DOI: 10.2420/ACT-BOK-AF
ISSN: 2309-1940

Acta Futura

Issue 9

AI in Space Workshop at IJCAI 2013

Advanced Concepts Team

http://www.esa.int/act

European Space Agency
Agence spatiale européenne

Publication: Acta Futura, Issue 9 (2014)

Editor-in-chief: Duncan James Barker

Associate editors: Dario Izzo
Jose M. Llorens Montolio
Pacôme Delva
Francesco Biscani
Camilla Pandolfi
Guido de Croon
Luís F. Simões
Daniel Hennes

Editorial assistant: Zivile Dalikaite

Published and distributed by: Advanced Concepts Team
ESTEC, European Space Research and Technology Centre
2201 AZ, Noordwijk
The Netherlands
`www.esa.int/actafutura`
Fax: +31 71 565 8018

Cover image: Four-wheel skid-steered robot used in data collection experiments from Kohler et al. in this issue.

ISSN: 2309-1940
D.O.I.:10.2420/ACT-BOK-AF

Typeset with XeTeX

Contents

Foreword

Artificial Intelligence is making significant inroads in the space sector. Artificial Intelligence (AI) systems are contributing to numerous space missions including the Hubble Space Telescope, Mars Exploration Rovers, the International Space Station, and Mars Express. Many different areas of AI are already, or may be in the near future, of particular interest from a space applications point of view. These include topics as diverse as: intelligent search and optimization methods in aerospace applications, image analysis for guidance navigation and control, autonomous exploration of interplanetary and planetary environments, intelligent algorithms for fault identification, as well as data mining and visual presentation of large data sets.

This issue contains selected papers from the 2013 AI in Space workshop, held as a satellite event at the 23rd International Conference on Artificial Intelligence (IJCAI) in Beijing, China. Now in its fourth edition, this workshop was co-organized by the Advanced Concepts Team of the European Space Agency and the Artificial Intelligence Group of NASA's Jet Propulsion Laboratory. The goal of the *AI in Space* workshop is to highlight the most recent AI applications related to space, encourage collaboration between the two areas, and to provide an overview of current research. The following 9 papers were presented at the *AI in Space* workshop and selected for revised submission. As a result, this issue provides an interesting compilation of current research directions in the area of artificial intelligence applied to the space domain.

Sensor Fault Detection and Compensation in Lunar/Planetary Robot Missions Using Time-Series Prediction Based on Machine Learning by Tim Köhler et al. introduces a system that detects sensor drop outs and compensates missing sensor signals. *Planning Mars Rovers with Hierarchical Timeline Networks* by Juan M. Delfa Victoria et al. presents a heuristic planner to solve highly constrained temporal problems for autonomous on-ground and on-board planning. *Onboard Autonomous Response for UAVSAR as a Demonstration Platform for Future Space-Based Radar Missions* by Joshua Doubleday et al. discusses a system for onboard processing and interpretation of radar data as well as autonomous retasking of the vehicle and instrument. *Lunar Crater Identification from Machine Learning Perspective* by Chak Pong Chung, Cheuk On Chung, and Chit Hong Yam explores different methods for lunar crater detection based on digital elevation model data provided by the Chinese Chang'E-1 and Chang'E-2 spacecrafts. *GPU Accelerated Genetic Algorithm for Low-thrust GEO Transfer Maneuvers* by Kai Yu and Ming Xu presents a GPU based implementation of a novel genetic algorithm for low-thrust trajectory design. *How Ants Can Manage Your Satellites* by Claudio Iacopino et al. discusses the design of an innovative ground-based automated planning and scheduling system based on ant colony optimization. *Pattern-Based Modeling for Timeline Planning in Space Domains* by Simone Fratini, Nicola Policella and Alessandro Donati describes a system for pattern-based modeling to bridge the gap between AI planning languages and the current practice in the space operational context. *Procedural Onboard Science Autonomy for Primitive Bodies Exploration* by Steve Chien et al. presents a close loop autonomous approach for onboard science target detection and response for time constraint missions to primitive bodies. *Space Hopper: a Serious Game Crowdsourcing the Design of Interplanetary Trajectories* by Wiktor Piotrowski et al. introduces an online crowdsourcing experiment that aims to improve automated trajectory design.

Daniel Hennes and Dario Izzo
(Associate editors)

Acta Futura 9 (2014) 9-20
DOI: 10.2420/AF09.2014.9

Acta Futura

Sensor Fault Detection and Compensation in Lunar/Planetary Robot Missions Using Time-Series Prediction Based on Machine Learning

Tim Köhler[*1], Elmar Berghöfer[1], Christian Rauch[2] and Frank Kirchner[1]

[1] *DFKI GmbH, Robotics Innovation Center, Robert-Hooke-Straße 5, 28359 Bremen, Germany*

[2] *University of Bremen, Robotics Research Group, Robert-Hooke-Straße 5, 28359 Bremen, Germany*

Abstract. Mobile robots operating in a lunar or planetary space mission can usually neither be repaired from nor brought back to earth. In case of sensor damages or drop outs an overhaul of the hardware leading to a properly working sensor is not possible in most cases. Instead, the system has to continue working as reliable as possible. In the special case of an autonomous space robot this means that the robot, first, needs to detect a sensor drop out automatically. Second, the missing sensor signal needs to be compensated. In typical mobile robot setups this is possible by using other sensor modalities. Presented is a method to detect single sensor faults by model-based predictions covering multiple sensor modalities. The methods are learned using one of two methods: tested and compared are a multi-layer perceptron (MLP) and a "Neural Gas (NG)" vector quantization method. The test use case is a turning skid-steered robot with four different sensor modalities (velocities left and right wheels, gyroscope Z-axis, and horizontal optical flow). With the collected training and test data the model predictions turns out to be accurate enough for the purpose of a sensor fault detection. Moreover, by using learned models a compensation in case of a sensor fault can be possible.

[*]Corresponding author. E-mail: tim.koehler@dfki.de

1 Introduction

Robotic applications in space have a high demand on the system's reliability: there are only short communication windows, a high latency in communication, and there is nearly no way of recovering a system when it is in a fault state. Different hierarchical behavior architectures were developed (e.g., [2]) to enable the robot to carry out high-level plans and to supervise the behavior execution. However, faults in the sensor hardware would still be a critical problem in plan execution – although there might be several cases of single sensor drop-outs that could be compensated. For a typical lunar or planetary space mission, an example could be a wheeled mobile robot with a three-axes gyroscope, a camera or laser range finder, and wheel encoders. A malfunctioning single sensor in such a setup could be compensated (at least partly) by a combination of the motor commands and the data obtained from the other sensors.

By learning the correlation between actions executed by the robot and the corresponding sensor feedback, a prediction of sensor signals could be possible. Thus, a model of the motor-to-sensor relation is learned which is used to generate expected sensor values when executing a learned action again. Such a model is specific for

the learned action (or actions) and might depend on certain properties of the environment. A model-based prediction with three different methods of model generation and with an application of detecting environmental conditions rather than sensor failures is presented by Rauch et al. [7].

Comparing the prediction error (between the learned, expected state and the actual sensor feedback) of several sensor modalities could lead to an identification of single sensor drop-outs. Furthermore, if a sensor failure is identified, a compensation could be possible by using the predicted sensor values instead of the measured ones.

Presented is the application of a vector-quantization-based method and a neural network model to generate prediction models for typical sensor modalities and typical motions of a mobile robot in a lunar environment. The motor-sensor model is learned with training data in a so called *normal case*: All variations and disturbances that occur in this training set are expected to be covered by the model. Predictions based on this trained model are evaluated on test cases where different sensor faults are simulated. The aim is therefore, first, to model the correlation of actions and perception (the sensorimotor loop) for the normal case, i.e. for situations where no sensor drop-outs occur, second, to recognize when expectations in the sensorimotor loop do not match the measured values, e.g. when a sensor drop-out occurs, and, third, to use predicted sensor values instead of the faulty real sensor values. Depending on the overall architecture, this switching could also be done transparently. However, in many cases a signaling of the error condition to higher layers is demanded.

In the field of fault detection, several publications can be found. For example, a more general review on process fault detection is published by Venkatasubramanian et al. ([10] and following two parts). Some publications concentrate on the sensor fault detection without giving a possibility to correct or compensate faulty sensor signals, e.g., using a classifier. The combination of detection and correction can be done using an analytical model based on prior knowledge or using a learned model. Lishuang et al. present a very interesting solution including even an identification of bias faults and drift faults of a sensor [4]. However, their method uses a relatively complex setup based on least squares support vector machine. With the methods presented in this work, an alternative solution is proposed. Furthermore, especially the combination of multiple sensor modalities and the comparison of different sensors' behaviours is examined.

The first method used to train the prediction model is called "Neural Gas". Neural gas (NG, Martinetz et al. 1993, [6]) is a vector quantization method which preserves the structure of the action-perception relation and does not fit a functional description to this relation. This especially can be useful if the prediction model has to cover different situations where the sensor feedback differs depending on environmental properties only (i.e. if ambiguities exist in one modality).

A second method used in comparison to the NG is the "Multi Layer Perceptron" (MLP) (Rummelhart et al. 1986, [8]). The MLP is a neuronal method that is capable of learning functional correlations between input and output values. Once an MLP is trained on example data, like described for the NG above, it can be used to predict sensor values. However, in contrast to NG there might be multiple MLP networks needed if different environments lead to ambiguous data.

This work covers (1) learning of models for a sensor value prediction, (2) evaluation of prediction and sensor measurements, and (3) compensation of a sensor fault. In Section 2 the methods are described. Section 3 presents the results which are discussed in Section 4. The paper finishes with a Conclusion and an Outlook.

2 Methods

An overview of the single components of the fault detection and compensation is given in Figure 1. Shown is the setup when predicting sensor responses using learned models. The configuration to train a model is not depicted. In the figure, three different ways of operation can be seen.

Using a prediction based on the motor commands only (method A in the figure) is an application of the biological model called reafference principle proposed by von Holst and Mittelstaedt [11, 12]. We use this configuration for the tests described in Section 3. As input the history of the last 15 motor commands is used. In variant B, the predictions are based on the sensor states. The advantage of this configuration is that the sensor state expectations can also been matched to robot states in cases where the motion was not initiated by the robot. An example is a mobile robot rolling down a slope. But one disadvantage is that presumably the matching of the predictions and actual sensor states is worse without the motion commands. Furthermore, a sensor fault would

 DOI: 10.2420/AF09.2014.9

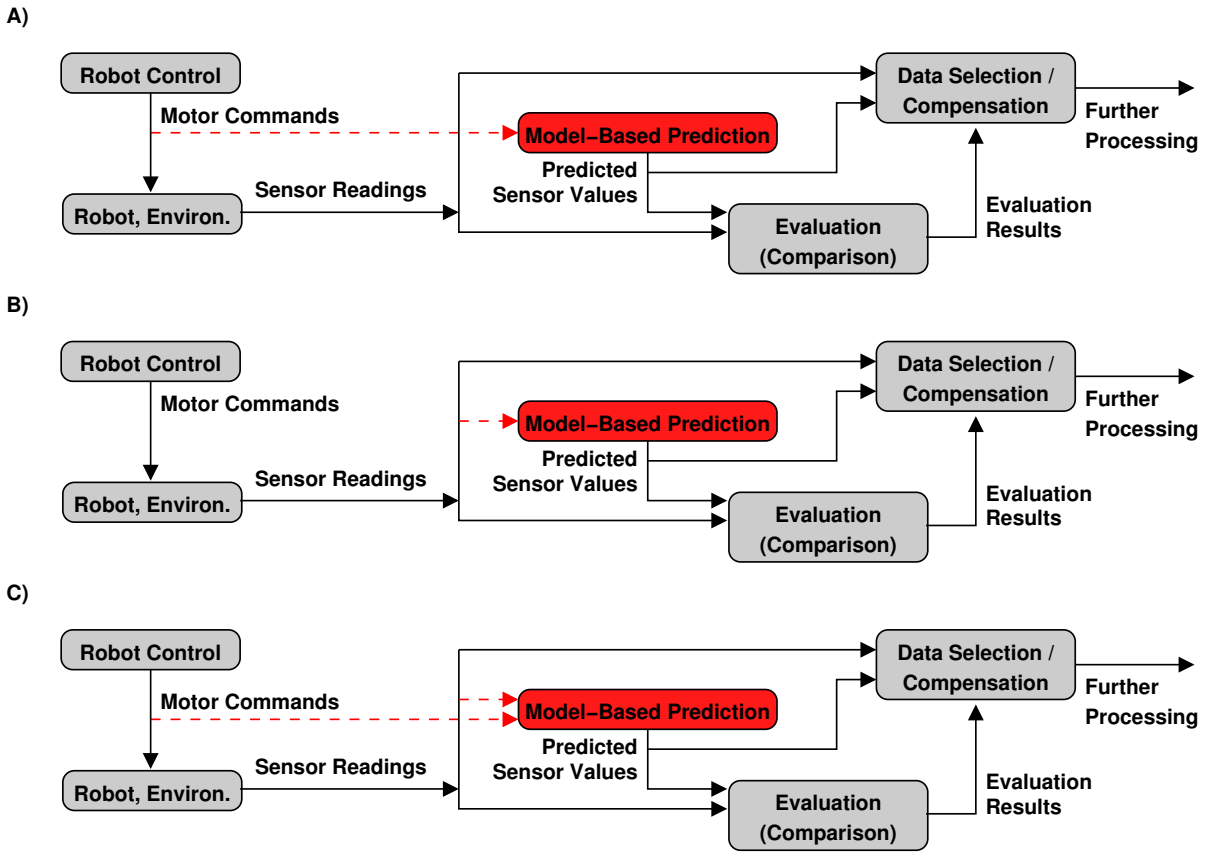


Figure 1: Overview of the proposed fault detection and compensation. Three different ways of operation are discussed.

have an influence on the prediction result. Finally, variant C combines both sources as a basis for the predictions. However in this work, variant C is not studied in more detail because the main difference to configuration A, i.e. tolerating externally induced or passive motions, is actually a disadvantage in the chosen application. If a mobile robot in unknown lunar or planetary terrain for example slips down a crater rim the predictions could perhaps be matched in configuration C. But the state of slipping is to be avoided or at least detected. Comparisons of the prediction results of variant A and C have yet been carried out, see Section 2.3.

2.1 Prediction Using MLP

One of the methods we used for the sensor value prediction is known as "Multi-Layer Perceptron" (Rummelhart *et al.* 1986, [8]) which is a well known method in the field of artificial neural networks. The architecture of a standard MLP is that it has one input layer that has a number of neurons equal to the dimension of the desired input and an output layer containing a number of output neurons equal to the dimension of the desired output. In between those input and output layers the network has a predefined number of "hidden" layers which can contain each an independently chosen number of neurons which are normally fully forward connected. The MLP can be seen as a general function approximator which can in theory approximate every arbitrary function with only one hidden layer as long as this layer has enough neurons. The network is trained to minimize an error function (least mean square error) on a given training dataset, in this paper we used the *back propagation* approach and an optimization method called *Levenberg-Marquardt* method which is a combination of the methods described by Levenberg in [3] and extended by Marquardt in [5]. As the activation function for the hidden

layers we used $\tanh$ and for the output layer a linear function.

Using the MLP for sensor value prediction we have to define input and output sensor channels for which we want to train an MLP. As an input channel the motion commands of the robot (including a short history of past commands), the values of any of the other sensors, or a combination of both can be used to predict one or some of the other sensors. At least of course the sensor values selected as output should have a correlation to at least one of the input channels. We tested different variants where only a single sensor or a combination of multiple ones was predicted given only the motion commands as well as variants where also information of other sensors was available as input. The results of these analysis will be described in Section 3.1.

2.2 Prediction Using NG

The second method to learn a model studied for the proposed fault detection is the vector quantization method "Neural Gas" (NG) by Martinetz et al. [6]. In contrast to MLP, this method is able to preserve any distribution of the action-perception relation (in the training data set). Especially it is possible to cover ambiguities in the motor-sensor relation like sensor responses which vary depending on environmental properties (e.g., turning rate depending on the type of ground).

In the application of NG, the NG center vectors (CV) are adapted to a multi-dimensional training data set with n input dimensions (motor commands) and m output dimensions (sensor modalities to be predicted) combined in one $n + m$-dimensional space. The input/output modalities and the training set can be the same as for the MLP (see above). At recall, the current (and p past) motor commands are used to find the closest trained CV. The winning CV gives the sensor value (single or multi dimensional) which is used as prediction value (expected sensor response). This method was already proposed by Hoffmann et al. [1] and Schenck et al. [9]. One advantage of using a vector quantization like NG in this application is that after learning further dimensions can be used for recall. For example, the winning CV for a sensor value prediction can be selected by the p past motor commands and the $m-1$ other sensor values.

2.3 Evaluation and Compensation

Fault Types There are different kinds of sensor faults possible. Independent of the specific type of sensor, general misbehaviours can be distinguished. First of all, the temporal conditions of a fault can be classified in persistent and temporarily. When a failure is active typically three cases can be distingushed: a) the sensor reading stays at a fixed value (e.g. 0, maximum, or last reading before the fault), b) the sensor reading is arbitrarily changing (free floating), and c) the defective sensor reading follows another signal (e.g. a connection to another sensor). For simulation and test, fault models have been designed which cover typical misbehaviours. Examples for the three cases are a) stuck-at-0 fault model, b) open fault model, and c) bridging fault model. The stuck-at-0 case and the bridging case are used for the tests described below. Open faults can in principle be detected in the same way. However, the appearance of such a fault can vary a lot from trial to trial. Thus, an evaluation with artificial data would have to be done with a wide range of different sensor reading frequency spectra and sensor reading courses. Furthermore, an identification of such a fault could be possible more easily by frequency spectrum analyses.

Detection Given there are m predictions (expectations) $\hat{x}_i$ of m current sensor data x_i, the sensor-specific absolute error $e_i = |x_i - \hat{x}_i|$ is used for sensor fault detection. Depending on the sensor fault type to be detected and depending on the application, the single normalized absolute errors $e_{norm,i} = e_i/\sigma_i$ (where σ_i is the standard deviation of the training data in sensor channel i), the single relative errors $e_{rel,i} = m * e_i / \sum_j^m e_j$ or a combination of both needs to be used.

The distinction between the sensor fault condition and the no fault condition can be drawn based on single error values or based on the history of a number of past error values. In both cases either fixed or dynamically adapting thresholds are possible solutions. In the tests described below, a fixed threshold can be applied.

Compensation Using prediction models not only allows to detect a sensor fault but also offers a way for compensation. Depending on error model and application, a fault detection can respond instantaneous or after a buffering period (e.g. to accumulate error). When a fault is detected the further sensor processing components (e.g. robot motion control) can either be triggered (e.g. to switch to a safe system hold mode) or

 DOI: 10.2420/AF09.2014.9

Figure 2: The mobile robot platform used for data collection. The environment is a lunar crater model.

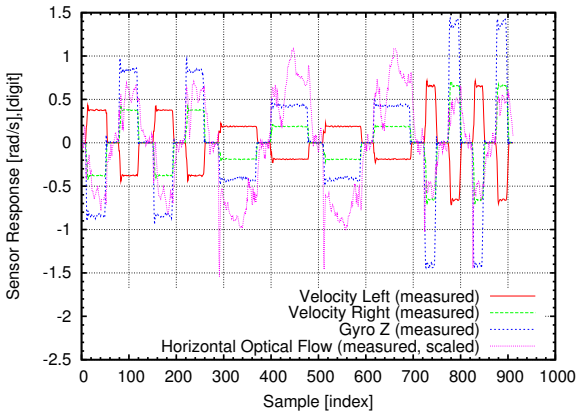


Figure 3: Test data to evaluate the prediction results.

they can be supplied with the expected sensor readings generated by the prediction. In cases where just a single sensor fails and where other sensors deliver enough information such a combination of real measurements and expectations might lead to a still controllable system – depending on the system possibly even without any changes in the control architecture.

3 Experimental Results

The two prediction approaches are trained on real sensor data of a four-wheel skid-steered robot (see Figure 2). The robot is equipped with a three-axes gyroscope, cameras, and wheel encoders. The system was placed on flat ground in a black hall which contains the model of a moon crater rim. In the crater model, the lighting can be controlled to cover typical cases occurring in space missions, like glares or harsh shadows (as can be seen in the figure). In training, the robot carried out turning motions at different speeds, each repeated multiple times. During the whole data recording the robot was placed so that the cameras mainly recording only the crater rim. Since we analysed the optical flow, this represents an environment that could be found in a similar way on a real moon mission.The test data is generated by simulating different types of sensor faults within the sampled real sensor and actuator data.

To analyse and compare the performance of the prediction on sensor data in the normal case in which all sensors working properly. The data of different sensors are recorded which are, an inertial measuring unit (IMU) with a gyroscope which provides a measure of the turning speed of the robot, the robot odometer giving values for the speed of the right and the left wheels, and finally optical flow generated on the camera data. The motion commands which tell the robot how to move will also be recorded. The recorded data are divided into training and test data. The training dataset contains 8 repetitions of each of the different turning speeds that were tested while the test data set contains the remaining 2 repetitions of each speed. This guarantees, that both methods are trained and tested on the same data and therefore, render the results comparable. The test data is shown in Figure 3.

The data of different sensor streams was aligned with a frequency of 10 Hz while for each sensor the latest datum is used. During training the algorithms received a history of the last 15 motion commands which contains a translation and rotation command along with sensor data for the next future time step which should be predicted by the algorithms. These sensor data which shall be predicted contain data of 4 different sensor modalities which are the rotation velocity given by the IMU gyroscope Z axis, the velocity of the left wheels, the velocity of the right wheels, and the horizontal optical flow. The results for the NG and MLP methods on these test dataset are described in the following.

3.1 Results of MLP Prediction

As described in 2.1 we used an MLP network with $\tanh$ as the activation function of the neurons in the hidden layers and a linear activation function for the neurons in the output layer. We used two hidden layers both with 20 neurons which are fully forward connected. The network is trained using the Levenberg-Marquardt optimization method to minimize the mean square error

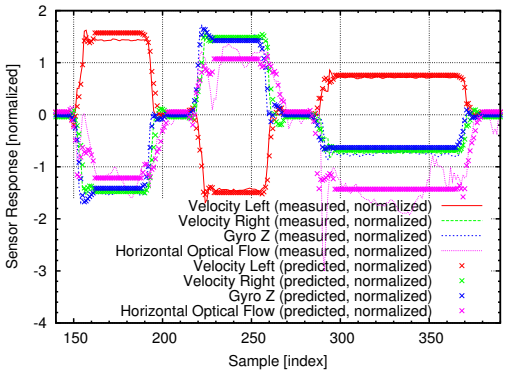


Figure 4: Part of the test data showing measured sensor values and predictions of the MLP trained on all four sensor modalities.

