A Robust Approach for 3D Cars Reconstruction

Adrien Auclair[1], Laurent Cohen[2], and Nicole Vincent[1]

[1] CRIP5-SIP, University Paris-Descartes,
45 rue des Saint-Pères, 75006 Paris, France
{adrien.auclair,nicole.vincent}@math-info.univ-paris5.fr
[2] CEREMADE, University Paris-Dauphine,
Place du Maréchal De Lattre De Tassigny 75775 PARIS, France
cohen@ceremade.dauphine.fr

Abstract. Computing high quality 3D models from multi-view stereo reconstruction is an active topic as can be seen in a recent review [15]. Most approaches make the strong assumption that the surface is Lambertian. In the case of a car, this hypothesis is not satisfied. Cars contain transparent parts and metallic surfaces that are highly reflective. To face these difficulties, we propose an approach to robustly reconstruct a 3D object in translation. Our contribution is a one-dimensional tracker that uses the vanishing point computed in a first pass. We applied it to video sequences of cars recorded from a static camera. Then, we introduce a local frame for the car and use it for creating a 3D rough model. The final result is sufficient for some applications where it is needed to estimate the size of the vehicle. This model can also be used as an initialization for more precise algorithms.

Key words: Structure From Motion, Feature Tracking, Surface Fitting, RANSAC

1 Introduction

Automatic solutions for creating 3D digital representation of objects are needed in many domains. A practical approach is to use a video as input. Having a moving camera recording a static scene [14] can be useful to build a 3D model of outdoor scenes, or object like statues in archeology. In this article, we deal with the dual approach of having a static camera recording a moving rigid object. Some articles use this configuration to build a high quality model of an object that rotates on a turn-table in front of a static camera [5]. Many methods perform well in the case the object has a rich enough texture and does not contain much reflection or transparency. The recent review in [15] compares several algorithms that give sub-millimeter precision with these conditions.

Most of the structure-from-motion algorithms make the hypothesis that the object surface is Lambertian. This ensures that two images taken from two lightly different points of view are not very different. With this hypothesis, and assuming a rich texture on the surface object, a Lucas-Kanade based point tracker coupled with robust pose estimation method (e.g., RANSAC [6]) and non linear

minimization (e.g., Bundle Adjustment step [18]) leads to good quality 3D point cloud. Several books detail precisely these methods ([9],[13]).

Motivated by some industrial application, we chose to reconstruct a 3D model of a car. Cars are challenging objects because the Lambertian assumption is strongly violated, parts of the objects are transparent and large parts have no texture. The lights for example are highly reflective surfaces behind a transparent plastic surface. The wheels are parts of the object that are moving relatively to the car. This leads to many difficulties for 3D reconstruction without priors. A feature tracker would not be able to make any difference between a moving part of the vehicle and an object of the environment reflected on the car surface. If there is much reflection, the number of outliers among the tracked points could exceed 50 percent. Moreover, many steps of tracking or reconstruction measures are based on photo-consistency over time. For example, the Lucas-Kanade feature tracker [17] assumes that a window around a point remains constant within a small interval of time. In some recent reconstruction algorithm, the depth of a point is correct if its corresponding 2D positions in movie frames have a high Normalized Cross Correlation score ([7],[5]). This photo-consistency measure cannot be used on cars because of highly reflective parts and transparency.

Our new approach is to use a two-pass features tracking to analyze the motion. The presented method can be used for building a 3D point cloud from video sequence of an object in translation in front of a static camera. The translation hypothesis makes the algorithm robust by using the vanishing point and achieves to track much more features. Then, we show that for cars, results still contain outliers. We propose methods to filter the 3D point cloud. Our second contribution is an approach to compute a local 3D frame for the car. Then, this local frame is used to fit the point cloud to a simple 3D car model. Figure 1 shows some images of an input sequence we used. The overall advantage of our method is to robustly build an approximate 3D model whereas other methods could fail because of the reflections and the transparencies.

Fig. 1. Frames 20,40,60 and 100 of an input movie.

2 Structure from Motion for an Object in Translation

In the following section, we introduce a one dimensional feature tracker. Compared to a traditional Lucas-Kanade based feature tracker, this two-pass tracker

is more robust and tracks many more points. The output is used in a classical structure from motion algorithm. At first, a two-views reconstruction is established and then, other views are added iteratively, in a manner close to [14]. Internal parameters of the camera are computed off-line by using a chessboard pattern.

