robotics

MDPI

Article

A Matlab-Based Testbed for Integration, Evaluation and Comparison of Heterogeneous Stereo Vision Matching Algorithms

Raul Correal [1,*], Gonzalo Pajares [1] and Jose Jaime Ruz [2]

[1] Software Engineering and Artificial Intelligence, University Complutense of Madrid, Madrid 28040, Spain; pajares@ucm.es

[2] Computer Architecture and Automation, University Complutense of Madrid, Madrid 28040, Spain; jjruz@fis.ucm.es

* Correspondence: rcorreal@ucm.es; Tel.: +34-91-394-7501

Academic Editor: Huosheng Hu

Received: 5 July 2016; Accepted: 4 November 2016; Published: 9 November 2016

Abstract: Stereo matching is a heavily researched area with a prolific published literature and a broad spectrum of heterogeneous algorithms available in diverse programming languages. This paper presents a Matlab-based testbed that aims to centralize and standardize this variety of both current and prospective stereo matching approaches. The proposed testbed aims to facilitate the application of stereo-based methods to real situations. It allows for configuring and executing algorithms, as well as comparing results, in a fast, easy and friendly setting. Algorithms can be combined so that a series of processes can be chained and executed consecutively, using the output of a process as input for the next; some additional filtering and image processing techniques have been included within the testbed for this purpose. A use case is included to illustrate how these processes are sequenced and its effect on the results for real applications. The testbed has been conceived as a collaborative and incremental open-source project, where its code is accessible and modifiable, with the objective of receiving contributions and releasing future versions to include new algorithms and features. It is currently available online for the research community.

Keywords: stereovision; 3D scene reconstruction; matching algorithms; correspondence; computer vision; image processing; robotics; testbed; open-source; Matlab

1. Introduction

Stereoscopic vision is a mechanism to obtain depth or range data based on images. It consists of two cameras separated by a given distance so that two differing views of a scene are obtained, similar to human binocular vision. This pair of images is the input to the matching process. By comparing both images, relative depth information is obtained, in the form of disparities, which are inversely proportional to distance to objects. The difference in position of a set of features or pixels from one image relative to the other is then computed, usually along the horizontal axis. Depth can be established by triangulation of the disparities obtained from the matching process, provided that the position of centres of projection, the focal length and the orientation of the optical axis are known. It can then be reprojected to 3D space using a perspective transformation, obtaining a set of world coordinates.

According to [1] and [2], the classical problem of stereo analysis consist mainly of the following steps: image acquisition, camera modelling, feature extraction, image matching and depth determination. Of these, the matching process—the process of identifying features in both images and searching for correspondences—is the key step. This paper is devoted solely to the matching step, to which the scientific community has devoted a major effort.

The problem of correspondence in stereoscopic systems stems from the fact that images from cameras, although similar, show different intensity levels for the same physical entity in the 3D scene. The main reason for this feature lies in the different response from the camera sensors to the signal light from the scene and also from the different mapping of the scene over each image due to the different relative points of view of each camera. That makes it necessary to devote a major research effort to correct these deviations typical of any stereo system. This problem is not yet satisfactorily solved, particularly in unstructured and uncontrolled environments, which is the main reason why literature about this topic is so broad.

This paper describes the design and implementation experience of an open-source testbed to centralize, standardize and facilitate the evaluation, functional analysis and comparison of heterogeneous stereo matching algorithms. It allows the integration of different approaches and algorithms, both existing and forthcoming ones, developed by different authors in diverse languages, together under a common framework, representing a new approach in relation to previous works in the field. Additionally, this testbed includes a set of pre- and post-filtering processes in order to correct problems derived from the illumination conditions in outdoor environments and remove spurious matched pixels, both representing important contributions.

Some other contributions of the presented testbed to the computer vision community with respect to previous related works are: portability, as the testbed is architecture and operative system independent; openness of source code allowing analysis of algorithms and adaptations; scalability of the system to incorporate new published algorithms and contributions in a collaborative and incremental approach; flexibility in the inclusion of these algorithms in different programming languages; use of diverse image formats; a graphical interface for friendly interaction and configuration of tests and algorithms; and a graphical presentation of results. This is intended with the aim of providing an efficient tool to apply stereovision-based strategies in real applications with high degree of efficiency and accuracy. Researchers and developers will find a useful tool to guarantee the success and progress of their works when stereo-vision based methods are to be applied. This makes the main contribution of this paper with a high content of applicability. The first version of this stereo testbed is already available online [3], and have had one thousand downloads up to now, denoting a strong interest by the research community.

The present paper is structured as follows: Section 2 presents an introduction to the stereo vision state of the art and the justification for the need of this research. The stereo testbed is introduced in Section 3, which outlines its main requirements and design. The set of algorithms and processes integrated into the testbed are described in Section 4. Next, in Section 5, the testbed usage is illustrated; a use case is described to show how different processes, filters and algorithms, can be sequenced and its effect on the results. Section 6 summarizes conclusions, experiences, lessons learned and future work.

2. Revision of Methods

Stereo vision is a heavily researched field; a vast amount of published literature and a wide range of algorithms, techniques, implementations and libraries are available [4,5]. However, there is no single approach that could be considered the best one, able to solve every possible problem. Usually, each technique is suitable for a set of conditions but not for others.

A review of the state of the art in stereo matching allows us to distinguish two main classes of techniques: feature-based (sparse) and area-based (dense) methods [6]. Feature-based methods use sets of pixels with similar attributes, picking out feature points of high distinctiveness, either pixels belonging to edges [7] or the corresponding edges themselves [8]; leading to a sparse depth map used mainly for object segmentation and recognition, sometimes reconstructing the rest of the surface by interpolation [9,10]. Examples of feature-based approaches are SIFT (scale-invariant feature transform) or FAST (features from accelerated segment test). Area-based stereo algorithms are used instead to find matches for all points in the images; they use correlation between brightness (intensity) patterns in the local neighborhood of a pixel in one image with respect to the other [11]; the number of possible

matches is intrinsically high. These methods are useful if the system needs to recover the detailed geometry of the scene, in order to facilitate obstacle avoidance or object recognition. This paper is devoted solely to this last approach.

For any project that uses stereo vision, there is usually a set of requirements to be met. The first step is to analyze the state of the art, verify if there is already any algorithm or technique fulfilling the needs and to what extent. In case there is not any, a new strategy has to be developed, either from scratch or based on previous developments and existing approaches. This evaluation phase is not trivial at all. Given the range of available resources and literature, it is necessary to make a considerable effort to analyze and determine what strategy may fit each concrete set of requirements. This becomes especially crucial when stereo-based techniques are to be applied to real situations. Several problems arise when theoretical methods or strategies are to be investigated for posterior application in real scenarios. This paper analyses such problems and provides a framework for solving them.