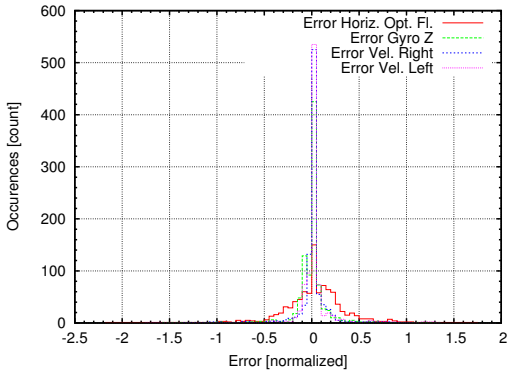


Figure 5: Error histogram of the MLP-based prediction using a single MLP for all four sensor modalities.

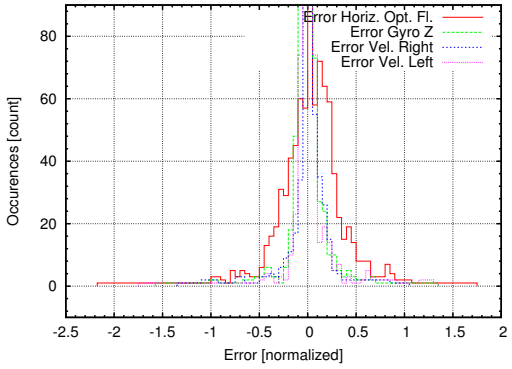


Figure 6: Error histogram of the MLP-based prediction using a single MLP for all four sensor modalities, lower part.

in 50 training epochs on the training dataset.

For the MLP we tested two different variants, to predict all target sensor modality with a single MLP and as reference one MLP was trained just to predict the optical flow value separately. For the prediction of all sensor modalities using one MLP it turned out to be useful to normalise the values of the optical flow first. In doing this on the training data the mean value for all optical flow values and their standard deviation were calculated and all training and test data were normalised by subtracting the mean and dividing them by the standard deviation.

As it can be seen in Figure 4 the MLP is able to predict all sensor modalities quite well. The prediction of the optical flow values in the multi modal dataset produced a root mean square error (RMSE) of 68.44 digits and for the single modality case a RMSE of 66.91 digits. These results seem to be quite small compared to the large absolute variance of the target signal.

In Figure 5 a histogram of the occurred errors while predicting the test data is shown. It can be seen, that in most of the times the prediction is quite close to the true sensor values and therefore, the histogram plot has high peaks at or around an error of zero. Figure 6 shows a rescaled version of the same plot "zoomed" at the lower part of the plot. Comparing these two plots it can be seen, that the prediction of the different sensor channels has also different quality. The values for the right and left velocity as well as the gyro Z-axis rotation have mostly very low errors, shown by the high peaks in the middle of the histogram. In contrast to that it can be seen, that the optical flow more often has a higher error and therefore a wider distribution of occurred errors in the histogram. Comparing these with Figure 4 we can see that the optical flow values are also much more noisy than the values from the other sensors. Since also the error is higher for the optical flow we can assume that the correlation between the motion commands of the robot and the resulting optical flow values is more loose than for the other sensors, and therefore, also the prediction is not as accurate as for the other sensors.

3.2 Results of NG Prediction

The NG center vectors are adapted to a 34-dimensional training data set (history of 15 two-dimensional motor commands, four exemplary sensor dimensions: wheel speed left and right, gyroscope response Z-axis, horizontal optical flow). For these tests, 300 center vectors

 DOI: 10.2420/AF09.2014.9

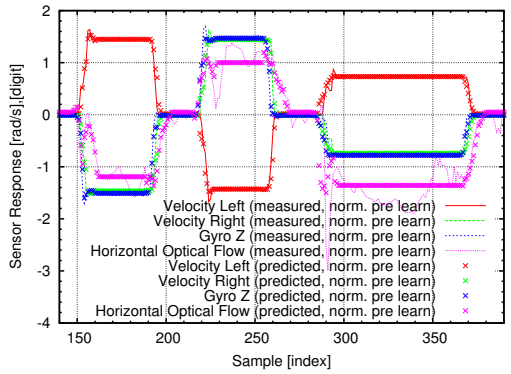


Figure 7: Example of the test data showing measured sensor values and predictions of the NG model trained with normalized training data. The test data depicted here was adapted with mean and standard deviation of the training data normalization.

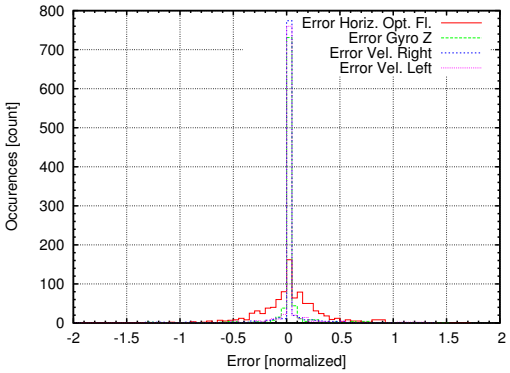


Figure 8: Error histogram of the NG-based prediction using normalized training data.

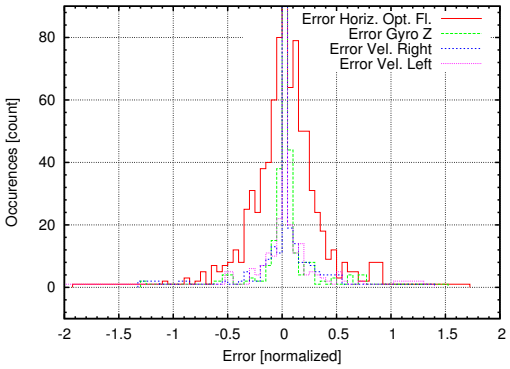


Figure 9: Error histogram of the NG-based prediction using normalized training data, bottom part.

were used with the NG parameters learning step size ϵ and neighbourhood decay constant λ decreasing from 1.0 to 0.001 and from 150 to 0.01, respectively (see [6]). 1,000,000 adaptation steps were carried out. In a first training test, the measured test data was used without any preprocessing. In a second trial, the measured data was normalized to a mean of 0.0 and a standard deviation of 1.0. Such a normalization is needed for the MLP prediction especially in case of the optical flow data. However, in the given training and test data also the NG results improved. In Figure 7 an example part of the test data and the NG-based predictions is shown.

Like for the MLP results an error histogram for the NG-model trained on normalized data is given in Figure 8. The lower part can be seen in Figure 9. MLP and NG histograms are comparable. Especially, again the higher errors of the optical flow can be seen.

In case normalized data is used for training, the same adaptation needs to be carried out for the test data (i.e. the current sensor measurements in the application). Mean and standard deviation need to be stored together with the trained NG center vectors (see Section 2.3).

3.3 Comparison of Predictions

In Table 1 the RMS errors of MLP and NG, each in two different configurations are given. To be comparable (over method and over sensor modality), all error values have been normalized using the training data standard deviations.

As can be seen, MLP and NG show similar performances with the chosen settings and the chosen computational effort. Both results could of course be improved. For both methods, the acceleration dynamics when starting and stopping the motion is covered by the learning in average. However, varying latency in motion start/stop from trial to trial and the noise in the steady state phases are not covered in learning. This leads to the prediction error, especially for the optical flow sensor channel.

In the last row in Table 1 the NG results for variant C of Figure 1 are given. In this configuration, in recall all dimensions are used to find the corresponding center vector. This means that besides the motor commands also the current sensor values have an influence on the winning center vector. This leads to better matching results, as can be seen in Table 1. However, in case of a

Method	*Condition*	*RMSE Modality 1*	*RMSE Modality 2*	*RMSE Modality 3*	*RMSE Modality 4*
MLP	single MLP	0.173	0.160	0.185	0.333
MLP	separate MLP optical flow				0.326
NG	raw training data	0.225	0.217	0.272	0.482
NG	normalized training data	0.165	0.145	0.172	0.331
NG	norm. train. data, var. "C"	0.144	0.114	0.150	0.279

Table 1: Comparison of the prediction methods' errors. To compare the different methods and modalities the RMS errors of the results for raw training data (rows 1 to 3) have been scaled by the training data set standard deviations. The same scaling was done for the test data in the evaluation of the NG run with normalized training data (rows 4, 5). In row 5 the results for variant C in Figure 1 are given, see text. The four sensor modalities are (1) velocity left, (2) velocity right, (3) gyroscope Z-axis, and (4) horizontal optical flow.

sensor fault the center vector selection is influenced in the same way. Thus, the error might be smaller and a fault could be harder to be detected. Training data and learning was like in the run for row 4 in the table.

3.4 Results Evaluation and Compensation

In the following tests, the NG method with normalized test and training data is used for the prediction. As shown in Section 3.3 the difference to the other MLP or NG methods is not that large. Their main practical disadvantage is the need to normalize the prediction error before comparing the sensor channels.

At first, a stuck-at-0 fault in a setup of three sensor modalities (channel 1: velocity left, channel 2: velocity right, channel 3: gyroscope Z-axis) is tested. Test and training data was sampled with the turning robot (see the evaluation of the prediction methods). Afterwards test and training sets were normalized with the training set's mean and standard deviation. In Figure 10 the absolute error for a stuck-at-0 of channel 1 is shown. Figure 11 shows three of the 12 runs in more detail (compare Figure 4 for start and stop of the motion). As can be seen, an identification of the sensor fault is easily possible (when the robot is moving). The difference between the error of the defective channel 1 and the other two channels is very pronounced in these runs. However, the error is varying depending on the motion speed.

To avoid such a dependence, the relative errors can be compared. Figure 12 shows this metric for the three sensors. A value of 3 for one sensor means that there is no error on the other channels. A value of 1 means the sensor shows an average error.

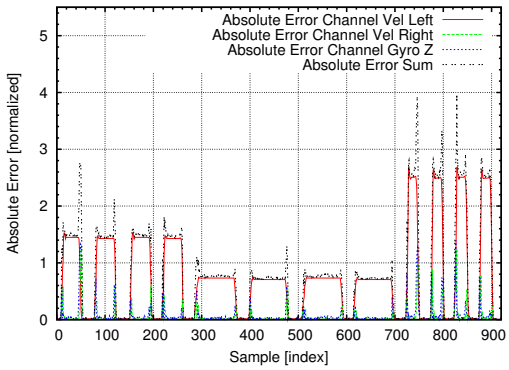


Figure 10: Stuck-at-0 fault of channel 1 (velocity left) in a setup of three sensor modalities. Shown are the three absolute errors and the absolute error sum.

Using the relative error metric is especially helpful in applications where an external "fault" condition is possible and should not lead to a sensor fault detection (like in the application described in [7]). Characteristical for such external "failure" conditions is (1) that the current situation differs from the situation when learning the prediction models and thus the predictions do not match the current sensor readings and (2) that this is the case for all sensor modalities. In the test depicted in Figure 13 that was simulated by having all sensor channels stuck at 0. Of course this is an artificial "ideal" case. Nevertheless it is important to notice how close the relative errors approache 1.0 (and how far they differ while

 DOI: 10.2420/AF09.2014.9

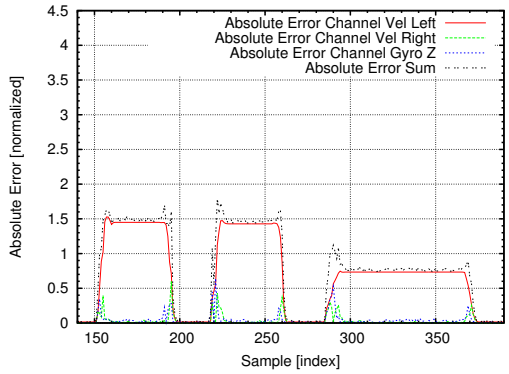


Figure 11: Stuck-at-0 fault of channel 1 (velocity left) in a setup of three sensor modalities. Shown are the three absolute errors and the absolute error sum for only three motion trials. Compare Figure 4.

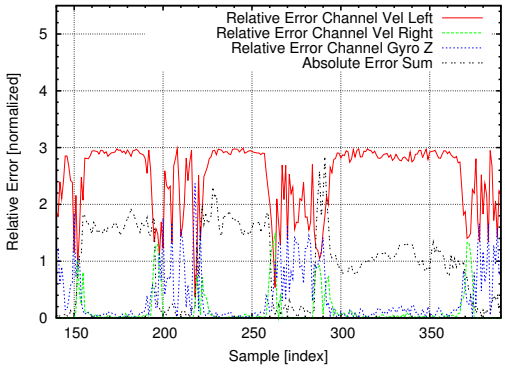


Figure 12: Stuck-at-0 fault of channel 1 (velocity left) in a setup of three sensor modalities. Shown are the three realtive errors and the absolute error sum for only three motion trials.

the robot is not moving).

One problem occures (especially for the relative error) if one of the sensors is very noisy. In the test data this is the case for the optical flow measurements. Here, even in the standing phases motion in the camera images was noticed. This of course can happen in several applications. The results of using all four sensor modalities are depicted in the Figure 14 for the absolute error and in the Figure 15 for the relative error. While in motion phases the relative error is still a good detection measure, in standing phases the absolute error sum needs to be taken into account to reject the fault detection by the optical flow channel relative error.

As test case of a bridging fault a connection between channels 3 (gyroscope Z-axis) and 4 (horizontal optical flow) was chosen. The values assumed to be read from channel 3 were the values of channel 4. While typically a bridging fault is simulated with a bit-wise logical "or" or "and", an overwriting of one channel is no usual simulation. However, it mimics the possible case of a short-circuit of a strong and a weak driver. Moreover, this failure is not an easy detectable problem because the faulty sensor behaviour itself may have the same properties of the correct signal and is even correctly related to the robot motion. The channels 3 and 4 chosen in this test case show a relatively similar behaviour and furthermore the noisy properties of channel 4 are here present in two of four channels. This leads to hard test case, as can be seen in the Figures 16 and 17. Given this data, a fault detection by the absolute error values is not easily possible. A detection by the relative errors would be

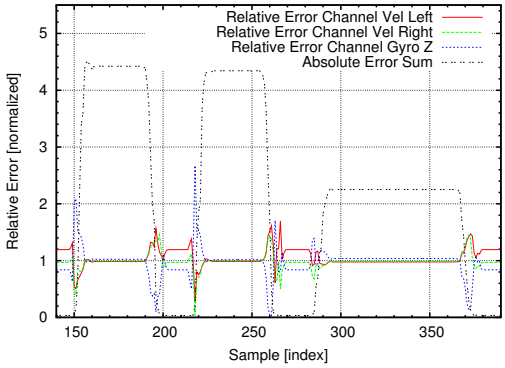


Figure 13: Stuck-at-0 fault at all three channels 1, 2, and 3 in a setup of three sensor modalities. Shown are the three relative errors and the absolute error sum for only three motion trials.

possible at least in most of the trials.

In Figures 18 and 19 the compensation of a sensor fault is shown. In the simple example used here, at each sample just the relative error is considered. Moreover, a fixed threshold of 2.0 is used (center value between minimum and maximum possible threshold values 1.0 and $m = 3.0$). Thus, whenever a single sample of the relative error exceeds the threshold then the predicted sensor value is used as output (signal "Vel., Left (comp.)" in Figure 19). If the threshold is not exceeded then the measured value can be used by, e.g., the robot motion control.

Such a very simple evaluation can be improved in two

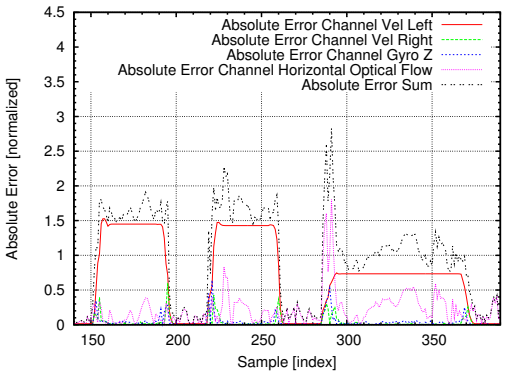


Figure 14: Stuck-at-0 fault of channel 1 (velocity left) in a setup of four sensor modalities. Shown are the four absolute errors and the absolute error sum for only three motion trials.

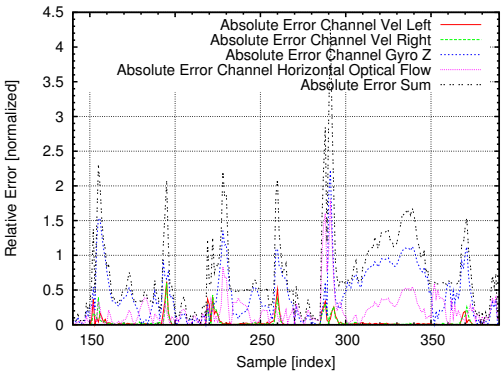


Figure 16: Bridging fault of channel 3 (gyro Z) in a setup of four sensor modalities. Shown are the four absolute errors and the absolute error sum for only three motion trials.

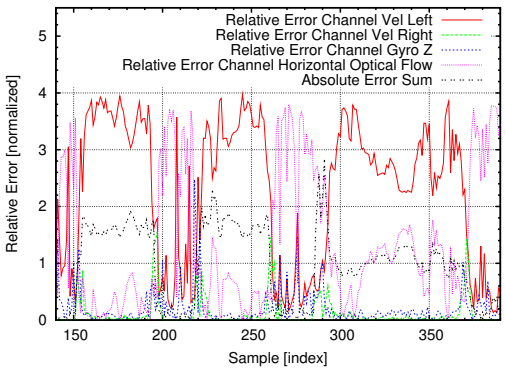


Figure 15: Stuck-at-0 fault of channel 1 (velocity left) in a setup of four sensor modalities. Shown are the four relative errors and the absolute error sum for only three motion trials.

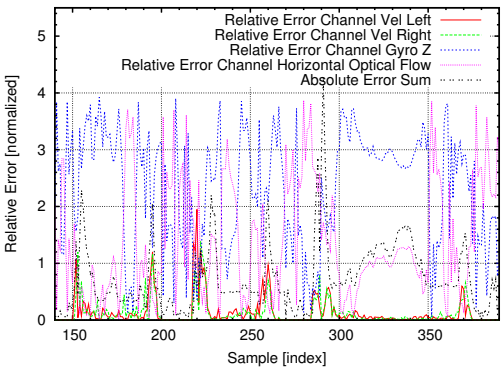


Figure 17: Bridging fault of channel 3 (gyro Z) in a setup of four sensor modalities. Shown are the four relative errors and the absolute error sum for only three motion trials.

 DOI: 10.2420/AF09.2014.9

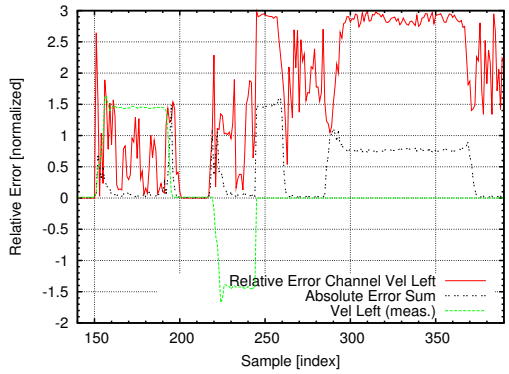


Figure 18: Demonstration of compensation: signal "Velocity, Left" is stuck at 0 starting from sample 245. The absolute error sum is again high during the fault. The relative error of the channel "Velocity, Left" is closed to 3 while the robot is moving. Compare with the following figure.

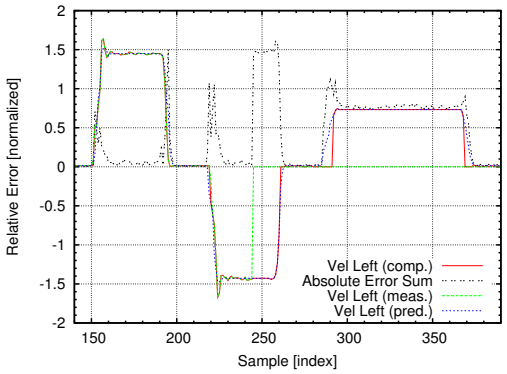


Figure 19: Demonstration of compensation: When a sensor fault is detected, the sensor data given to robot control is switched from measured to predicted values.

ways. As can be seen in the plots including the optical flow signal, a noisy sensor can lead to several short peaks of the relative error. Thus, filtering the error values before their evaluation is an improvement needed for typical noisy setups. The second improvement is especially easily possible if using a vector qantization method like NG. Here, when a sensor fault is detected, the correct sensor modalities can be used in recall, too. Thus, the winning center vector could potentially follow the actual current dynamics – as far as this is encoded in the other sensor signals.

4 Discussion

While a fusioning of multiple sensor modalities could also be carried out in a different way, e.g., within the higher robot control layers, the proposed method has certain advantages. First, if the sensor fault detection and compensation is done separately from the behavior control and plan execution system these components can be used without any adaptations. They do not have to care about the described drop-out cases. Second, as machine learning methods are used here the definition of the normal case can be chosen arbitrarily – depending on the specific application. And third, switching between different models or combining multiple models can be done online, e.g., depending on current plans or behaviors.

Comparing the two prediction methods does not show a clear winner. Both, MLP and NG produce comparable results and both are comparably simple in implementation and computational effort. For the application shown in this paper, a vector quantization method has two advantages. First, the method can be used in different configurations in recall: Learned once, the model can be used as a forward model as proposed by von Holst et al. or Wolpert et al. and as an inverse model fed with sensory input or one of the combinations shown in Figure 1 [11, 12, 13]. Second, a vector quantization method is in principle be able to adapt to any kind of data distribution. Thus, for example ambiguities in the training data set could be covered, too.

The evaluation and fault models presented above can be seen as first results showing the general applicability of the proposed fault detection. The simple combination of a fixed threshold and a single-sample test works for test data shown here. However, an evaluation based on multiple past error values, i.e., a filtering of the error, is needed for setups with sensor signals with more noise.

5 Conclusions and Outlook

Presented was a method to detect single sensor faults by predictions based on learned models. As learning methods a multi-layer perceptron (MLP) and a "Neural Gas (NG)" vector quantization method was tested. The test use case was a turning skid-steered robot with four different sensor modalities (velocities left and right wheels, gyroscope Z-axis, and horizontal optical flow). With the collected training and test data the model predictions turned out to be accurate enough for the purpose

of a sensor fault detection. Moreover, by the learned models a compensation in case of a sensor fault is possible.

Besides some further tests with other motion conditions and fault models, an integration in a robot behaviour control architecture is a crucial test. Therefore, especially the switching between real measurements and predicted (expected) sensor values in an unadapted control architecture is the next test case.

Acknowledgments

Supported by the Federal Ministry of Economics and Technology on the basis of a decision by the German Bundestag, grant no. 50RA1113 and 50RA1114.

References

[1] H. Hoffmann and R. Möller. Unsupervised learning of a kinematic arm model. In O. Kaynak, E. Alpaydin, E. Oja, and L. Xu, editors, *Artificial Neural Networks and Neural Information Processing — ICANN/ICONIP 2003*, Lecture Notes in Computer Sciences 2714, pages 463–470. Springer, 2003.

[2] T. Köhler, C. Rauch, M. Schröer, E. Berghöfer, and F. Kirchner. Concept of a biologically inspired robust behaviour control system. In *Proceedings of International Conference on Intelligent Robotics and Applications 2012 (ICIRA-12), October 3-5, Montreal„ Québec, Canada*, pages 486–495. Springer Berlin / Heidelberg, 10 2012.