2.1 Feature Tracking

The point tracker of Lucas-Kanade [17] is a well known algorithm. Its underlying assumption is that a window around a point stays constant between two frames. When applied to features on cars, this suffers from lighting changes, orientation changes, reflections. Moreover, good features to track must contain gradients in both directions, meaning they are corners. When using a Harris detector [8] on a car image, there are only few of these points. This is a consequence of the lack of texture on surfaces. Still, when using a pyramidal implementation of the Lucas Kanade, as described in [3], a proportion of the corners are correctly tracked.

To increase the number of tracked points, the translation hypothesis is used. In the case the object is translating in front of a static camera, and assuming that radial distortion has been removed, each feature track is ideally projected as a line on the focal plane of the static camera. All the feature lines are meeting at a vanishing point. Knowing this point would allow to apply a one dimensional tracking. In practice, the result of the bi-dimensional feature tracking does not lead to perfect lines meeting at a single point. Still, some parts of the tracks can easily be approximated by linear segments. A robust algorithm must then be used to approximate the vanishing point. Some high precision algorithms have been proposed [1]. In practice a RANSAC [6] framework is very efficient. Every pair of lines is a sufficient subsample to compute a potential vanishing point. The vanishing point is the one with the largest consensus.

In the traditional Lucas-Kanade tracker, the two dimensional displacement d of a feature is solution of the linear system

$$\left(\sum_{W_t} gg^\top\right).d = \sum_{W_t}(I_t - I_{t+1})g \ , \tag{1}$$

where W_t is the window of interest around a corner at time t, I_t the image intensity value at time t, I_{t+1} the image intensity at time $t+1$, g the image intensity gradient. This system is applied recursively in a Newton-Raphson manner to minimize the difference between windows at time t and $t+1$. But once the vanishing point is known, the bi-dimensional tracking is over-parametrized. With this knowledge, the equation (1) can be simplified (with a proof very close to the one from [17]) to a one dimensional equation :

$$\left(\sum_{W_t} (g.dir)^2\right).d = \sum_{W_t}\left((I_t - I_{t+1}).(g.dir)\right) \ , \tag{2}$$

where *dir* is the direction from the feature to the vanishing point and d is now a scalar representing the displacement along the vanishing direction.

In practice, we track features in two passes. At first, a bi-dimensional tracking is done to compute the vanishing point. In a second pass, the features are tracked by the one-dimensional tracker. As good features to track only need to have gradient along the vanishing direction, this leads to much more tracks, resulting in a better input for the structure from motion algorithm. The table 1 shows the number of inlier points that are valid for reconstruction between two frames. For the selected sequences and pairs of frames, the gain of our method is between 2.4 and 5.1. And on some sequences, the result of the bi-dimensional tracker is too poor to be used directly in a structure from motion algorithm.

Table 1. Number of tracked points declared inliers by the reconstruction algorithm

	2D tracker	1D tracker	gain
Seq 1	138	405	2.9
Seq 2	118	508	4.3
Seq 3	73	370	5.1
Seq 4	55	134	2.4
Seq 5	75	348	4.6
Seq 6	37	120	3.2

2.2 3D Point Cloud Reconstruction

The algorithm we implemented is close to [14], except that it is adapted to translation. At first, two frames are used for reconstruction. Then views are added iteratively to complete and refine the 3D point cloud.

Initial Reconstruction from Two Frames A point x in a frame f_i and its correspondence x' in the frame f_j are linked by the fundamental equation :

$$x'^{\mathrm{T}} F x = 0 \,, \tag{3}$$

where F is called the fundamental matrix. It encodes the spatial relationship between the two cameras positions. In the particular case of translation, it can be seen [9] that the fundamental matrix has the particular form of a skew-symmetric 3x3 matrix. If the vanishing point, which is equivalent in this special case to the second epipole, is noted $e' = (x_e, y_e, z_e)$, the fundamental matrix F is :

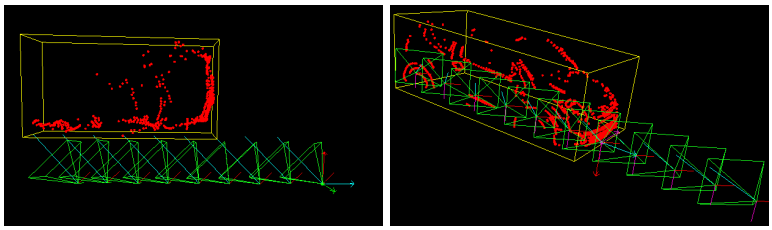
$$F = [e']_x = \begin{bmatrix} 0 & -z_e & y_e \\ z_e & 0 & -x_e \\ -y_e & x_e & 0 \end{bmatrix}$$

Because computing the fundamental matrix can be unstable, this is very valuable to get it directly from the vanishing point. The problem is that all the

points conform to the fundamental equation (3). And thus it is impossible to use the fundamental matrix as a filter between two views.

