clouds thus obtained frequently present the shortcoming of being too sparse for realistic visualisation and, moreover, provide insufficient support for a subsequent surface reconstruction.

In this paper we explore the generation of detailed point clouds suitable for surface reconstruction using alternative feature point detectors and its associated computational challenges, including an efficient implementation, both in CPU and GPU, of a geometry-driven feature matcher to enable the computation of the aforementioned detailed point clouds in reasonable time.

II. Detailed point clouds

One possible solution to the sparsity of the SIFT-based point clouds may be using a patch-based point cloud densification algorithm based on photo-consistency constraints such as PMVS/CMVS [4] but, apart from being very costly, these algorithms still leave many holes in the scene and frequently introduce wrong patches in the scene due to photo-consistency between non-homologue areas (notably the sky). In addition, it may be preferable in many scenarios to obtain standard mesh-based models, which are easier to process (e.g., simplify for mobile devices) and render. In order to reliably obtain a mesh using techniques such as Poisson surface reconstruction [5], SIFT-based point clouds are usually not dense enough, so we have chosen to experiment with Accelerated KAZE (A-KAZE) [6], [7], a novel multi-scale feature detector that yields much denser coverage of the images with feature points. This in turn should result in a denser point cloud but, since all the feature points must satisfy the geometric restrictions of epipolar geometry during the SfM process rather than an arguably weaker photo-consistency check, the number of outliers should remain low.

As we expected, the point clouds resulting from the A-KAZE features were more detailed than those stemming from the SIFT features. However, as Fig. 1 shows, the A-KAZE feature detector largely failed to extract key points in the shaded areas of buildings, resulting in very poor reconstructions of those portions. Consequently, we decided to try image enhancing algorithms prior to the detection of feature points to

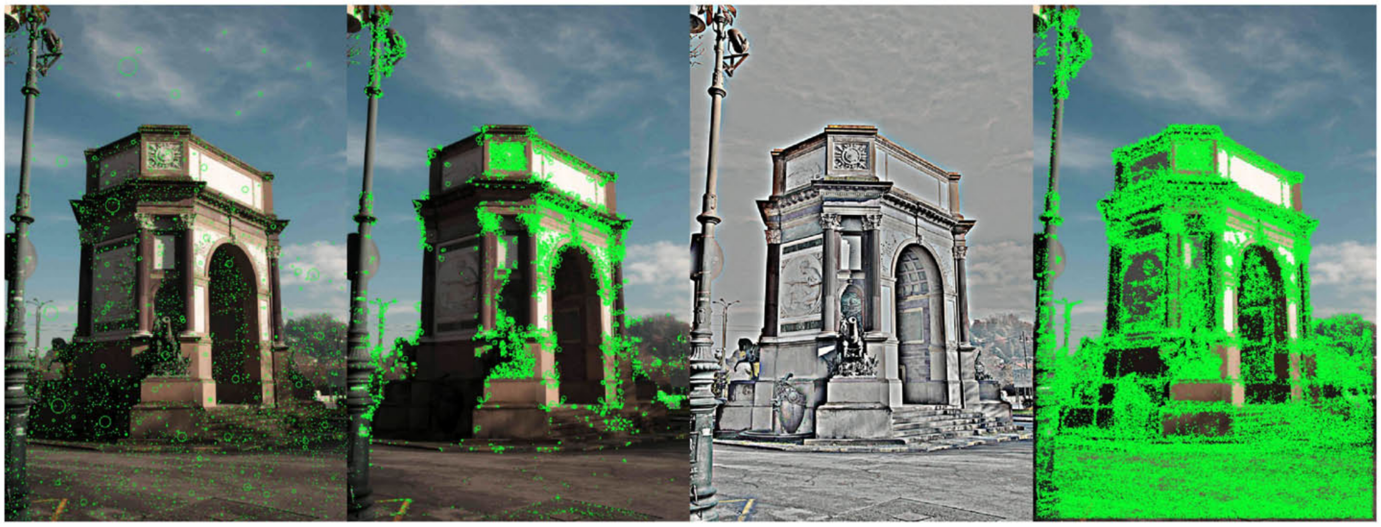

Fig. 1. From left to right: 6133 feature points detected by SIFT; 15025 feature points detected by A-KAZE; image enhanced using MSRCR; 138846 feature points detected by A-KAZE on the enhanced image, shown over the original. The A-KAZE detector failed to detect points in shaded areas of the original image but worked extremely well on the MSRCR-enhanced one.