[3] K. Levenberg. A method for the solution of certain problems in least squares. *Quarterly of Applied Mathematics*, 2:164–168, 1944.

[4] X. Lishuang, C. Tao, and D. Fang. Sensor fault diagnosis based on least squares support vector machine online prediction. In *Robotics, Automation and Mechatronics (RAM), 2011 IEEE Conference on*, pages 275–279, 2011.

[5] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 11(2):431–441, 1963.

[6] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten. "neural-gas" network for vector quantization and its application to time-series prediction. *Neural Networks, IEEE Transactions on*, 4(4):558–569, 1993.

[7] C. Rauch, E. Berghöfer, T. Köhler, and F. Kirchner. Comparison of sensor-feedback prediction methods for robust behavior execution. In *Proceedings of the German Conference on Artificial Intelligence KI 2013, Lecture Notes in Artificial Intelligence*, volume 8077. Springer, (accepted).

[8] D. E. Rumelhart, G. E. Hintont, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[9] W. Schenck, H. Hoffmann, and R. Möller. Grasping to extrafoveal targets: A robotic model. *New Ideas in Psychology*, 29(3):235–259, 2011.

[10] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri. A review of process fault detection and diagnosis: Part i: Quantitative model-based methods. *Computers & chemical engineering*, 27(3):293–311, 2003.

[11] E. von Holst and H. Mittelstaedt. Das Reafferenzprinzip. *Naturwissenschaften*, 37(20):464–476, 1950.

[12] E. von Holst and H. Mittelstaedt. *The reafference principle. Interaction between the central nervous system and the periphery.*, pages 39–73. The behavioural physiology of animals and man. Selected papers of Erich von Holst, Teil vol. 1. Methuen., London, 1973.

[13] D. Wolpert and M. Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7–8):1317 – 1329, 1998.

 DOI: 10.2420/AF09.2014.9

Acta Futura 9 (2014) 21-29
DOI: 10.2420/AF09.2014.21

Acta Futura

Planning Mars Rovers with Hierarchical Timeline Networks

Juan M. Delfa Victoria[*1,2,3], Simone Fratini[2], Nicola Policella[2], Oskar von Stryk[1], Yang Gao[3] and Alessandro Donati[2]

[1] *Technische Universität Darmstadt, 64289 Darmstadt, Germany*

[2] *European Space Agency, 64293 Darmstadt, Germany*

[3] *University of Surrey, GU2 7XH Guildford, United Kingdom*

Abstract. Surface operations in distant bodies like Mars present a lot of challenges. Among them, lack of direct communications and a complex environment are the main drivers that push for more autonomy both on-ground and on-board. This paper presents a timeline, hierarchical, heuristically-based and domain-independent planner called QuijoteExpress oriented to solve highly constrained temporal problems.

1 Introduction

Even though automated planning has been used since long in practical problems like traffic control in cities, package transportation or spacecraft operations, many recent scenarios such as Fukushima, the on-going Darpa Robotic Challenge or the Curiosity rover are not using planning techniques beyond path planning.

In Fukushima, robots were mainly tele-operated due to the high complexity of the scenario, but limited communication in some areas heavily shielded, lack of situation awareness or difficult synchronization between the human operators (for some robots an operator guides the robot-base and another the arm) were part of the reasons for the poor performance displayed [12]. In the DRC, automated planning is optional and most teams have focused in control under human supervision, where the operator performs the action in a virtual world and the movements are sent then to the "real" robot, with the exception of walking, for which some teams implemented autonomous systems. Finally, Curiosity plans are generated in a highly manual process involving a huge human team. As in DRC, the most remarkable autonomous system is the Autonav.

In this paper, a new planner called QuijoteExpress is described. The motivation is to provide a planner ready to be used in real-world scenarios. More specifically, we try to fulfil the following goals:

1. Performance: Improve the performance of the planner is key to achieve a good scalability as the complexity of the problem grows and to be able to quickly react by means of re-planning to changing conditions.

2. Handle uncertainty: Due to the lack of information in partially-observable, stochastic and dynamic environments such as Mars or noise coming from faulty sensors, sometimes problems cannot be fully stated. A planner for such domains should be able to generate valid plans in these conditions.

3. User friendly: The communication between the user and the planner is critical in real systems.

[*]Corresponding author. E-mail: delfa@sim.tu-darmstadt.de.

Users should be able to define problems in terms of high level goals that hide the complexity and planners should provide plans easy to understand and verify.

4. Domain-independent: Even though we focus our research on planetary rovers, QuijoteExpress is also suitable for any kind of highly constraint scenario with temporal requirements. The approach is to develop domain-independent search algorithms, encoding the specific knowledge in both Hierarchical Task Network [8] (HTN) *methods* and dedicated heuristics whenever it is required. In this way, knowledge is perfectly isolated and reusability is improved.

QE incorporates two novelties oriented to achieve these goals. First, Hierarchical Timeline Networks (HTLN [6]) are intended to improve the planner performance and the way users interact with the planner (see next section). Second, it can generate partial plans to handle uncertainty.

The rest of the paper is organized as follows. Section 2: short overview of the concepts introduced in HTLN. Section 3: describes the rover scenario used during the tests. Section 4: description of the planner QuijoteExpress and the concept of sufficient plan. Section 5: initial results in comparison to AP^2. Section 6: conclusions and future work.

2 Background

QuijoteExpress is based on APSI*, and extension of APSI [11] that combines temporal and HTN planning in a new planning paradigm called HTLN [7]. The main concepts of HTLN are briefly introduced in the appendix at the end of the paper. Central to HTLN and QuijoteExpress is the way in which complex goals[1] are represented and managed. Most HTN planners like SIPE-2 [14] or O-Plan [13] replace in the problem the complex task by the set of sub-tasks of the method selected. In consequence, some information is lost as the problem does not retain the hierarchical structure of the domain. This information can be in fact useful for a given resolver in charge of fixing flaws of the problem or during backtracking as we will see later in this section.

[1]The conventional name in HTN nomenclature is compound. Along the paper we will use complex instead, as we think it is more representative

QE is based on the construction of hierarchical structures rather than goals replacement, adding new levels of detail as complex tasks are refined into sub-tasks [6]. Given a problem represented as a *decision network* $\mathtt{dn}$ (a hypergraph), it is composed by a set of *decisions* (the nodes), each containing a value and a list of parameters, and *relations* (the hyperedges) among them. The decisions can be complex ($\mathtt{d}^{\mathtt{c}}$) or primitive ($\mathtt{d}^{\mathtt{p}}$). Primitive decisions are always represented in HTLN as component decisions ($\mathtt{cd}$), that is a simple node in the problem graph or in other words constraints on timelines. A $\mathtt{cd}$, that is, a simple node. However, a complex goal $\mathtt{d}^{\mathtt{c}}$ can be represented as a component decision $\mathtt{cd}$ in case it is not decomposed, or as a decision network $\mathtt{dn}$ otherwise. In the last case, $\mathtt{d}^{\mathtt{c}}$ retains the value and parameters of the original (not-decomposed) decision plus the sub-network ($\mathtt{dn}_{\mathtt{dec}}$) which contains a list of sub-tasks and relations among them. A planner can use a decomposed $\mathtt{d}^{\mathtt{c}}$ as an atomic decision ignoring the details, or as a sub-problem. This approach provides several advantages:

- Better constraint propagation: As a $\mathtt{cd}$, single modifications on $\mathtt{d}^{\mathtt{c}}$ affect directly the whole sub-network. In the particular case in which $\mathtt{d}^{\mathtt{c}}$ is *self-contained*, that is, no sub-task $\mathtt{d}^{\mathtt{sub}}$ of $\mathtt{d}^{\mathtt{c}}$ is related to any decision out of $\mathtt{d}^{\mathtt{c}}$, then no constraint propagation will be required in case a relation involving $\mathtt{d}^{\mathtt{c}}$ is modified. For example, changing the position of $\mathtt{d}^{\mathtt{c}}$ will also change the position of all its sub-tasks, as they are ordered relatively to $\mathtt{d}^{\mathtt{c}}$. Besides, it also represent a powerful technique to handle uncertainty. In case it is unknown how to achieve a complex goal $\mathtt{d}^{\mathtt{c}}$ during planning time, $\mathtt{d}^{\mathtt{c}}$ is modelled as a $\mathtt{cd}$. During execution, once the required information is available, a decomposition method is chosen for $\mathtt{d}^{\mathtt{c}}$ and the plan is completed.

- Parallelism: If the problem has a $\mathtt{d}^{\mathtt{c}}$ which is an articulation point of the hypergraph, that is, the hypergraph is divided in two if this node is removed, then $\mathtt{d}^{\mathtt{c}}$ sub-network ($\mathtt{dn}_{\mathtt{dec}}(\mathtt{d}^{\mathtt{c}})$) can be planned in parallel. Moreover, if the result is optimal for each $\mathtt{dn}_{\mathtt{dec}}$, the overall solution will be also optimal [6].

With respect to relations, HTLN supports temporal relations such as $\mathtt{drive}\ \mathtt{before}[\mathtt{l},\mathtt{u}]\ \mathtt{communication}$ [1] and parameter relations such as $\mathtt{pointing}_{\mathtt{camera}} = \mathtt{position}_{\mathtt{target}}$.

 DOI: 10.2420/AF09.2014.21

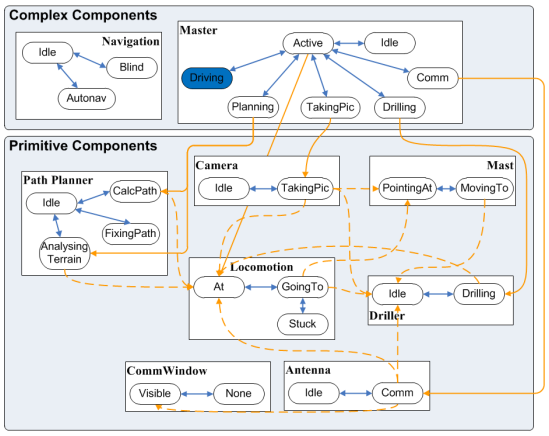


Figure 1: Hierarchical rover model

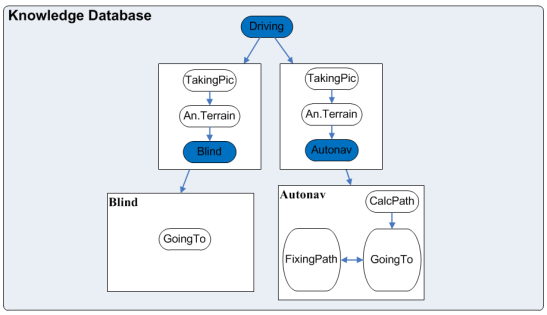


Figure 2: Knowledge database for a Mars Rover

3 Scenario

The rover is a type of robot equipped with a locomotion system to move across hazardous terrain. Its hardware is divided between *payload* and *platform*. The former includes all the instrumentation dedicated to perform science, while the remaining sub-systems in support of these activities are considered the platform. They can serve different purposes both on Earth and space, such as rescue missions, surveillance or planetary exploration.

The same problems that make planetary rover missions very challenging form a planning point of view such as uncertain environments, highly constrained plans or no real-time communications represent the main arguments in favour of more autonomy. Given the increasing complexity of future missions, the advantages in terms of science return and costs are undeniable. As an example, MER would have never been so successful without the AUTONAV system [2]. While autonomous navigation is already well understood, other aspects like opportunistic science start to be addressed by new automated systems like AEGIS [9].

Figure 1 presents the model used for our experiments. It emphasizes two properties: a hierarchical structure and complex inter-dependencies between different components of the robot.

The components are divided in two categories. Those primitive are used to model real rover sub-systems such as the antenna or the driller, while the complex are used to model complex behaviours that involve different primitive components. An extensive description of each component and constraint is out of the scope of this paper. However, some components require a brief description.

Within the primitive components, Camera, Mast, Locomotion, Driller and Antenna represent the main rover hardware. Locomotion and Driller are specially relevant, as they impose a lot of constraints to the rest. CommWindow models the communication windows with a satellite, which are imposed as constraints to the planner.

With respect to complex components, Master represents a set of the main activities the rover can accomplish. Even though in this example most of them have a 1:1 mapping with primitive components, Driving and Idle are special. Idle requires all the components to be in the Idle or equivalent initial state while Driving is hierarchically decomposed in either Blind or Autonav as showed in figure 2. Both require as initial step to take a picture of the surroundings and analyse the traversability of the terrain. In case the terrain is easy, the rover blindly goes directly to the target. In case the terrain is complex, a path with intermediate waypoints is calculated and, while driving, it is continuously corrected with information gathered from the sensors. These decomposition recipes are stored as `methods` in the Knowledge Database (`kdb`).

4 QuijoteExpress

QuijoteExpress (QE) is a timeline, hierarchical, heuristically-based and domain-independent planner which extends the AP^2 [4] planner developed in the frame of the ESA Goal Oriented Autonomous Controller Study (GOAC [3]) and is backwards compatible with AP^2, thus allowing QE to run already existing domains. QE is organised in two packages: Planners and Heuristics. The planners are themselves divided in

a strategic and a tactical planner, the last one composed of four *resolvers* ρ, each dedicated to solve a specific type of flaw ϕ. The heuristics are organised in four groups: choose the next SolvingSpaceNode, choose the next flaw in a given SolvingSpaceNode, choose a resolver and choose a decomposition method for a complex goal.

Three resolvers, Unfolder, Scheduler and TimelineCompleter are inherited from AP^2 with some modifications. The unfolder is in charge of adding the supporters of a given goal by applying the domain theory. The scheduler is in charge of adding ordering constraints in the form of temporal constraints while the timeline completer is called to fill up the holes of the timelines, that is, the parts where no decision has been yet assigned.

The rest of this section focuses on the description of the two new planners: the strategic and decomposer.

4.1 QE-Strategic

This planner is in charge of guiding the search. It has been designed as a Producer-Consumer, each running in a separate thread to favour parallel computing thanks to its flaw-oriented approach. The producer is in charge of deciding the next SolvingSpaceNode to be solved, while the consumer analyse the partial solutions provided by the resolvers.

The algorithm 1 shows the cycle of the producer.

```
begin
    while (¬exit_cond) do
        if (pendingJobs(shared_info)) then
            wait_to_finish()
        else
            node ←
            sel_next_solv_node(search_space)
            ρ ← sel_next_resolver(node)
            create_thread(ρ, node, domain)
```

Algorithm 1: producer(search_space, domain, exit_cond)

Given a search-space, the producer chooses the deepest node in the tree as it is expected to be the most evolved and closer to the solution. However, in the future it will be replaced by an A* algorithm with an heuristic based on the number of pending flaws to be resolved. Once the SolvingSpaceNode is selected, a resolver is chosen. Given the fact that each resolver tries to solve all the flaws of its type in the node, the selection follows a cycle depicted in figure 3. First, the Unfolder add new sub-goals to support the goals already existing in the problem and calls the scheduler to order them. Once it can't go any further, the decomposer is called to decompose all complex-goals that must be decomposed (a complex goal might not be required to be decomposed as explained in section 4.2). When the decomposer finishes, the unfolder is called back to add supporters for the new sub-goals. This process iterates until the point where a partial solution is found, in which case the TimelineCompleter is called, or exit_cond becomes true, in which case the algorithm exits. In case the TimelineCompleter adds new sub-goals, then the unfolder is called back and the process is repeated. The exit_cond of the main loops becomes true when there is no pending nodes (partial solutions) in the search space and there is no pending jobs, that means, no thread is still running.

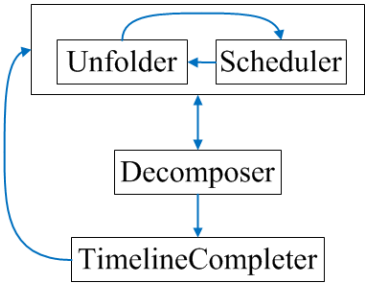


Figure 3: Cycle of calls to the different resolvers

```
begin
    while (¬exit_cond) do
        solution ← wait_next_solution()
        node ← create_new_node(solution)
        add_to_search_space(node)
```

Algorithm 2: consumer(search_space, domain, exit_cond)

The consumer (algorithm 2) waits in a loop until any solver produces a new solution. Equally to the producer, the consumer exits the loop when there is no pending solutions to be managed nor any pending job. Once a job ends and a solver produces a solution, the consumer analyse it. In case it is partial, a new node is created and added to the search space. Otherwise, it is added to the list of solutions.

To guarantee the consistency of the information, both producer and consumer share a thread-safe data struc-

 DOI: 10.2420/AF09.2014.21

ture containing information used by both of them, including the search space, list of pending jobs and solutions.

4.2 QE-Decomposer

QE-Decomposer is in charge of decomposing high level goals into sub-goals.

```
begin
    for (all goals to decompose) do
        goal ← goals
        methods ←
        get_decompositions(goal)
        dn_dec ← select_candidate(methods)
    solution ← create_solution(dn_dec)
    return solution
```

Algorithm 3: decomposer()

The decomposer (algorithm 3) receives as input a dn with at least one complex node to be decomposed d^c_{dec}. Even though the resolver will have to decompose all $d^c_{dec} \in dn$, the order is important as the decomposition of one node might restrict the possibilities of the next ones. Therefore, the next node d^c to be decomposed is heuristically selected based on the number of methods in which it can be decomposed. Then, one of the methods is selected, also heuristically. In this case, domain-dependent heuristics are preferred over general ones because the different methods encode expert knowledge that require domain-dependent information to choose among them. Once the method is selected, the decomposition network dn_{dec} is created and added to the list of decompositions that represent the solution. When the planner finishes, the solution containing all the decompositions is sent back and captured by the consumer which will analyse it.

4.3 Sufficient planning

HTLN allows the planners to create partial solutions in which not every complex node has to be decomposed. This solution is called *Sufficient Plan* [7]. To support this capability, the problem description language used in APSI (PDL) had to be slightly modified. For each d^c of the problem, the user can specify an *exclusion list* that indicates to the planner which decisions, in case they are used as sub-tasks of d^c, do not need to be decomposed. As an example, the first Driving goal g1 shown bellow indicates that it should not be decomposed at all, while g2 express that in case it contains either Autonav or Blind, these sub-goals should not been decomposed.

- g1 < goal, Driving > Master.Drive (?xi, ?yi, ?xf, ?yf, ?travers);
- g2 < goal, Auto, Blind > Master.Drive (?xi, ?yi, ?xf, ?yf, ?travers);

Internally, each decision d of a problem dn_i contains two variables. mustDecomposed is a boolean condition set to false in case the goal was in the list mentioned before or true otherwise. decompositionLevel indicates the present level of decomposition of a complex goal, which is calculated with two different algorithms depending on whether d is a component decision (algorithm 4) or a decision network (algorithm 5).

```
begin
    if (¬d ∈ Goals) then
        decompositionLevel ← TOTALLY
    else if (isPrimitive(d)) then
        decompositionLevel ← TOTALLY
    else if (mustDecomposed) then
        decompositionLevel ←
        NOT_SUFFICIENTLY
    else
        decompositionLevel ←
        SUFFICIENTLY
```

Algorithm 4: recalculateDecomposition(d ∈ CD)

For a d ∈ CD, if the decision is primitive or a fact (is not in the goals list), its decomposition level is TOTALLY. If d ∈ CD must be decomposed, then its level of decomposition is NOT_SUFFICIENTLY because a decomposed decision should be d ∈ DN. In any other case, d is SUFFICIENTLY decomposed.

```
begin
    min_elem ← lowest decomposition level among all
    d ∈ dn_i
    if (mustDecomposed(dn_i)) then
        minim_dn ← NOT_SUFFICIENTLY
    else
        minim_dn ← SUFFICIENTLY
    decompositionLevel(dn_i) ←
    max(min_elem, min_dn)
```

Algorithm 5: recalculateDecomposition(d ∈ DN)

For a d ∈ DN, it is first calculated the minimum decomposition level of all its sub-tasks and the minimum possible value for d, which is NOT_SUFFICIENTLY in

case it must be decomposed and SUFFICIENTLY otherwise. The level of decomposition of d is the maximum of these two values. In case the new value changes respect the previous one, the new value is propagated up to the parents of d, $dn^{sup}(d)$.

5 Initial results

We have implemented two different rover domains in a hierarchical and flattened style in order to compare QE and AP^2. The only modification between the two models has been the translation of the decompositions as synchronizations, which have similar semantics, and no additional constraints have been required. Some properties of the model are shown in table 1.

Property	RD-C	RD-S
Number of timelines	9	7
Total number states	27	19
Number decompositions	4	2
Number synchronizations	14	6

Table 1: Rover domain properties

The first domain, called *RD-C* (Rover Domain Complex) fully represents the model presented in section 3. The second one named *RD-S* (Rover Domain Simple) has removed a level of decompositions, as Driving is directly decomposed in the primitive tasks without the Blind and Autonav intermediate layer. Moreover, the Navigation and Drilling components have been removed.

The tests have been run in an Intel Core i5 M540 at 2.53 GHz computer with 3 GB RAM. The OS is Windows 7 Enterprise 32 bits. To understand the level of impact of the different decomposition methods in the performance an, both QE and AP^2 were configured to choose randomly the decomposition/synchronizations to apply. Notice that for the rover domain, the decomposition of a Driving goal as Autonav impose way more constraints than Blind.

We have created 7 problems named *Problem-1* to *Problem-7* with increasing level of complexity. Each problem *Problem-x* contains a list of *facts* defining the initial state for each component and *x* high-level goals, i.e. goals from the Master component. Being Driving the most complex goal, we have alternated between Blind and Autonav driving goals to cover all the possibilities during testing.

A simplified instantiation of *Problem-4* for the domain *RD-C* is shown bellow.

```
PROBLEM Rover-Problem (DOMAIN Rover_Domain){
 f1 <fact> Locomotion.At(?x=0, ?y=0);
 f2 <fact> Mast.PointingAt(?pan=0, ?tilt=0);
 f3 <fact> Camera.CamIdle();
 f4 <fact> Driller.DrillIdle();
 f5 <fact> Planner.PlannerIdle();
 f6 <fact> Antenna.CommIdle();
 f7 <fact> Navigation.NavIdle();
 f8 <fact> Master.Idle();

 f9 <fact> CommWindow.Visible() AT [0,5];
 f10 <fact> CommWindow.Visible() AT [40,80];

 g1 <goal> Master.Drive(?x=1, ?y=1, ?trav=easy);
 g2 <goal> Master.Pic(?x=1,?y=1,?pan=2,?tilt=2);
 g3 <goal> Master.Drill(?x=1, ?y=1, ?depth=1);
 g4 <goal> Master.Drive(?x=4, ?y=4, ?trav=hard);

 g1 BEFORE [1,+INF] g2;
 g2 BEFORE [1,+INF] g3;
 g3 BEFORE [1,+INF] g4;
```

Most of the parameters are self-explanatory. In order to simulate the inputs from sensors and the output from AnalyseTerrain in charge of deciding which type of navigation the rover should do, we added an extra parameter to driving that specifies by hand the type of terrain. Few modifications were required for the simpler domain *RD-S*. Drilling activities are replaced by TakingPicture and facts *f4* and *f6* were removed from the initial state.