One of the difficulties is checking the availability of a suitable implementation. Many algorithms and techniques are outlined within the published literatures, described in more or less detail depending on each publication; sometimes the level of detail covered is enough to implement it using any programming language, but sometimes it is not. There exist also some online sites containing stereo vision algorithms; in some cases they are available to be downloaded just in binary format, which may require a concrete platform such as Linux, Windows or any other system. In other cases source code access may be granted, finding algorithms implemented in C/C++, Java, Matlab or any other language, requiring compilation before using it. There are a series of computer vision libraries in a variety of languages, where some code has to be written to make use of them. This heterogeneity in the available resources and the lack of standardization makes the state of the art analysis, algorithms evaluation and comparison a hard, complicated and time consuming task. Therefore, it is crucial to establish a common framework where this broad spectrum of heterogeneous resources could be put together, facilitating its analysis and comparison, and where they can be integrated to obtain even more sophisticated approaches.

A previous work in classification and characterization of stereo correspondence algorithms was made at Middlebury College by [4], where the authors presented a taxonomy of two-frame dense methods. They developed a stand-alone software containing some representative stereo algorithms implemented in C++ and a collection of multi-frame data sets. Despite the fact that it is a very interesting, useful approach and a reference work in the field, it presents some significant drawbacks. The source code has to be compiled by the user, which is a significant step. As the authors point out, while it is still being maintained, it does not represent an implementation of state of the art stereo methods. Also, in case a new algorithm is to be added in the future, as new approaches are constantly coming up, the internal software structure should be analyzed to understand where to properly insert it and the interdependencies with other modules, and recompile all together again. As the authors point out, there is no documentation provided beyond the comments embedded in the source code, what may be challenging for non-expert users. In addition, it does not allow including algorithms developed in different languages; despite most of the available algorithms are implemented in C++, there are also a large amount of resources developed using other technologies such as Java or Matlab. In this case, just C++ code can be integrated. Finally, given the lack of a graphical user interface, they proposed a sophisticated mechanism for specifying parameter values and input images that supports recursive script files, the syntax of which is quite complex. New scripts shall be created in case additional algorithms or images are included in a predefined structure of directories. Results are stored in text files, lacking of any graphical representation.

3. Stereo Testbed

3.1. Requirements

The main objective of the testbed is to focus on the problems of distribution of resources and lack of standardization, centralizing heterogeneous resources under a common framework. It purports to serve as a starting point, bringing different algorithms and approaches together in an organized way, facilitating its analysis, evaluation and comparison of results and reducing the necessary effort. The lack of standardization is one of the main issues. Implementations of some of the algorithms proposed in the literature can be found, however they are written in different languages and for different platforms, making their evaluation hard and time consuming. For that reason, this testbed aims to serve as a common platform to integrate such a variety, being language and platform transparent.

To allow algorithms and processes to interface with each other, inputs and outputs must be standardized, so that results from some processes can be used as inputs for others. It eases also the evaluation and comparison of algorithms by presenting results from different approaches in a common format, such as disparities obtained, right and wrong matches, or processing time. The testbed should support working with images in several formats (e.g., jpg, bmp, png, pnm, etc.), to use them as input for the algorithms, including and managing them in a straightforward way. It allows several algorithms to be compared using the same input images or a given algorithm to be tested using different images. Ideally each stereo pair would include ground truth information to verify the obtained results with trustworthy data, although this should not be a requirement as this data is not always available.

Given the parametric nature of the stereo matching process and the large amount and variety of algorithms that can be included in the testbed, the selection and configuration of the algorithm should be done in an easy and straightforward way. This is important for testing purposes, as algorithms' parameters have to be frequently adjusted, given they critically influence the result and performance. There should be no need to get into the source code for parameters definition or to modify its values, as it would require to recompile, implying a considerable effort to analyze and understand the code. Ideally it should provide a graphical interface, both for configuration and presentation of results, avoiding the use of scripts and input/output text mode.

As researchers in the stereo vision community work in a heterogeneity of platforms, the testbed should be easily portable, capable of work in a multitude of systems, ideally avoiding recompilation of sources, which is often a challenge, achieving the highest possible degree of platform independency and portability. Regarding source code, although C++ is the dominant language for computer vision algorithms, many researchers use languages such as Java or Matlab to implement their works. The testbed should allow for the inclusion of these heterogeneous resources without requiring to rewrite them in any other language, taking advantage of the work already done; it eases the necessary effort to integrate new resources into the testbed and allow to concentrate efforts in its evaluation instead. This leads to a core requirement of the testbed, which is openness. Researchers may have access to the sources of the algorithms included on it to extend or adapt any piece of code to each concrete needs, as well as to include new algorithms and processes. However, it should also be possible to include resources in a closed form, such as libraries or compiled code, for cases where an author decides not to release its code or they are just not allowed to do it.

The testbed is expected to be a dynamic and evolutionary platform growing over time, in a collaborative and incremental development, releasing new and more sophisticated versions periodically including more algorithms and capabilities. For that reason, besides the capability of including new algorithms on a given local copy of the testbed, there should be a way to report or send new resources to the testbed maintainers to be included in subsequent releases.

3.2. Design

After a thoughtful analysis, Matlab has been chosen as the development platform to build up the testbed, as it allows meeting the set of requirements introduced in the previous section. Matlab is a well-known programming environment for algorithm development, data analysis, visualization

and numerical computation. It is widely used in academic and research institutions and industrial enterprises, with a large community of users and a broad range of specific packages and toolboxes that help to reduce considerably the development efforts. Figure 1 shows the stereo testbed architecture.