Using the internal calibration computed off-line, the fundamental matrix directly leads to external calibration ([9],[13]). Once the camera poses are known for two frames, using the two 2D positions of a point to find its 3D position is called triangulation. Again, because camera calibration is known, triangulation is achieved in Euclidean space and thus, a simple linear method leads to good quality 3D point cloud ([9], [10]).

Adding a View From the cloud of 3D points and their corresponding 2D positions in a new frame, there are several methods to compute the camera pose for this frame. In case of pure translation, there are only three unknowns for the camera external calibration. That could be reduced to one parameter as the motion direction is known. But for not being too dependent of the vanishing point computation, we look for the full 3D translation. The projection equations lead to two equations for each couple 3D-2D. Thus we only need two 3D-2D correspondences to compute a pose. Because of the small number of data samples needed to estimate a model, a RANSAC approach [6] fits very well to this problem. Figure 2 shows a 3D point cloud with all the locations and orientations of cameras used for reconstruction.



(a) (b)

Fig. 2. (a) Top view of the 3D point cloud and its bounding box. Cameras locations are on a line on the side of the car. (b) Another reconstruction example. For nicer display, point clouds shown here have been filtered using steps of section 3.

3 Cloud Filtering

Figure 3(a) shows the final result of the previous algorithm. There are still many outliers. This is because there is no filtering on points yet. A first cloud is computed from two views. Then other views are robustly added and the point positions are refined but none is rejected. In this section, we introduce two simples filter that achieve efficient filtering for our 3D clouds.

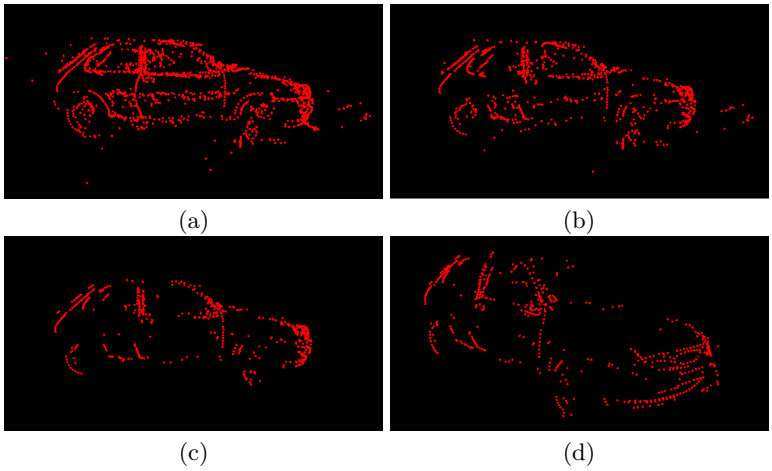


Fig. 3. (a) Initial cloud. (b) Cloud filtered by retro-projection error below one pixel (c) cloud filtered by retro-projection error below one pixel and feature tracked for at least three frames. (d) Perspective view of the final filtered cloud

The first filter is to impose a threshold τ on the retro-projection error. For a 3D point X_i that has been reconstructed from a subset of views $\mathcal{F}$, the retro-projection error $err(X_i)$ is defined as :

$$err(X_i) = max_{c \in \mathcal{F}}(\|P_c(X_i) - x_c\|) \ ,$$

where P_c is the projection matrix for frame c and x_c the 2D position of the point in this frame. A 3D point X_i is declared outlier if $err(X_i) > \tau$.

Figure 3(b) shows the result with a threshold τ of one pixel. With our experiments, one pixel is the lowest acceptable value and setting a sub-pixel threshold remove too many points. This is mostly because there is no non-linear optimization in the presented algorithm. When using the bundle adjustment implementation of [12], the average value of the $err(X_i)$ defined above can get below 0.1 pixels for more than 200 points. But this optimization is time consuming and because our goal is to compute an approximate model, we skip it.

Filtering with the retro-projection error does not remove all the outliers. Figure 3(b) still contains large outliers. Analyzing these points leads to several potential sources of error. Some points are tracked only for the two initial reconstruction frames. Thus, their retro-projection error is zero. Other points belong to the shadow in front of the car. For a short distance, they have a displacement that is very close to the car's move. To filter these outliers, the chosen heuristic is to reject a point if its triangulation has been done from less than n frames. Figure 3(b) shows results for n=3. In practice, using n=4 or 5 ensures more robustness.