It is important to mention that QE has been used without exploiting its multi-threading capability. The results showed that both planners were using intensively just one of the four cores of the processor, therefore we consider that this capability can boost the planner performance in the future.

For each run, we have registered the execution time to the first solution found and the number of nodes and edges in the solution network. The results comparing QE and AP^2 are shown in Figure 4.

In the first domain, *RD-C*, AP^2 starts to have problems even with very simple problems due to the big branching factor. While AP^2 has to branch for all possible disjunctive synchronizations, QE goes straight to the solution by using the decomposition methods.

In the second scenario, AP^2 managed to solve up to *Problem-5*. The number of edges increments notoriously faster than the number of nodes, crucial to understand the degradation in AP^2 performance as the problem complexity grows. Moreover, QE consistently generate less edges than AP^2. It is also interesting to remark that the peak in time doesn't come with the second Driving activity, which impose itself a number

 DOI: 10.2420/AF09.2014.21

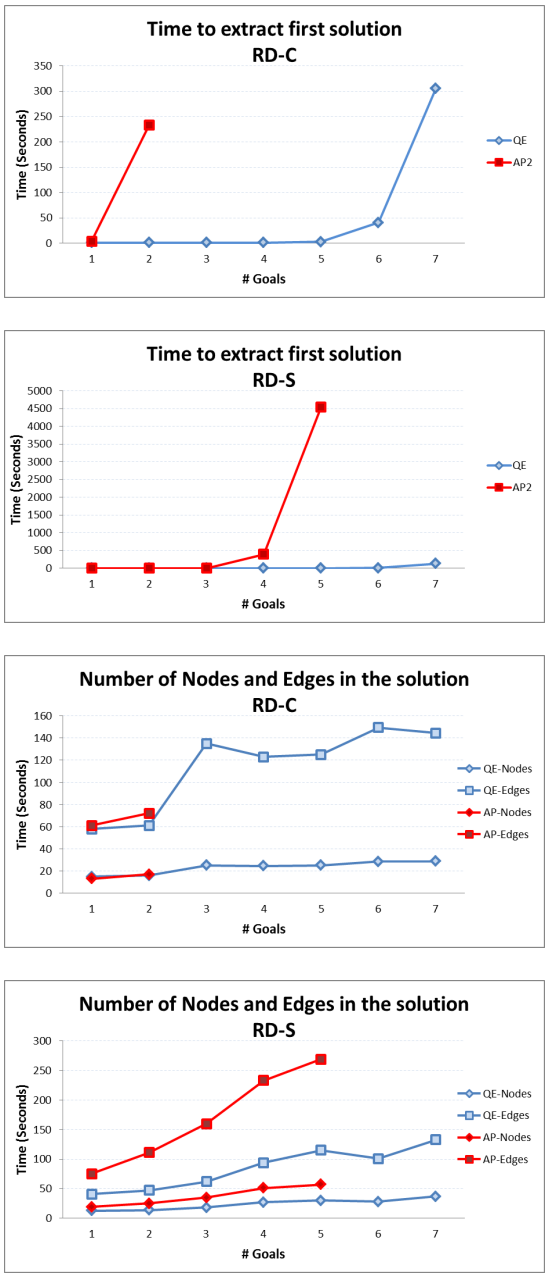


Figure 4: Comparison between QE and AP^2.

of sub-tasks and sub-relations, but with the next activity which is `TakingPicture`. Even though further analysis should be accomplish, our first impression is that after the second `Driving`, the planners (Unfolder and Scheduler) need to do a lot of processing to insert `TakingPicture` and unify some of its parameters. The same effect can be seen in for QE in the first scenario.

Nonetheless, these represent preliminary results that require more experiments to have a better understanding of how both planners explore the search space and generate the solutions.

6 Conclusions and future work

We have presented QuijoteExpress, a solver based in APSI*, an extension of the APSI framework that exploits Hierarchical Timeline Networks for temporal problems. Even though further experiments should be conducted, the results provided from running 6 different problems for 2 different domains in both QE and AP^2 are even better than our original expectations, showing an important improvement on performance. We will conduct more experiments in the future to analyse which are the stronger and weaker points for each planner and evaluate the effect of sufficient planning and parallel planning approaches.

There is a number of lessons learnt extracted from the tests with the rover domain.

- The capability of HTN planners to reduce the branching factor and therefore the size of the search space is crucial to understand the great benefit in terms of performance.

- HTN represents an overhead during the modelling process. The design and validation/verification of the four models implemented (hierarchical and non-hierarchical) was tedious and error prone. This fact is particularly relevant in HTN, where the model can present hidden constraints/dependencies between elements in different levels of abstractions. In consequence, we consider that it will be crucial in the future to develop new tools able to assist during the construction of complex models.

- Debugging and understanding the output of the planner is almost impossible for a non-expert. HTN plays an important role in this two aspects. During the the runs with the hierarchical model, we could identify a fault condition just observing the high-level goals and their relations. It would have taken much more time in case we would have needed to go through all the primi-

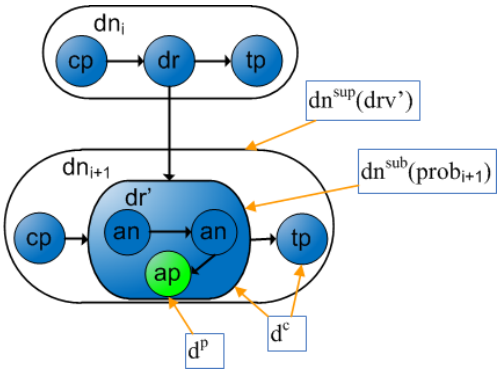


Figure 5: Hierarchical dn structure

tive components in a flat domain. In the same way, it is easier for the user to understand the plan by examining only the timelines related with complex components.

- Modern domain description languages are not expressive enough. In our case, even with the help of a time-oriented language, we were highly constrained. Extending expressiveness or moving towards the use of standard programming languages, as it happens with SMASH for ROS, will be required to construct realistic models.

- It is important to provide tools that allow users to represent knowledge. Domain-dependent knowledge must be contained in dedicated structures to maintain the system reusable, in our case in the methods and heuristics. Regarding method selection, we believe that general heuristics such as method timespan, number of complex tasks, etc. are not appropriate because the selection use to depend on the same specific knowledge the method represents. For example, in the rover domain, the selection between Blind and Autonav should be done by an expert-system according to the information gathered by the rover sensors.

In the future, our intention is to further improve the planner performance in two ways: implementing search heuristics and enable parallel planning. Creating an agile, anytime planner is critical to use it in scenarios that demand continuous re-planning such as Mars Rovers, telescopes or rescue robots. Comparing QE with other planners using similar techniques such as ASPEN [5] or Europa [10] would be important to understand where we are. So far we have just run simulations with virtual models, but it is our intention to start soon doing tests with real robots which should be of great relevance to understand how to evolve the system.

Acknowledgments

This research has been co-funded by the Networking/Partnering Initiative (NPI) between ESA-ESOC, TU Darmstadt and University of Surrey. It also receives support from the German Research Foundation (DFG) within the Research Training Group 1362 “Cooperative, adaptive and responsive monitoring in mixed mode environments”.

APPENDIX A – Nomenclature

Acronym	Description
C_i	Component i
cd	Component Decision
rlt	Relation
dn	Decision Network
$w = (N, E)$	Hypergraph used to represent a dn. N is the list of nodes and E the list of hyperedges
d^p	Primitive task
d^c	Complex task
D^p	Set of all primitive tasks
D^c	Set of all complex tasks
D^{all}	Set of all tasks
m	Method, implemented in HTLN as a dn
$dn_{dec}(d)$	Method selected to decompose a decision $d \in D^c$
$dn^{sup}(d)$	Parent decision of d. It will be either a dn or $\emptyset$ in case d is the problem network
$dn^{sub}(d)$	Child decision of d. It will be either $dn^{sub}(d) = \emptyset$ if $d \in D^p$ or $dn^{sub}(d) = dn$
ic	Initial condition
ρ	Procedure intended to fix a flaw ϕ in a given problem dn_i
d^+, d^-	List of decisions returned by the resolver ρ to be added/retracted to the problem dn_i
rlt^+, rlt^-	List of relations returned by the resolver ρ to be added/retracted to the problem dn_i
$\rho(from, \sigma)$	Decomposition resolver that decomposes the decision $from \in dn_i$ in the sub-network to using the substitution σ

APPENDIX B – Hierarchical dn structure

Figure 5 illustrates the concepts presented before. dn's are represented as ellipses, cd's as circles and rlt's as segments. The problem contains three compound goals

 DOI: 10.2420/AF09.2014.21

(cp, dr, tp) which stand for *calculate path*, *drive* and *take picture* respectively, ordered between them by *before* temporal constraints (represented as arrows). Drive is the only d^c already decomposed. Therefore, it is represented not as a cd but as a dn (dr') containing three sub-tasks, where an stands for *autonav* and ap *approach*. While ap is primitive, an is compound and will require to be further decomposed adding another nested dn into drive. $dn^{sup}(d)$ represents the dn one layer up of d and $dn^{sub}(d)$ one layer bellow d. Being dn_i the problem network, $dn^{sup}(dn_i) = \emptyset$, that is, a problem has no layer above and $dn^{sub}(d) = \emptyset$ if d is in the last layer.

References

[1] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, Nov. 1983.

[2] J. J. Biesiadecki and M. W. Maimone. The mars exploration rover surface mobility flight software driving ambition. In *IEEE Aerospace Conference (IAC)*, number March, 2006.

[3] A. Ceballos, S. Bensalem, A. Cesta, L. S Silva, S. Fratini, F. Ingrand, J. Ocon, A. Orlandini, F. Py, K. Rajan, R. Rasconi, and M. V. Winnendael. A goal-oriented autonomous controller for space exploration. *ASTRA*, 2011.

[4] A. Cesta, S. Fratini, A. Orlandini, and R. Rasconi. Continuous Planning and Execution with Timelines. In *International Symposium on Artificial Intelligence (i-SAIRAS)*, number 1, 2012.

[5] S. Chien, G. R. Rabideau, R. L. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. A. Estlin, B. Smith, F. W. Fisher, A. C. Barrett, G. Stebbins, and D. Tran. ASPEN - Automated Planning and Scheduling for Space Mission Operations. In *International Conference on Space Operations (SpaceOps)*, pages 1–10, 2000.

[6] J. M. Delfa Victoria, N. Policella, G. Yang, and O. V. Stryk. Design Concepts for a new Temporal Planning Paradigm. In *ICAPS Planning & Scheduling for Timelines (PSTL), 2012*, 2012.

[7] J. M. Delfa Victoria, N. Policella, G. Yang, and O. V. Stryk. QuijoteExpress - A Novel APSI Planning System For Future Space Robotic Missions. In *ASTRA*, 2013.

[8] K. Erol, J. Hendler, and D. S. Nau. Semantics for Hierarchical Task-Network Planning. Technical report, 1994.

[9] T. A. Estlin, B. J. Bornstein, D. Gaines, R. C. Anderson, D. R. Thompson, M. Burl, R. Castano, and M. Judd. AEGIS Automated Science Targeting for the MER Opportunity Rover. *International Symposium on Artificial Intelligence Robotics and Automation in Space (i-SAIRAS)*, pages 1–25, 2010.

[10] J. D. Frank and A. K. Jonsson. Constraint-based Attribute and Interval Planning. *Journal of Constraints Special Issue on Constraints and Planning*, 8(4):339–364, 2003.

[11] S. Fratini and A. Cesta. The APSI Framework: A Platform for Timeline Synthesis. *Proceedings of the 1st Workshops on Planning and Scheduling with Timelines PSTL-12*, 2012.

[12] K. Nagatani, S. Kiribayashi, Y. Okada, K. Otake, K. Yoshida, S. Tadokoro, T. Nishimura, T. Yoshida, E. Koyanagi, M. Fukushima, and S. Kawatsuma. Emergency response to the nuclear accident at the fukushima daiichi nuclear power plants using mobile rescue robots. *J. Field Robot.*, 30(1):44–63, Jan. 2013.

[13] A. Tate, B. Drabble, and R. Kirby. O-Plan2: an open architecture for command, planning and control. In *Intelligent Scheduling*, volume 1, pages 213–239. 1994.

[14] D. E. Wilkins, K. L. Myers, J. D. Lowrance, and L. P. Wesley. Planning and reacting in uncertain and dynamic environments. *Journal of Experimental Theoretical Artificial Intelligence*, 7(1):121–152, 1995.

Acta Futura 9 (2014) 31-39
DOI: 10.2420/AF09.2014.31

Acta Futura

Onboard Autonomous Response for UAVSAR as a Demonstration Platform for Future Space-Based Radar Missions

Joshua Doubleday, Steve Chien*, Yunling Lou, Duane Clark and Ron Muellerschoen
Jet Propulsion Laboratory, California Institute of Technology, 91109-8099, Pasadena, CA (USA)

Abstract. The paper discusses software to enable: 1. onboard processing and interpretation of radar data and 2. autonomous response to retask vehicle and instrument based upon interpretation of this data. We first discuss scenarios in which space-based radar could benefit from this autonomous interpretation and response capabilitiy. Next we discuss the Uninhabited Aerial Vehicle Synthetic Aperture Radar (UAVSAR) airborne testbed and its use as a surrogate for a spaceborne testbed. We then discuss a range of onboard processing products that have been investigated and produced onboard. We then discuss the retasking model and process for UAVSAR. We discuss a flight demonstration of the onboard data processing and retasking that occured in January 2012. Finally we discuss related work and areas for future work.

1 Introduction

Space based radar has been found to have a wide range of science and humanitarian applications [1, 17] including but not limited to: vegetation and ecosystem studies [2, 5, 14], wildfires [13], flooding, soil moisture, and hydrology [16, 21], solid earth (earthquakes, landslides, volcanoes) [3, 19, 22] and cryosphere (glaciers, climate change). NASA and other space agencies have flown a number of missions that provide this space-based radar capability for these science and applications areas. These missions include but are not limited to RADARSAT-1 and RADARSAT-2 (Canadian Space Agency), TerraSAR-X and TanDEM-X (German Space Agency), ALOS/PALAR (Japanese Space Agency), Shuttle Imaging Radar (National Aeronautics and Space Administration), Envisat/ASAR (European Space Agency). These same agencies are also studying possible future missions to provide this unique capability for science and applications.

Synthetic aperture radars utilize the flight path of the instrument to synthesize a large aperture to form radar/backscatter images of the earth. Interferometric processing of radar images can utilize multiple instruments or multiple overflights to synthesize altimetry and/or change data for a surface. Below we show a sample UAVSAR image acquired of a glacier in Greenland from June 2009 (Figure 1) and a change detection interferogram product showing land displacement in Baja California from October 2009 to April 2010 (Figure 2).

One challenge of space-based imaging radars is that radar data can be extremely large. Onboard processing offers several advantages over traditional ground-based processing. Onboard products can be used to deliver much smaller notifications (alerts) and or summary products to ground personnel and assets using more convenient data low capacity links. Onboard prod-

*Corresponding author. E-mail: steve.chien@jpl.nasa.gov

Figure 1: Image of Greenland Glacier acquired by UAVSAR in June 2009. Image courtesy JPL/NASA.

ucts can be used to retask the acquiring asset or other assets to acquire followup imagery. Analysis of the onboard products can be used to selectively not downlink or delete the full aquired imagery, thereby relieving storage and downlink resources.

In the remainder of this paper, we describe efforts to develop and demonstrate the capability to develop radar products onboard and retask assets based on interpretation of these products.

2 The Uninhabited Aerial Vehicle Synthetic Aperture Radar (UAVSAR)

The UAVSAR is an airborne platform developed by NASA to study earth science and emergency response potential [11] using remote sensing radar. UAVSAR is a Gulfstream-III jet with the radar mounted in a pod below the fuselage (see Figure 3). While UAVSAR is a piloted aircraft it is intended as a precursor testbed to the deployment of the UAVSAR radar onto unpiloted aerial vehicles as well as space vehicles.

Within the UAVASAR aircraft computers and instrument processing hardware are mounted to enable onboard processing of the radar imagery. Onboard general purpose computing hardware are also used for onboard image/product analysis and retasking the UAVSAR. The onboard processing hardware is shown to the left of the aisle in the image in Figure 4 below.

Special purpose cards (shown below in Figure 5) including field programmable gate array (FPGA) capability are used to process the radar images in near real time.

3 Autonomous Response Scenario: Space-Based Radar and Airborne Radar

Enabling onboard interpretation and response has many applications for space-based and airborne radar. In each case the first step is to form the radar image. Luckily, for our onboard autonomy work, the UAVSAR project has been addressing exactly this challenging task [15]. Onboard the UAVSAR, the raw radar data is streamed to recorders and is simultaneously streamed to the *Onboard Processor* which forms the synthetic aperture radar image. This radar image can be formed in a range of polarizations (e.g, HH, HV, etc.). Once the radar image is formed, the backscatter image data can be interpreted using application-specific algorithms. Based on the mission at hand, this interpretation can then be used to direct future operations of the space or air vehicle. For space-based radar, applications of onboard autonomy abound. For example, detection of volcanic activity by detection of ash emissions might trigger followup imagery on a later orbital overflight. Or mapping of the flooded area might be used to direct the same or different radar to acquire higher resolution imagery of the boundary of the flooded area. Or the same area might be imaged on a subsequent overflight to map out a timeseries of the progression of the flood. Alternatively, biomass analysis might be used to map out the progression of a forest fire. All of these scenarios involve the same baseic operations pattern of:

- form radar image
- analyze radar image
- generate new target requests
- assimilate new target requests into operational plan as appropriate based on prioritization.

These scenarios are highlighted by the operations flow in Figure 6.

 DOI: 10.2420/AF09.2014.31

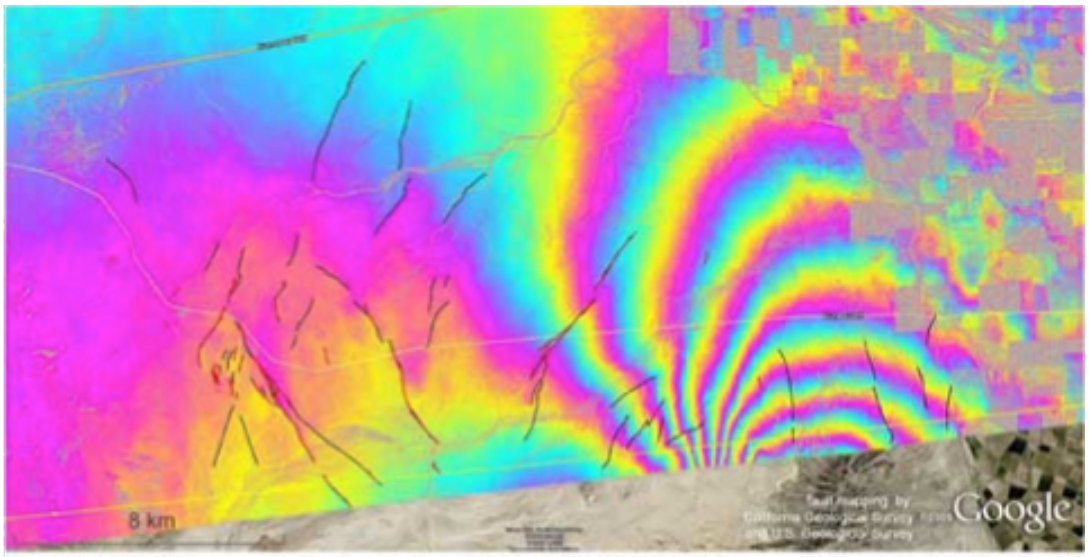


Figure 2: Interferogram showing displacement (change) from an earthquake in Baja California, Mexico. First image acquired in October 2009. Second image acquired in April 2010. Black lines indicate interpreted faults and red lines show where geologists in the field confirmed surface rupture. Image courtesy NASA JPL/United States Geological Survey/California Geological Survey/Google.

4 Onboard processing of radar imagery

Once the SAR image has been formed [15] it can be used as the basis of interpretation for many science and application purposes [7, 8]. The multiple available polarizations contain significant information that can reveal surface smoothness properties, structure properties, as well as water content/moisture properties of the substances being imaged. While interferometric analysis can also provide tremendous amounts of data for many targets, since our autonomy thus far has focused on individual overflights (and UAVSAR cannot do single pass interferometry) we have focused on SAR imaging capabilities. The table below in Figure 7 shows a number of onboard processing analysis products that have been considered in this and other efforts.

In our specific autonomy demonstration, we used an amplitude segmentation algorithm that is sensitive to changes in surface roughness/smoothness features of the surface being imaged. This algorithm is useful for detecting differences in the substance being imaged (e.g. liquid versus solid land) as well as covering layers (e.g. oil on the surface of water). This algorithm consists of a number of processing steps.

- Each base image is multilooked/subsampled at different resolutions into a stack of images. The remaining procedure's results will vary with the incoming resolution due to pixel-to-pixel value variation and/or perceived sharpness of edges in the image.

- Each subsampled image is then segmented using a Felzenswalb/graph based segmentation routine, which generates a minimum spanning forest over the 4-neighborhood graph of pixel-magnitudes. This step is nearly linear run-time on the image size. This step produces regions/super-pixels of similar radar response that can be measured in area-of-coverage, geographic location, average intensity, and other statistical quantities.

- The resulting regions/segmentations are checked for "interest" criteria, such as minimum size, and in the case of our "oil-slick" demonstration, an average backscatter intensity below a tunable threshold (-28db).

- The stack of segmentation images are expanded to the greatest common resolution and a composite (pixel average) image is created as a highly compressible image containing qualitative visual features at various scales.

Figure 3: The UAVSAR aircraft. Note the radar mounted in the pod directly underneath the fuselage

Figure 4: The interior of the UAVSAR aircraft. The rack to the left of the aisle is the rack for the onboard processing equipment.

Below in Figure 8 we show images of the Deep Horizons oil spill in the Gulf of Mexico along with the corresponding products processed with a range of segmentation scales.

5 Demonstration of onboard autonomy on UAVSAR

In this section we describe the autonomy software onboard the UAVSAR and its use in an onboard autonomy demonstration in January 2012. In this demonstration the UAVSAR successfully executed a number of steps.

- processed radar data into SAR imagery,
- autonomously analyzed the imagery,
- derived new observation goals from this analysis,
- developed a new flight plan to achieve the new observation goals, and
- when concurred by the radar operator and pilot UAVSAR flew the new flight plan to achieve the observation goal.

5.1 UAVSAR Autonomy System

The UAVSAR Autonomy System consists of a number of modules which are described below.

