Recovering Camera Motion in a Sequence of Underwater Images through Mosaicking

Rafael Garcia, Xevi Cufí, and Viorela Ila

Computer Vision and Robotics Group, Institute of Informatics and Applications,
University of Girona, Campus de Montilivi,
17071, Girona, Spain
{rafa, xcuf, viorela}@eia.udg.es

Abstract. A procedure for automatic mosaic construction over long image sequences is presented. This mosaic is used by an underwater vehicle to estimate its motion with respect to the ocean floor. The system exploits texture cues to solve the correspondence problem. The dynamic selection of a reference image extracted from the mosaic improves motion estimation, bounding accumulated error. Experiments with real images are reported.

1 Introduction

A composite image constructed by aligning and combining a set of smaller images is known as *mosaic* [1]. In most cases the construction of a mosaic involves recovering the relative motion between the camera and the scene. Mosaics of the ocean floor are very useful in undersea exploration, creation of visual maps, underwater navigation, etc. In this context an underwater vehicle carries a down-looking camera, taking images of the ocean floor to build a mosaic as it performs the mission. Quite often, the mosaicking systems found in the literature perform local gray level correlation [2] or compute optical flow [3] to align the images which form the mosaic. Although these techniques provide good results in standard images [4], they may lead to detection of incorrect correspondences in underwater sequences. The special properties of the medium makes underwater images difficult to process [5]: the elements of the image get blurred, some regions of interest present high clutter and lack of distinct features. Although most of the techniques neglect the use of textural information, considering only image intensity, texture provides a rich source of information to solve image alignment [6]. This paper extends our previous work [7, 8] to construct more accurate mosaics of the ocean floor and over longer image sequences. In [7], every image of the sequence was registered to the previous one. Therefore, when an inaccuracy is introduced in the transformation between both images, this error affects not only the current registration, but all the following ones.

In this paper, we address the problem of building mosaics of the ocean floor to reduce the error associated to the position of an underwater vehicle when it performs a mission. In particular, our mosaicking system has been upgraded in several ways to overcome the difficulties described above: (*i*) texture cues are considered to improve feature correspondences, thus reducing inaccuracies; and (*ii*) selection of a reference image extracted from the mosaic, in contrast to processing every pair of consecutive images.

The remainder of the paper is structured as follows: Section 2 outlines our mosaicking methodology, detailing the characterization procedure and the methodology applied to the selection of a reference image. Then, experimental results proving the validity of our proposal appear in Section 3. Finally, Section 4 presents the conclusions of this work.

2 Building a Mosaic

Our mosaicking system is divided into two main blocks, namely: *mosaic controller* and *mosaic engine*. The mosaic controller keeps the state of the mosaicking system and takes decisions according to this state. It is responsible of the mosaic data structure, *i.e.*, updating the mosaic image (I_m) according to the estimated motion. On the other hand, the motion is estimated by the mosaic engine. It considers the current image (I_c) and a reference image (I_r) and computes a planar homography which describes the motion between both. Selection of the reference image is performed by the mosaic controller. Figure 1 shows the relationship between both modules.

2.1 Mosaic Controller

This module aims to analyze how the vehicle is moving and generates the pertinent commands to control the mosaic engine. The mosaic controller provides the engine module with the images which will be used to estimate the motion of the vehicle. One of these images is the current frame acquired by the camera (I_c). The second one is a reference image (I_r), extracted from the mosaic image I_m by the controller.

Every iteration of the algorithm starts when current image I_c is acquired. Then, the geometric distortion caused by the lens (and the water-glass-air interface of the camera housing) is corrected through a simplification of the Faugeras-Toscani algorithm [9] to correct uniquely radial distortion, instead of performing full camera calibration [10]. Once lens distortion has been corrected, the current image at time instant k, denoted $I_c(k)$, is rotated and scaled so that its orientation and scale matches that of the reference image $I_r(k)$.