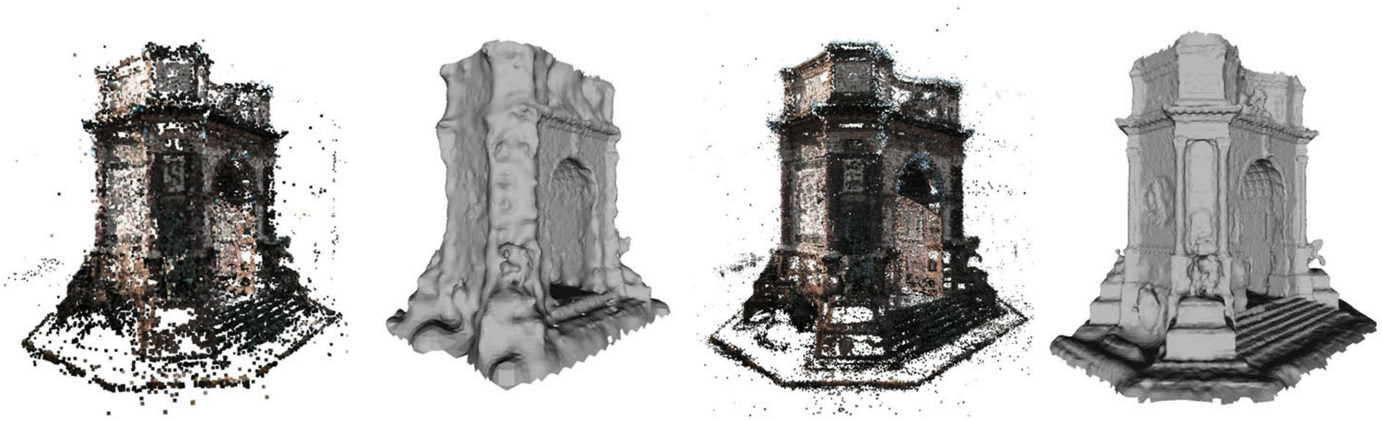

Fig. 2. Point-cloud- and mesh-based reconstructions of *Arco de Trionfo* at *Parco del Valentino,* Turin, Italy. The dataset has 333 pictures. Left: SIFT-based point cloud (88147 vertices) and its associated Poisson surface reconstruction; right: point cloud (1742247 vertices) and Poisson surface reconstruction obtained using A-KAZE feature extractor on MSRCR-enhanced images.

improve local contrast in all areas of the image. Firstly we tried contrast-limited adaptive histogram equalisation [8], a simple and inexpensive technique that improved results somehow, but still exhibited obvious differences between sunlit and shaded areas. Then we tried multi-scale retinex with colour restoration (MSRCR) [9], configured to boost dark areas rather than obtaining an aesthetically pleasing result and the detection improved markedly, as depicted in Fig. 1. The final results of such pipeline can be seen in Fig. 2, where the increase in detail and precision is apparent compared to the SIFT-based models.

III. Reduction of computational cost

In addition to balancing the results between sunlit and shaded areas, the enhancement of source images produced the side effect of dramatically increasing the number of detected feature points. As a rough approximation, we have found photos of buildings to yield around 5K feature points using SIFT and around 25K-30K feature points using A-KAZE on original images (the exact figures vary depending on the texture and shape of the subject), and over 125K feature points using A-KAZE on MSRCR-enhanced images (no such effect was observed using SIFT). While this is most welcome in terms of results, because the resulting point clouds are more detailed, it constitutes a very significant increase in computational demands because exhaustive pairwise matching between two sets of feature points takes quadratic time on the number of feature points. In addition, the number of possible pairs of images in a set also grows quadratically on the number of images. Therefore, we have focussed on performance optimisation to alleviate the problem and cut down on processing time, both trying to reduce the number of pairs of images to be matched and the per-pair computational cost.