- The **OnBoard Processor** (OBP) takes a stream of the radar instrument raw image data and also data indicating the position, orientation, and velocity of the aircraft derived from GPS and inertial navigation unit (INU) data. It uses this information to form the SAR image of the relevant area of ground underneath the aircraft. The OBP can form images of various polarizations as needed by the subsequent analysis (e.g. HH, HV).
- The **Onboard Analysis** (OA) module is responsible for interpreting the formed SAR image. In our specific demonstration we utilized a surface smoothness analysis algorithm. The Onboard Analysis software takes the formed radar polarization and/or backscatter image and produces the analysis product.
- The **Target Generator** (TG) software uses the analysis product to generate new observation goals

 DOI: 10.2420/AF09.2014.31

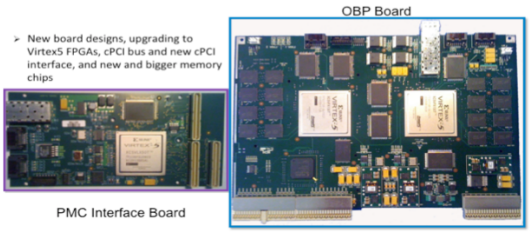


Figure 5: Closeup of the onboard radar processing cards.

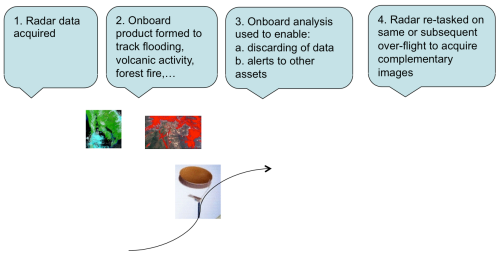


Figure 6: Autonomous interpretation and response scenario for space-based radar

for the planners to achieve. In our demonstration, the TG analyzes the imagery and selects smooth surface targets for followup imagery. For each area greater than a threshold size and smoothness, it generates a Point of Interest (POI) which in effect is an observation goal to image the area.

- The **Flight Line Analyzer** (FLA) takes the current flight plan and new observation goals (POIs) and first checks if all of the newly requested observation goals are achieved by the existing flight plan. If not, it selects a new flight line to propose for addition to the flight plan that covers the highest priority new observation goal. This new flight line is then submitted to the Flight Planner below.

- The **Flight Planner** (FP) attempts to add the new flight line to the current flight plan. It uses the Solomon insertion heuristic that tries insertions of the new flight line into the existing flight plan that minimize the makespan (total duration) of the resultant schedule.

- The **CASPER Planner** (CP) is a time, state, and resource planner that has been applied to a range of mission planning applications including onboard spacecraft control [6], rover control [9], aerobot control [10], and autonomous underwater vehicle planning [20]. CASPER is used to model UAVSAR operational constraints as well as deadlines. If the new flight plan passes these constraints modelled in CASPER it is passed on to the Radar Operatior Workstation.

Algorithm	Applications	Notes
Soil moisture	Agriculture, Water resource management	Currently requires sparse vegetation; generalization of algorithms to variable/ dense vegetation future work.
Surface water extent	Flood mitigation	Extensions to varied topography, rough waters, and smooth land are future work.
Repeat-pass disturbance	General	Requires expert/interpreter and prior imagery onboard.
Snow/ice vs. land SVM classification	Transportation, Freeze-thaw monitoring	Vegetation can complicate classification.
Amplitude Correlation	Sea transportation, Glacial movement monitoring	Strong transportation application if provided repeat-pass data at rates of ~1hr. Glacial studies would desire higher fidelity (sub-pixel motion, per pixel).
Biomass	Wildfires, Invasive species, avalanche, blowdown, landslide, subsidence, tsunami	

Figure 7: A number of potential onboard radar products and their applications.

- The **Radar Operator Workstation** (ROW) manages the radar for the data acquisitions and also serves as the interface to the autopilot. The ROW receives the new CASPER-approved flight plan from CASPER and at the ROW the human radar operator can approve the flight plan for submission to the UAVSAR autopilot.

- The **UAVSAR Autopilot** (UA) actually flies the flight lines to the tolerances required by the UAVSAR SAR instrument. The UA receives the new flight plan from the ROW and allows the pilots to view the new flight plan. The pilots then review and accept or reject the new proposed flight plan. If the new flight plan is accepted then the autopilot (and UAVSAR) will then fly the new flight plan (including the new flight line for the new observation goal).

This operations flow is detailed below in a flowchart in Figure 9.

5.2 Autonomy Demonstration

In January 2012, the UAVSAR autonomy system was exercised in a flight demonstration. In this flight, an

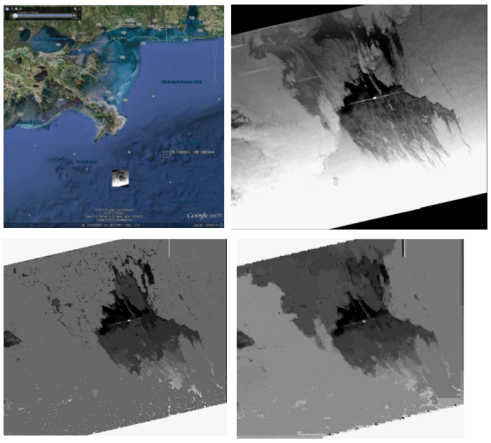


Figure 8: Oil Detection/Amplitude Segmentation Analysis Product on Gulf Of Mexico Oil Spill data. Upper Left: Context image showing location of oil spill south of the Mississippi delat in the Gulf of Mexico. Upper Right: Original browse image of HH polarization backscatter. Lower Left 60m segmented image Lower right: 60m composite image.

initial set of points of interest (POI's, e.g. areas to observe), flight plan of flight segments and CASPER plan was loaded into the UAVSAR autonomy system. As the UAVSAR flies the flight plan, it takes observations according to the plan and the Onboard Processor forms the radar images as shown in Figure 10.

As per the original plan, the Onboad Analysis Software also generates the Amplitude Segmentation product with the corresponding detections for each newly acquired image. These producs are shown in Figure 11. Using these products, the Target Generator generates a new observation goal for each detection. The Flight Line Analyzer then checks each of these new observation goals against the planned flight lines remaining in the current observation plan. Any goals that are not covered by existing segments in the remaining plan are flagged and the Flight Line Analyzer searches for flight lines in the current catalogue to cover these observation goals. For operational reasons all flyable flight lines must be pre-filed before flight, therefore preventing the flight line analyzer from creating novel flight lines.. In this case a flight line is found that covers the new observation goals. The Flight Planner (FP) then sequences the additional flight segment into the remaining plan minimizing overall flight time. This proposed plan is then

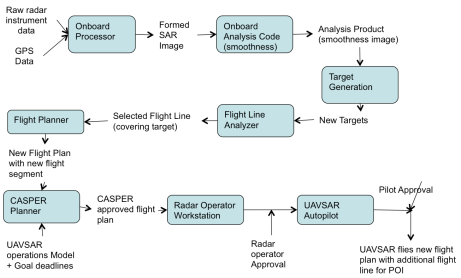


Figure 9: The operations flow for onboard processing and retasking the UAVSAR.

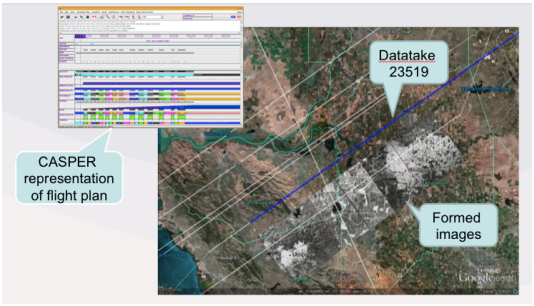


Figure 10: UAVSAR flies the original flight plan and acquires data according to plan. The Onboard Processor forms images as shown.

sent to CASPER which validates that the plan does not violate UAVSAR operations state, resource, and flight deadline constraints. CASPER also includes new processing and analysis goals for the new data acquisitions. The CASPER Graphical User Interface is shown in Figure 12 displaying the CASPER plan including the new flight line.

Next, the CASPER validated plan is passed onwards to the ROW for radar operator approval. Once this is complete, it is passed on to the Autopilot where after pilot review it is accepted and the new plan is flown by the UAVSAR. This successful flight demonstration illustrates how the UAVSAR can be operated in a highly autonomous fashion to close the loop of data acqusition, analysis, and retasking.

6 Discussion and Conclusions

6.1 Related Work

Considerable work has focused on Unpiloted Aerial Vehicle path planning either for single or multiple vehi-

 DOI: 10.2420/AF09.2014.31

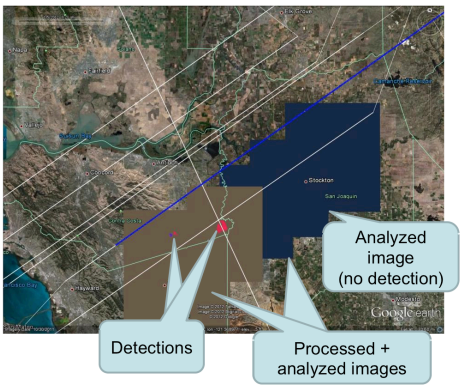


Figure 11: UAVSAR Onboard Analysis Software generates analysis of image. Target Generator then produces observation goals for new detections.

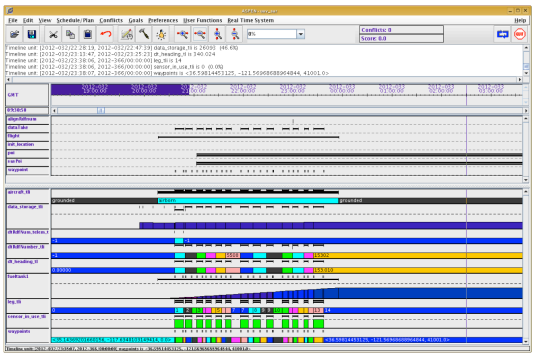


Figure 12: CASPER validates that new flight plan does not violate UAVSAR state, resource, and timing operational constraints and obeys UAVSAR flight deadlines.

cle configurations [18, 12, 4]. However this work has generally started with inputs that are waypoint goals. In contrast, this work has focused on integration of complex instrument data interpretation into the overall onboard autonomy including timing, resource, and path planning constraints (while not incorporating more complex path planning constraints such as avoidance zones, maneuvers, uncertain terrain, and other constraints).

6.2 Conclusions

In this paper we have described an overall approach to mission autonomy designed for space and airborne radar platforms. In this approach, onboard hardware and software is used to enable onboard processing of radar data into images and then to analysis products. These analysis products are used to drive generation of new observations goals, such as to enable tracking of dynamic phenomena such as flooding, oils spills, wildfires, or lava flows. Onboard flight planning, and state, resource, and deadline mission planning are then used to replan current mission and flight plans to accomodate these new goals as priorities warrant. This overall closed loop control enables more rapid response to capture key science and applications data for fast moving science and applications phenomena.

Acknowledgements

This work was perfomed by the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration.

References

[1] NASA Earth Observing Missions Applications Workshop, February 2010.

[2] K. Bergen, S. Goetz, R. Dubayah, G. Henebry, C. Hunsaker, M. Imhoff, R. Nelson, G. Parker, and V. Radeloff. Remote sensing of vegetation 3-D structure for biodiversity and habitat: Review and implications for lidar and radar spaceborne missions. *Journal of Geophysical Research: Biogeosciences (2005–2012)*, 114(G2), 2009.

[3] J. Biggs, E. Robertson, and M. Mace. ISMER Active Magmatic Processes in the East African Rift: A Satellite Radar Perspective. In *Remote Sensing Advances for Earth System Science*, pages 81–91. Springer, 2013.

[4] S. A. Bortoff. Path planning for UAVs. In *Proceedings of the American Control Conference.*, volume 1, pages 364–368. IEEE, 2000.

[5] M. Burgin, D. Clewley, R. M. Lucas, and M. Moghaddam. A generalized radar backscattering model based on wave theory for multilayer multispecies vegetation. *IEEE Transactions on Geoscience and Remote Sensing*, 49(12):4832–4845, 2011.

[6] S. Chien, R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davis, D. Mandl, B. Trout,

S. Shulman, et al. Using autonomy flight software to improve science return on Earth Observing One. *Journal of Aerospace Computing, Information, and Communication*, 2(4):196–216, 2005.

[7] J. Doubleday, S. Chien, and Y. Lou. Low-latency DESDYNI data products for disaster response, resource management, and other applications. In *Proc. 34th International Symposium on Remote Sensing of Environment.*, Sydney, Australia, April 2011.

[8] J. Doubleday, D. Mclaren, S. Chien, and Y. Lou. Using Support Vector Machine Learning to automatically interpret MODIS, ALI, and L-band SAR remotely sensed imagery for hydrology, land cover, and cryosphere applications. In *Proc. International Joint Conference on Artificial Intelligence Workshop on Artificial Intelligence in Space (IJCAI 2011)*, Barcelona, Spain, July 2011.

[9] T. Estlin, D. Gaines, C. Chouinard, R. Castano, B. Bornstein, M. Judd, I. Nesnas, and R. Anderson. Increased Mars rover autonomy using AI planning, scheduling and execution. In *IEEE International Conference on Robotics and Automation*, pages 4911–4918, Rome, Italy, 2007. IEEE.

[10] D. Gaines, C. Chouinard, S. Schaffer, T. Estlin, and A. Elfes. Autonomous planning and execution for a future titan aerobot. In *Third IEEE International Conference on Space Mission Challenges for Information Technology*, pages 264–269, Pasadena, CA, 2009. IEEE.

[11] S. Hensley, C. Jones, and Y. Lou. Prospects for operational use of airborne polarimetric sar for disaster response and management. In *Proceedings IEEE Geoscience and Remote Sensing Symposium*, Munich, Germany, July 2012.

[12] J. How, E. King, and Y. Kuwata. Flight demonstrations of cooperative control for UAV teams. In *AIAA 3rd unmanned unlimited technical conference, workshop and exhibit*, 2004.

[13] D. P. Jorgensen, M. N. Hanshaw, K. M. Schmidt, J. L. Laber, D. M. Staley, J. W. Kean, and P. J. Restrepo. Value of a dual-polarized gap-filling radar in support of southern California post-fire debris-flow warnings. *Journal of Hydrometeorology*, 12(6):1581–1595, 2011.

[14] Y. Kim, T. Jackson, R. Bindlish, H. Lee, and S. Hong. Radar vegetation index for estimating the vegetation water content of rice and soybean. *Geoscience and Remote Sensing Letters, IEEE*, 9(4):564–568, 2012.

[15] Y. Lou, S. Chien, D. Clark, and J. Doubleday. Onboard Radar Processing Concepts for the DESDynI Mission. In *Proceedings of the Earth Sciences Technology Forum*, College Park, MD, June 2011.

[16] D. Mason, G.-P. Schumann, J. Neal, J. Garcia-Pintado, and P. Bates. Automatic near real-time selection of flood water levels from high resolution synthetic aperture radar images for assimilation into hydraulic models: a case study. *Remote Sensing of Environment*, 124:705–716, 2012.

[17] NASA. Report of the 2009 DESDynI Applications Workshop, October 2008.

[18] D. Rathbun, S. Kragelund, A. Pongpunwattana, and B. Capozzi. An evolution based path planning algorithm for autonomous motion of a UAV through uncertain environments. In *Proceedings of the 21st Digital Avionics Systems Conference*, volume 2, pages 8D2–1. IEEE, 2002.

[19] L. Scharff, F. Ziemen, M. Hort, A. Gerst, and J. Johnson. A detailed view into the eruption clouds of Santiaguito volcano, Guatemala, using Doppler radar. *Journal of Geophysical Research: Solid Earth (1978–2012)*, 117(B4), 2012.

[20] D. R. Thompson, S. Chien, Y. Chao, P. Li, B. Cahill, J. Levin, O. Schofield, A. Balasuriya, S. Petillo, M. Arrott, et al. Spatiotemporal path planning in strong, dynamic, uncertain currents. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4778–4783, Anchorage, AK, 2010. IEEE.

[21] G. Villarini, J. A. Smith, M. Lynn Baeck, P. Sturdevant-Rees, and W. F. Krajewski. Radar analyses of extreme rainfall and flooding in urban drainage basins. *Journal of hydrology*, 381(3):266–286, 2010.

[22] G. Wadge, P. Cole, A. Stinton, J.-C. Komorowski, R. Stewart, A. Toombs, and Y. Legendre. Rapid topographic change measured by high-resolution satellite radar at Soufriere Hills

 DOI: 10.2420/AF09.2014.31

Volcano, Montserrat, 2008–2010. *Journal of Volcanology and Geothermal Research*, 199(1):142–152, 2011.

Acta Futura 9 (2014) 41-47
DOI: 10.2420/AF09.2014.41

Acta Futura

Lunar Crater Identification from Machine Learning Perspective

Chak Pong Chung[*1], Cheuk On Chung [2] and Chit Hong Yam [1]
[1] *The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong*
[2] *Sun Yat-sen University, Guangzhou, China*

Abstract. Automatic crater detection from orbital imagery and altitude data has wide applications in planetary science. Many missions, like the Chinese Lunar Exploration Program (CLEP), also known as the Chang'E program, provide valuable imagery and altitude data from the Digital Elevation Model (DEM). Our goal is to recognize the craters on the Moon and related information about their landform using the DEM data and gray scale images provided by the Chinese Chang'E-1 and Chang'E-2 spacecraft. In this paper, we explore four methods to search for craters with DEM data and gray scale crater images, as follows: 1) SVM with feature construction derived from DEM data; 2) A mathematical ratio, based on which we can classify the geographical landform of the moon surface; 3) Image processing techniques to detect edge and corner of the crater structure with gray scale images; and 4) Scale Invariant Feature Transform (SIFT). The density problem of the DEM data from Chang'E spacecraft is also discussed. We expect our method can be applied to detect craters on other celestial bodies such as Mercury, asteroids, and other planetary moons using geographical information from future missions.

1 Introduction

Craters are typical geographical surface feature on the Moon. Extensive studies of the Moon surface keep drawing attention with increasingly accurate information from lunar space missions. Many papers have been published in detecting the lunar crater with images, e.g. Segmentation-based approach by Bue and Stepinski [1] discussed the pros and cons of Hough Transform and Edge-based methods. Morphological construction using ellipse can be found in paper from Kim el al. [7]. Also, feature extraction combined with Support Vector Machine (SVM) approach was discussed by Vinogradova et al. [12]. But the crater detection problem is not completely solved. For example, image-based techniques are limited by sun incidence angles which create edge information. SVM approach is not easy because features of a crater from which SVM can learn is hard to employ and subjective. In view of the difficulties, less restrictive methods from the image and altitude data provided by China's Lunar Exploration Program [13] [9] and assumption of the crater geographical feature is needed to handle the problem for wider applications.

2 Motivation of the Problem

Consider a lunar landing mission scenario. Before a lunar lander attempts to land on the Moon, it can take many photos and transmit them back to control center for analysis. Engineers in the control center can choose a landing site from photos based on the landform and location information. Then the spacecraft will be in-

[*]Corresponding author. E-mail: cpchung@stu.ust.hk

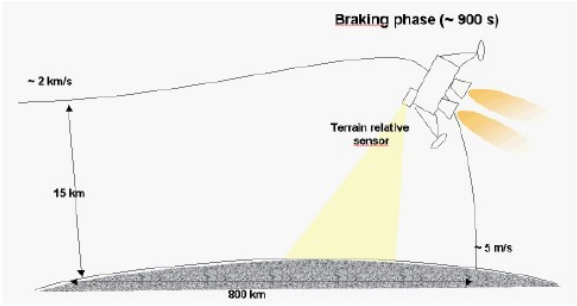


Figure 1: Schematic of a Spacecraft Landing on the Moon [11].

training set	Einstein	Tsiolkovsky
testing set	Tsiolkovsky	Einstein
correctly classified feature vectors	9363	8970
total feature vectors	12093	10530
accuracy	77.43%	85.19%

Figure 2: SVM result summary

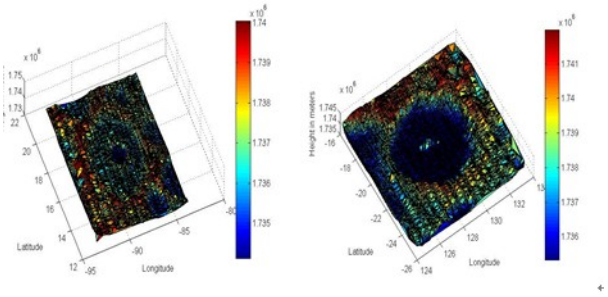


Figure 3: visualization with TRISURF of the corresponding altitude data. Left image is from Einstein and right image is from Tsiokovsky(with central peak)

structed to seek for the proximity of the landing site and then approach to the exact landing location by matching images or using other image processing techniques. This process is illustrated in figure 1

3 Methods

3.1 Support Vector Machine (SVM)

In machine learning, support vector machines [2] are supervised learning models with associated learning algorithms that analyse data and recognize patterns, used for classification and regression analysis. If a hidden and unknown model of data is existed, we can construct feature from the data to discover the model.

The features we try to construct are mean and standard deviation of altitude from Digital Elevation Model (DEM) data. First we choose a specified rectangular region within certain longitude and latitude. Then we partition the data within the rectangular region into square cell with given cell size. For each cell, we calculate the mean and standard deviation of the altitude and then attach these two variables into a feature vector. We will also add one more entry as label. Since we already have ground truth which area is crater and which is not, we manually label it 1 if it belongs to a cell in a crater and 0 if it does not belongs to a cell which belongs to a crater. Then we repeat the same procedure for another rectangular region. But this time we choose to leave the label empty for a region that we in advance know what is its shape from corresponding gray scale images. This complete the data preparation.

Given the data in the format prepared, we can apply SVM to train the model with the first set of data as training set and then test the accuracy of the model trained with the second set of data which is training set. In our experiment we use LIBSVM [5] with default parameters.

Here we present the result of two trials. First we train the model with the crater Einstein and test this model with crater Tsiolkovsky. The accuracy of the model is 77.4% (9363 feature vectors correctly classified out of 12093), as shown in Figure 2. Then we switch the role that we train the model with crater Tsiolkovsky and test the model with crater Einstein. This time the accuracy is 85.2% (8970 feature vectors correctly classified out of 10530), as shown in figure 2). Visualization of the altitude data is shown in figure 3

3.2 The proposed number for estimating landform of specified area

In order to estimate the landform of specified area, we design a number (RVS number) as follows

$$\mathrm{RVS} = \sum \frac{E(h_i)}{H}, i \in I$$

Where RVS represents Relative Volume Scale, H is the maximum relative height of the chosen area, which is the difference between the actual maximum DEM height and minimum DEM height;

$$E(h_i)$$

is the average height of i-th cell of the chosen area. h_i is the relative altitude(difference between the actual DEM

DOI: 10.2420/AF09.2014.41

height and minimum DEM height), I is the index set representing the partitions of the chosen area. So RVS is a ratio between 0 and 1.

We illustrate the use of this number with a simple example. Assume the DEM data is uniformly distributed over the chosen area and also dense enough. Then the h_i could be simply as the relative altitude of each element in i . The number of elements in I is equal to the number of DEM data entries in the chosen area. In our experiment with DEM data, if an area is of bowl shape or mountain peaks, the RVS will be around 0.55. If it is an almost a flat surface with tiny holes and peaks, the number will approach to 1. In case the chosen area is around the central peak of a large crater, the number should approach to 0. Brief derivation of this number comes from the idea of estimating the volume of a terrain.