Consider a 3×3 matrix ${}^r\mathbf{H}_c(k)$ as the homography which transforms the coordinates of a point in image $I_c(k)$ into its corresponding coordinates in the reference image $I_r(k)$. The motion estimated at the previous time instant ${}^r\mathbf{H}_c(k-1)$ is assumed to be valid as an "a priori" motion estimation for instant k, since motion between two consecutive images is rather small due to the high frame-rate of the sequence. Then, images $I_r(k)$ and $I_c(k)$, together with "a priori" motion estimation matrix ${}^r\mathbf{H}_c(k-1)$ are passed to the mosaic engine, and it is told to execute. The output of the mosaic engine is the homography matrix ${}^r\mathbf{H}_c(k)$, which estimates the motion between $I_c(k)$ and $I_r(k)$ at time instant k. It should be noted that the engine is executed only when the controller requires it.

Once the engine has finished its execution, the controller decides if I_m should be updated. The controller can be configured to actualize the mosaic every α images, with $\alpha = 1..20$. It uses equation (1) to update the mosaic image $I_m(k)$ with the current image $I_c(k)$. I_m is only updated in those areas which have not been updated before by the previous images. Therefore, the first available information for every pixel is used to actualize the mosaic image. This strategy of using the less recent information to construct the mosaic is known in the literature as "use first" [2].

$$^{m}\mathbf{H}_c(k) = {}^{m}\mathbf{H}_r(k) \cdot {}^{r}\mathbf{H}_c(k) \tag{1}$$

The next step consists of deciding whether a new reference image I_r has to be selected for the next iteration. The controller uses matrix $^{r}\mathbf{H}_c(k)$ to check if the overlapping between the reference image $I_r(k)$ and current image $I_c(k)$ is below a given threshold (*e.g.* 40% of the size of the image). In this case, it has to select a new reference image $I_r(k+1)$ for the next iteration of the algorithm. The new reference image will be extracted from the mosaic image $I_m(k)$ at the same position and orientation as that of the last image added to the mosaic (at time $k - mod(k/\alpha)$). Following this methodology, drift in the estimation of the trajectory of the vehicle increases more slowly than registering every pair of consecutive images.

On the other hand, if the overlap between images $I_c(k)$ and $I_r(k)$ is bigger than the threshold, the reference image will not change, *i.e.* $I_r(k+1) = I_r(k)$.

2.2 Mosaic Engine

The engine begins its execution by detecting interest points in image I_c. The goal of the interest point detector is to find scene features which can be reliably detected and matched when the camera moves from one location to another. Moreover, these features should be stable when lighting conditions of the scene change somewhat. A slightly modified version of the Harris corner detector [11] is used to detect the interest points. Once the relevant features of image I_c have been detected, the next step consists of finding their correspondences in the reference image I_r. Before searching for correspondences, both images are smoothed with a 3×3 Gaussian mask. Given an interest point $^{c}\mathbf{m}$ in image I_c, instead of considering the point as an individual feature, we select an $n \times n$ region $R(^{c}\mathbf{m})$ centered at this point. Then, the system aims to find a point $^{r}\mathbf{m}$ in the reference image I_r, surrounded by an $n \times n$ area which presents a high degree of similarity to $^{c}\mathbf{m}$. This "similarity" is computed as a correlation function [4]:

$$\mathrm{corr}\left\{R(^{c}\mathbf{m}), R(^{r}\mathbf{m})\right\} = \frac{\mathrm{cov}\left\{R(^{c}\mathbf{m}), R(^{r}\mathbf{m})\right\}}{\sigma\left\{R(^{c}\mathbf{m})\right\} \cdot \sigma\left\{R(^{r}\mathbf{m})\right\}} \tag{2}$$

From equation (2) we can observe that the correlation between two points is defined as the covariance between the grey levels of region $R(^{c}\mathbf{m})$ in the current image and region $R({}^r\mathbf{m})$ defined in I_r, normalized by the product of the standard deviation of these regions. In practice, these regions are subsampled by a factor q, reducing the processed pixels from $n \times n$ to $m \times m$, where $m = ((n-1)/q)+1$, and, therefore, decreasing the computational burden.

