Adaptive SLAM with Synthetic Stereo Dataset Generation for Real-time Dense 3D Reconstruction

Antoine Billy[1,2], Sébastien Pouteau[1], Pascal Desbarats[1], Serge Chaumette[1] and Jean-Philippe Domenger[1]

[1]*Laboratoire Bordelais de Recherches en Informatique, Université de Bordeaux, France*

[2]*Innovative Imaging Solutions, Pessac, France*

Keywords: SLAM, Stereo Vision, Synthetic Dataset, Alastor, Adaptive Frame Rate Selection.

Abstract: In robotic mapping and navigation, of prime importance today with the trend for autonomous cars, simultaneous localization and mapping (SLAM) algorithms often use stereo vision to extract 3D information of the surrounding world. Whereas the number of creative methods for stereo-based SLAM is continuously increasing, the variety of datasets is relatively poor and the size of their contents relatively small. This size issue is increasingly problematic, with the recent explosion of deep learning based approaches, several methods require an important amount of data. Those multiple techniques contribute to enhance the precision of both localization estimation and mapping estimation to a point where the accuracy of the sensors used to get the ground truth might be questioned. Finally, because today most of these technologies are embedded on on-board systems, the power consumption and real-time constraints turn to be key requirements. Our contribution is twofold: we propose an adaptive SLAM method that reduces the number of processed frame with minimum impact error, and we make available a synthetic flexible stereo dataset with absolute ground truth, which allows to run new benchmarks for visual odometry challenges. This dataset is available online at http://alastor.labri.fr/.

1 INTRODUCTION

The ability for a machine to navigate autonomously has become a central challenge for nowadays robotic science. Building a model of the surrounding world and having the capacity to work out its location (position and orientation) in the generated map allows a robot to ”understand” information of an unknown area. This feature is called Simultaneous Localization And Mapping (SLAM). A machine with such a skill can estimate a path, provide ergonomic visualization for human users and even reset its localization estimation by revisiting known areas. Its application for autonomous vehicles such as cars (Geiger et al., 2012; Howard, 2008; Lategahn et al.,) or UAV (Karlsson et al., ; Cvišić et al., 2017; Caballero et al.,), has been illustrated many times over. To be aware of its environment, the robot must obviously be equipped with sensors. Several types of sensors give birth to different SLAM algorithms. Although the state of the robot is often described in similar ways (its position and rotation, sometimes with additional information such as its velocity, sensor orientation or biases, battery or any other accessible information), the map representation is still debated. They go from sparse representations with only landmarks, connected graphs or obstacles to dense (Geiger et al., 2011) or semi-dense (Engel et al., 2014) reconstruction with (*e.g.* stereo vision) or without colour information (*e.g.* most LiDAR systems).

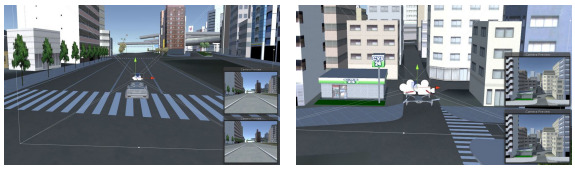
Tactile sensors outputs are extremely sparse as they contain only information about points very close to the agent, so they require strong prior models to compensate. Thus, most modern methods use laser scans or visual features. They provide details of many points within an area, sometimes rendering SLAM inference unnecessary because shapes in these clouds of points can be easily and unambiguously aligned at each step via image registration. Optical sensors may be one-dimensional or 2D laser range-finders, 3D High Definition LiDAR, 2D or 3D sonar sensors, one or more 2D cameras. Since 2005, there has been intense research into visual SLAM primarily using cameras, because of the increasing availability of cameras such as those in mobile devices or UAV. Moreover, the use of two 2D cameras to mimic the human stereo vision system has become very popular in a great number of robotic applications. Besides being relatively cheap systems compared to LiDAR or sonar based approaches, stereo-based methods possess the huge advantage of avoiding scale drift which typically occurs in monocular SLAM. They have the

Billy, A., Pouteau, S., Desbarats, P., Chaumette, S. and Domenger, J.
Adaptive SLAM with Synthetic Stereo Dataset Generation for Real-time Dense 3D Reconstruction.
DOI: 10.5220/0007386508400848
In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2019)*, pages 840-848
ISBN: 978-989-758-354-4

ability to quickly generate a dense point cloud estimation. These two main advantages have made stereo based systems a vast field of research and a well-established approach for SLAM algorithms (Howard, 2008; Cvišić et al., 2017; Geiger et al., 2011; Engel et al., ; Kuschk et al., 2017; Tanner et al., ; Sanfourche et al., ; Kaess et al., ; Zhu, 2017; Pire et al., 2018; Cvišić and Petrović, 2015; Kitt et al., 2010).