In practice, uniformly distributed DEM data is almost unlikely, but $E(h_i)$ in the RVS approach still reserve the possibility to handle this situation. If the number of DEM entries within a partition with given area is no less than a lower bound, we can use triangle mesh, which is constructed by randomly distributed data point as triangle vertex ,instead of square cell as basis within the chosen partition, to fully utilize all the DEM data to give us the RVS.

Preliminary results of our own method are shown in the first and the second row. The first row in figure 4 is about two flat surfaces on the Moon and the RVS of them are 0.814 and 0.824, respectively. The second row in figure 4 is craters named Tsikovsky(left) and Mendeleev(right) and the RVS for them are 0.377 and 0.557, respectively, as shown in figure 4.

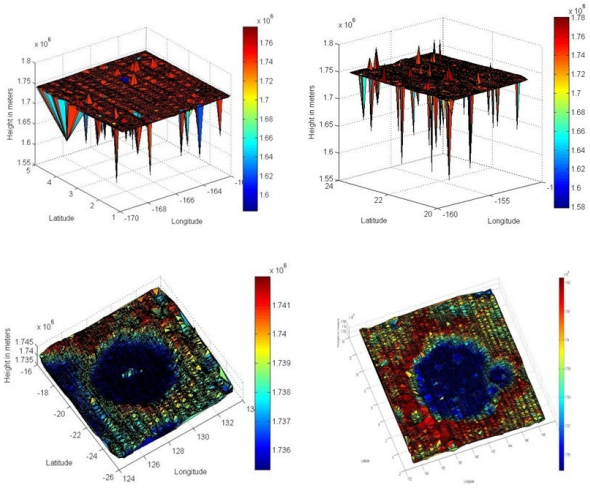


Figure 4: Visualization of the landform. Images in the first row are flat A (left) and flat B (right).Images from second row are Tsiolkovsky (left) and Mendeleev (right).

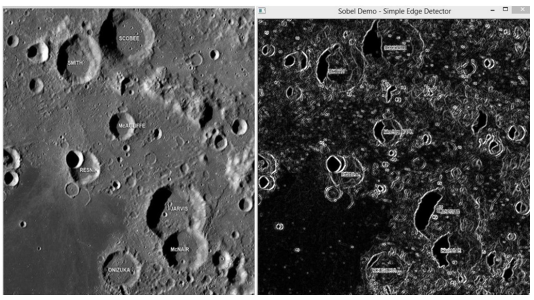


Figure 5: Comparison between raw image (left) and the result from Sobel operator (right)

3.3 Edge and Corner detection

With the help of RVS, we can eliminate a lot of images that are not likely to have craters. After we attain a chosen area with satisfactory RVS value, we can label the shadow around the crater with edge detection techniques.Sobel operator is one simple choice. In case we have many tiny components of noise. Threshold can be set to eliminate part of them with finely tuned threshold parameter. In ideal cases, we can confirm the crater location with images with different sun angles.

To detect the edge, we apply sobel operator [6] and canny edge detection [5]. The result are presented in figure 5, 6 and 7.

Sobel operator

Technically, Sobel operator is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. Below we apply the operator to the crater images as shown in figure 5.

Canny edge detection

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. The result is shown in figure 6.

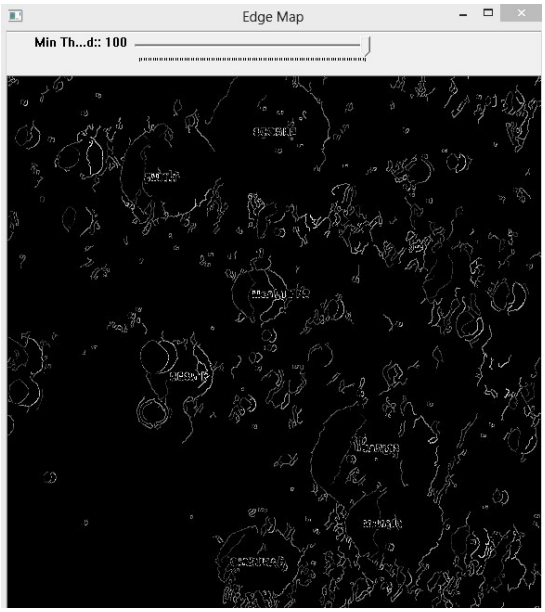


Figure 6: Comparison between the result from canny edge detection

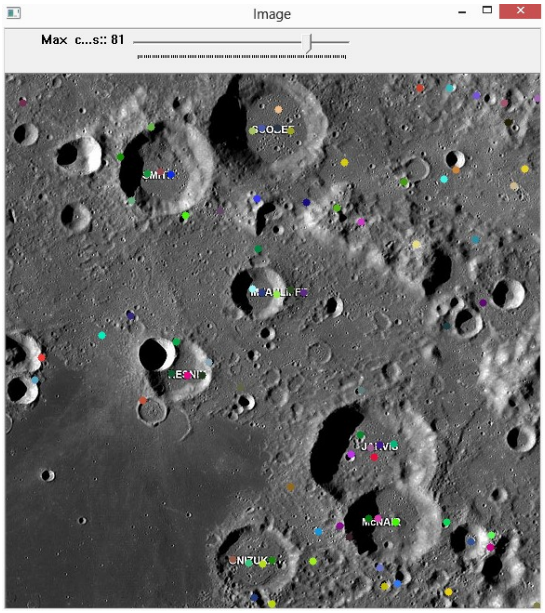


Figure 7: Result from raw image in Fig 7 with Shi-Tomasi corner detector.

Shi-Tomasi corner detector

Shi-Tomasi [10] corner detector and ellipse fitting [4] are also applied in figure 7

Ellipse fitting

One of the most commonly used models to fit craters is the ellipse which, being the perspective projection of the circle, is of great use to find the craters, as shown in figure 8 and figure 9.

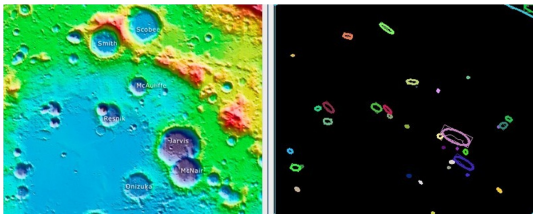


Figure 8: Comparison between the raw images (left) and the result of ellipse fitting (right)

3.4 SIFT

Scale-invariant feature transform (or SIFT)[8] is an algorithm to detect and describe local features in images.

In our experiment we apply two versions of SIFT algorithm. One is without RANdom SAmple Consensus (RANSAC [3]) to eliminate the mismatched key point descriptor. The other is to use SIFT with RANSAC to eliminate the mismatched key point descriptors.

SIFT key point descriptors:

Here we compare the result as shown in figure 10

Crater matching

We also try to use the key point descriptor to match two craters from the same location but under changes in image scale, noise and illumination. The results are shown in figure 13 and 14

4 Comparisons and Implication

In our experiment, image of different resolution have been tested. For corner, edge detection and ellipse algorithm, high resolution (7 meters per pixel) gray scale images from Chang'E-1 is too blur for naked eyes and too smooth to be detected. While low resolution image is too sharp that noises are included. Those noises are formed by tiny craters in low resolution images. They

 DOI: 10.2420/AF09.2014.41

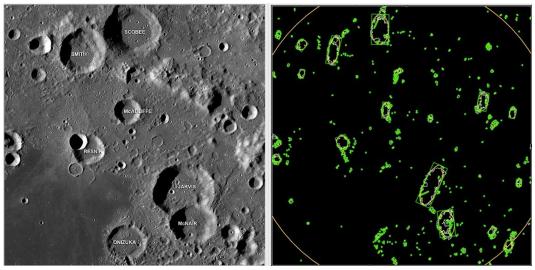


Figure 9: Comparison between the raw images (left) and the of result ellipse fitting with chosen thresholding parameter (right)

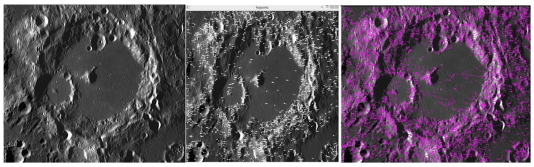


Figure 10: Comparison between SIFT key point descriptor: raw image (left), image applied with SIFT without RANSAC (middle) and the one with SIFT with RANSAC (right)

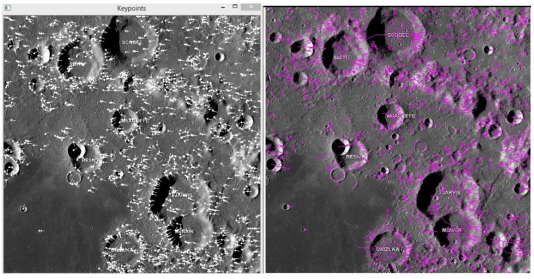


Figure 11: Comparison between SIFT key point descriptor: image applied with SIFT without RANSAC (left) and the one with SIFT with RANSAC (right)

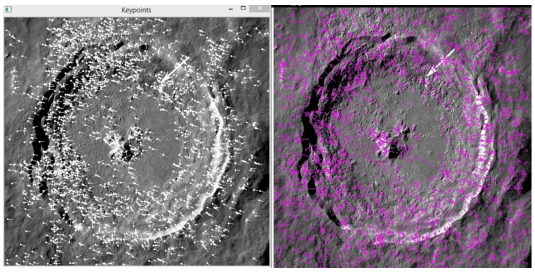


Figure 12: Comparison between SIFT key point descriptor: image applied with SIFT without RANSAC (left) and the one with SIFT with RANSAC (right)

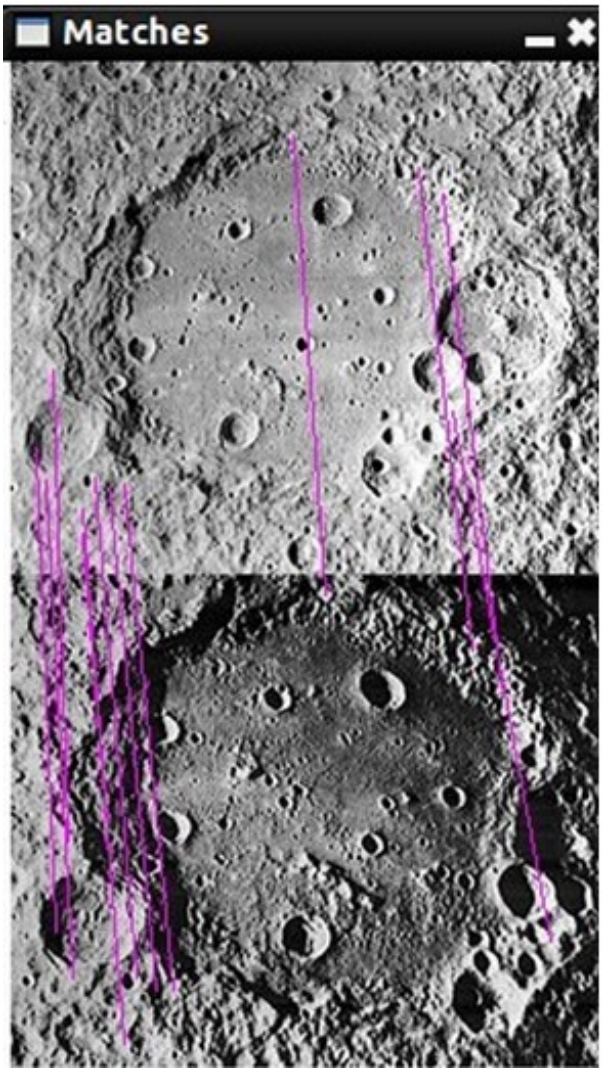


Figure 13: Matching two images for the same crater with different sun incidence angle using SIFT with RANSAC

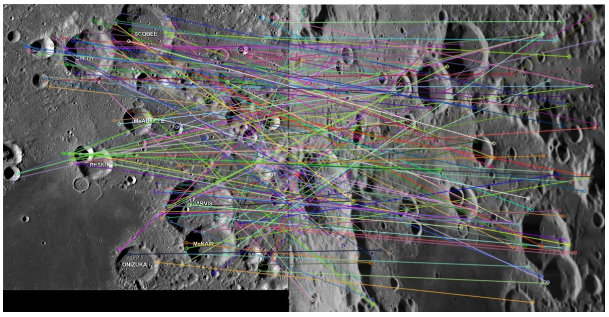


Figure 14: Matching two images for the same crater with different sun incidence angle using SIFT with RANSAC

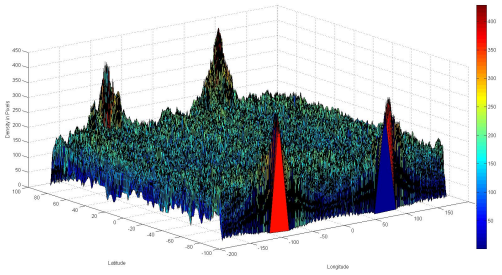


Figure 15: the distribution of the Chang'E DEM data across different location

can easily lower the performance of the edge, corner and ellipse detection algorithm since they are mostly gradient based approach. So preprocessing of images is needed. This explain why SIFT method are relatively effective in matching. But SIFT method can also create undesired key point descriptors. With the help of RANSAC to eliminate the undesired key point, SIFT seems to be the better approach with high probability to match objects. In the spacecraft landing navigation problems, for example, SIFT provides as basis a good starting point to further improve the performance. As all other image-based approach, the choice of image to be applied is also significant. To achieve any specific goal, craters or landform that satisfy certain condition which can be easier recognized by image technique should also be carefully chosen.

For those non-image based approach, density of the DEM data and feature construction from DEM data are the difficult parts. The features we use in our experiment are mean and standard deviation of the altitude. We have surprisingly high accuracy for several chosen craters. If we can choose better feature and discover the hidden pattern of landform, SVM based approach can also be further improved. Figure 15 is the distribution of the DEM data across different location.

From rough estimate for each 30 square km sub cell, the highest data point count is around 330. So roughly every 75 meters, there is one point surveyed by the Chang'E-1 in the DEM data set. To fully utilize the DEM data with given precision, RVS with triangle vertex mesh should be considered. So far the result is promising since landform of different shape have different range of RVS number. With higher density DEM data, we think this method have wider application since so far the only limitation is the density of DEM data.

Acknowledgements

We thank the China's Lunar Exploration Program, National Astronomical Observatories for imagery and DEM data from Chang'E spacecraft. We also thank Prof. Y. C. Zheng, Prof. S. W. Cheng and Prof. K. L. Chan for helpful discussion.

References

[1] B. D. Bue and T. F. Stepinski. Machine detection of martian impact craters from digital topography data. *Geoscience and Remote Sensing, IEEE Transactions on*, 45(1):265–274, 2007.

[2] C. J. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[3] M. A. Fischler and R. 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.

[4] A. Fitzgibbon, M. Pilu, and R. B. Fisher. Direct least square fitting of ellipses. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):476–480, 1999.

[5] B. Green. Canny edge detection tutorial. `http://dasl.mem.drexel.edu/alumni/bGreen/www.pages.drexel.edu/_weg22/can_tut.html`, 2002.

[6] N. Kazakova, M. Margala, and N. G. Durdle. Sobel edge detection processor for a real-time volume rendering system. In *Circuits and Systems, 2004. ISCAS'04. Proceedings of the 2004 International Symposium on*, volume 2, pages 913–916. IEEE, 2004.

[7] J. R. Kim, J. Muller, S. van Gasselt, J. G. Morley, G. Neukum, et al. Automated crater detection, a new tool for Mars cartography and chronology. *Photogrammetric engineering and remote sensing*, 71(10):1205, 2005.

[8] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. IEEE, 1999.

 DOI: 10.2420/AF09.2014.41

[9] Z. Ouyang, C. Li, Y. Zou, H. Zhang, C. Lü, J. Liu, J. Liu, W. Zuo, Y. Su, W. Wen, et al. Primary scientific results of ChangÉ-1 lunar mission. *Science China Earth Sciences*, 53(11):1565–1581, 2010.

[10] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[11] M. Spigai, S. Clerc, and V. Simard-Bilodeau. Crater detection with segmentation-based image processing algorithm. Presentation from Thales Alenia Space, 2010.

[12] T. Vinogradova, M. Burl, and E. Mjolsness. Training of a crater detection algorithm for Mars crater imagery. In *Aerospace Conference Proceedings, 2002. IEEE*, volume 7, pages 7–3201. IEEE, 2002.

[13] Zheng, Yongchun and Ouyang, Ziyuan and Li, Chunlai and Liu, Jianzhong and Zou, Yongliao. China's lunar exploration program: Present and future. *Planetary and Space Science*, 56(7):881–886, 2008.

DOI: 10.2420/AF09.2014.41

Acta Futura 9 (2014) 49-57
DOI: 10.2420/AF09.2014.49

Acta Futura

GPU Accelerated Genetic Algorithm for Low-thrust GEO Transfer Maneuvers

Kai Yu and Ming Xu*
School of Astronautics, Beihang University, Beijing 100191

Abstract. Compared with the classical chemical thrusters, the ion thruster has high specific impulse and is employed to delivery the satellite platform within a wide range. A systematic design on a communication satellite equipped with powerful ion thruster is presented in this paper, which is required to transfer from the geosynchronous transfer orbit (abbr. GTO) to the geosynchronous orbit (abbr. GEO). A high-level bottoms-up approach originating from the value-centric design methodologies (abbr. VCDM) is used to derive the optimal weight and power of the satellite, within some constraints from orbital transfers. A maneuver strategy based upon the feedback of the latitude amplitude is achieved to parameterize the on/off time of the ion thruster. Furthermore, the GPU accelerated genetic algorithm is constructed to optimize all the variables from the closed-loop maneuver controller and systematic design on weight and power. The numerical results show that the ion thruster can reduce the propellant and guide the satellite to its targeting orbit.

Keywords: ion thruster; orbit transfer; GPU accelerated genetic algorithm; value-centric design

1 Introduction

The solar-energy ion propulsion system will be applied in future space missions, and its specific impulse is much higher than the classical chemical systems. Recently, the electric propulsion system (abbr. EPS) can achieve 3-5 kW in power, which has been proposed for some interplanetary missions to raise orbital altitude, like ESA's Cornerstone-class and Solar Orbiter Flexi-class missions. Moreover, the interest in guiding a communication satellite from GTO to GEO is growing by the next-generation EPS, which is expected to be 10-15kW in power [1]. SMART-1 developed by ESA as well, was propelled by a Hall thruster using xenon propellant and cost 82kg Xenon in the end of its mission. The fuel efficiency is 10 times higher than the tradition chemical fuels.

The value-centric design methodology is used by the U.S. Defense Advanced Research Projects Agency on the concept of separation module technology development and demonstration projects. F6 project concept is built around the mission tasks, payload, energy, communication, navigation, and other functional units decomposed into multiple modules, rather than mechanically splitting spacecraft subsystems. During the project, the researchers proposed value-centered spacecraft design methods firstly. By the methods, the spacecraft valuation emphasizes on an input-output ratio and pays attention to get the greatest value with the lowest cost. During the valuation, the concepts of the system's flexibility, fast response, robustness are redefined to cope with risks such as the value of factors and to consider the value of the rich connotation [2].

*Corresponding author. E-mail: xuming@buaa.edu.cn

VCDM bases on life period, technical risk, uncertainty factors to obtain the highest input-output ratio, in order to achieve the minimum cost and obtain the maximum value goal. The high-level bottoms-up approach derives the optimal cost and performance of a satellite by designing the optimal weight and power. For instance, the process of calculating total mass based on individual subsystem mass, where inter-dependencies between subsystems requires an iterative process to converge on a total mass. One of the contributors to the total mass is the propulsion subsystem. The mass of the propulsion system is partly dependent on the size of the propellant tank, which in turn is dependent on the amount of required fuel, or the mass of the spacecraft [9]. By this method, we can design a satellite to complete the complex mission.

Due to the low-thrust, the orbit transfer takes a long time from the parking orbit to the target orbit transfer. The researches are focusing on how to design the control strategy to make the use of the fuel efficiently. Recently, the strategy of the low-thrust transfer orbit has been designed by the open-loop control methods for optimal control and path planning. The low-thrust propulsion is continuous, which is different from the traditional chemical propulsion, so the impulse control cannot be used to design the strategy. The finite thrust model is applied in the problem, in which Hill model and Gauss model are adopted. And Hill model is suitable for the short distance or time maneuver problem. In the Gauss model, there is the singular problem rarely used to calculate. To solve this problem, Walker had given a set of modified equinoctial orbit elements, which are suitable for perturbation analysis of all kinds of orbit. The Legendre pseudo-spectral method had been used to design the orbital transfer with finite thrust [8].

A famous platform is developed by Chinese Academy of Space Technology for small satellites named as CAST968. According to orbit and power requirements, the main parameters of the CAST968 platform are listed as following [10]: the weight about platform ranges from 200 to 300kg, the platform size is fixed as 1.2m $\times$ 1.1m $\times$ 0.5m, the weight of load bears from 100 to 250kg, and the attitude control adopts three-axis stabilized mode.

In this paper, the electric propulsion system is equipped onto this platform to improve a new one for the GTO transfer mission, named as CAST968-Ion. Firstly, an automatic control strategy is designed to reduce the pressure of artificial monitoring during the flight operation. Secondly, all the subsystems are defined in the constraints of mass and power based on the small satellite platform CAST968. Thirdly, calculate the mass and power of the subsystems optimized to reduce the fuel during the mission.

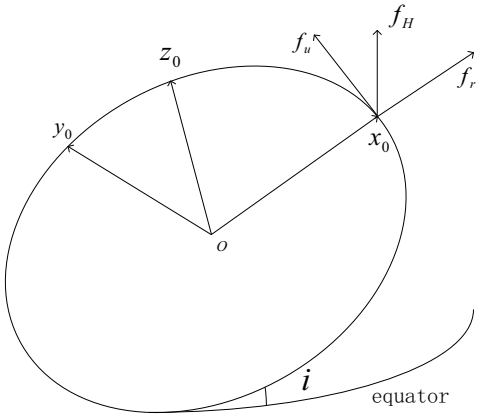


Figure 1: Thrust force decomposed in the orbital coordinate system.

2 Control Strategies

In this chapter, we design a control strategy to find the optimal transfers that minimize transfer time and mass consumed by the spacecraft. This is the key point of low-thrust propulsion, because the chemical rockets are more inefficient and there is an upper limit on the mass of payload that can be delivered to high orbit (although the transfer time is smaller). Therefore the time and mass are two parameters to be taken into account. Compared with a range of transfers with Hohmann mode, the low-thrust propulsion to higher orbit is employing a perturbation method in this paper, i.e., Gauss method. Due to the low orbit transfer mission, we adopt two-body model that neglects the gravities of solar, moon, the earth oblateness and others.