A. Prioritisation of pairs of images

In the absence of any previous information, feature points should be extracted on every input image and matches sought between all possible pairs of images, resulting in $O(N^2)$, N being the number of images. However, in practice many pairs of images will have no overlap (i.e., they do not depict the same portion of the object) and therefore not only feature matches between them will be irrelevant but also their computation time will be wasted, so the aim would be not to compute them. This problem is also present when using SIFT, but it is much less severe because each image has significantly fewer points. Even so, there are proposals in the literature such as vocabulary trees [10] or preemptive feature matching [3] to aggressively prune the number of image pairs to compute.

SfM algorithms jointly estimate camera poses and positions of points in the cloud to minimise the global error. However, if camera poses are known, estimating just the position of points in the cloud from their projections (i.e., feature points) is a considerably easier problem. The key observation here is that, even though SIFT-based point clouds are too sparse to mesh, camera poses are well estimated. Therefore, it makes sense to first obtain, relatively cheaply, a sparse point cloud using any SIFT-based SfM framework to get camera poses and only then compute high-density feature points with A-KAZE to obtain the detailed cloud using the known camera poses.

Additionally, we can indirectly *piggyback* onto whatever strategy the SfM engine has used to prune the image pairs to cut down on the number of image pairs we will have to compute. The results of a SfM module typically include the camera parameters, the 3D points and, crucially, their projections onto the appropriate source images or, more precisely, the 2D feature points that originated each 3D point, so we can use it as a statistical sample of sorts to decide what image pairs are relevant and should be computed with A-KAZE to contribute to the detailed point cloud.

Thus, we can estimate the amount of overlap between two images simply by counting the number of points that project onto those two images. If a given pair of images shares no 3D points, we can safely discard it because they will have no or very little overlap. For each image, we count the number of points it shares with each of the other images and sort them in decreasing order, so that we can compute matches against the images with the most significant overlap first and possibly discard pairs with negligible overlap.

Finally, pairwise matches can be computed in either depth-first order (i.e., for each of the images we compute matches with all its significant pairs) or breadth-first order (i.e., we compute the most significant pairs for every image first, then the second most significant and so on). Best results are obtained if all significant pairs are computed, but if the user cannot afford to wait, breadth-first order gives the best results for an allotted time. If the users are not satisfied with the results, they can resume the computation where it was left off to refine the cloud.

B. Reduction of per-pair computational cost

Once a pair of images has been selected, the standard procedure to compute feature matches between both images is to exhaustively try to find the best (significant) match between all feature points from image I_1 and all feature points from image I_2, which results in quadratic complexity, then estimate a geometric model to reject outliers (estimation of the fundamental matrix F [11] with RANSAC [12]). The bulk of the procedure (pairwise feature matching) is completely regular and hence amenable to efficient parallelisation, which we have implemented both on CPU and GPU. However, its inherent time-complexity is excessive, therefore algorithmic optimisation is required in order to further reduce processing time.

As previously stated, while point clouds derived from SIFT feature detection and matching are too sparse for reliable meshing, camera pose estimation and intrinsic parameters are essentially correct, which means that we can obtain [13] the fundamental matrix F, which enables us to perform smarter matching as we will see next, for any pair of cameras:

$$F = [P_2 \mathbf{C}_1]_\times P_2 P_1^+, \tag{1}$$

where P_1 and P_2 are the projection matrices of the cameras, P_1^+ is the Moore-Penrose pseudo-inverse of P_1 and $\mathbf{C}_1$ is the centre of the first camera.

