

OpenREC: A Framework for 3D Reconstruction of Models from Photographs

G. Arroyo[†] and D. Martín[‡]

Department of Software Engineering
University of Granada, Spain

Abstract

In this paper we introduce openREC to the scientific community, an extendable framework for reconstruction of 3D models from photographs. This system provides a framework designed for archaeologist, art restorers, architects, and other professionals, hiding not relevant details from the underlying complex commands that are only suitable for expert computer scientists. There is a huge amount of free and commercial tools for photogrammetry available, but none of them are really suitable for experts in other fields different from computer science. Free tools are not free of issues due to their huge complexity – most of them are commands based –, while commercial software are under strict close-source models that hide important details of their underlying algorithms, avoiding any tuning of their parameters. This article presents the core of openREC: a complex task manager that is able to chain different sub-tasks in a transparent way for the user, and that is able to deal with remote tasks through network without the need of any specific servers.

CCS Concepts

• **Computing methodologies** → **Computational photography**; Mesh models; Point-based models;

1. Introduction

3D reconstruction from photographs is an area of growing interest in computer graphics and computer vision due to its great versatility. However, the software used for this reconstruction is complicated for archaeologists, art restorers, architects, and most of other non computer scientist professionals. Frequently this software is operated by commands in a console, or it is a closed commercial tool which hides the underlying algorithms and their parameters. When reconstructing a 3D model it is important, not only to simplify the user interface for standard users, but also to offer the possibility of customising all the options of the algorithms to better adjust the final result.

Our contribution consists in collecting a great diversity of free tools, generating a running environment with an easy to use user interface. This is possible by means of our task manager, which transparently schedules all the tasks in the system and their dependencies, allowing cloud computing without the necessity of specific servers. We provide this software both for development and as a teaching tool.

2. Related Works

Photogrammetry techniques have been a complement for cultural heritage professionals during the recent years. The advantages derived from developing low-cost methodologies for the conservation of cultural heritage and archeology sites compared to high-cost alternatives such as 3D scanning have been proven successful during the last years [ÖGG09, LFG⁺10, McC14]. In the next section we review different photogrammetry software for that purpose.

2.1. Available Software

There are several tools commercially available that allow us to 3D reconstruct models from photographs.

Probably one of the most famous tools is Recap^{®1}. Recap has gained some popularity for being one of the best 3D reconstruction software on the market today, producing high resolution and quality models. One of the problems with this tool is that the user interface is not easy to handle. In addition, the underlying algorithms remain hidden, although there is some adjustment capacity in generic parameters.

Agisoft Photoscan^{®2} is the direct competitor of Recap[®]. It

[†] arroyo@ugr.es
[‡] dmartin@ugr.es

¹ <http://www.autodesk.com/products/recap/overview>
² <http://www.agisoft.com>

is also a very popular software because it was the first to use Bundler [WACS11]. Again, this software hides all the details of the algorithms it uses.

Remake[®], formerly called Memento[®], successor of Autodesk ImageModeler[®]³, is a software that focuses on simplicity for the user. This tool focuses on simplicity for the user rather than the quality of the reconstruction.

Acute3D[®]⁴ allows, unlike the previous ones, to distribute the work between several servers. It is a software intended for cloud computing rather than a personal computer. Though like the above we lack knowledge about the details of the underlying algorithms.

Neitra3D[®]⁵ is freeware software, that means that as long as we do not have commercial purposes it can be freely used. The interface is extremely simple, and the quality of the models is not great, but is suitable for beginner students or academics.

Pix4D[®]⁶ is software that moves away from the generic photo-reconstruction software. One of the strengths of Pix4D is that it is designed for reconstructing 3D models from photographs taken by autonomous drones (UAVs), which allows not only to automate the entire capture process, but also to track the drones even from mobile devices.

An common issue with all these commercial software is that their underlying algorithms have been deliberately obscured, and that makes it difficult to adjust their parameters in the process of reconstruction. This avoid the user to experiment and to get better results than the default configuration, contrarily of what occurs with other 3D scanner software, where tuning is a crucial factor in the process.