The thrust can be decomposed with three vectors in F_0 frame (i.e., the orbital coordinate system). The Ion-thrust vector f_r is allocated along the radial (positive outwards) direction, the ion-thrust vector f_u is along with track (tangential) direction of the satellite motion, and the ion-thrust vector f_H is normal to the orbital plane in direction to the positive angular momentum (cross track) direction, as shown in Fig.1.

This article uses the no-singular vernal equinox elements to model the satellite dynamics of low-thrust [5, 3]. The attitude angle will be applied to feedback the

 DOI: 10.2420/AF09.2014.49

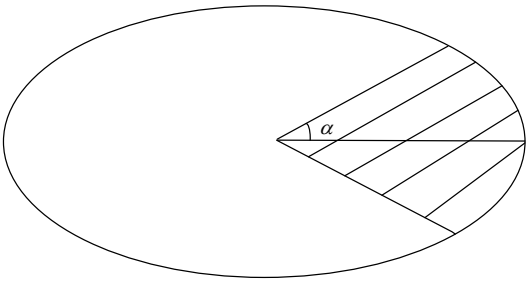


Figure 2: f_u is expected in the shaded area.

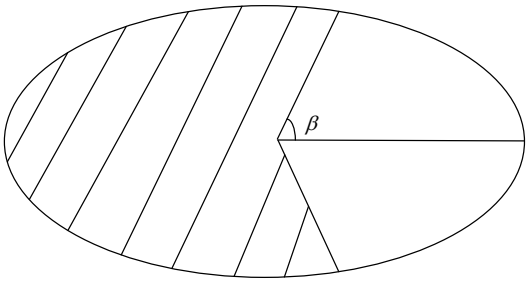


Figure 3: f_H is expected in the shaded area.

thruster in on/off state. In the F_0 frame, the perturbation equations are listed as follows:

$$\frac{da}{dt}=\frac{2a^2}{\sqrt{\mu P}}[ef_r \sin\theta + (1+e\cos\theta)f_u] \tag{1}$$

$$\frac{de_x}{dt}=\sqrt{\frac{P}{\mu}}\left\{\begin{matrix} f_r \sin u + [(1+\frac{r}{P})\cos u \\ +\frac{r}{P}e_x]f_u + f_H \sin u \frac{r}{P} e_y ctgi \end{matrix}\right\} \tag{2}$$

$$\frac{de_y}{dt}=\sqrt{\frac{P}{\mu}}\left\{\begin{matrix} -f_r \sin u + [(1+\frac{r}{P})\cos u \\ +\frac{r}{P}e_y]f_u - f_H \sin u \frac{r}{P} e_x ctgi \end{matrix}\right\} \tag{3}$$

$$\frac{di}{dt} = \frac{r\cos(\omega+\theta)}{\sqrt{\mu P}} f_H \tag{4}$$

where P is the semilatus rectum. From the equation (1), we can derive $\int_0^{2\pi} e\sin\theta f_r = 0$.

Therefore the vector f_r has no contribution to the semi major axis, but the vector f_H does. From the equation (2), we can find the vector f_u makes more efficient contribution to the eccentricity two times than the vector f_r. It is concluded from the above that f_r along the radial direction isn't used to the orbit transfer strategy. From the equation (3), we can find that the vector f_u can change the orbit inclination and does the great influence on inclination at the latitude argument of 0 or 180 degree. Based on the platform, there is only one thruster equipped on the satellite, so f_u and f_H should work in different time.

To sum up, the closed-loop control strategy is presented as follows (as shown in Fig 2 and Fig 3):

$$f = \begin{cases} 0, & u \notin \left[\beta, 360^0-\beta\right] \\ & \text{and } [-\alpha,+\alpha] \\ f_h, & u \in \left[\beta, 360^0-\beta\right] \\ f_u, & u \in [-\alpha,+\alpha] \end{cases} \alpha \leqslant \beta \tag{5}$$

Similar to the attitude control system, the closed-loop control strategy can effectively decrease the errors and then improve the control ability.

3 The Heuristic Design for the subsystems and Their Masses and Powers

The total mass of satellite bases on individual subsystems dependent on each other, which require an iterative algorithm to converge on a total mass. The spacecraft power is dependent on the individual powers and the spacecraft mass. Once power requirements are increasing from the altitude determination and control system (abbr. ADCS), it will lead to increase the solar array and battery size, then to increase the electrical power system(abbr, EPS) mass, and then to increase the total mass, in turn to increase ADCS power requirement. The power iteration stops when the total mass loop converges. The iterative algorithm is illustrated as follows:

Step1: give the mass of the payload, telemetry tracking and command system (abbr. TTC) and command and data housekeeping system (abbr. CDH) respectively, $m_{payload}$, m_{TTC} and m_{CDH}, ; then the initial total mass can be assigned equal to above the three parts mass but the initial power is equal to the power of payload, as:

$$m_{SC} = m_{payload} + m_{TTC} + m_{CDH} \tag{6}$$

$$P_{require} = P_{payload} \tag{7}$$

Step2: according to the CAST968 platform, the power of TTC and CDH is listed as follows:

$$P_{TTC} = 2.29 \times 10^{-2} m_{SC} \tag{8}$$

$$P_{CDH} = 2.14 \times 10^{-2} m_{SC} \tag{9}$$

Step3: the mass and power of structure and mechanism

subsystem can be calculated as follows:

$$m_{structure} = 0.25 \times m_{SC} \quad (10)$$
$$P_{structure} = 0 \quad (11)$$

Step4: the mass and power of thermal control system can be calculated as follows:

$$m_{thermal} = 0.1 \times m_{structure} \quad (12)$$
$$P_{thermal} = 0 \quad (13)$$

Step5: the mass and power of ADCS is refined from the control accuracy. For example, when the accuracy is less 0.1 degree, we can have:

$$m_{ADCS} = 20 + 0.02 \times m_{SC} \quad (14)$$
$$P_{ADCS} = 20W \quad (15)$$

Step6: the propulsion system is determined by the ion engine and fuel. The fuel consumption of orbit transfer cannot be achieved in advance from the delta-V, but from the numerical simulation (I_s is the specific impulse and η is the parameter of engine).

$$m_{thruster} = 96kg \quad (16)$$
$$F_{thruster} = 500mN \quad (17)$$
$$m_{propellent} = f\,(F_{thruster}, m_{SC}) \quad (18)$$
$$P_{pro} = \frac{1}{2} \times F_{thruster} \times I_s \times \eta \quad (19)$$

Step7: electrical power system is made up of five parts, which are listed as following:

$$P_{pro} = \frac{1}{2} \times F_{thruster} \times I_s \times \eta \quad (20)$$
$$m_{array} = P_{BOL}/60, m_{battery} = Cr/40,$$
$$m_{PCU} = 0.02 \times P_{BOL} \quad (21)$$
$$m_{reg} = 0.025 \times P_{BOL},$$
$$m_{wiring} = 0.04 \times m_{total} \quad (22)$$

Step8: summate all the subsystem mass and power above as the initial parameter for the next iteration.

Step9: go to Step (3) until the relative error in the successive iterations is less than 1%.

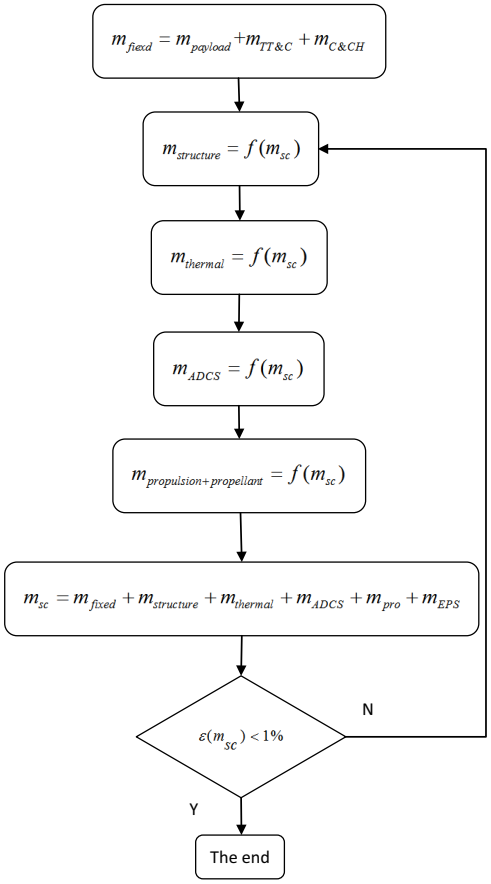


Figure 4: The iteration algorithm for the spacecraft mass.

4 GPU accelerated genetic algorithm

Genetic algorithm (abbr. GA) is a kind of random search technology which simulates the natural selection, species evolvement and population genetics, and also suits to provide a solution to problems such as combination and optimization [6]. Although GA is rather effective in solving many practical problems in science, engineering, and business domains, it needs a long time to find optimal solutions for huge problems, as several fitness evaluations must be performed. Then many improved genetic algorithm models are given to solve complicated problem. Wong proposed to implement an evolutionary computing on consumer-level graphics cards and achieved better speed-up [4]. But in his method, massive data was transferred from GPU to CPU. The reading back of GPU is currently slowed down by the hardware restrictions. Li proposed a fine-grained par-

 DOI: 10.2420/AF09.2014.49

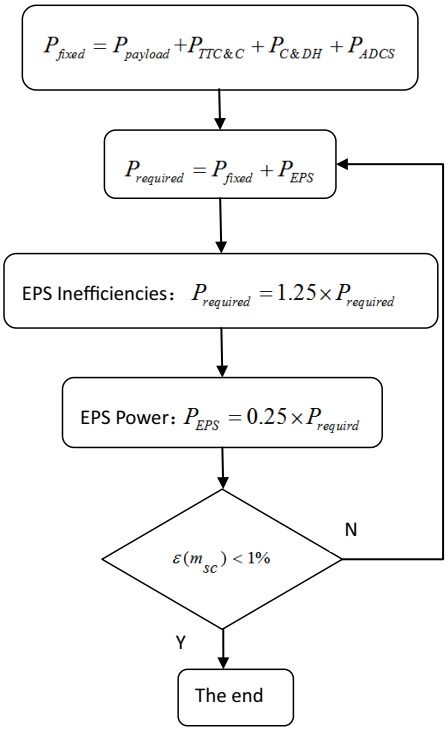


Figure 5: The iteration algorithm for the spacecraft power.

allel genetic algorithm (abbr. FGPGA) which is a model based on GPU-acceleration, which maps parallel GA algorithm to texture-rendering on consumer-level graphics cards [7]. In his method, he presents a new method to implement FGPGA on GPU, which is used to avoid massive data transfer. But, he implement binary encoding scheme of GA and all steps of FGPGA are executed on GPU. Although the above methods have many advantages, for some people, they maybe not understand the hardware of GPU, or they want to use the GPU as simple as possible.

We know that the well-designed CUDA code may be around 20% faster than Jacket and poor CUDA code may perform at only 5% of what well designed code delivers, also the CUDA code costs too much energy of designers. So in this paper, MATLAB and JACKET are running on a GPU compute cluster and the GPU is NVIDIA Tesla C2050.

The workflow of parallel GA on GPU is listed as follows:

Step1: read parameters: crossover rate P_1, mutation rate P_2, individual encoding length L.

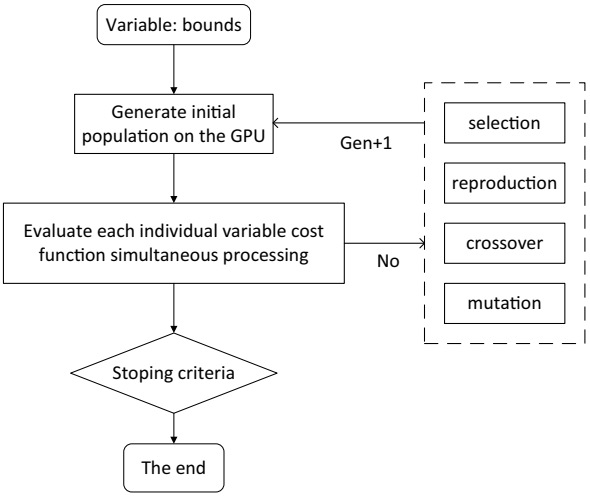


Figure 6: The process of GA on GPU.

Step2: Generate initial population on the GPU and calculate cost fitness function f (x) of each individual.

Step3: search for the fitness individual and the optimal fitness value.

Step4: if the condition is satisfied then output the result and stop the criteria, or else jump to the step 5.

Step5: selection operation: the selection operator is used to ensure that the best fit solutions are chosen to pass on their genes. In this paper, the better fit individuals are chosen to for further genetic operations.

Step6: Crossover operation: this operation insures that the new and unique gene is created.

Step7: mutation operation: When it does many generations, the population may be lose genetic diversity and stagnate at a local minimum. Mutations can help to prevent it. The mutation probability should be small, typically less than 0.05. Once the mutation probability is set too high the genetic algorithm will start to resemble a simple random search. If the user doesn't wish to make use of the mutation operator a probability of 0 can be entered as well.

Step8: Jump to step3 to continue the computation.

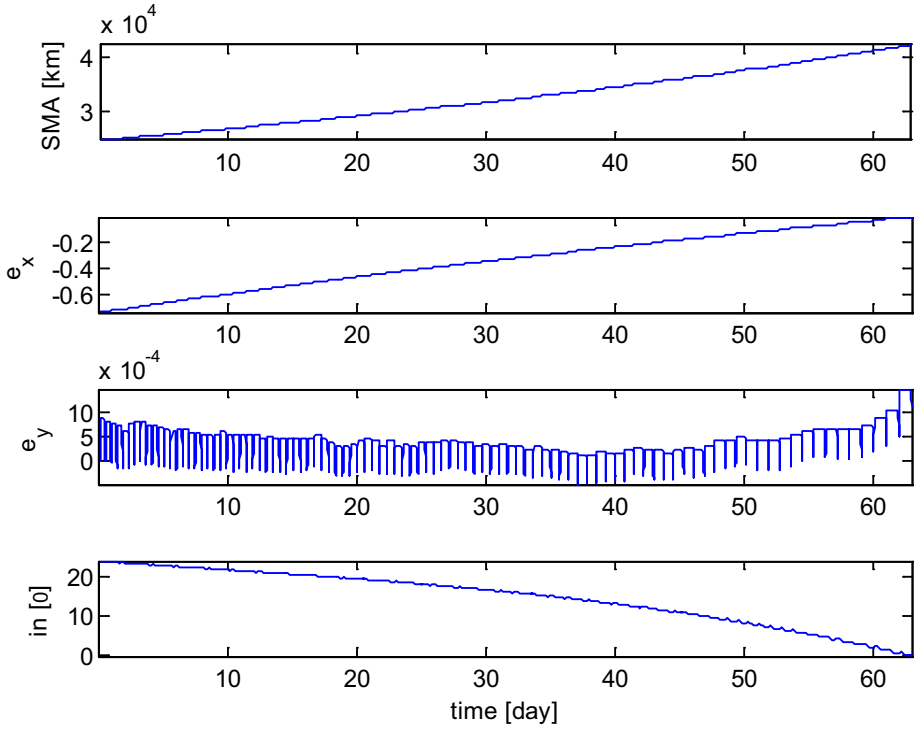


Figure 7: Histories of semi-major axis, eccentricity and inclination.

Table 1: The parameters of GTO and GEO

	GTO	GEO
Argument of perigee	180	
Inclination	24	0
eccentricity	0.731216	0
semi-major axis	24473.653	42166

Table 2: The parameters of Ion-thruster

parameters	Ion-thruster
The power input power P/W	2500
propellant	xenon
impulse $I_{sp}/(m \cdot s^{-1})$	20000
pushing force F/mN	500

5 The simulation calculation and analysis

For the transfer mission, we design the CAST968-Ion based on the platform CAST968. We use an ion engine to replace the original thruster, and the parameters are shown in figure 7. The CAST968-Ion Satellite by LM-3A launch to the orbit of GTO, whose parameters are shown in tab1. The parameters of GTO and GEO orbit are shown in table 2.

Table 3: The power and the mass of CAST968-Ion

Power	(W)	Mass	(kg)
$P_{thermal}$	0	$m_{thermal}$	23.75
P_{TTC}	21.755	m_{TTC}	87
P_{CDH}	20.33	m_{CDH}	
P_{EPS}	347.5	m_{EPS}	39
P_{ADCS}	0	m_{ADCS}	39
P_{pro}	2450	m_{pro}	272.5
$P_{payload}$	896	$m_{payload}$	175
$P_{structure}$	0	$m_{struture}$	327.5
P_0	4490		
m_{sc}			873.75

5.1 Calculate the power and mass

Through the above calculation, we can get the quality and power parameters of each subsystem and total system:

DOI: 10.2420/AF09.2014.49

Table 4: Comparison of GPU and CPU

	Time(s)	α (degree)	β (degree)	J	M	G
CPU	115828	13.4	47.42	0.006	80	100
GPU	795	13.4	47.34	0.006	80	100

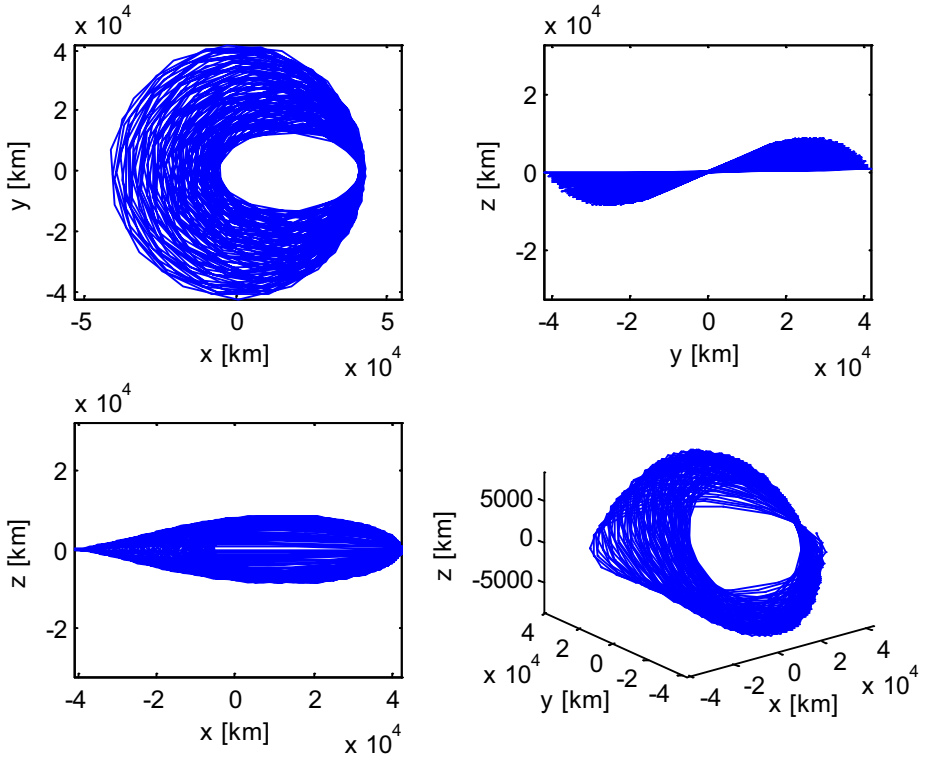


Figure 8: 3-D trajectory of transfer orbit.

5.2 Search the optimal parameter of orbit transfer

From the chapter II, we know that the parameter of α/β is key point of control strategy, which are relative to the mission consumption of fuel and time. We hope that the final inclination and eccentricity are minimums. So we define the fit function as follows:

$$J = f(e_x, e_y, i) = \min_{\alpha,\beta} \sqrt{e_x^2 + e_y^2 + i^2} \begin{cases} \alpha \in [0, 180^0) \\ \beta \in (0, 180^0] \\ \alpha \leqslant \beta \end{cases} \quad (23)$$

We define that the parameter of GA as follows:

Population size of GA: $M = 80$, population algebra of GA: $G = 100$, crossover probability of GA: $P_c = 0.6$, mutation probability of GA: $P_m = 0.01$.

First, we operate the simulation on CPU. Second, we change the code according to JACKET rules and operate on GPU and CPU.

5.3 The optimal results

From the GTO to GEO, the mission consumption of 63-day flight and the finial inclination equal to 0.0661 degree and eccentricity equal to 0.0087.The parameter changing process of the semi-major axis and eccentricity are shown in figure 7 and the trajectory of transfer orbit are shown in figure 8.

5.4 The control strategy for global feature analysis

Genetic algorithm can get the optimal solution of control strategy, but it is unable to present the robustness

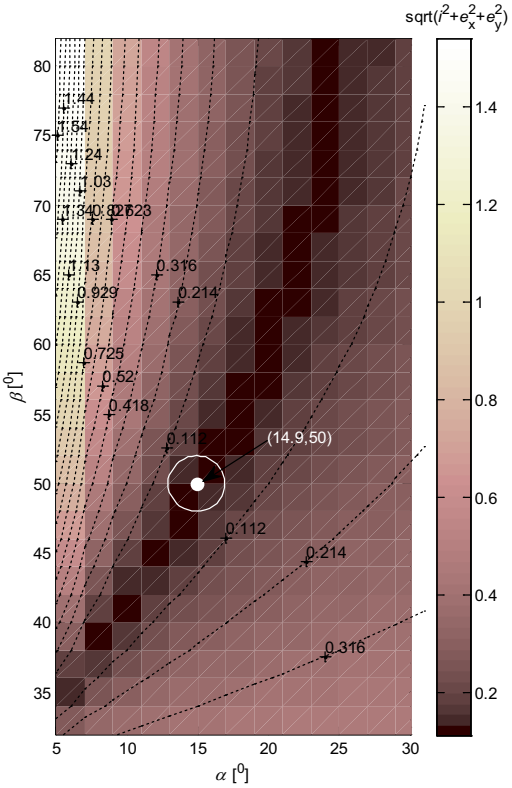


Figure 9: Global features of closed loop control strategy.

of the optimal parameters. The latitude argument is derived from the onboard recursion algorithm, which works as variables to feedback control. As the error in recursive algorithm, it is necessary to implement the error distribution of global features.

With the ergodic investigation on the final target $\sqrt{i^2 + e_x^2 + e_y^2}$ by the different thresholds of α and β, the global features of the closed-loop control strategy is shown in Fig. 9:

The best parameter pair of (α, β) are indicated by the blue point in Fig.9, i.e., (50, 14.9). Due to the error in latitude argument (2^0 is set in this paper), the maximum deviations in inclination and eccentricity are 0.15° and 0.0017 respectively, which are bounded by the blue circle in Fig.9.

6 Conclusions

This paper presented a new genetic algorithm based on GPU. The improved GA is used for the aerospace mission design such as transferring to geosynchronous orbit and is successful in optimizing in only a few minutes. Firstly, an automatic control strategy is designed to reduce the pressure of artificial monitoring during the flight operation. Secondly, all the subsystems are defined in the constraints of mass and power based on the small satellite platform CAST968. Thirdly, calculate the mass and power of the subsystems optimized to reduce the fuel during the mission. The numerical results show that the ion thruster can reduce the propellant and guide the satellite to its targeting orbit.