Equation (2) is computed for all the points of the reference image which fall inside a small search window. In order to locate this window, the system takes into account the previous estimated motion ${}^r\mathbf{H}_c(k-1)$. Consider an interest point ${}^c\widetilde{\mathbf{m}}$, defined in the current image and expressed in homogeneous coordinates. The search window is centered at ${}^r\tilde{\mathbf{c}}$, as shown in equation (3).

$$ {}^r\tilde{\mathbf{c}} = {}^r\mathbf{H}_c(k-1) \cdot {}^c\widetilde{\mathbf{m}} \tag{3} $$

being ${}^r\tilde{\mathbf{c}}$ the projection of the interest point ${}^c\widetilde{\mathbf{m}}$ into the reference image. The coordinates provided by ${}^r\tilde{\mathbf{c}}$ are uniquely used to open the window where equation (2) is applied to search for the correct correspondence ${}^r\widetilde{\mathbf{m}}$ of interest point ${}^c\widetilde{\mathbf{m}}$.

Equation (2) is normalized by substracting the mean and dividing by a factor which takes into account the dispersion of the gray levels in the considered regions. For this reason, this measurement of correlation is very adequate for underwater imaging, where lighting inhomogeneities are frequent. Unfortunately, although equation (2) produces good results in absence of rotations, its reliability decreases as images I_c and I_r present a higher rotational component. For this reason the mosaic controller rotates and scales the current image, prior to pass it to the engine.

According to equation (2), given an interest point ${}^c\mathbf{m}$ in the current image I_c, its correspondence ${}^r\mathbf{m}$ in I_r should be the point which has obtained the highest correlation score. Those pairs (*interest_point*, *matching*) which have obtained a correlation score lower than a given threshold are deleted. However, experimental work with underwater images has proved that in some cases the true correspondence is not the one with the highest correlation score [6]. Therefore, the incorrect correspondences (known as "*outliers*") are detected through a two-step approach: first, discrimination of false matches through textural analysis; and next, elimination of points describing non-dominant motion by means of a *robust estimator*.

In order to characterize incorrect correspondences through textural analysis, the textural properties of the neighborhood of both the interest point ${}^c\mathbf{m}$ and its estimated correspondence ${}^r\mathbf{m}$ are computed. In this way, the regions $R({}^c\mathbf{m})$ and $R({}^r\mathbf{m})$ are characterized by two feature vectors (${}^c\mathbf{v}$ and ${}^r\mathbf{v}$), which encode their textural properties. Some of the *Energy filters* defined by Laws (*e.g.* L5S5, E3E3, etc.) are used to perform the textural analysis [12]. Depending on the parametrization of the system, this textural characterization may consist, for instance, of measuring the texture at some neighboring locations $(g_0, g_1, ..., g_8)$ as shown in Figure 2. If the Euclidean distance between both vectors is smaller than a selected threshold, the pair (*interest_point*, *matching*) is considered to be an *outlier*. In fact, this approach is based in the assumption that interest points (and their correspondences) are located at the border between, at least, two regions with different textural properties. It is a reasonable assumption since interest points are detected by finding areas of high variation of the image gradient through a corner detector, *i.e.*, located in the border of different image textures.

The second step consists on applying a robust estimation method, the Least Median of Squares (LMedS), to detect those feature points describing a trajectory which is different from the dominant motion of the image [13]. These “bad” correspondences are in fact correctly matched points belonging to some moving object of the scene, such as a moving fish, algae, etc.

Next, the motion estimation ${}^{r}\mathbf{H}_c(k)$ between current and reference images is computed from the remaining pairs of points applying equation (4).

$$ {}^{r}\widetilde{\mathbf{m}} = {}^{r}\mathbf{H}_c \cdot {}^{c}\widetilde{\mathbf{m}} \quad \text{or} \quad \begin{bmatrix} \lambda \cdot {}^{r}x \\ \lambda \cdot {}^{r}y \\ \lambda \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \cdot \begin{bmatrix} {}^{c}x \\ {}^{c}y \\ 1 \end{bmatrix} \tag{4} $$

where λ is an arbitrary non-zero constant. Therefore, solving the homography of equation (4) involves the estimation of 8 unknowns. By using inhomogeneous coordinates instead of the homogeneous coordinates of the points, and operating the terms, the projective transformation of equation (4) can be written as:

$$ \begin{bmatrix} {}^{c}x_1 & {}^{c}y_1 & 1 & 0 & 0 & 0 & -{}^{r}x_1 \cdot {}^{c}x_1 & -{}^{r}x_1 \cdot {}^{c}y_1 \\ 0 & 0 & 0 & {}^{c}x_1 & {}^{c}y_1 & 1 & -{}^{r}y_1 \cdot {}^{c}x_1 & -{}^{r}y_1 \cdot {}^{c}y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ {}^{c}x_n & {}^{c}y_n & 1 & 0 & 0 & 0 & -{}^{r}x_n \cdot {}^{c}x_n & -{}^{r}x_n \cdot {}^{c}y_n \\ 0 & 0 & 1 & {}^{c}x_n & {}^{c}y_n & 1 & -{}^{r}y_n \cdot {}^{c}x_n & -{}^{r}y_n \cdot {}^{c}y_n \end{bmatrix} \cdot \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} {}^{r}x_1 \\ {}^{r}y_1 \\ \vdots \\ {}^{r}x_n \\ {}^{r}y_n \end{bmatrix} \tag{5} $$

When the engine completes its execution, it gives back the control to the mosaic controller.

3 Experimental Results

In order to evaluate the proposed technique, several sea trials have been carried out under real conditions, using URIS, a small Unmanned Underwater Vehicle (UUV) developed at the University of Girona. The vehicle carries a downward-looking camera, which takes images of the seafloor. As the vehicle moves, the acquired images are sent to the surface through an umbilical tether, where they are stored on a tape to be processed off-line.

The sea trial reported here was carried out in July 2001. This experiment shows a trajectory performed by the vehicle in an area of the sea floor formed by rocks and algae. The original trajectory is formed by 4.380 images at a frame rate of 25 i.p.s. The sequence has been sub-sampled, taking only one image of every five, thus the mosaicking system processes 876 images.

Figure 3 shows the resulting mosaic. It can be observed in this Figure that image alignment is quite good, although the underwater terrain is not flat. Unfortunately, it is not possible to quantify the errors which are produced in real sea trials since the real trajectory cannot be recovered from any other sensors available in the UUV.

4 Conclusions

In this paper we have presented a methodology to construct mosaics of the ocean floor to estimate the motion of an underwater vehicle. The construction of visual mosaics of the floor can provide accurate position estimates for local navigation of the vehicle. A new texture-based characterization and matching methodology has been proposed, reducing the number of incorrect correspondences in image pairs. Moreover, a dynamic selection of the reference image improves, to a large extent, the performance of the system.

References

1. R. Szeliski, Image mosaicing for tele-reality applications, in: IEEE Workshop on Applications of Computer Vision, 1994, pp. 44–53.
2. N. Gracias, J. Santos-Victor, Underwater video mosaics as visual navigation maps, Computer Vision and Image Understanding 79 (1) (2000) 66–91.
3. S. Negahdaripour, X. Xu, A. Khamene, A vision system for real-time positioning, navigation and video mosaicing of sea floor imagery in the application of ROVs/AUVs, in: IEEE Workshop on Applications of Computer Vision, 1998, pp. 248–249.
4. A. Giachetti, Matching techniques to compute image motion, Image and Vision Computing 18 (3) (2000) 247–260.
5. J. Jaffe, The domains of underwater visibility, in: SPIE Ocean Optics VIII, 1986, pp. 287–293.
6. R. Garcia, X. Cufí, J. Batlle, Detection of matchings in a sequence of underwater images through texture analysis, in: IEEE International Conference on Image Processing, Vol. 1, Thessaloniki, Greece, 2001, pp. 361–364.
7. R. Garcia, J. Batlle, X. Cufí, J. Amat, Positioning an underwater vehicle through image mosaicking, in: IEEE International Conference on Robotics and Automation, Vol. 3, Seoul, Rep. of Korea, 2001, pp. 2779–2784.
8. X. Cufí, R. Garcia, P. Ridao, An approach to vision-based station keeping for an unmanned underwater vehicle, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 1, Lausanne, Switzerland, 2002, pp. 799–804.
9. R. Garcia, J. Batlle, X. Cufí, A system to evaluate the accuracy of a visual mosaicking methodology, in: MTS/IEEE OCEANS Conference, Vol. 4, Honolulu, Hawaii, 2001, pp. 2570–2576.
10. O. Faugeras, G. Toscani, The calibration problem for stereo, in: IEEE Conference on Computer Vision and Pattern Recognition, 1986, pp. 15–20.
11. C. Harris, M. Stephens, A combined corner and edge detector, in: Alvey Vision Conference, Manchester, U.K., 1988, pp. 147–151.
12. K. Laws, Textured image segmentation, Tech. Rep. 940, Image Processing Institute, University of Southern California, Los Angeles, California (1980).
13. P. J. Rousseeuw, A. M. Leroy, Robust Regression and Outlier Detection, John Wiley and Sons, New York, 1987.