These algorithms use stereo visual odometry to solve the SLAM problem as accurately as possible, and the results of most of them are really impressive. Due to their objective, SLAM algorithms are often embedded on on-board systems, making them truly autonomous. However, the robots have their own limitations in terms of memory, computation power and battery. In that respect, whereas most of modern SLAM methods tend to exploit every available piece of computational power to improve their accuracy, our first contribution in this paper is to suggest an adaptive SLAM algorithm that modulates its input frame rate to avoid unnecessary computation. Saving time, power and thus making the autonomy of the system that embeds it.

Despite the popular success of stereo based SLAM, the number of available benchmarks on which they can be tested is relatively poor. Indeed, building such a real-world dataset with a high precision level can be an extremely time-consuming task and requires complex equipment and consequently expensive human work. And even then, the ground truth is highly sensitive to the manual labelling method that is used and/or the precision of the sensors. This turns out to be a real issue because of the constantly increasing precision of new SLAM algorithms: their evaluation requires more precise ground truth to which their results could be compared. Finally, aiming at proving the efficiency of our adaptive SLAM method, a dataset with tunable parameters such as modular frame rate or customizable resolution was required, which unfortunately did not exist. The second contribution of this paper is thus the presentation of a synthetic benchmark generation tool that we have designed and developed: Alastor. In addition to fulfilling the previously unavailable requirements described above, it offers two major advantages compared to real-world datasets: 1. it comes with a perfect ground truth and it ensures no error coming from the sensors; 2. it eases the generation of an extensive amount of stereo data, which is an insatiable request from neural networks based methods that become more and more popular. An example of the Alastor output is shown on figure1.



(a) *(b)*

Figure 1: Alastor simulator engine output. (a) Simulation of a stereo system embedded on a car. (b) same system on an UAV.

2 RELATED WORKS

As mentioned in section 1, a large number of methods that use stereo visual odometry already exist. These methods build a 3D map of their environment in real-time. Among them, three main categories can be distinguished:

1. Sparse map generation methods (Visual-SLAM (Cvišić et al., 2017; Cvišić and Petrović, 2015));
2. Semi-dense map generation methods, (LSD-SLAM or ORB-SLAM (Engel et al., 2014; Mur-Artal et al.,));
3. Dense map generation methods (StereoScan (Geiger et al., 2011)).

Among the several stereo datasets that have been created for training and evaluating stereo algorithms, only a few of them are commonly used. Middlebury stereo dataset (Baker et al., 2011) is widely used for indoor scenes; it provides high-resolution stereo pairs with dense disparity ground truth. However it does not offer any data usable for SLAM algorithms, as all the given images are totally independent. The KITTI stereo dataset (Geiger et al., 2012) is a benchmark consisting of urban video sequences with semi-dense disparity ground truth along with semantic labels. The EuRoC MAV dataset (Burri et al., 2016) presents two visual-inertial datasets containing stereo images collected on-board on a Micro Aerial Vehicle (MAV).

Because of the complex equipment and expensive human ressource required to build them, these three real-world datasets have relatively small sizes: for instance, the largest one, the KITTI dataset, has only aounrd 400 labelled stereo pairs in total for public use. Another disadvantage of real-world datasets is the limited precision of 3D sensors and LIDAR that prohibits high-quality ground truth. Finally, because of physical limitations of acquisition and recording, the frame rates of these datasets are relatively low (around 10 frames per second) compared to actual cameras. This was a major drawback to evaluate our adaptive

SLAM proof of concept. Additionally, with the recent interest in deep convolutional neural networks to resolve the SLAM problem, the actual amount of data is unsatisfactory for those greedy methods.

For all the reasons described above, the need for synthetic stereo datasets arose. Virtual KITTI (Gaidon et al.,) offers a dataset automatically labelled with accurate ground truth for object detection, tracking, scene and instance segmentation, depth acquired by LiDAR, and optical flow. They unfortunately do not allow users to extract any stereo data, making it strongly powerful for monocular SLAM method, but irrelevant for our context. On the other hand, Unreal stereo (Zhang et al.,) and (Mayer et al.,) offer a real answer to synthetic stereo data generation. However, none of them provides any consecutive frame, and they are thus ineffective to achieve SLAM. Our dataset is therefore, to our knowledge, the only synthetic stereo dataset that could be used for SLAM algorithms.

3 ADAPTIVE SLAM METHODOLOGY

This section presents the adaptive SLAM method that we have created and its integration in the pipeline for 3D dense reconstruction.