VisualSfM⁷ is a free software that makes use of research software (described in Section 2.2) for 3D reconstruction from photographs, but it takes years without maintenance. Unlike the previous ones, the algorithms are known, but still their parameters can not be adjusted from the user interface.

Its strong point is that the interface calls to other software, so the user can know at any moment the software that is being used below. The problem is that since VisualSfM is not open software, we do not have the code that would allow us to update the algorithms and to expand its options, in addition some of the links to the underlying software are not updated to the new software, making difficult to run it in recent hardware or having a functional updated version. Like the above mentioned tools, this tool predefines the parameters, so users do not have any control of the underlying algorithms.

VisualSfM can only work with Bundler and SiftGPU [SFPG06] or SIFT as core algorithms, which makes its application limited especially for academics of other areas different from computer science.

2.2. Open Research Software

In addition to commercial and freeware software discussed in Section 2.1, there are many powerful utilities for the reconstruction of photographs, most of them based on console commands.

One of the more classic implementations of the SfM algorithm is Bundler⁸, which started as part of the Photo Tourism project [SSS06, SSS08], and which can be combined with feature detectors such as SIFT (the original software) or SiftGPU⁹ [SFPG06]. Although one of the problems is that the base algorithm remains the same as it was about four years ago – basically the transcription of the algorithm was published in 2008 – and it has not been even updated for at least one year.

Another post-Bundler implementation, called OpenMVG [MMMO, MMM12] has relegated Bundler to oblivion, rendering VisualSfM obsolete in terms of the quality of the algorithms used. OpenMVG implements all the necessary steps to obtain a scattered cloud, including feature detection. Unlike VisualSfM – which can only work with SIFT – OpenMVG implements both SIFT and SURF [BTVG06] algorithms as feature detectors. Same as other free open software, openMVG can only be used in a command console, making very difficult its use by any standard user.

Regard3D¹⁰ is a remarkable attempt to integrate openMVG and PVMS under a simple user interface. But it lacks the possibility of cloud computing.

Regard3D, OpenMVG and VisualSfM are able to be integrated with PMVS or CMVS¹¹, but VisualSfM cannot integrate OpenMVS¹², a new version that improves the previous algorithm. Both tools allow to build the dense cloud from the reconstruction generated by SfM in the previous step.

This is the reason because it seems necessary a tool that provides the mechanisms to simplify the task of reconstruction. The following sections describe the architecture of OpenREC, as an intended replacement of VisualSfM.

3. Architecture of OpenREC

The user interface of OpenREC divides the window in three parts, as shown in Figure 1. On the left side the user interface groups all tasks in four different high level operations:

1. Pictures: In this stage photographs are treated, so features [Low04] are detected, matching [MM12] is done, and paths to the files are stored for the next stage. In this stage, the parameters of the cameras are treated by OpenMVG automatically based on the meta information of the photographs.
2. 3D Cameras: In this stage one of the algorithms based on SfM is computed, and the points cloud is coloured from the photographs. So, after this stage, a 3D model is generated.

³ <http://remake.autodesk.com>

⁴ <http://www.acute3d.com>

⁵ <http://triayaam.com/>

⁶ <https://pix4d.com/>

⁷ <http://ccwu.me/vsfm/>

⁸ <http://www.cs.cornell.edu/~snaveily/bundler>

⁹ <https://github.com/pitzer/SiftGPU>

¹⁰ <http://www.regard3d.org>

¹¹ <https://github.com/pmoulon/CMVS-PMVS>

¹² <https://github.com/cdcseacave/openMVS>

3. Final Clouds: In this stage, the information is converted to the software that reconstruct the dense cloud, and the dense cloud is computed.
4. Final Meshes: In this final stage, the dense cloud is converted to a surface [KBH06, BMR*99, GKS00, CL96].