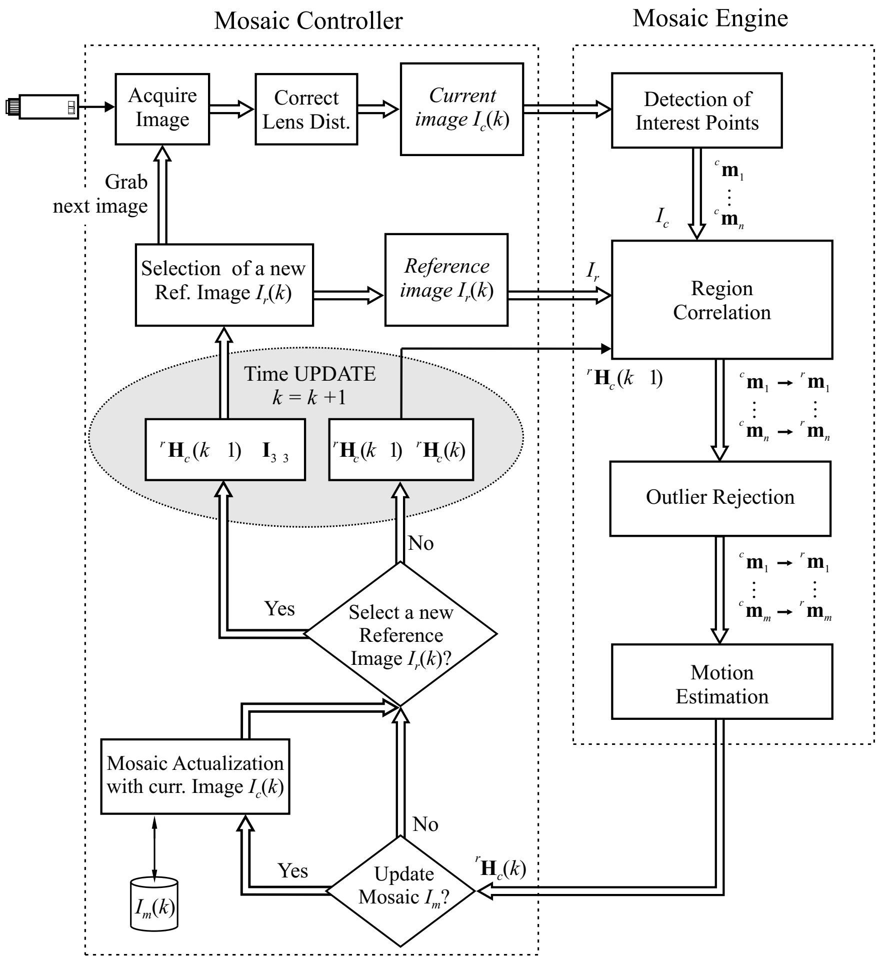
Mosaic Controller
Mosaic Engine
Acquire Image
Correct Lens Dist.
Current image $I_c(k)$
Detection of Interest Points
$^c\mathbf{m}_1$ … $^c\mathbf{m}_n$
I_c
I_r
Grab next image
Selection of a new Ref. Image $I_r(k)$
Reference image $I_r(k)$
Region Correlation
Time UPDATE $k = k + 1$
$^r\mathbf{H}_c(k-1) = \mathbf{I}_{3\times3}$
$^r\mathbf{H}_c(k-1) = {}^r\mathbf{H}_c(k)$
$^r\mathbf{H}_c(k-1)$
$^c\mathbf{m}_1 \rightarrow {}^r\mathbf{m}_1$ … $^c\mathbf{m}_n \rightarrow {}^r\mathbf{m}_n$
Outlier Rejection
$^c\mathbf{m}_1 \rightarrow {}^r\mathbf{m}_1$ … $^c\mathbf{m}_m \rightarrow {}^r\mathbf{m}_m$
Motion Estimation
No
Yes
Select a new Reference Image $I_r(k)$?
Mosaic Actualization with curr. Image $I_c(k)$
$I_m(k)$
No
Yes
Update Mosaic I_m?
$^r\mathbf{H}_c(k)$

Fig. 1. Bloc diagram illustrating the relationship between the *mosaic controller* and the *mosaic engine*.