3.1 SLAM Optimization

As shown in the previous section, actual slam methods are mostly based on consecutive pairwise analysis, what means that they directly rely on the camera frame rate. Let us consider an image sequence composed of frames $f_0, f_1, \ldots, f_n$. We will note f_i the i^{th} frame. The objective of a SLAM algorithm is to find the transformation matrix M (rotation and translation) that allows a vector to go from f_i to f_{i+1}:

$$M = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & R & & T \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

With r_{ab} the rotation coefficients, t_a the translation coefficients o the corresponding axis. R and T are thus respectively the Rotation and Translation matrix. We can derive from the R matrix the rotation angle in each dimension r_x, r_y, r_z. The role of a SLAM algorithm is to estimate these matrices so that:

$$f_{i+1} = R \times f_i + T$$

With f_i the set of feature points extracted for the i^{th}frame. In order to ensure the highest possible precision, recent methods include every frame in the process. Indeed, minimizing the distance between two consecutive frames reduces the matrix estimation error. Moreover, the feature selection process from two close frames ensure a large number of matching landmarks, which is suitable for a robust path estimation. However, such a computation is both time and power consuming. With the advent of embedded systems such as mobile phones or small unmanned vehicles, autonomy became a strong parameter to take into consideration for SLAM applications, in addition to real-time process requirements. It is thus unthinkable to deal with a 30 fps frame rate within an on-board system still fulfilling all the requirements. To make things possible, the frame rates are strongly reduced, compared to standard acquisition devices, ineluctably incorporating biases in the process.

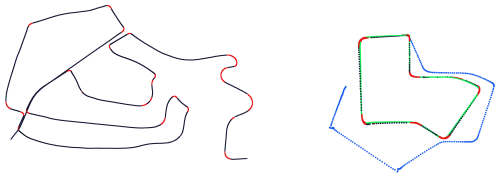
It is a known fact that rotations are the actual reason why SLAM is still an actively searched problem (Carlone et al., ; Pirchheim et al.,). Rotation estimation (*a.k.a.* rotation averaging) has been widely studied in computer vision, where accurate camera orientation estimation is critical to solve Bundle Adjustment in Structure from Motion. It has also been investigated in the control theory community, where it has applications for vehicle coordination, sensor networks localization, camera network calibration and more broadly in robotic science. The idea that we suggest is thus to reduce the number of frames processed while the trajectory remains straight, in order to reduce power consumption to increase processing speed, and to measure the frame rate when the robot is carrying out a rotation. Instead of considering the whole set of images, we only consider a subset of it as long as the trajectory stays straight. Let σ be the number of skipped frames and ε a minimum rotation angle threshold. If at the i^{th} frame the detected rotation angle is above the threshold, we process the subset $f_{i-\sigma}$ to f_i back into the SLAM algorithm. Doing so, we strongly reduce the total number of processed frame still minimising the critical errors in the dangerous rotation parts. The pseudo-code of this simple process is available on figure 1.

With this method, two parameters have to be considered: the number of skipped frames σ and the rotation threshold ε. These two factors are strongly dependent on the original frame rate of the sensor, the current speed of the vehicle when turning. So to avoid the laborious task of parameters adjustment, we have developed an adaptive method the role of which is to continuously adjust σ and ε during the exploration process.

Algorithm 1: Adaptive SLAM frame selection pseudo code.

```
1:  procedure ADAPTIVESLAM
2:  i ← 0
3:  σ ← 1
4:      while i < n do:
5:      (r_x, r_y, r_z, t_x, t_y, t_z) ←
    computeSLAM(f_{i−σ}, f_i)
6:      if r_x > ε or r_y > ε or r_z > ε then
7:          for j ← [i − σ + 1, i] do:
8:                                  (r_x, r_y, r_z, t_x, t_y, t_z) ←
    computeSLAM(f_{j−1}, f_j)
9:              save(r_x, r_y, r_z, t_x, t_y, t_z)
10:         σ ← 1
11:     else
12:         save(r_x, r_y, r_z, t_x, t_y, t_z)
13:         σ ← σ + 1
14:     i ← i + σ
```

As shown in figure 2 (*a*), a less energy consuming frame rate is applied to most of the trajectory (black dots) and our method re-samples to a higher frame rate when a rotation occurs (red dots). The structure of the trajectory is conserved, still a significant number of frames have been skipped. More information about these numbers are given in the results section of this paper (section 5). Figure 2 (*(b)*) illustrates the differences between the estimation of the trajectory with and without Adaptive SLAM with the same number of processed frames.The blue doted line is the output of SLAM algorithm with a naive reduced frame rate. The black and red doted line is the output of AdaptiveSLAM algorithm with the same number of dots. The green doted line corresponds to the ground truth delivered by KITII. This example clearly emphasizes the need for high frame rates in the rotation areas, and its non necessity for straight lines.



(a) *(b)*

Figure 2: Illustration of Adaptive SLAM frame selection. The black+red points represent the trajectory given by adaptive SLAM. We can easily see in *(a)* that the curves are processed with a highest frame rate than the straight lines. *(b)* illustrates in blue what would happen if we keep the reduced frame rate for the entire sequence. The computed trajectory strongly spreads off the represented ground truth (green dots), whereas the adaptive SLAM remains very close to it.

This method allows us to quickly get a coarse estimation of the transformation between two selected frames. In the next subsection, we will use the output of the adaptive SLAM algorithm to generate a 3D dense reconstruction of the world surrounding the cameras.