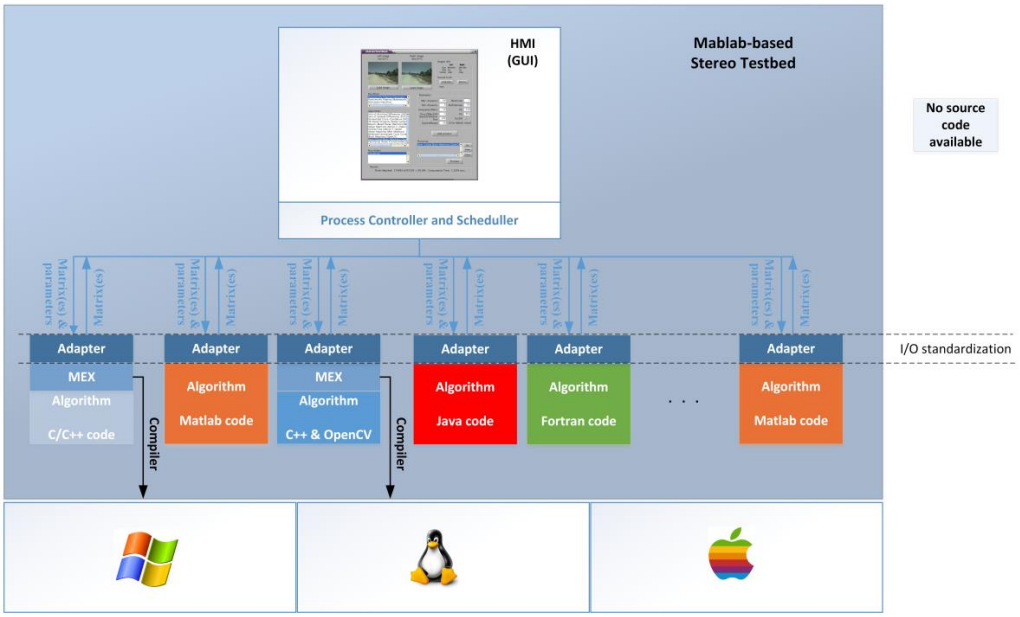


Figure 1. Stereo testbed architecture.

Regarding the requirement of addressing the heterogeneity of available resources, Matlab allows interfacing with programs written in other languages, such as C, C++, Java and Fortran, avoiding the need to rewrite them. Currently, there is a large number of stereo vision algorithms available implemented in C++. However, the amount of stereo and computer vision resources available for the Matlab community is growing, as every day more researchers implement their algorithms directly using Matlab code. Computer vision libraries, such as the well-known OpenCV [12] can be also interfaced with Matlab. It also allows addressing the fact the research community works in heterogeneity of platforms as there exist Matlab versions for different platforms such as Windows, Linux and Mac, so the stereo testbed and the algorithms included on it are easily portable, and capable of executing in a multitude of systems, avoiding recompilation of sources and meeting the transparency and OS independency requirements.

The testbed code itself is open and accessible, as it has been developed using Matlab code, as well as most of the algorithms included in the testbed, meeting the openness requirement, so they can both be extended and adapted to each concrete need. But it allows also for the inclusion of closed-source resources, such as libraries or compiled files, using Matlab Executable (MEX)-files. They are dynamically linked subroutines produced from C, C++ or Fortran that, when compiled, can be run from within Matlab in the same way as .m files—open source files—or built-in functions. Besides protecting the source code when access to it is not allowed, this format is more efficient, as these files execute much faster than interpret code. Although the speed of the stereo process is usually crucial and it is known code executed from Matlab is slower than native languages like C/C++, even for MEX-compiled files, it is important to emphasize the main focus of the presented stereo tested is not to include or evaluate the most efficient or real-time algorithmic implementation. In contrast, it is to facilitate the functional analysis of algorithms, checking performance and validating approaches, saving the effort of implementing algorithms just to test how they work. Once a concrete algorithm or process has been identified as functionally appropriate for a given set of requirements, efforts can be focused in coding just that one and in the most efficient way. There exist multiple approaches to optimize algorithms for embedded and real-time systems such as using low level languages (e.g. C,

Assembler) using multi-processor boards for parallel computing or coding the algorithm in a field programmable gate-array (FPGA) [13,14].

Inputs and outputs have been standardized to ease algorithms integration with the testbed and to interface processes with each other. As introduced before, besides the stereo matching algorithms, the testbed includes a set of pre- and post-filtering processes to correct problems derived from the illumination conditions and remove spurious matched pixels. Input to any matching algorithm has been established as a pair of stereo images and a structure with corresponding parameters' values, while the expected output is a disparity matrix. Any process or filter applicable prior to the matching step receives the stereo pair and parameters as input and return the same pair of images modified according to each process' functionality; post-filters take the disparities matrix computed in the matching process and return the improved and enhanced results, also as a disparities matrix.

To include any given algorithm within the testbed, an intimate understanding of the algorithm is not required. Actually, it can be used as a black box. The only requirement is to adapt the parameters of the algorithm, generally by creating a wrapper, to transform the standardized inputs from the testbed described above to the format required by the concrete algorithm and the equivalent for the outputs. This standardization mechanism allows not just an easy integration of algorithms with the testbed but the possibility to interface with each other and chain processes. The only steps left are the inclusions of the algorithm in the appropriate list within the user interface, described below, including its call in the testbed code and adding a graphical panel to tune its parameters, if any. Places in code to accomplish these steps are localized and details are clearly indicated within the testbed source code. In any case, as indicated in the requirements section, new algorithms can be sent to the maintainers of the project for its inclusion in future versions. Once an algorithm has been integrated into the testbed, just a few clicks are necessary to select, configure and execute it.

To meet the requirement of creating a friendly user interface, the Matlab Graphical User Interface Development Environment (GUIDE) has been employed. It includes common components such as textboxes, labels, lists, radio/check buttons and panel, allowing the user to easily select, configure and execute any of the algorithms included. A series of contextual panels are presented to the user to set parameters' values in a straightforward way, avoiding any source code recompilation to adjust these values. Figure 2 shows the testbed graphical user interface.

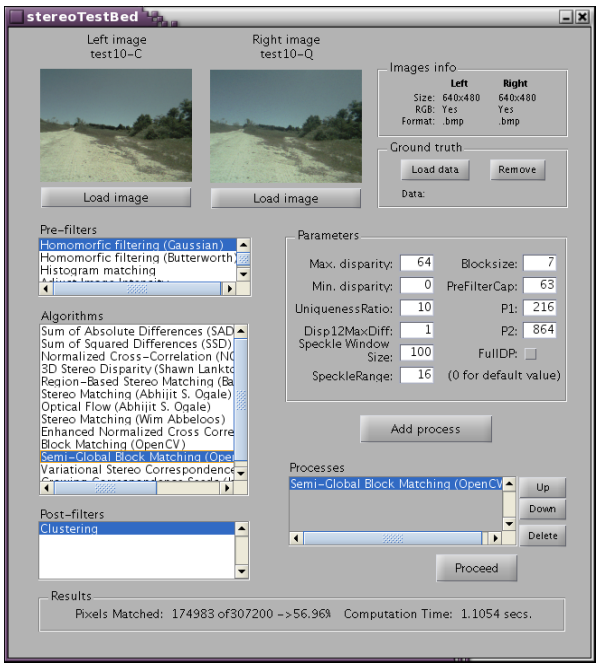


Figure 2. Stereo testbed user interface.