4 Building a 3D Car Model from the Point Cloud

The previous algorithm leads to a complex cloud of points. Some points on the car surface are correctly reconstructed. Some interior parts of the car were reconstructed as well. Several points on the wheels were forced to have a linear track and surprisingly, it happens that they pass the previous filters and are reconstructed at a wrong depth. Because of the complexity of the surface, some reflected parts cannot be filtered by algebraic criteria ([16]). Thus we need to robustly approximate the point cloud by a simple model. We propose an approach to establish first a local frame which has the same directions as the car. Working in this local frame, we approximate side part of the car by a second degree polynomial function and front part by a higher degree polynomial function.

4.1 Computing the Local Frame

Because the camera positions are the dual of the object positions, the vector between successive camera centers gives the inverse 3D direction of the translation of the car. We present here an approach to compute the missing horizontal direction. Our underlying hypothesis is that a front view of the car would contain many horizontal lines. At least, top and bottom lines of the license plate should be detected. Thus, using the 3D point cloud, we generate a textured synthetic front view of the car and then analyze it to find a major direction.

A front view virtual camera is positioned on an axis aligned with the motion direction, and far enough from the object to include the bounding box in its field of view. Its optical axis is pointing toward the point cloud (figure 4.a). To render a textured synthetic view, a 3D surface is needed. But approximating the 3D point cloud by a surface is a difficult problem. There exist general approaches based on Delaunay tetrahedrization ([2],[4]). For the cars, we consider that the 3D surface is a range image from focal plane of the first camera in the sequence. This allows to fit the points with a spline surface using the Multilevel B-Splines Approximation algorithm of [11]. Figure 4.b shows a line version of the 3D spline surface. Once this 3D surface is obtained, one can use any frame of the movie to texture it. Figure 4.c shows the textured 3D spline surface. To reduce texture projection error, it is a good choice to work with the camera whose optical axis is the most aligned with the vehicle motion. In general, this is the first camera of the movie. This figure shows that the result is good enough on the central part of the front of the car.

Once this 3D textured surface is obtained, we used the virtual front camera described above to render it for a front view. Edge detection is applied on the synthetic view and then lines are detected using an Hough transform on this edge image. By clustering the direction of the lines, the major direction (i.e., the horizontal one) is found. Figure 4.d shows a synthetic front view with the detected lines and the major direction.

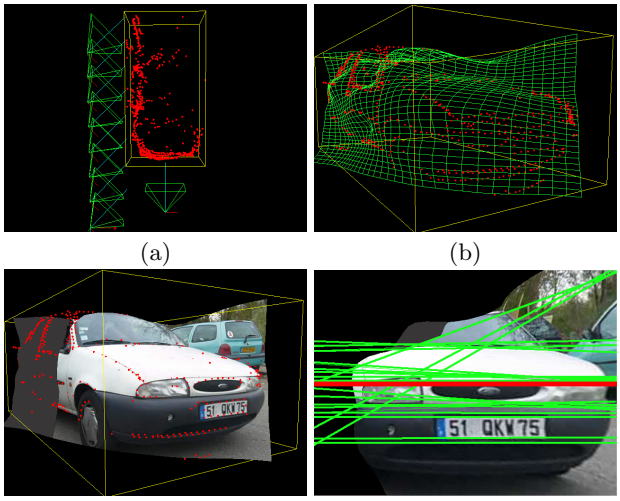


Fig. 4. (a) Virtual front view camera facing the point cloud. (b) The spline fitting on the point cloud. (c) Same view but the spline has been textured with one frame on the video sequence. Dark grey parts are spline surface nor reached by the projection of the texture. (d) The textured spline of (c) has been rendered with camera of (a). All detected lines are in green. Major direction is wider, in red.

4.2 Fitting a Polynomial Model on the Point Cloud

The 3D bounding box is now aligned on the frame of the car. Thus, its faces have a high-level meaning relatively to the car (e.g, front face, side face). Using this knowledge, our approach is to fit the side and front 2D profiles of the car by two polynomial functions.