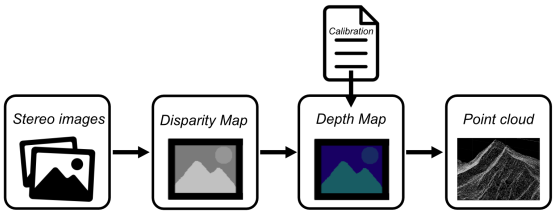


Figure 3: Pipeline to generate a cloud of points from a stereo image pair.

3.2 A Pipeline for 3D Dense Reconstruction in Real-time

In this section we present a full pipeline for real-time 3D dense reconstruction from stereo visual odometry, without any other sensors than our two calibrated cameras.

The very first step of the pipeline consists in building an initial point cloud at a time t on which the other generated clouds will be grafted. To do so, we preliminary have to calibrate the cameras. This step is an essential requirement of the pipeline. This calibration process will generate intrinsic parameters (focal length, distortion, principal points and axis skew) and extrinsic parameters(baseline length, translation and rotation matrices from a camera to the other). A standard procedure to calibrate the camera is to use a chessboard pattern. An average of significant number of different poses is used to get better calibration results. This calibration is a mandatory step for the rectification process used to project the two images onto a common image plane. We can then apply a stereo vision algorithm to build a disparity map. We decided to compute our maps using an implementation of the so-called Semi-Global Block Matching SGBM from (Hirschmuller, 2006). Using the previously computed calibration parameters, we can then integrate the disparity map into a depth map. We can eventually extract a point cloud from the depth map. This process is illustrated in figure 3.

One of the main advantages of this method is that the point cloud is generated directly from the image, and every 3D point is associated to one pixel in the image. This provides a quick method to build a dense 3D point cloud in real-time scenarios. A critical drawback is that the quality of the generated cloud is really sensitive to the calibration process. This step should thus be done with great precision in order to limit noise generation, but this is out of the scope of this paper. Once the initial point cloud has been gene-

rated, we apply the proposed adaptive SLAM method to the input images sequence. From the trajectory estimation, we then extract a key frame and compute its corresponding point cloud. the key frame is selected when a displacement threshold has been reached. After a filtering step, the transformation matrix estimated by the adaptive SLAM algorithm is applied to the new point cloud and merged within the global cloud, giving suitable estimation. Then, in order to minimize the difference between the global point cloud and the new one, an Iterative Closest Point (ICP) algorithm (Chetverikov et al.,) is run to refine the final registration. This step is done simultaneously with the adaptive SLAM algorithm so it does not slow down the global pipeline (figure 4). Results will be presented in section 5.

4 ALASTOR DATASET

As shown in the related work section, only a few datasets exist for stereo vision based odometry. And even then, the volume of data they provide remains limited due to the hard task of collecting such data in a real-world scenario (Geiger et al., 2012; Baker et al., 2011; Burri et al., 2016). Moreover, existing synthetic datasets are ineffective for stereo (Gaidon et al.,) or SLAM (Zhang et al., ; Mayer et al.,) scenarios. As the first one only provides images from mono camera, and the second does not provide any images sequence mandatory to apply SLAM algorithms. The second main contribution of this article is thus to offer the first, to our knowledge, synthetic stereo dataset for SLAM algorithms.

To generate such a dataset, several solutions exist in the literature. The Modular OpenRobots Simulation Engine (MORSE) (Echeverria et al.,) seemed to fulfill most of the needed requirements to perform our simulation. MORSE is an open-source software which offers an extended set of predefined sensors and controllers that cover reasonably well common simulation needs. Nevertheless, even though the camera physics were ideal for our needs, the vector manipulation lacked of realism. We thus decided to implement our own solution using Unity 3D[1], a powerful cross-platform engine we are familiar with. Still, we should give a closer look to MORSE in our future works.

We developed a solution which allows users to load any 3D environment model (forest, city, road, *etc*.) and to insert one or several vectors (car, UAV, boat, person, *etc*.) equipped with a stereo acquisition system. The simulation lets the user play with a large number of parameters (resolution, baseline, Field of View, baseline, FPS). Then, with a regular game controller, the user simply wanders inside its 3D scene (see figure 1), letting the simulation capture the stereo images and record the ground truth trajectory. The generated dataset can then be downloaded and given as input to any SLAM algorithm, with a perfect ground truth to compare with. As shown in figure 5 *(a)*, the simulation also enables the user to generate a calibration scene.

The synthetic datasets that we generated offer the following advantages:

1. Perfect ground truth.
2. Complete vehicle parametrization (dimensions, speed, turning angle, ability to fly).
3. Full acquisition system parametrization (resolution, frame rate, baseline).
4. Multi-scale and multi-scene adaptability.

The figure 5 shows an example of a 3D model of a city used in the Alastor simulation. The full dataset is available on-line at http://alastor.labri.fr/ The figure 6 represents 10 simulated drives, from very simple straight trajectories to more complex scenarios.