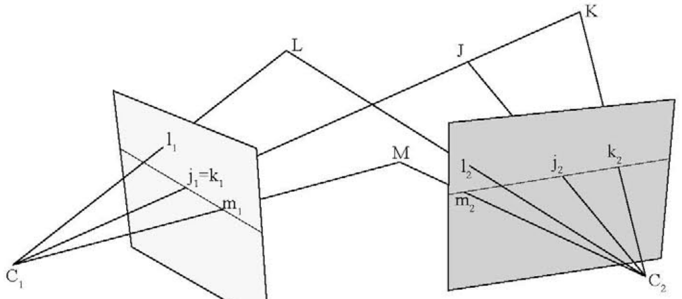

Fig. 3. Epipolar geometry helps rule out incompatible projections. The idea is that only projections that lie on the plane determined by the centres of both cameras and the projection whose match is to be determined are geometrically possible. Thus, j_1 is geometrically compatible with j_2, k_2 and m_2, but never with l_2.

Since we now have the F matrix, we do not need to compute all potential point-wise matches, we can first check whether the locations of the candidate feature points are viable, as illustrated in Fig. 3, and only if they are we proceed to compute the distance between their descriptors. Of course, the situation pictured in Fig. 3 is ideal but in real cases 3D points do not necessarily project exactly onto their corresponding 2D features. Specifically, for each pair of potential matching points $\mathbf{x}_1$ and $\mathbf{x}_2$ we compute their epipolar distances

$$d_2 = d(\mathbf{x}_2, F\mathbf{x}_1) = \mathbf{x}_2^\top F \mathbf{x}_1 \left(\lambda_1^2 + \mu_1^2\right)^{-1/2} \tag{2}$$

$$d_1 = d(\mathbf{x}_1, F^\top \mathbf{x}_2) = \mathbf{x}_1^\top F \mathbf{x}_2 \left(\lambda_2^2 + \mu_2^2\right)^{-1/2}, \tag{3}$$

where $F\mathbf{x}_1 = (\lambda_1, \mu_1, \nu_1)$ and $F^\top \mathbf{x}_2 = (\lambda_2, \mu_2, \nu_2)$. Note that in general $d(\mathbf{x}_2, F\mathbf{x}_1) \neq d(\mathbf{x}_1, F^\top \mathbf{x}_2)$, so we consider plausible a match between pairs of points for which both $|d_1|$ and $|d_2|$, pictured in Fig. 4, are sufficiently small.

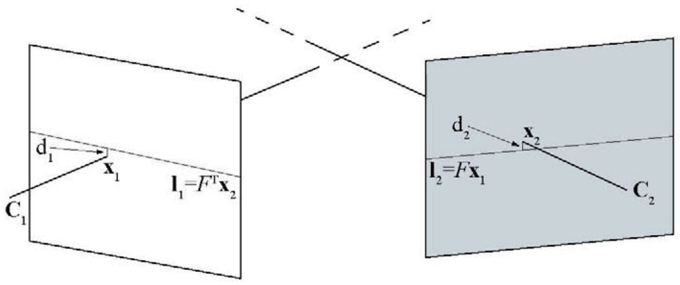

Fig. 4. Epipolar distances between (potentially matching) points $\mathbf{x}_1$ and $\mathbf{x}_2$.

This approach results in more than a tenfold speed increase in the CPU, despite the fact that we are now factoring in the locations of the points in the point-wise comparison. Strictly speaking, this procedure still has quadratic time complexity, but we have exchanged (for most points) a data- and compute-intensive computation (comparing descriptors) for a lighter one (vector product to compute epipolar distance). As a bonus, we do not need to apply RANSAC to estimate F and filter outliers afterwards because we have eliminated them from the beginning.

In the CPU, this can be parallelised reasonably well by mapping one execution thread to each feature point in image I_1 and iterating through all the points in image I_2 so that there are no possible race conditions. Unfortunately, such naive mapping does not work so well on a GPU because execution threads are scheduled together in groups (also called *warps*, typically 32 threads) that must execute the same instructions synchronously and, for maximum efficiency, reuse or coalesce memory transactions. In our particular case, this means that all the threads in the same warp iterate synchronously over the points in image I_2 but each thread finds different points to be compatible with its own reference point from image I_1. Since all threads in a warp must execute synchronously, they must all wait for the thread that does want to compare its reference descriptor to the one all threads are currently iterating over, thereby decreasing the parallelism drastically.