Processes are organized in three categories within the user interface: pre-filters, algorithms and post-filters. Pre-filters refer to processes performed on the images before executing the matching algorithm, like adjusting images' intensities. The "Algorithms" list includes the set of stereo matching algorithms available within the current version of the testbed. Processes listed in the "Post-filters" list check the computed disparities and filter out wrong matches, improving and optimizing results. To set a chain of processes, each one is selected and configured independently. Once an algorithm has been selected from its list, its parameters, if any, are displayed within the graphical interface along with its default values, so they can be tuned to be adapted to the input images and application. It is then introduced in the "Processes" list, which contains the chain of algorithms to be executed so far. The order of execution and parameters of any process can be modified at any time using the appropriate controls within the user interface. Also, processes can be removed and included in the execution chain at any time, easily checking how it affects the whole process.

Input images can be loaded from a standard open dialog box. As required, many different formats are supported: bmp, gif, jpg, pbm, pcx, pgm, png, pnm, ppm and tiff among others. Ground truth information can be optionally specified and loaded if available; it is actually an image representing real disparities. Ground truth data, although not mandatory, is very useful for algorithms evaluation. After an algorithm is executed, the testbed reports the total number of potential matches, the number of right, wrong, false and missed correspondences obtained according to this ground truth data, both in absolute values and percentages. Final results and intermediate products are displayed as images, graphs or messages in the console, including information such as the number of correspondences found, number of right, wrong and missed matches or execution time, easily comparing results from different algorithms; extended information is presented in case ground truth data is available.

4. Integrating Algorithms within the Testbed

For the first version of the testbed we have included a number of stereo matching algorithms and some additional image processes. From the two approaches for stereo algorithms introduced previously, this paper focuses on the area-based (dense) methods, given its applicability to our research domain, which is robotic autonomous navigation in outdoor terrains. However, the same approach is valid for feature-based and other stereo algorithms. They are listed below, grouped in three categories: pre-filters, processes executed on the input images; matching algorithms, the actual correlation between matching algorithms; and post-filters, processes that filter out wrong matches and improve results.

Pre-filters:

- Homomorphic filtering [15,16]: this is based on the assumption that images can be expressed as the product of illumination and reflectance components. Illumination is associated with low frequencies of the Fourier transform and reflectance with high frequencies. Thanks to the product and applying logarithms followed by high pass filtering, the illumination component can be eliminated. This allows the retention of only the reflectance associated with intrinsic properties of the objects in the scene. In this implementation we have two different filters available: a Butterworth High Pass [17] and a Gaussian filters. They can be applied in the RGB or HSV color models.
- Histogram matching [18,19]: this normalizes the images of the stereo pair, adjusting the color distribution of one image with respect to the other. It can be performed in the greyscale, RGB or HSV color models.
- Adjust image intensity: this adjusts the intensity values such that 1% of data is saturated at low and high intensities, increasing the contrast of each image of the stereo pair.

Matching algorithms:

- Sum of absolute differences, SAD [20]: this is a classical approach. It measures similarity between image blocks by taking the absolute difference between each pixel in the original block and the corresponding pixel in the block being used for comparison. These differences are summed to create a simple metric of block similarity. In this implementation, three similarity measures can be applied: SAD, zero-mean SAD or locally scaled SAD.

- Sum of squared differences, SSD [21]: this measures similarity between image blocks by squaring the differences between each pixel in the original block and the corresponding pixel in the block being used for comparison. In this implementation, three similarity measures can be applied: SSD, zero-mean SSD or locally scaled SSD.
- Normalized cross correlation, NCC [22]: this is a common matching technique to tolerate radiometric differences between stereo images. In this implementation, two similarity measures can be applied: NCC or zero-mean NCC.
- 3D stereo disparity [23]: this estimates pixel disparity by sliding an image over another one used as a reference, subtracting their intensity values and gradient information (spatial derivatives). It also performs a post filtering phase, segmenting the reference image using a "mean shift segmentation" technique, creating clusters with the same disparity value, computed as the median disparity of all the pixels within that segment.
- Region-based stereo matching [24]: this provides two different algorithms based on region growing: a) Global Error Energy Minimization by Smoothing Functions method, using a block-matching technique to construct an error energy matrix for every disparity found and b) line growing method, locating starting points from which regions will grow and then expanding them according to a predefined rule.
- Stereo matching [25,26]: this develops a compositional matching algorithm using binary local evidence which can match images containing slanted surfaces and images having different contrast. It finds connected components and for each pixel, picks the shift which corresponds to the largest connected component containing that pixel, while respecting the uniqueness constraint.
- Optical flow [25,26]: this is similar to the previous stereo matching algorithm, but the main difference is this considers the shifts are two dimensional, unlike the previous one that considered just horizontal shifts, and connections across edges parallel to the flow being considered must be severed.
- Stereo matching [27]: this implements a fast algorithm based on sum of absolute differences (SAD), matching two rectified and undistorted stereo images with subpixel accuracy.
- Enhanced normalized cross correlation [28]: this implements a local (window-based) algorithm that estimates the disparity using a linear kernel that is embodied in the normalized cross correlation function, leading to a continuous function of subpixel correction parameter for each candidate right window. It presents two modes: (a) a typical local mode where the disparity decision regards the central pixel only of each investigated left window, and (b) a shiftable-window mode where each pixel is matched based on the best window that participates into, no matter what its position inside the window is.
- Block-matching [29]: this is based on sum of absolute differences between pixels of the left and right images. This algorithm is optimized by applying the epipolar constraint, searching for matches only over the same horizontal lines which cross the two images, what necessarily implies the two images of the stereo pair must have previously been rectified. The OpenCV 2.3.0 [12] implementation is used for this process.
- Semi-global block-matching [30]: this is a dense hybrid approach that divides the image into windows or blocks of pixels and looks for matches trying to minimize a global energy function, based on the content of the window, instead of looking for similarities between individual pixels. It includes two parameters that control the penalty for disparity changes between neighboring pixels, regulating the "softness" in the changes of disparity. The OpenCV 2.3.0 [12] implementation is used for this process.
- Variational stereo correspondence: this implements a modification of the variational stereo correspondence algorithm described in [31], consisting on a multi-grid approach combined with a multi-level adaptive technique that refines the grid only at peculiarities in the solution, allowing the variational algorithm to achieve real-time performance. It includes also a technique that adapts the regularizer used in the variational approach, as a function of the current state of the optimization. The OpenCV 2.3.0 [12] implementation is used for this process.

- Growing correspondence seeds [32]: this is a fast matching algorithm for binocular stereo suitable for large images. It avoids visiting the entire disparity space by growing high similarity components. The final decision is performed by Confidently Stable Matching, which selects among competing correspondence hypotheses. The algorithm is not a direct combination of unconstrained growth followed by a filtration step. Instead, the growing procedure is designed to fit the final matching algorithm in the sense the growing is stopped whenever the correspondence hypothesis cannot win the final matching competition.