Our architecture makes it possible to easily tune acquisition parameters. We have included a rain simulator to deal with noisy particles as well as a daylight manipulation tool allowing to work with illumination changes. An output example is given in fig. 7 *(a,b)*.

To go even further, a rotary LiDAR has been added in the simulation, as shown in fig. 8. Once more, as the datasets are fully synthetic, the precision of the generated cloud of points is strongly higher than a real sensor subject to real-life acquisition noise.

5 RESULTS

In order to evaluate our model, we tested the proposed adaptive SLAM algorithm on the so-called KITTI dataset (Geiger et al., 2012) and on the Alastor dataset. The SLAM algorithm on top of which is implemented the Adaptive SLAM algorithm is generated by libviso2 (Geiger, 2012), an open source library for stereo visual odometry. The goal of the following results is to demonstrate that the same results are achievable with less resources. We do not focus on precision improvement here. The following results will consider the computed trajectory given by libviso2 as the ground truth for our adaptive SLAM estimation. Thanks to the Alastor dataset, our results are obtained on a high frame rate (50fps) for all the synthetic data,

[1]Unity 3D Engine: https://unity3d.com/

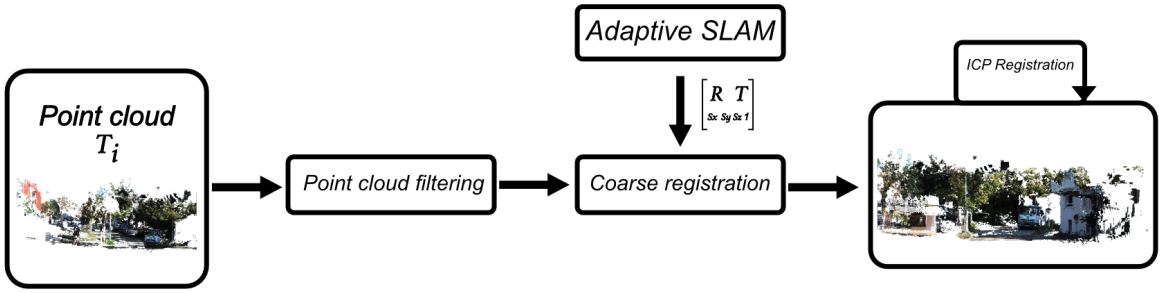
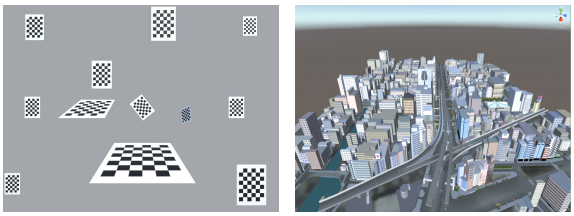


Figure 4: Pipeline of global 3D dense reconstruction.



(a) *(b)*

Figure 5: Alastor simulation example scenes. *(a)* Example of the calibration scene used to compute the intrinsic and extrinsic parameters of the stereo cameras. *(b)* Example of a 3D model used as the global scene in Alastor simulation.

whereas the KITTI dataset was used at its maximum of 10 fps.

We used two main metrics to evaluate the performances of our method: the first one is the time saved between libviso2 SLAM and our adaptive SLAM expressed in seconds. The evaluation have been run on a i7-4770 CPU with 16Go of RAM. The second metrics is the error Δ generated by our adaptive SLAM. The error represents the average error with respect to the total travelled distance (as a percentage):

$$\Delta = \frac{\sum_{i=0}^{n} \sqrt{(x_i - x_i')^2 + (y_i - y_i')^2 + (z_i - z_i')^2}/n}{\sum_{j=1}^{n} \sqrt{(x_j - x_{j-1})^2 + (y_j - y_{j-1})^2 + (z_j - z_{j-1})^2}}$$

where n is the total number of processed frames, x, y, z are the 3D coordinates given by libviso2 for a given frame i. x', y', z' are the 3D coordinates given by the Adaptive SLAM for the same frame. We first tested our algorithm on the KITTI Dataset.

5.1 Evaluation based on the KITTI Dataset

Due to the frame rate used in the KITTI datasets, we set the parameters to $\varepsilon = 0.01$ and $\sigma_{max} = 4$. This way, we ensure a low frame rate at 2,5 fps with a maximum speed of the vehicle at 30 km/h. The results using the KITTI dataset are illustrated on an histogram figure 9. The black bar corresponds to the execution time of the SLAM algorithm of libViso2. The blue one corresponds to the execution time of the Adaptive SLAM algorithm. As it can be seen on the histogram, the error rate is very low (the highest error is 0.34%) with saved time of 32.8% on average. These values represent considerable gains for embedded systems.