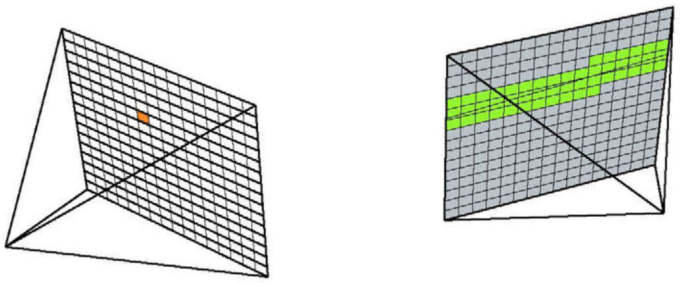

Fig. 5. All feature points in the orange tile can only find their matches roughly around the corresponding epipolar line in the right image, i.e., within the green tiles. All feature points outside the green tiles are ignored altogether.

The solution is to make use of the fact that in a GPU we have some degree of control in thread scheduling. Thus, instead of using one flat list of features per image, we divide the images into tiles and make one list of features per tile (or, equivalently, reorder the list of feature points), effectively grouping neighbouring feature points which will roughly satisfy geometric constraints in a per-tile basis instead of per-point, as illustrated in Fig. 5. This results in two benefits: all the feature points located in tiles that do not satisfy geometric constraints can be disregarded in a single operation and all the feature points located in tiles that do satisfy geometric constraints are relevant candidates to all the feature points mapped to the same thread block, enabling shared reading and making much more efficient use of memory bandwidth. Moreover, since the location of each block is implicit, we do not need to even read the locations of the feature points, just compare the descriptors of pairs of points contained into compatible blocks.

IV. Experiments

To illustrate the benefits of the proposed method, we have performed feature matching on several datasets using different strategies and measured the execution times. All tests have been done on a machine equipped with an Intel Core i7-4790 processor, which has 4 processing cores (8 virtual cores via hyper-threading) clocked at 3.6 GHz, 32 GiB of memory and a NVIDIA GeForce GTX 980 Ti GPU, with 6 GiB of onboard memory and a peak processing power of 5632 GFLOPs. The proposed matching algorithms have been programmed in C++11, using OpenMP to exploit the multiple cores in the CPU, and CUDA 7.5 to access the GPU.

Fig. 6 shows a comparison of the point clouds obtained using standard SIFT-based SfM (specifically, we have employed VisualSfM [3]) and using, as proposed, the A-KAZE features extracted from pictures enhanced using MSRCR. This combination clearly yields more detailed clouds, while the necessary processing times remain reasonable, as reported in Table I, using the proposed tile-based matching strategy on a GPU. Note that all matching strategies have been applied onto the same set of pairs of images, substantially pruned with respect to the set of all possible pairs, as explained in section III-A.

We can observe that the processing times using the brute force matching strategy in the GPU or the epipolar-distance-based matching strategy in the CPU are roughly proportional to the number of computed pairs divided by the square of the mean number of points per image in each dataset. However, this proportionality is not so clear in the tile-based matching strategy. The main reason for this behaviour is that every block/tile must have the same number of threads, each mapped to one feature point in the reference image, and the minimum thread allocation unit is one warp (32 threads). Since it is very unlikely that every tile contains the same number of feature points, and even more unlikely that all tiles contain a number of points that is a multiple of the warp, a fraction of the threads allocated to each block is wasted. Thus, the speedup compared to the brute force matching strategy varies depending on the dataset. The other strategies, on the other hand, achieve nearly