Post-filters:

- Clustering: this removes false matches after disparities computation. The algorithm groups pixels in connected components based on the principle of spatial continuity, where pixels belonging to the same region are all reachable from any other pixel from the same region, either by direct contact or at very short distance. Once clusters have been computed, they are filtered out depending on certain configurable criteria, such as cluster size or density.

All of these algorithms have been adapted to ensure compatibility to be included within the testbed; their inputs and outputs have been standardized to interact with each other and with the testbed itself (see Figure 1). For extended details on the insights of each algorithm, understanding of their performance, implementation details and how they differ from the other to confirm which one could better suit each own needs and requirements, it is recommended to consult the available literature associated to each algorithm, as a thoughtful analysis of these aspects is out of the scope of this paper and heavily depends on the concrete application and constraints of each project.

For an initial version of the testbed, the selection of the algorithms to be included in the testbed does not try to represent a sampling of state of the art algorithms. The main objective is to demonstrate the integration of heterogeneous algorithms developed by different authors using different programming languages. Some of them have been implemented by us, while others have an implementation already available in the network; some are older, classical approaches, although still very commonly applied, while others are quite recent; others, such as the block-matching [29], semi-global block-matching [30] and variational stereo correspondence, are part of well-known and popular libraries in the computer vision community, in this case part of the OpenCV library [12]. This variety of algorithms was chosen that way on purpose, in an attempt to balance approaches from different ages and diverse technologies, to be integrated together under the common testbed framework.

As shown in [33,34], the immense number of dense stereo matching algorithms makes the idea of implementing all of them a titanic, and almost unachievable, effort, and definitely not attainable for an initial version. Besides, new approaches are continuously coming up and being published. For that reason the testbed has been conceived as an incremental development, open to collaborations to produce more sophisticated versions with more algorithms and features over time. The algorithms selection presented here constitutes just a first attempt to demonstrate how a set of fairly random and diverse algorithms are integrated together.

5. Usage of the Testbed

Figure 3 illustrates the usage of the testbed. Initially, a pair of input images, in any of the multiple formats accepted, are loaded into the testbed. Optionally, ground truth data, if available, can be also loaded to automatically be compared with results obtained from algorithms. Some stereo pairs with its corresponding ground truth data can be found in the Middlebury site [33]. Next, a series of algorithms and filters can be selected to be chain executed. This is accomplished by an iterative process of selecting one of the available algorithms, from the pre- or post-filters or algorithms lists, configuring its parameters and adding it to the processes list. This list can be reordered at any time, causing an alteration in the execution order in the scheduler. Once this chain of processes is executed, the testbed shows the results obtained in the form of number of computed correspondences, including details on right and wrong matches, false positives, and missed matches if ground truth data is available.

Parameters of a given algorithm in the processes chain can be adjusted at any time and re-execute the chain to compare results and verify the effects of parameters. This is a very fast and effective way to tune algorithms' parameters, with no need to modify source code, recompile or edit a parameters file. It is done graphically, preserving the processes chain and other algorithms' configuration. This chain can also be altered at any time by introducing or removing algorithms at any point in the list, quickly and easily checking its effects.

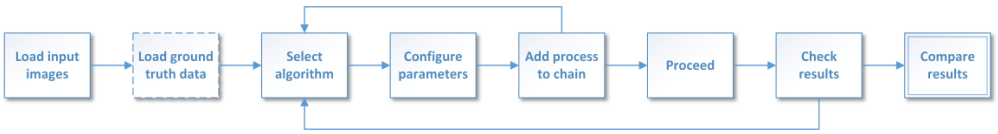


Figure 3. Testbed usage flowchart.

Use Case: Experiments and Results

This section details a use case to illustrate the usage of the testbed and its utility and application to a real problem. As introduced before, one of the main focuses of our research group is autonomous navigation of Unmanned Ground Vehicles (UGVs) in natural, sharp terrain and unstructured environments. We have created a framework to support simulations in this domain [35] that allows us the creation of simulated models of robots and outdoor environments. We have created also a visual-based autonomous navigation software that allows our robots to move safely through an unknown terrain [36]. It uses stereo cameras mounted on the robot and processes pair of images to build a 3D reconstruction of the terrain. To do that, an area-based (dense) stereo matching algorithm has to be employed in order to compute disparities.

In the initial phases, there was no vehicle available, and so it was emulated using the simulation capabilities of the framework, where vehicles and terrains were replicated. Images of simulated terrains were captured from simulated cameras. Several stereo algorithms were evaluated to produce the disparity maps of those images. The stereo vision process was based on the block-matching (BM) algorithm [29]. It produced very satisfactory results, finding almost 100% of the potential matches. Figure 4 shows an example of simulated images and the disparity map obtained using this algorithm.

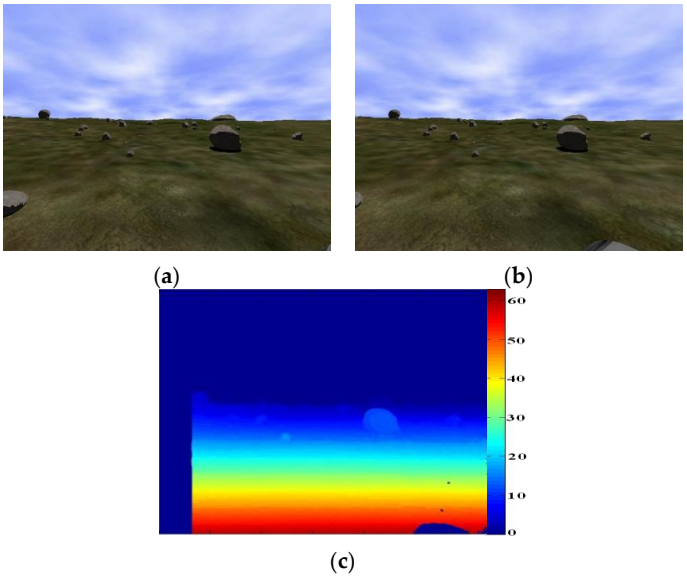


Figure 4. (**a**,**b**) Synthetic stereo images obtained from the simulated environment and (**c**) disparities map computed with the block-matching (BM) algorithm (right side bar represents disparity values, maximum: 64 pixels).

The problem arose when the navigation software was ported to a real robot, equipped with a stereoscopic system of parallel optical axes Videre STH-DCSG-9 color 640 × 480 pixels, with two three millimeter lenses. The same algorithm that obtained very satisfactory results with the synthetic images did not performed so well with the real ones. Figure 5 shows a pair of real images obtained with this system and the disparity map the BM algorithm produces. Images shown in the figure contain 370,200 pixels, and 157,049 of them have potential matches; the other pixels belong to remote areas with no computable disparity, like the sky. The BM algorithm is able to find 57,598 correspondences, representing 36.67% of the total possible matches. This results in a poor 3D reconstruction of the environment, producing a map with large empty areas, where accurate trajectories for robot navigation cannot be computed. This problem appears systematically in all stereo pairs analyzed from the real system, due to some factors present in the real world that do not occur in simulation, such as the different response from the camera sensors to the light from the scene, which produces different intensity levels, critically affecting the disparity computation.