5.2 Evaluation based on the Alastor Dataset

On the basis of the results on the KITTI dataset, we then tested the Adaptive SLAM algorithm on the Alastor dataset. Due to the used frame rate in the simulation, we set our parameters to $\varepsilon = 0.01$ and $\sigma_{max} = 10$. This way, we ensure a low frame rate at 5 fps with a maximum vehicle speed of 60 km/h. The number of skipped frames and the corresponding error are displayed on figure 10. The average saved time is 73.8%. This good result is due to the original frame rate of the simulation. In the other hand, the maximum generated error reached 1.75% for the dataset number 00. This is mainly due to a bad estimation of the elevation close to the first frames.

Tested on a i7-4770 computer with 16Go of RAM, a regular dataset of 1000 frames with a resolution of 1392 x 512 pixels (similar to KITTI (Geiger et al., 2012)) takes around 420s (7 minutes) with libviso2 raw algorithm. Three scenarios can be considered with our Adaptive SLAM algorithm:

1. **Worst Case Scenario:** The trajectory is entirely composed of rotations: no single frame is skipped, the computation time thus remains thus the same (7 minutes).
2. **Regular Case:** The trajectory is composed of rotations and straight lines. In average, the execution time is shorten by 30%. The whole SLAM generation takes 4 minutes and 54s.
3. **Optimal Case:** The trajectory is a single straight line. The percentage of skipped frames is in this case is around 90%. The execution time is 42 seconds. This situation occurs, for instance, in one of the KITTI datasets see figure 11.

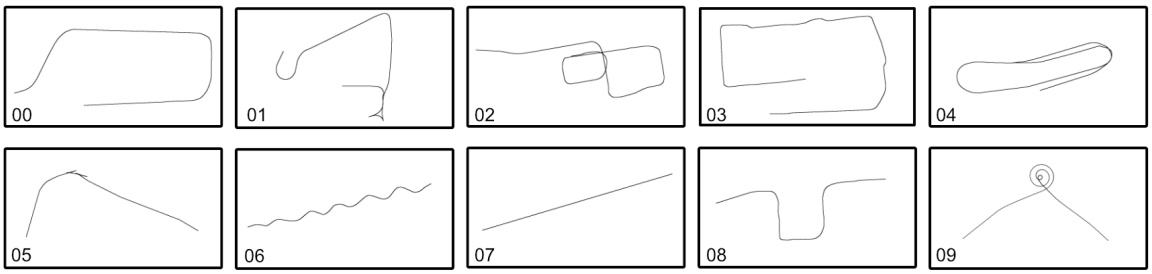
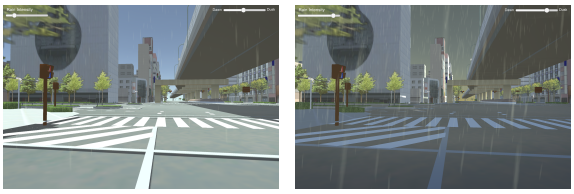


Figure 6: Some simulated trajectories available in the Alastor dataset. These 10 datasets correspond to scenarios with different complexity levels. The results of the adaptive SLAM algorithm on these datasets are shown in the next section.



(a) *(b)*

Figure 7: Alastor simulation engine outputs. With different weather and lighting conditions: *(a)* Simulation during daylight with light rain, *(b)* during dawn with strong rain.

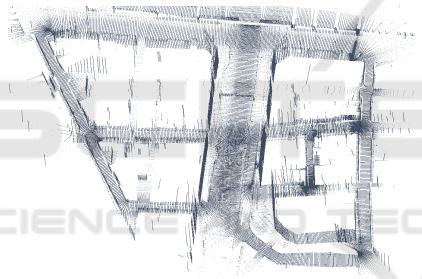


Figure 8: Example of a cloud of point generated by the rotary LiDAR. Thanks to the synthetic nature of the datasets, its reliability is very high.

To ensure the stability of the Adaptive algorithm, it was tested on more complex scenarios. For instance one of the biggest datasets available in KITTI. The corresponding trajectory is shown on figure 12 *(a)*. This examples highlights the efficiency of the frame selection process. Indeed, most of the straight parts are sparse (black dots) whereas the rotation parts are more dense (red dots). This phenomenon is even more noticeable in the simulated spinning trajectory of figure 12 *(b)*. This dataset clearly put forward the process of the adaptive frame selection.

The Adaptive SLAM computes a coarse estimation of the movements of the camera from one frame to an other. The final merge between the current point cloud and the previously computed global one is consistent. With a precise enough initialization, ICP (Chetverikov et al.,) algorithm perfectly merges the input clouds. As no ground truth exists for such a dense generated cloud, this observation may

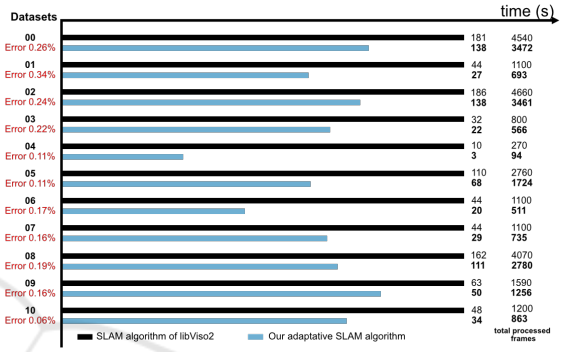


Figure 9: Evaluation of adaptive SLAM algorithm compared to libviso2 on the KITTI dataset. This array clearly highlights the saved computational time on real data. It also confirms that the error is negligible in comparison.