First, points are projected in the front plane of the bounding box. The car side profile is then computed by fitting these 2D points by a second degree polynomial function. Fitting polynomial functions is made as a least square problem but because of the outliers in the cloud, it is required to use M-estimators (e.g., Tukey or Huber). These estimators are used in an iterative re-weighted least square algorithm. The obtained 2D profile is extruded in the car motion direction to obtain the side 3D surface of the car model. Then, we apply the same procedure for front profile. The 3D points are projected on the side face of the bounding box. The front profile is given by the fitting of these points with a higher degree polynomial function. This 2D profile is then extruded in the width direction to obtain the complete front car 3D surface. Final step is to merge the side and front surfaces to obtain the correct model (figure 5).

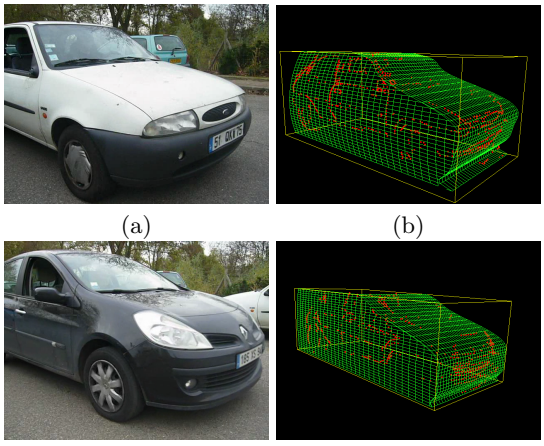


Fig. 5. A frame (a) of an input sequence and the reconstructed polynomial 3D model (b). Another frame (c) from another sequence with the corresponding 3D model (d).

5 Conclusion

In this article, we introduced a complete algorithm to construct approximate 3D models of cars from video sequences. Our first contribution is to introduce a one-dimensional tracker that makes use of the vanishing point computed in a first pass. Then, we showed that some basic filters can clean-up the outliers that were introduced by the translation hypothesis. Our second contribution is to propose a method to work in a local frame for building the car model. This local frame is computed without any external informations as markers on the road. The obtained model is coarse but still useful for many applications. It can be used directly to classify vehicles according to their dimensions. Moreover, having established the local frame orientation is a strong and meaningful knowledge for further algorithms. For example, the cloud bounding box dimensions correspond to actual height, length and width of the car (at a certain scale). And one can generate synthetic views from a point of view relative to the car (i.e., front view, side view...) for higher-level interpretations. Our future work consists of exploring methods to use this coarse model in a deformable scheme to achieve high quality 3D cars reconstruction. It could be used as an initialization surface and also to generate a model-driven force to avoid collapsing through transparent surfaces of the car.

References

1. A. Almansa, A. Desolneux, and S. Vamech. Vanishing points detection without any a priori information. 25(4):502–507, april 2003.

2. Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry*, 19(2-3):127–153, 2001.
3. Jean-Yves Bouguet. Pyramidal implementation of the Lucas Kanade feature tracker, 2000.
4. T. Dey and S. Goswami. Tight cocone: A water-tight surface reconstructor, 2003.
5. Carlos Hernandez Esteban and Francis Schmitt. Silhouette and stereo fusion for 3d object modeling. *Comput. Vis. Image Underst.*, 96(3):367–392, 2004.
6. Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
7. Michael Goesele, Brian Curless, and Steven M. Seitz. Multi-view stereo revisited. *cvpr*, 2:2402–2409, 2006.
8. C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *4th ALVEY Vision Conference*, pages 147–151, 1988.
9. R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
10. Richard I. Hartley and Peter Sturm. Triangulation. *Computer Vision and Image Understanding: CVIU*, 68(2):146–157, 1997.
11. S. Lee, G. Wolberg, and S. Y. Shin. Scattered data interpolation with multilevel B-splines. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):228–244, 1997.
12. M.I.A. Lourakis and A.A. Argyros. The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, Aug. 2004. Available from `http://www.ics.forth.gr/~lourakis/sba`.
13. Yi Ma, Stefano Soatto, Jana Kosecka, and S. Shankar Sastry. *An Invitation to 3-D Vision*. Springer Verlag, 2004.
14. Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *Int. J. Comput. Vision*, 59(3):207–232, 2004.
15. Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. *cvpr*, 1:519–528, 2006.
16. Rahul Swaminathan, Shree B. Kang, Richard Szeliski, Antonio Criminisi, and Shree K. Nayar. On the motion and appearance of specularities in image sequences. In *ECCV (1)*, pages 508–523, 2002.
17. Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.
18. Bill Triggs, Philip McLauchlan, Richard Hartley, and Andrew Fitzgibbon. Bundle adjustment – A modern synthesis. pages 298–375, 2000.