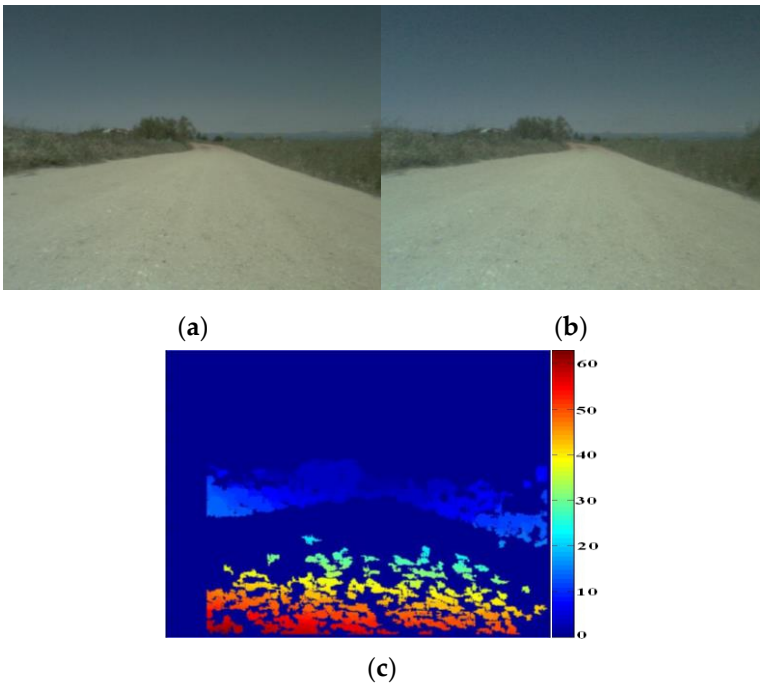
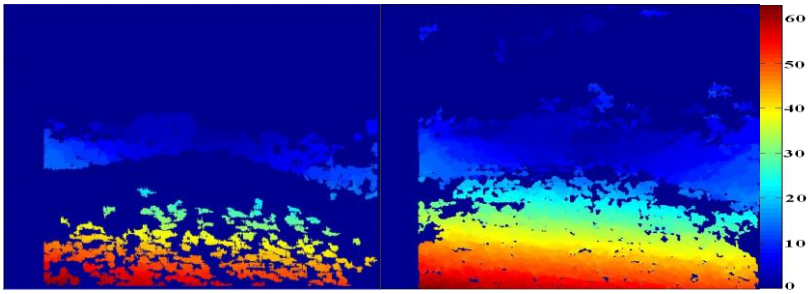


Figure 5. (**a**,**b**) Real images and (**c**) disparities computed with the BM algorithm.

Therefore, some other algorithm has to be employed. Again, an analysis of the stereo matching algorithms must be made. Prior to implementing an algorithm, resources available must be tested, and the problems mentioned previously have to be faced: algorithms are implemented in different languages, input/output formats differ from one to another, tuning parameters requires sometimes recompilation, slowing down the process, etc. It is for this point that the design and creation of the stereo testbed was motivated: different algorithms were analyzed and integrated into the tested to be tested. It was identified that this situation happens frequently, where the nature of the input images, the camera manufacturer or the operational conditions changes and the results are highly affected, requiring to adapt the algorithms and its parameters or even rebuild the solution and implement a brand new and different approach and algorithm. This is where the stereo testbed provides a real advantage, to test many different algorithms with the same input images, tuning its parameters and comparing results in a very short time with just a few clicks, saving lots of time. It allows checking the performance of the algorithms included in the testbed without the need to write any code, avoiding unnecessary efforts, and it is expected to be more useful as more algorithms are included and integrated into the testbed over time.

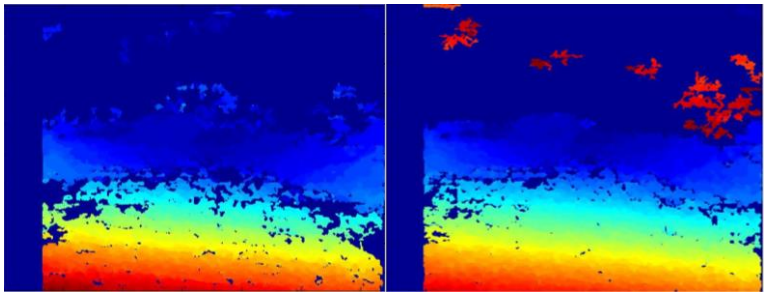
As a result, from the set of algorithms tested and integrated, it was verified that the best results for the real input images were obtained using the semi-global block-matching (SGBM) algorithm [30]. This works well for image areas where texture is more or less uniform and that have low variation in the spectral levels of the pixels that define them. The energy reduction function is able to detect correspondences in the regions where the block-matching algorithm fails. The disparities map obtained using this algorithm is shown in Figure 6. The SGBM algorithm finds 144,827 matches, achieving a great improvement with respect to the 57,598 correspondences obtained from the BM algorithm. As a result, the BM algorithm was replaced by the SGBM algorithm in the robot onboard software.



(**a**) (**b**)

Figure 6. Disparities obtained using the (**a**) BM and (**b**) semi-global block-matching (SGBM) algorithms.

Despite the fact that the SGBM algorithm achieves better results computing disparities, it computes some mistaken correspondences; for the case of the example images shown previously, it computes 10,738 miss matches. To correct these errors, certain techniques can be applied to the input images, prior to the matching process. Some have been implemented and included within the testbed, the so called pre-filters. One such treatment for the correction of deviations is homomorphic filtering [15,16]. This process eliminates the illumination component of the input images while preserving reflectance. The idea is that the same object must produce similar intensity levels on both the right and left images, achieving maximum similarity in the spectral levels of the stereoscopic pair, as happens in the case of simulated images. Using the testbed it can be achieved by configuring a chain of processes; the pre-filter "homomorphic filtering" is first applied to the input images and the result is passed to the SGBM matching algorithm for disparities computation. The effects of applying this filter to the images shown in Figure 5 can be seen in Figure 7. It can be observed that the disparity computation has been improved. The matching algorithm is able to find 11,497 more correspondences using the filtered images than the real ones; in this case it computes 145,586 matches out of the 157,049 possible, or 92.70%, representing an improvement of 7.32%.