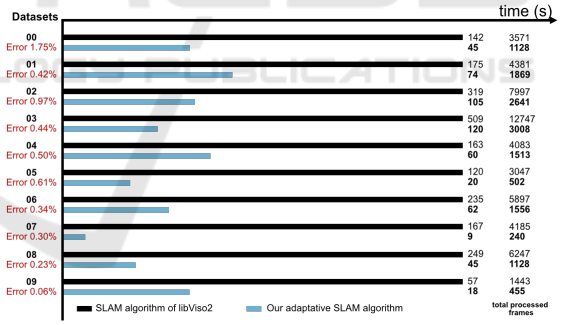


Figure 10: Evaluation of adaptive SLAM algorithm compared to libviso2 on the Alastor dataset. The gain is here even more impressive.

only be made subjectively. As displayed in figure 13, the trajectory (in red) fits perfectly the generated point cloud. A zoom inside it allows us to observe more closely the efficiency of the registration.

In order to ease the estimation of the readers, two videos are provided with this paper as supplemental materials, one for each generated point cloud.

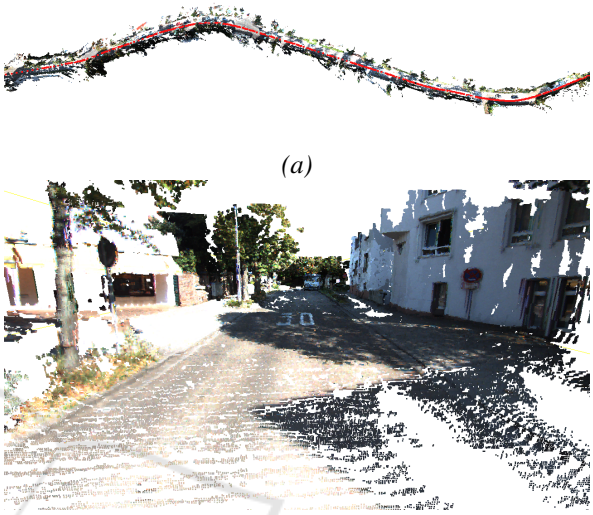
Figure 11: Illustration of the number of skipped frame. The two lines represent the same trajectory. The upper one is given by libviso2 whereas the lower one is generated by adaptive SLAM. This trajectory is extracted from a real KITTI sequence.

Figure 12: Comparison between the output of our adaptive SLAM algorithm and the ground truth of one of the KITTI datasets. Yellow lines correspond to the perfect ground truth computed during the simulation. Black and red dots are the adaptive SLAM output.

6 CONCLUSION

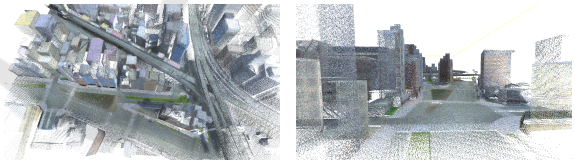
In this paper, we have presented an optimization of SLAM algorithms that reduces the number of processed frames without increasing the resulting error. We have shown that this method significantly increases performances in real-time scenarios, and that it is applicable to well known datasets. We used Adaptive SLAM algorithm within a 3D dense reconstruction pipeline and we have shown that the results are perfectly suitable for dense reconstruction. Additionnaly, to support the validation of our concept and to answer the needs for stereo datasets for SLAM evaluation, we made available, what is to our knowledge, the first synthetic stereo dataset for SLAM applications: Alastor. The experiments that we have conducted have shown that adaptive SLAM performs great on simulated datasets. We hope that our dataset will help design and evaluate innovative methods that will take advantage of the tunable parameters in order to improve the effectiveness of SLAM solving algorithms in the future.

In terms of future work, we are also considering the integration of loop-closure algorithm in the reconstruction pipeline. This problem consists in detecting when the robot has returned to a past location after having discovered new terrain for a while. Such methods are able to greatly reduce the accumulated error by adjusting the generated cloud once a loop has been noticed. We also intend to improve the Alastor dataset by diversifying both its content and the sensors.



(a) *(b)*

Figure 13: 3D dense reconstruction based on the KITTI dataset. *(a)* Full 3D dense point cloud generated using the KITTI dataset. The red points show the trajectory computer by our adaptive SLAM. *(b)* Zoom inside the aside point cloud. Here, several depth maps have been merged to create a unique cloud.



(a) *(b)*

Figure 14: 3D dense reconstruction of Alastor dataset. *(a)* is a fully merged 3D dense point cloud generated from Alastor dataset overflight by drones. *(b)* is the Same point cloud but from an other point of view. The overlapping areas are totally invisible thanks to ICP (Chetverikov et al.,).