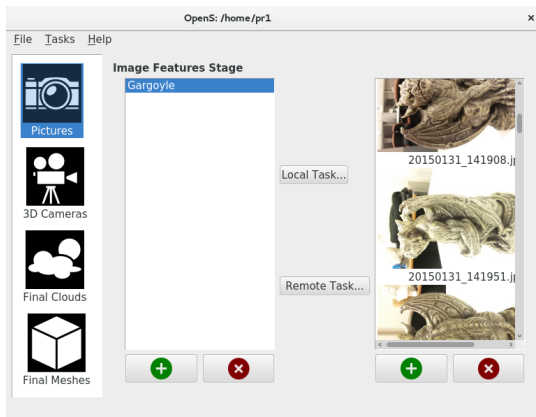


Figure 1: Main window of OpenREC: a) left side allows the user to select the stage of the reconstruction process; b) the middle section list the works available in the current stage; c) list the files attached to the current stage.

In order to launch a local work we simply press the button, then a window with all the parameters involved in all the underlying algorithms is displayed. Parameters are set to default in all cases, which allows a correct reconstruction without modifying any value. These parameters can be changed by means of the user interface, as shown in Figure 2.

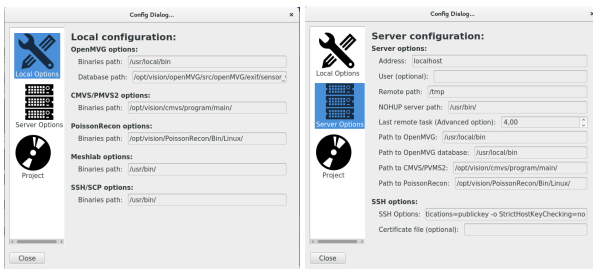


Figure 2: OpenREC defines several options for local and remote tasks, showing all the underlying algorithms and their paths in the installation.

Figure 3 shows a simple diagram with the more relevant parts of the architecture of OpenREC. The core of the system is the task manager, which executes all tasks in the correct order to reconstruct the 3D model. A task is a complex structure that is able to run local and remote commands when needed, each task has its own state and a list of dependencies. In the next sub-section tasks are further explained.

All information related to the operating systems and temporal data is stored the Cache Data, which is a dictionary type structure

that stores concepts such as extensions in files, or the way directories are split depending on the operating systems. OpenREC has been designed to be multi-platform, although currently it is not possible due to the remote capabilities of the software, as it is explained in Section 3.2.

All data of the project such as paths of images, are stored in the Project Data structure, while global information such as paths to the configuration file are stored in the Global Element. All these structures can be stored as JSON files in the system.

A wrapper is used as intermediate layer to transform the abstract process of the task in a real command in the operating system.

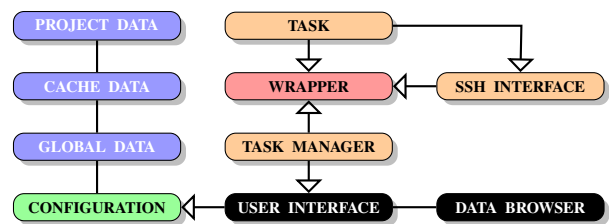


Figure 3: Diagram of the architecture of OpenREC.

Some of the outputs of the underlying algorithms provide information about the reconstruction in form of graphs, or matrices. Moreover, our intention is to add integrated help about the parameters and default configurations in the application. In order to visualize all this information, we have implemented a web browser – also called data browser – which is able to render SVG graphs, raster images and plain HTML.

In order to display more complex information, such as 3D models, Meshlab [CCC*08]¹³ is called by means of the wrapper.

3.1. The Task Manager

The task manager is the core of OpenREC. It maintains a list of active tasks, an active task is a task that has been started in the system by petition of the user. It maintains a list of active tasks: tasks that have been started in the system by petition of the user. A task is a complex structure that stores the necessary information for a process of the operating system to run, and to communicate with that process. The task manager stores not only the tasks that have been launched, but also the dependencies that every task must satisfy in order to be executed.

This way, the task stores its state that can be one of these:

- Running: the task has been launched and meet all the necessary requirements to be executed. When a task passes from the state Idle to the state Running, this task is inserted in the list of the task manager.
- Paused: the task has been launched but it does not meet the requirements to be executed. So the task is inserted in the list but is waiting until the resources are satisfied.