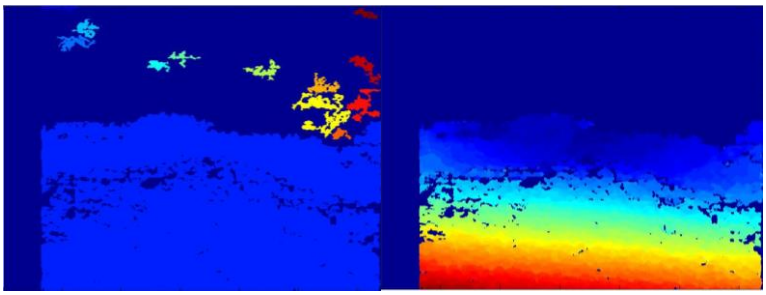


(**a**) (**b**)

Figure 7. Disparities maps obtained using the SGBM algorithm (**a**) directly on the original images and (**b**) applying a previous homomorphic filtering.

The reddish areas at the top of Figure 7b represent false matches. It is a collateral effect of using the homomorphic filtering; the same effect has been appreciated in all experiments using images captured with our stereo system. To filter out these false matches another process, called "clustering", has been included within the testbed, in the post-filters section. This technique consists of grouping pixels in connected components based on the principle of spatial continuity; any pixel belonging to a region must be reachable from any other pixel of the same region. It means a cluster is formed by pixels in direct contact with each other. The result of the process is a list of clusters. In this domain, in images from natural terrain, it can be assumed that the largest cluster is the terrain itself. This is true for all experiments we have performed with the large set of images captured with our stereoscopic system; virtually all except the main cluster are formed by false positives and therefore can be removed. Only in some specific cases do some clusters belong to small portions of terrain isolated from the rest. In such cases, the loss of pixels due to this filtering is very small, less than 0.1% of total pixels according to our experiments, which is acceptable and does not have a significant impact on the 3D reconstruction process. Regardless, this loss could be easily avoided. These clusters are usually very close to the main one; just a slight increase in the minimum connection distance parameter makes the algorithm consider them part of the principal group and they therefore will not be filtered out.

For the example shown in Figure 7, the process detects 13 clusters, represented in Figure 8a. A large cluster representing the terrain itself and a series of smaller clusters that form groups of false positives can be observed. The result of filtering all these clusters out is shown in Figure 8b; an improved and error-free disparity map is obtained. It is important to clarify that the presence of rocks or other objects in the images does not cause the detection of them as separate clusters. This method performs groupings based on spatial continuity and these objects, as they are in direct contact with the ground, do not generate a disconnected component that may form an independent cluster [37].



(**a**) (**b**)

Figure 8. (**a**) Clusters detected by the algorithm and (**b**) final disparities computation after filtering out errors.

It is possible to conclude that the application of a homomorphic filtering process to remove the illumination component of the stereo pair prior to the matching algorithm, and a subsequent clustering process to filter out wrong matches, introduce significant improvements in the disparity map computation. This has an important impact and represents an advance in the matching process for real images.

Images used in this experiment are just a representative example to illustrate the differences found in the algorithms when using synthetic or real images and the applicability of some image filtering; analogous results have been obtained from a large set experiments using 60 different stereo pairs, taken with a general-purpose low-cost stereo system under different illumination conditions, days and locations.

6. Conclusions and Future Work

The broad range of publications and the heterogeneity of the available resources about stereo vision make the analysis of the state of the art a hard and complex task. A flexible framework with a friendly interface has been designed to facilitate the task.

Matlab has been chosen as the development platform. A series of heterogeneous stereo matching algorithms, and some pre- and post-filtering techniques, have been adapted, standardized and integrated into the testbed, to make use of the large collection of existing resources and the work already done by the research community. This integration and standardization process did not require a major effort as the testbed is quite flexible and allows for the inclusion of algorithms written in different languages and writing a wrapper for inputs/outputs is usually a straightforward task.

Major contributions of the presented stereo testbed with respect to previous related works, like the Middlebury stereo correspondence software [4], are: a graphical interface that allow a friendly interaction for selection and configuration of algorithms and its parameters and the avoidance of the use of scripts; it allows to perform a multitude of tests in very short time; graphical representation of results for easy analysis and comparison; use of different formats of input images, not constrained to the set of images distributed with the testbed; portability, architecture and operative system independence, as the testbed can been executed in different platforms without the recompiling the source code or configuring a runtime environment, representing one of the most important contributions; flexibility in the inclusion of algorithms written in different source languages, as well as easy connection with computer vision libraries such as OpenCV; openness of source code, so users can adapt both the algorithms and the testbed or its graphical interface to their concrete needs, and include new features, which represents a crucial aspect, as published algorithms may serve as inspiration for others, who can extend or adapt them to achieve more robust and optimized approaches over time, which the whole community can benefit from; scalability, as it has been designed, and it is expected, to evolve and include more algorithms, image processes, and functionality over time, receiving contributions from the research community as new works are published; once integrated, new algorithms can be easily tested and compared with others, analyzing its strengths and weaknesses to assess its contribution to the state of the art.

A use case has been presented to illustrate the utility and usability of the testbed. Several algorithms were tested to analyze their performance computing correspondences in a set of real stereo images, in contrast to using synthetic images. It allowed us to test many algorithms with different sets of parameters in a very short time, until satisfactory results were obtained, saving us from implementing several algorithms just for testing purposes, and representing a considerable saving of time and effort, demonstrating the utility of the testbed. Once a concrete algorithm was identified as the most suitable one for our concrete application, it was optimally implemented and adapted to be integrated within our robot onboard software. Also, several processes were combined to be executed consecutively, including a previous treatment on the input images and a filtering process following the matching phase, resulting in a major improvement of results.

For future versions of the testbed, it is expected to include new matching algorithms, image processes and functionality, such as test automation capabilities to arrange a series of tests to be executed without supervision, scheduling a test harness to be executed during low activity periods or by night. This way, lots of data from different tests can be gathered together for later analysis, automatically storing it to be presented in a structured mode to ease its analysis. Some automatic analysis capabilities are also envisioned, reporting evaluations, comparisons and elaborated conclusions from the raw gathered data. These features, along with a set of contributions already received from a number of authors, are currently being developed and integrated into the next version of the testbed.