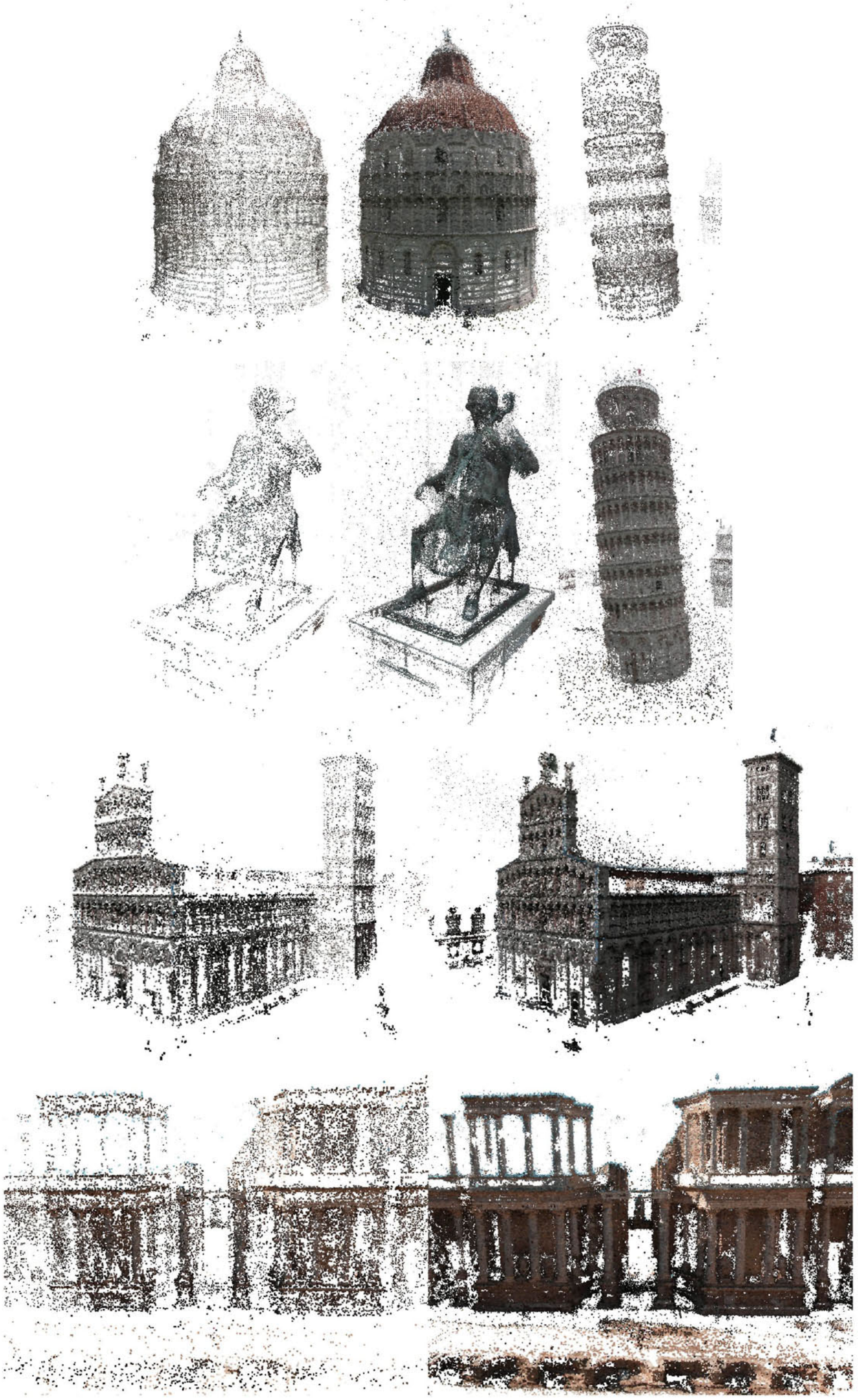

Fig. 6. Point clouds generated using SIFT and the proposed MSRCR + A-KAZE method. From top to bottom: Baptistery, Pisa, IT (SIFT: 44367 points; A-KAZE: 747899 points); Leaning Tower, Pisa, IT (SIFT: 18100 points; A-KAZE: 393436 points); Boccherini statue, Lucca, IT (SIFT: 12307 points; A-KAZE: 210846 points); San Michele in Borgo church, Lucca, IT (SIFT, 33117 points; A-KAZE: 590465 points); Roman Theatre, Mérida, ES (SIFT: 47684 points, A-KAZE: 902346 points).