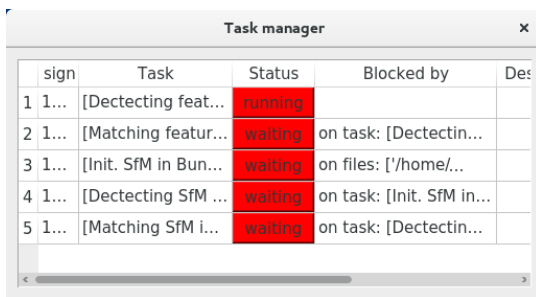
¹³ <http://www.meshlab.net/>

- Aborted: the task has been manually aborted by the user or the operating system has killed it. In this case, the task and all its dependent tasks are aborted and removed from the list of the task manager.
- Done: the task finished successfully, so it can be safely removed of the list in the task manager.

When the task manager launches a task, a unique identifier is assigned to that task and its dependencies are constantly monitored. Dependencies can be one of four types:

- No dependencies: the task automatically runs when it is inserted in the task manager.
- Dependency on task: the task will wait until other indicated task has finished.
- Dependency on files: the task will wait until all the depending files have been created.
- Dependency on event: this task will wait until some event has been successfully executed.

If any on the dependencies reports an error, all the dependant tasks will be aborted by the task manager. Figure 4 shows the user interface monitoring the dependencies of several tasks.



sign	Task	Status	Blocked by	Des
1 1...	[Dectecting feat...	running		
2 1...	[Matching featur...	waiting	on task: [Dectectin...	
3 1...	[Init. SfM in Bun...	waiting	on files: [/home/...	
4 1...	[Dectecting SfM ...	waiting	on task: [Init. SfM in...	
5 1...	[Matching SfM i...	waiting	on task: [Dectectin...	

Figure 4: The activity of the task manager can be monitored in the user interface.

The communication of the processes with the task manager is done by the standard error codes of the processes and the standard output, but not using pipes of the operating system. This is the reason that this software cannot be easily ported to non-POSIX operative systems operating systems, such as Microsoft Windows[®], though we are currently searching for another solution.

The task manager not only concurrently controls all threads for each process, but by means of an extension, it is also able to manage remote tasks, as described in the next section.

3.2. The Remote Task Manager

In general, cloud computing provides flexibility, so the user can send its work to one or more powerful computers without worrying about the amount of models he or she wants to reconstruct. In order to allow cloud computing in OpenREC, the task manager has been extended.

When the task manager sets a task as remote, the wrapper changes all the underlying commands to execute *ssh* and *nohup*. Although the operating system takes over of users and permissions, there are some problems that the operating systems do not solve.

One problem to avoid is inconsistency through concurrency, so the files of one task should not be modified by the files of another task. OpenREC solves this problem by using a unique identifier that is increased once a remote task is launched. The real PID of the remote underlying command and a unique identifier are also stored in the task with the data regarding to the server and the remote path.

Another problem is that operating systems do not provide any mechanism to know if some process has finished or not. To check its remote PID is not enough: in practice the underlying algorithms of the operating systems can re-use PIDs of dead processes in new ones. OpenREC solves this problem by creating a file once the process has finished. It stores whether it has finished with errors or not. In that case, we can also know whether the process aborted: if no file exists and the remote PID is no more in the system we can safely guess that the process has been killed.

Before sending the files back to the client, the task manager converts the remote paths to the local (the reversed process is carried out before sending the files to the server).

This way, it is possible to combine local and remote tasks without problems.

This extension is far from being perfect, for example, some of the files are signed by OpenMVG, avoiding any software modifying paths in these kind of files without corrupt them, although one possible solution is to launch all tasks in block against the server, that would kill the idea of combining local and remote tasks indistinctly. Other problem is the way *ssh* works in Microsoft Windows[®] is slightly different to other operating systems, making difficult to migrate this capability to that operating system.

4. Conclusions and Future Work

In this article we have described OpenREC, a new software that implements all the necessary stages to reconstruct 3D surfaces from photographs. This paper also describes the core of the software that is able, not only to launch local tasks, but also to use the SSH protocol to execute some parts of the process remotely.

Though the software is not complete, and there are issues that avoid to execute remotely all tasks of the reconstruction process, we plan as future to work some additional server-side software that mitigates this problem and that would allow us to complete a multi-platform version. We also want to extend OpenREC to manage OpenMVS and Bundler, and to extend the compatibility with commercial software such as Recap or Agisoft Photoscan via scripts.