Author Contributions: R.C. designed and built the testbed, performed the experiments, analyzed the data and wrote the paper. G.P. and J.J.R. assisted in the design of trials, algorithms selection and preparing the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Barnard, S.; Fishler, M. Computational stereo. *ACM Comput. Surv.* **1982**, *14*, 553–572. [CrossRef]
2. Cochran, S.D.; Medioni, G. 3-D surface description from binocular stereo. *IEEE Trans. Pattern Anal. Mach. Intell.* **1992**, *14*, 981–994. [CrossRef]
3. Correal, R. Matlab Central. 2012. Available online: http://www.mathworks.com/matlabcentral/fileexchange/36433 (accessed on 8 November 2016).
4. Scharstein, G.; Szeliski, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vis.* **2002**, *47*, 7–42. [CrossRef]
5. Tombari, F.; Gori, F.; Di Stefano, L. Evaluation of Stereo Algorithms for 3D Object Recognition. In Proceedings of the 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), Barcelona, Spain, 6–13 November 2011.
6. Ozanian, T. Approaches for Stereo Matching—A Review. *Model. Identif. Control* **1995**, *16*, 65–94. [CrossRef]
7. Grimson, W.E.L. Computational Experiments with a Feature-based Stereo Algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.* **1985**, *7*, 17–34. [CrossRef] [PubMed]
8. Ayache, N.; Faverjon, B. Efficient Registration of Stereo Images by Matching Graph Descriptions of Edge Segments. *Int. J. Comput. Vis.* **1987**, *1*, 107–131. [CrossRef]
9. Pajares, G.; Cruz, J.M.; López-Orozco, J.A. Relaxation Labeling in Stereo Image Matching. *Pattern Recognit.* **2000**, *33*, 53–68. [CrossRef]
10. Pajares, G.; Cruz, J.M. Fuzzy Cognitive Maps for Stereovision Matching. *Pattern Recognit.* **2006**, *39*, 2101–2114. [CrossRef]
11. Baker, H.H. Building and Using Scene Representations in Image Understanding. In *Machine Perception*; Prentice-Hall: Englewood Cliffs, NJ, USA, 1982; pp. 1–11.
12. Bradski, G.; Kaehler, A. *Learning OpenCV: Computer Vision with the OpenCV Library*; O'Reilly Media: Sebastopol, CA, USA, 2008.
13. Darabiha, A.; MacLean, W.J.; Rose, J. Reconfigurable hardware implementation of a phase-correlation stereoalgorithm. *Mach. Vis. Appl.* **2006**, *17*, 116–132. [CrossRef]
14. Lim, Y.K.; Kleeman, L.; Drummond, T. Algorithmic methodologies for FPGA-based vision. *Mach. Vis. Appl.* **2013**, *24*, 1197–1211. [CrossRef]
15. Gonzalez, R.C.; Woods, R.E. *Digital Image Processing*; Prentice-Hall: Englewood Cliffs, NJ, USA, 2008.
16. Pajares, G.; de la Cruz, J.M. *Visión Por Computador: Imágenes Digitales y Aplicaciones*; RA-Ma: Madrid, Spain, 2007; pp. 102–105.
17. Butterworth, S. On the Theory of Filter Amplifiers. *Wirel. Eng.* **1930**, *7*, 536–541.
18. Jensen, J.R. *Introductory Digital Image Processing*; Prentice-Hall: Englewood Cliffs, NJ, USA, 1982.
19. Gonzalez, R.C.; Wintz, P. *Digital Image Processing*; Addison-Wesly: Reading, MA, USA, 1987.
20. Barnea, D.I.; Silverman, H.F. A Class of algorithms for fast digital image registration. *IEEE Trans. Comput.* **1972**, *21*, 179–186. [CrossRef]
21. Shirai, Y. *Three-Dimensional Computer Vision*; Springer: Berlin, Germany, 1987.
22. Faugeras, O.; Keriven, R. Variational Principles, Surface Evolution, PDE's, Level Set Methods and the Stereo Problem. *IEEE Trans. Image Process.* **1998**, *7*, 336–344. [CrossRef] [PubMed]
23. Lankton, S. 3D Vision with Stereo Disparity. 2007. Available online: http://www.shawnlankton.com/2007/12/3d-vision-with-stereo-disparity (accessed on 8 November 2016).
24. Alagoz, B. Obtaining Depth Maps from Color Images by Region Based Stereo Matching Algorithms. *OncuBilim Algorithm Syst. Labs* **2008**, *8*, 1–13.
25. Ogale, A.S.; Aloimonos, Y. Shape and the Stereo Correspondence Problem. *Int. J. Comput. Vis.* **2005**, *65*, 147–162. [CrossRef]
26. Ogale, A.S.; Aloimonos, Y. A Roadmap to the Integration of Early Visual Modules. *Int. J. Comput. Vis.* **2007**, *72*, 9–25. [CrossRef]
27. Abbeloos, W. Stereo Matching. 2010. Available online: http://www.mathworks.com/matlabcentral/fileexchange/28522-stereo-matching (accessed on 8 November 2016).
28. Psarakis, E.Z.; Evangelidis, G.D. An enhanced correlation-based method for stereo correspondence with subpixel accuracy. In Proceedings of the Tenth IEEE International Conference on Computer Vision, Beijing, China, 17–21 October 2005.

29. Konolige, K. Small vision system: Hardware and Implementation. In Proceedings of the International Symposium on Robotics Research, Hayama, Japan, 19–22 October 1997; pp. 111–116.
30. Hirschmüller, H. Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Diego, CA, USA, 20–25 June 2005.
31. Kosov, S.; Thormaehlen, T.; Seidel, H.P. Accurate Real-Time Disparity Estimation with Variational Methods. In Proceedings of the International Symposium on Visual Computing, Las Vegas, NV, USA, 30 November–2 December 2009.
32. Cech, J.; Sara, R. Efficient Sampling of Disparity Space for Fast and Accurate Matching. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Minneapolis, MN, USA, 17–22 June 2007.
33. Scharstein, D.; Blasiak, A. Middlebury stereo evaluation site. 2014. Available online: http://vision.middlebury.edu/stereo/eval/ (accessed on 8 November 2016).
34. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. The KITTI Vision Benchmark Suite. 2014. Available online: http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=stereo (accessed on 8 November 2016).
35. Correal, R.; Pajares, G. Modeling, Simulation and Onboard Autonomy Software for Robotic Exploration on Planetary Environments. In Proceedings of the 2011 International Conference on DAta Systems in Aerospace, San Anton, Malta, 17–20 May 2011.
36. Correal, R.; Pajares, G. Onboard Autonomous Navigation Architecture for a Planetary Surface Exploration Rover and Functional Validation Using Open-Source Tools. In Proceedings of the ESA International Conference on Advanced Space Technologies in Robotics and Automation, Noordwijk, The Netherlands, 12–14 April 2011.
37. Correal, R.; Pajares, G.; Ruz, J.J. Stereo Images Matching Process Enhancement by Homomorphic Filtering and Disparity Clustering. *Rev. Iberoam. Autom. Inform. Ind.* **2013**, *10*, 178–184. [CrossRef]