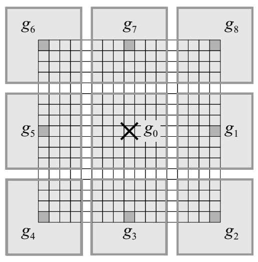

Fig. 2. Point characterization is performed by computing texture at 9 neighboring locations $(g_0, g_1, ..., g_8)$.

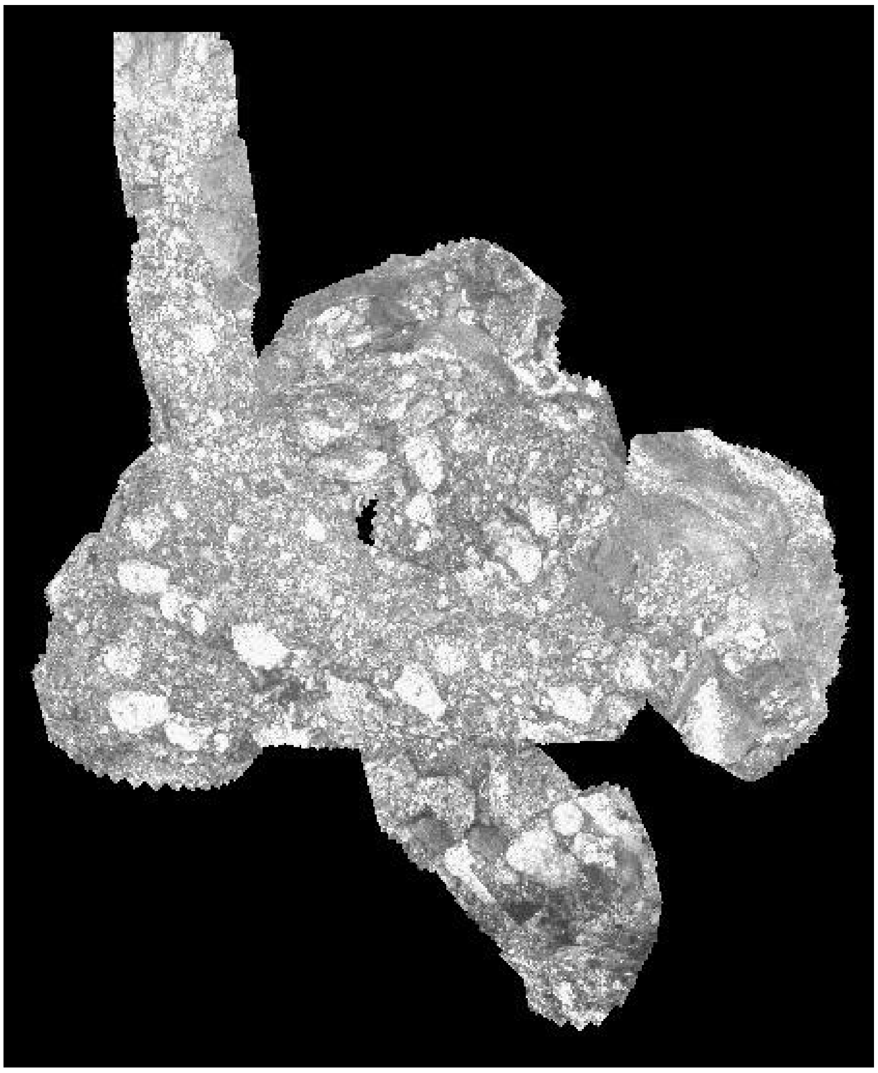

Fig. 3. Resulting mosaic after processing a sequence of 876 images. The vehicle starts its motion at the top-left of the image and then moves down performing several loops.