TABLE I
TOTAL PROCESSING TIMES FOR FEATURE MATCHING COMPUTATION

Dataset	# pics.	Avg. points per pic.	# pairs	Brute force, multi-core CPU	Epipolar distance-driven, multi-core CPU	Brute force, GPU	Tile-based, GPU
Baptistery, Pisa, IT	93	167769	1263	188453 s	14380 s	5820 s	288 s
Leaning Tower, Pisa, IT	57	127304	596	N/A	3895 s	1676 s	230 s
Boccherini statue, Lucca, IT	32	108657	131	N/A	638 s	273 s	56 s
S. Michele church, Lucca, IT	75	144149	1285	N/A	10634 s	4478 s	446 s
Roman Theatre, Mérida, ES	150	187799	6543	N/A	89411 s	43835 s	2906 s

full utilisation of the processing cores, which explains their more regular processing time.

V. Conclusions

We have presented a strategy for obtaining detailed point clouds from unstructured sets of pictures. Instead of the commonplace SIFT features, we have found the alternative key point detector A-KAZE to provide much denser coverage of pictures, especially when operating on images previously enhanced with a multi-scale retinex algorithm. Consequently, feeding these feature points to a standard Structure from Motion pipeline results in much more detailed point clouds.

Unfortunately, matching such numerous feature points across the set of input images constitutes a significant workload. Therefore, we propose a strategy for saving computing resources: a first coarse reconstruction is leveraged to decide which image pairs should be matched and to obtain the camera calibration matrices, which enable us to perform smarter (and shorter) pairwise feature matching by applying restrictions based on epipolar geometry. We have also proposed a modified version of this geometry-driven matching strategy especially suitable for GPU implementation.

To prove the validity of our proposal, we have conducted several experiments using each of the proposed matching strategies over several datasets, obtaining clearly improved results compared to usual point cloud reconstructions and reasonable processing times.

Acknowledgment

This work was supported in part by the Ministerio de Economía y Competitividad of the Spanish Government under grant and TEC2013-48453 (project MR-UHDTV) and by the European Commission under grant 610691 (BRIDGET).

References

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[2] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3D," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 835–846, Jul. 2006.

[3] C. Wu, "Towards linear-time incremental structure from motion," in *International Conference on 3D Vision*, June 2013, pp. 127–134.

[4] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multiview stereopsis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 8, pp. 1362–1376, Aug 2010.

[5] M. Kazhdan and H. Hoppe, "Screened Poisson surface reconstruction," *ACM Transactions on Graphics*, vol. 32, no. 3, pp. 29:1–29:13, Jul. 2013.

[6] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, "KAZE features," in *12th European Conference on Computer Vision Proceedings, Part VI*, ser. Lecture Notes in Computer Science, vol. 7577. Springer, October 2012, pp. 214–227.

[7] P. F. Alcantarilla, J. Nuevo, and A. Bartoli, "Fast explicit diffusion for accelerated features in nonlinear scale spaces," in *British Machine Vision Conference*, 2013.

[8] S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. ter Haar Romeny, J. B. Zimmerman, and K. Zuiderveld, "Adaptive histogram equalization and its variations," *Computer Vision, Graphics, and Image Processing*, vol. 39, no. 3, pp. 355–368, 1987.

[9] D. J. Jobson, Z. Rahman, and G. A. Woodell, "A multiscale retinex for bridging the gap between color images and the human observation of scenes," *IEEE Transactions on Image Processing*, vol. 6, no. 7, pp. 965–976, Jul 1997.

[10] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski, "Building rome in a day," in *IEEE International Conference on Computer Vision*, Sept 2009, pp. 72–79.

[11] Q.-T. Luong and O. D. Faugeras, "The fundamental matrix: Theory, algorithms, and stability analysis," *International Journal of Computer Vision*, vol. 17, no. 1, pp. 43–75, 1996.

[12] 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*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[13] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.