REFERENCES

Baker, S., Scharstein, D., Lewis, J. P., Roth, S., Black, M. J., Szeliski, R., Baker, S., Szeliski, ·. R., Szeliski, R., Scharstein, D., Roth, S., and Black, M. J. (2011). A Database and Evaluation Methodology for Optical Flow. *Int J Comput Vis*, 92:1–31.

Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M. W., and Siegwart, R. (2016). The EuRoC micro aerial vehicle datasets. *The Inter-*

national Journal of Robotics Research, 35(10):1157–1163.

Caballero, F., Merino, L., Ferruz, J., And, A. O. J. o. I., and 2009. Vision-based odometry and SLAM for medium and high altitude flying UAVs. *Springer*.

Carlone, L., Tron, R., Daniilidis, K., and Dellaert, F. Initialization Techniques for 3D SLAM: a Survey on Rotation Estimation and its Use in Pose Graph Optimization.

Chetverikov, D., Svirko, D., Recognition, D. S. P., and 2002. The trimmed iterative closest point algorithm. *ieeexplore.ieee.org*.

Cvišić, I., Ćesić, J., Marković, I., and Petrović, I. (2017). SOFT-SLAM: Computationally Efficient Stereo Visual SLAM for Autonomous UAVs. *Journal of field robotics*.

Cvišić, I. and Petrović, I. (2015). Stereo odometry based on careful feature selection and tracking. In *Mobile Robots (ECMR), 2015 European Conference on*, pages 1–6. IEEE.

Echeverria, G., Lassabe, N., Robotics, A. D., Automation, and 2011. Modular open robots simulation engine: Morse. *Citeseer*.

Engel, J., Schöps, T., and Cremers, D. (2014). LSD-SLAM: Large-Scale Direct Monocular SLAM. pages 834–849.

Engel, J., Stückler, J., , Systems, D. C. I. R., and 2015. Large-scale direct SLAM with stereo cameras. *ieeexplore.ieee.org*.

Gaidon, A., Wang, Q., Cabon, Y., and Vig, E. Virtual Worlds as Proxy for Multi-Object Tracking Analysis.

Geiger, A. (2012). Libviso2: C++ library for visual odometry 2, www.cvlibs.net/software/libviso/.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE.

Geiger, A., Ziegler, J., and Stiller, C. (2011). Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 963–968. Ieee.

Hirschmuller, H. (2006). Stereo vision in structured environments by consistent semi-global matching. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2386–2393. IEEE.

Howard, A. (2008). Real-time stereo visual odometry for autonomous ground vehicles. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3946–3952. IEEE.

Kaess, M., Ni, K., , Automation, F. D. R., 2009, and 2009. Flow separation for fast and robust stereo odometry. *ieeexplore.ieee.org*.

Karlsson, R., Schon, T., Aerospace, D. T., and 2008. Utilizing model structure for efficient simultaneous localization and mapping for a UAV application. *ieeexplore.ieee.org*.

Kitt, B., Geiger, A., and Lategahn, H. (2010). Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 486–492. IEEE.

Kuschk, G., Bozic, A., and Cremers, D. (2017). Real-time variational stereo reconstruction with applications to large-scale dense SLAM. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1348–1355. IEEE.

Lategahn, H., Geiger, A., (ICRA), B. K. R., Automation, and 2011. Visual SLAM for autonomous ground vehicles. *ieeexplore.ieee.org*.

Mayer, N., Ilg, E., Häusser, P., Fischer, P., Cremers, D., Dosovitskiy, A., and Brox, T. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation.

Mur-Artal, R., on JMM Montiel IEEE Transactions, and 2015. ORB-SLAM: a versatile and accurate monocular SLAM system. *ieeexplore.ieee.org*.

Pirchheim, C., Mixed, D. S., Augmented, and 2013. Handling pure camera rotation in keyframe-based SLAM. *ieeexplore.ieee.org*.

Pire, T., Baravalle, R., D'Alessandro, A., and Civera, J. (2018). Real-time dense map fusion for stereo SLAM. *Robotica*, pages 1–17.

Sanfourche, M., , Robots, V. V. I., and 2013. evo: A real-time embedded stereo odometry for mav applications. *ieeexplore.ieee.org*.

Tanner, M., Piniés, P., Paz, L. M., and Newman, P. DENSER Cities: A System for Dense Efficient Reconstructions of Cities.

Zhang, Y., Qiu, W., Chen, Q., Hu, X., arXiv preprint ArXiv:1612.04647, A. Y., and 2016. Unrealstereo: A synthetic dataset for analyzing stereo vision. *arxiv.org*.

Zhu, J. (2017). Image Gradient-based Joint Direct Visual Odometry for Stereo Camera. In *Int. Jt. Conf. Artif. Intell*, pages 4558–4564.