In any case, we think that our software provides a very useful tool for professionals and academics in cultural heritage as a replacement of VisualSFM.

Acknowledgements

We want to thank the reviewers for their useful comments and suggestions.

This work has partially been funded by the Spanish Ministry of Economy and Competitiveness through the project TIN2014-60956-R with FEDER funds.

References

- [BMR*99] BERNARDINI F., MITTLEMAN J., RUSHMEIER H., SILVA C., TAUBIN G.: The ball-pivoting algorithm for surface reconstruction. *Visualization and Computer Graphics, IEEE Transactions on* 5, 4 (1999), 349–359. doi:10.1109/2945.817351. 3
- [BTVG06] BAY H., TUYTELAARS T., VAN GOOL L.: Surf: Speeded up robust features. *Computer vision—ECCV 2006* (2006), 404–417. 2
- [CCC*08] CIGNONI P., CALLIERI M., CORSINI M., DELLEPIANE M., GANOVELLI F., RANZUGLIA G.: MeshLab: an Open-Source Mesh Processing Tool. In *Eurographics Italian Chapter Conference* (2008), Scarano V., Chiara R. D., Erra U., (Eds.), The Eurographics Association. doi:10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136. 3
- [CL96] CURLESS B., LEVOY M.: A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 303–312. URL: <http://doi.acm.org/10.1145/237170.237269>, doi:10.1145/237170.237269. 3
- [GKS00] GOPI M., KRISHNAN S., SILVA C.: Surface reconstruction based on lower dimensional localized delaunay triangulation. *Computer Graphics Forum* 19, 3 (2000), 467–478. URL: <http://dx.doi.org/10.1111/1467-8659.00439>, doi:10.1111/1467-8659.00439. 3
- [KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2006), SGP '06, Eurographics Association, pp. 61–70. URL: <http://dl.acm.org/citation.cfm?id=1281957.1281965>. 3
- [LFG*10] LERONES P. M., FERNÁNDEZ J. L., GIL Á. M., GÓMEZ-GARCÍA-BERMEJO J., CASANOVA E. Z.: A practical approach to making accurate 3d layouts of interesting cultural heritage sites through digital models. *Journal of Cultural Heritage* 11, 1 (2010), 1–9. 1
- [Low04] LOWE D. G.: Distinctive image features from scale-invariant keypoints. *International journal of computer vision* 60, 2 (2004), 91–110. 2
- [McC14] MCCARTHY J.: Multi-image photogrammetry as a practical tool for cultural heritage survey and community engagement. *Journal of Archaeological Science* 43 (2014), 175–185. 1
- [MM12] MOULON P., MONASSE P.: Unordered feature tracking made fast and easy. In *CVMP 2012* (2012), p. 1. 2
- [MMM12] MOULON P., MONASSE P., MARLET R.: Adaptive structure from motion with a contrario model estimation. In *Asian Conference on Computer Vision* (2012), Springer, pp. 257–270. 2
- [MMMO] MOULON P., MONASSE P., MARLET R., OTHERS: Open-mvg. an open multiple view geometry library. <https://github.com/openMVG/openMVG>. 2
- [ÖGG09] ÖZTIRELI A. C., GUENNEBAUD G., GROSS M.: Feature preserving point set surfaces based on non-linear kernel regression. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 493–501. 1
- [SFPG06] SINHA S. N., FRAHM J.-M., POLLEFEYS M., GENÇ Y.: Gpu-based video feature tracking and matching. In *EDGE, Workshop on Edge Computing Using New Commodity Architectures* (2006), vol. 278, p. 4321. 2
- [SSS06] SNAVELY N., SEITZ S. M., SZELISKI R.: Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH Conference Proceedings* (New York, NY, USA, 2006), ACM Press, pp. 835–846. 2
- [SSS08] SNAVELY N., SEITZ S. M., SZELISKI R.: Modeling the world from Internet photo collections. *International Journal of Computer Vision* 80, 2 (November 2008), 189–210. URL: <http://phototour.cs.washington.edu/>. 2