Acknowledgments

The research is supported by the National Natural Science Foundation of China (11172020), the National High Technology Research and Development Program of China (863 Program: 2012AA120601), Talent Foundation supported by the Fundamental Research Funds for the Central Universities, Aerospace Science and Technology Innovation Foundation of China Aerospace Science Corporation, and Innovation Fund of China Academy of Space Technology.

References

[1] L. Biagioni, A. Passaro, and M. Andrenucci. Particle Simulation of Tailored Vacuum Pumping Configurations for Electric Propulsion Testing. In B. Schürmann, editor, *Fourth International Symposium Environmental Testing for Space Programmes*, volume 467 of *ESA Special Publication*, page 249, 2001.

[2] DARPA. F6 pivot pleiades innovative vcdm optimization tool. [2009-09-21] `http://www.darpa.mil/WorkArea/DownloadAsset.aspx?id=2652`.

[3] P. Enright and B. Conway. Optimal finite thrust spacecraft trajectories using collocation and nonlinear programming. *Journal of Guidance, Control, and Dynamics*, 14(5):981–985, 1991.

[4] K. Fok, T. Wong, and M. Wong. Evolutionary computing on consumer-level graphics hardware. *IEEE Intelligent Systems*, 22(2):69–78, 2005.

[5] Y. Gao and C. Kluever. Low-thrust interplanetary orbit transfers using hybrid trajectory optimization method with multiple shooting. In *Collection of Technical Papers-AIAA/AAS Astrodynam-*

 DOI: 10.2420/AF09.2014.49

ics Specialist Conference, volume 2, pages 726–747, 2004.

[6] D. E. Goldberg et al. *Genetic algorithms in search, optimization, and machine learning*, volume 412. Addison-wesley Reading Menlo Park, 1989.

[7] J.-M. Li, X.-J. Wang, R.-S. He, and Z.-X. Chi. An efficient fine-grained parallel genetic algorithm based on GPU-accelerated. In *Network and parallel computing workshops, 2007. NPC workshops. IFIP international conference on*, pages 855–862. IEEE, 2007.

[8] L. Tu, J. Yuan, and J. Luo. Optimal design of orbit transfer with finite thrust based on legendre pseudospectral method. *Journal Astronautics*, 29, July 2008.

[9] W. Voshell and D. Burms. SVMTool Algorithm Description Document. Technical report, Northrop Grumman Corporation, January 2009.

[10] Y. Zhu. Standardization of satellite platforms. *China Academic Journal Electronic Publishing House*, page S1, 2000.

Acta Futura 9 (2014) 59-72
DOI: 10.2420/AF09.2014.59

Acta Futura

How Ants Can Manage Your Satellites

Claudio Iacopino*[1], Phil Palmer[1], Nicola Policella[2], Alessandro Donati[2] and Andy Brewer[3]

[1] *Surrey Space Centre, University of Surrey, Guildford, GU2 7XH, United Kingdom*

[2] *ESOC, European Space Operation Centre, Robert-Bosch-Strasse 5, 64293 Darmstadt, Germany*

[3] *Surrey Satellite Technology Ltd., Surrey Research Park, Guildford, GU2 7YE, United Kingdom*

Abstract. In the last decade, the interest for space missions involving multiple spacecraft is rapidly growing. A number of areas in the space field are going to benefit from this new trend such as the Earth Observation (EO) field. The Global Monitoring for Environment and Security or the Disaster Monitoring Constellation are first examples of this increasing demand. The trend of multiple platforms is opening new challenges to the automated Mission Planning & Scheduling (MPS) systems as the current Operations concepts designed for individual spacecraft are not necessarily transplantable. The future generation of MPSs aims at gaining maximum value from the constellation of satellites by increasing the efficiency of onboard resources and coordinating the different spacecraft to provide a greater level of responsiveness and adaptability. Our research aims at designing an innovative ground-based automated planning & scheduling system for multiple platforms. The mission used as target for our design is the Disaster Monitoring Constellation. The novelty of this project is in designing an MPS as a self-organizing multi agent architecture, inspired by Ant Colony Optimization algorithms, offering a system adaptable to the problem changes and able to synchronize the satellites' plans in order to avoid duplications among the tasks planned. In this paper, we describe our system and how it can be applied to the planning problem of an EO constellation. Moreover, we present the results of a quantitate test phase aiming at comparing our solution against a standard optimization algorithm such as a genetic algorithm. This comparison is able to show the benefits of our system in terms of adaptability and coordination.

*Corresponding author. E-mail: claudio.iacopino@gmail.com

1 Introduction

The Earth Observation (EO) constellations are offering new challenges to the applications of Automated Mission Planning & Scheduling (MPS) systems as they present critical requirements such as coordination among the spacecraft, efficiency at constellation level and responsiveness in case of disaster management. The solutions developed for single platforms are not necessarily transplantable in this distributed context. A number of studies have recently shown interest for the disaster management, focusing on sensorweb [5] or just on Earth Observation constellations [20, 9, 21, 25]. Most of them faced the problem with classic techniques such as greedy [25, 20], backtracking [9] or simple heuristics. In these cases either they did not achieve efficient solutions either they considered small problems (reduced number of spacecraft). Moreover, a big limitation of these works is not considering the dynamics of the problem itself. This scenario needs to be faced as a dynamic problem, continuously shaped by the user requests, and

where a number of heterogeneous missions need to cooperate among them.

The EO scenario presented above motivates the investigation of the multi agent paradigms in the context of distributed platforms, to model the coordination and control aspects of such missions. The DAFA study [19] is one of the first attempts in this direction, developing a system based on negotiation paradigm and deliberative agents. The main limitation of this approach is in the lack of scalability and adaptability. Given the dynamics and the complexity of the scenario considered, our work looks at reactive approaches as they are highly suited with problems with uncertainty and they naturally lead to self-organizing systems. These types of systems are able to coordinate the actions of their components in a distributed manner, without central authority. Self-organization is therefore a highly desirable system's property if scalability is one of the requirements.

One of the most popular optimization techniques based on self-organization is Ant Colony Optimization (ACO) [7]. The inspiring idea of ACO is that the ants looking for food deposit pheromones along the path. These pheromones influence the following ants to get on the same path. However only the shortest path will end having the strongest pheromone distribution because is the one that requires the minimum travelling time. This is a self-organizing problem-solving strategy. The best path is expected to emerge with the strongest pheromone distribution. Thanks to their engineering benefits, ACO algorithms have been applied to a wide spectrum of problems: travelling salesman problem (TSP), assignment, subset such as the Knapsack problem and scheduling, the closest to the space mission planning problems [17, 13, 10, 4].

Another interesting aspect of the ACO self-organizing technique is its extension into the coordination mechanism of multi agent systems applied to industrial applications, called *Synthetic Ecosystems* [2]. Several works [12, 24, 6] have showed self-organizing manufacturing system based on artificial ants. A similar idea has been used in the on-board coordination system for cluster of satellites developed by Tripp and Palmer [23].

In this paper, we present the results of our investigations, aiming at designing a ground-based automated planning & scheduling system for multiple platforms based on self-organizing multi agent architectures. Such a solution is able to avoid conflicts between the satellites' plans while computing efficient solutions using the ACO communication system. This work is the result of a 3-years project that has seen the development of a novel theoretical model describing the self-organizing properties of an artificial ant colony [14]. Thanks to this model, a novel ant colony optimization algorithm has been designed to maximize the efficiency of each spacecraft [15]. It offers a high-level of adaptability and responsiveness, allowing the system to find good solutions, thanks to the collaboration of all the agents interacting and modifying the decision graphs which form the planning problem. Moreover, the decision graphs of each spacecraft are interconnected, allowing the synchronization among the satellites. This approach combines the optimization capabilities of ACO algorithms, together with the coordination properties of the Synthetic Ecosystem solutions.

The rest of the paper is organized as follows. Section 2 is going to describe the mission used as target of our design, the Disaster Monitoring Constellation operated by Surrey Satellite Technology Ltd. Section 3 presents the system developed. This section describes how we represent the constellation planning problem and how we use this representation to form the environment of a multi agent architecture inspired by ant colonies. Section 4 shows a quantitate analysis of how our system responds to dynamic problems and to multiple platform problems. A standard genetic algorithm is used to offer a performance comparison. Finally, Section 5 discusses the main benefits and drawbacks of our approach.

2 Case study

The mission considered is the Disaster Monitor Constellation (DMC). This platform is the first Earth Observation constellation of low cost small satellites; it provides daily images for a wide range of applications, commercial or of public interest including disaster monitoring. The DMC satellites are designed and built by a UK company, Surrey Satellite Technology Ltd, SSTL. The constellation is currently composed of 6 satellites, flying at about 700 km of altitude, (Beijing-1, NigeriaSat-1, UK-DMC-2, Deimos-1, Nigeriasat-NX, Nigeriasat-2) owned by different entities. DMC works within the International Charter "Space and Major Disasters" to provide free satellite imagery for humanitarian use, in the event of major international disasters. The national civil protection authorities of Algeria, China, Nigeria, Turkey and UK are direct authorized users of the Charter. This constellation faces a number of requests, quite varied in terms of typology and customers. This load ex-

 DOI: 10.2420/AF09.2014.59

ceeds the capabilities of the whole system and it keeps changing in time as new asynchronous targets are requested. This problem is defined as an imaging campaign planning & scheduling problem. In the following the terms *plan* and *solution* are going to be used without distinctions.

The costumers require to image specific targets within certain time windows. Depending on the costumers and on its legal agreements with the service provider, one or more satellites can be considered to image these targets. Given a planning horizon, we can calculate a set of imaging opportunities for each satellite. Finally, we have a set of imaging opportunities for each target but only one acquisition is usually required. More than one acquisition of the same target is considered a duplication, and it reduces the overall constellation's performance. Because of the limited memory on-board, time constraints between imaging opportunities and limited number of ground station passes, it is required to determine a subset of such imaging opportunities which satisfy all the constraints and maximize the performance metrics defined. Given this context, the requirements for the MPS go along three different dimensions:

- **Efficiency**, it needs to produce solutions that maximize the performance. This includes minimizing the duplications among the plans of each spacecraft.
- **Adaptability**, it needs to respond and adjust the solution when changes occur (new user requests, disaster management).
- **Scalability**, it needs to be scalable to the number of user requests and spacecraft considered.

The challenge is to build a system that satisfies all these requirements at the same time. They are often in contrast; a system very adaptable is often not very efficient and vice versa. Moreover, in order to avoid duplications among the tasks planned, a coordination mechanism needs to be part of the planning process. The coordination mechanism is closely related to the system's scalability because the complexity of this mechanism increases as we increase the number of spacecraft or the planning horizon considered. These duplications have an impact on the efficiency of the overall constellation because they represent resources that could be better exploited. It is important to note that two acquisitions of the same targets are not always seen as a duplication. Given a set of imaging opportunity, the operators need to define which among them are considered shared tasks, i.e. duplications if planned at the same time. The operators have a higher knowledge of the system's requirements, such as the revisit time requested by the customers. High priority requests, such as for disaster management, usually need many images in a very short time frame.

Given this context, the MPS needs to focus on the customers' requests. The level of uncertainty is quite low and the communication link is not a critical resource. The system is therefore foreseen to run centrally on the ground segment, abstracting from on-board processing and communication aspects among the satellites. Traditionally, the plan is generated only for a specific uplink opportunity and when all the inputs are available. For this problem, we want a setup offering a higher flexibility for the operators and a higher responsiveness to asynchronous events. We need therefore a system running continuously offering an updated plan at any time. The operators are then called to evaluate a number of equivalent plans during a specific time frame.

The following section introduces the system developed aiming at matching the requirements presented above.

3 Proposed approach

The case study presented above is a clear problem of planning and scheduling (P&S). Automated P&S systems applied to spacecraft operations generally involves generating autonomously sequences of spacecraft commands from a set of higher-level science and engineering goals. The sequences, called plans or schedules, have to comply with the system and the resources' constraints while optimizing the goals expressed as objective function. A common way to represent a P&S problem is the timeline representation where each resource is associated with a timeline. Such a timeline represents the operational schedule for that specific resource. The activities associated to each resource are represented as allocated slots in the relative timelines. Depending on the problem constraints, an activity might be linked with other activities of different timelines.

The MPS we aim to build is a multi agent architecture where ant-like agents are in charge to find solutions to the P&S problem considered. Though the timeline representation is quite intuitive and easy to read, we need to transform the P&S problem in a graph-like environment, which the ant-like agents can explore. Broadly speaking, we aim at implementing an MPS that be-

DOI: 10.2420/AF09.2014.59

haves as an ant colony, continuously exploring the environment, i.e. the P&S problem, and adapting to its changes.

In this section, we first examine the problem representation, how to translate the planning problem in a graph-like environment. We then focus on the logic of the ant colony algorithm that allows a single spacecraft to optimize its plan and to adapt to the environment's changes. Lastly, we present how this paradigm can be extended to offer a self-organizing coordination system for EO constellations.

3.1 Problem representation

Considering a single spacecraft, the problem domain can be modelled as a binary reusable resource, the spacecraft camera, strictly dependent on a depletable resource, the spacecraft on board storage memory. In the timeline representation, a single spacecraft can be modelled with just one timeline. An imaging opportunity is an activity that consumes memory while locking on the camera; we call these activities tasks. The ground station passes allow downloading data; they can be modelled as activities that produce memory. All these activities are on the camera timeline and are subjected to memory availability constraints and temporal constraints. The tasks are characterized by the memory needed and the *quality* which indicates the importance of the specific task; this last parameter is the results of a number of factors such as customer priority, weather forecast, rolling angle and so on. The ground station pass is indicated only with the memory that can be downloaded. That being defined, the problem can be seen as an assignment problem with resources and temporal constraints. It can be formulated as the following:

$$\max \quad Q(\bar{X}) \tag{1}$$

subject to

$$\begin{cases} \sum_{i=1}^{n} r_i x_i \leqslant a \\ \sum_{j=1}^{m} x_j = 1 \end{cases} \tag{2}$$

where $\bar{X}$ is a vector of $x_i \in \{0, 1\}, \quad i = 1 \dots n$, x_i is an assignment variable that indicates if the task *i* has been performed. Equation (1) is a generic objective function that needs to be maximized, taking in account the tasks selected and their relative qualities. This function represents the quality of a plan. Equations (2) express the memory constrains and the possible duplications defined by the operators respectively.

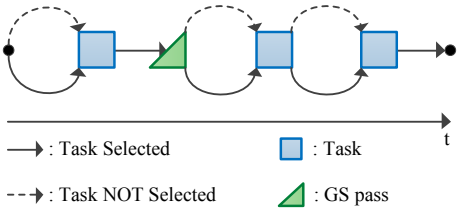


Figure 1: Problem represented by a binary chain.

A common way to translate an assignment problem into a graph is to use a binary representation called binary chain[11, 16, 26, 8]. Given binary variables, the two possible states are represented as distinct edges. The solution to problem therefore is the path connecting these edges. In the following, the terms *path* and *solution* are going to be used without distinctions. Figure. 1 shows the binary representation of the problem where the squares represent the task and the triangle the ground station passes. This binary chain can be seen as a discrete version of a timeline. Each path of the binary chain is a possible timeline solution. This representation is also convenient in the context of dynamic problems. It offers the possibility to express problem's events as minor changes in the graph. A change in the characteristics of one imaging opportunity affects only the parameters of the associated node. The adding or removal of one imaging opportunity affects the graph only locally.

It is important to note that we are not describing the pre-processing phase necessary to go from the mission specific details of each spacecraft, such as the instrument operations or the spacecraft maneuvering model, to the set of imaging opportunities. Such a phase is out of the scope of this paper because we are considering agile satellites with negligible slewing and setup time.

3.2 Ant Colony Optimization Paradigm

The algorithm developed is inspired by the ACO technique; its core workflow can be summarized in the following main steps:

Construction Phase: In this phase the ants construct the path. One ant at a time navigates the environment, i.e. the binary chain. At each time-step, the ant moves from the node where is located to the neighbour one. A transition rule is used to define the probability that the

 DOI: 10.2420/AF09.2014.59

ant chooses to move through the edge ij:

$$\mathcal{P}_{ij} = \frac{\tau_{i0}^{\alpha}}{\tau_{i0}^{\alpha} + \tau_{i1}^{\alpha}} \quad \tau : \text{pheromone variable} \qquad (3)$$

where τ_{i0} and τ_{i1} are variable associated to the edges connected to the node i, two edges in the binary chain. These variables represent the amount of pheromones lying on these edges. α represents the pheromone amplification parameter.

Update Phase: Once the ant reaches the end of the graph, a global pheromone update procedure takes place where the ant deposits on all the edges of its path a pheromone amount $\Delta\tau$. This amount is derived by the value of the objective function $f(\bar{X})$ on the path performed by the ant:

$$\tau_{ij}(t+1) = \begin{cases} (1-\rho)\tau_{ij}(t) + \Delta\tau, \text{ant path} \\ (1-\rho)\tau_{ij}(t), \quad \text{other edges} \end{cases} \qquad (4)$$

The pheromone on all the edges evaporates at the rate $\rho \in (0, 1)$.

The key element of the ACO metaheuristic is the combination of the construction phase with the update phase where the last increases the probability of some edges to be selected for further deposit. Thanks to perturbations and to this autocatalytic pheromone process, the colony can converge in the long-term to a specific path. This is regarded as a global solution.

3.3 Algorithm Implementation

This section introduces a new algorithm developed for the type of dynamic problems presented in this paper. It is important to clarify that we are interested in a system that continuously adapts its current global solution without knowledge on when a change occurs. The algorithm workflow is summarized in *Algorithm* 6: The function `TransitionRule()` is eq. (3). `path` represents the current solution under construction. After each move, the procedure `feasibilityCheck()` is in charge of verifying that the current solution complies with the problem constraints otherwise it activates a repairing procedure to solve the conflicting constraints. Each ant deposits a pheromone amount, `phDep`, on the edges forming the current path. This quantity is function of the `ObjectiveFuntion()`.

The key element of the algorithm is the cycle Exploration/Exploitation. The function `updatePhAmp()`

```
1: PheromoneInitialization();
2: PhAmpInit(), start Exploration Phase;
3: for all ants do
4:    for all nodes do
5:       path+=TransitionRule(); Construction Phase
6:       feasibilityCheck(path);
7:    end for
8:    phDep=ObjectiveFuntion(path);
9:    updateEnv(phDep); Update Phase
10:   if convergence then
11:      savePath();
12:      restartPhAmpCycle(); restart Exploration phase;
13:   else
14:      updatePhAmp();
15:   end if
16: end for
```

Algorithm 6: Algorithm Workflow

modifies the value of α, the pheromone amplification parameter, during the execution. This is a controller parameter and it is responsible to affect the system's dynamics. The current implementation modifies this parameter in order to reduce progressively the ants' exploration while increasing their exploitation. An extended mathematical analysis of the effects of α on the system's dynamics can be found in [14]. After a certain number of ants have navigated and deposited, the pheromone trail reaches levels allowing almost the entire colony to repeat always the same path. This is regarded as a global solution. Once there, the pheromone amplification cycle is reset to the exploration phase. The system is therefore continuously alternating exploration and exploitation phase improving its capabilities of adapting to new changes. It is clear that the system is going to move between equivalent solutions even if no changes occur. This is a desirable characteristic when there is uncertainty in the definition of the objective function.

With this algorithm, we envisage a different operational workflow where the ground segment operators are going to update the problem any time new information is available, from the users, from the spacecraft or from the real environment (weather forecast). The ground segment operators are then called to select the final schedule out of the solutions set produced by our system.

3.4 Coordination mechanism

The previous subsection was able to show the mechanisms behind the high-level of adaptability and effi-

ciency of our system. However a further step is necessary to handle the coordination of multiple spacecraft. The goal is to avoid duplications of the images acquired among the satellites and at the same time to optimize the performance of each spacecraft. Taking inspiration by the synthetic ecosystems seen in Section 1, each spacecraft is associated to an ant colony in charge of navigating a graph representing the planning problem of that spacecraft. These graphs are modelled as binary chains as explained above. The tasks shared among the satellites, representing possible duplications, are modelled as intersections among the satellites' binary chains. Figure 2 explains this representation.

To achieve coordination on the shared tasks, we exploit the pheromone fields generated by the ant colonies. We introduce a coupling similar to the one seen in section 3.3 between the construction and the deposit procedure. In this case, for each shared task, we add a link between the ants' deposit procedure of one colony and the ants' decisional process of the colonies sharing that task. Basically, when the ant of one spacecraft decides to perform a shared task, concurrently with the ants of the others spacecraft, it deposits pheromones on its path and also on the edges of the others binary chains intersecting that task. Specifically, the ant deposits only on the edges that inhibit the others colonies by choosing that task. This simple mechanism guarantees the coordination among the colonies, i.e., among the satellites, in a highly scalable manner. This approach does not have single point of failures; a common limitation of the hierarchical coordination systems. Moreover, this mechanism conserves the same dynamics of the system based on one single chain. This allows the reusing of the same algorithm explained above. We have been able to demonstrate mathematically the validity of this mechanism however this is outside the scope of this paper.

The following section presents a performance analysis used to demonstrate empirically the validity of our approach.

4 Experimental Results

A wide number of tests have been performed to evaluate quantitatively the system's properties. In this section, we analyse two scenarios, the first aims at showing the adaptability property of our system when facing dynamic problems. The second scenario regards constellation platforms, in order to show the system's coordination capability. In both the scenarios, we compare our system against a P&S system based on genetic algorithm. For sake of clarity, in the following sections, we refer to our system as the ant colony system, AC system, while we refer to the one based on genetic algorithm as the GA system.

4.1 P&S System based on Genetic Algorithm

In this paper we want to offer a comparison between our software and some more traditional system. Our main motivation is to show the properties of the AC system and its benefits therefore we propose a comparison with a system that is designed only for optimization of static problem for single spacecraft.

A wide number of techniques are available for optimization problems. We decided to use a standard Genetic Algorithm (GA) [18]. This technique has been used as a comparison to ant colony algorithms applied to binary chains [8]. Moreover a number of recent works applied it to problems of P&S for space [3, 22, 27, 1]. The GA algorithm is based on the concept of evolving a set of random generated solutions, identified as a population of individuals. Every step of the evolution process is called generation. During each generation a new population is created from the mating process among the best individuals of the previous population. The GA is a population-based algorithm as well as the ant colony algorithms. This simplifies some aspects of the comparison. However, it is important to clarify a point. In the GA each individual is a possible solution and generally only the best of a certain generation is considered as a system solution. In the AC system each ant is a possible solution but only the solutions where the entire colony converge to are considered system solutions. To make a fair comparison the following two rules are applied:

$$\mathtt{TotAnts} = \mathtt{Gen} \cdot \mathtt{PopSize} \tag{5}$$

where $\mathtt{TotAnts}$ is the number of ants used in one run by the AC system. $\mathtt{Gen}$ is the number of generations of the GA while $\mathtt{PopSize}$ is number of individuals in the population of the GA. Equation (5) regulates the runtime of both the systems. It takes care of providing the same amount of evaluations of the objective function. As said above, for each ant we evaluate the objective function. The same happens for an individual in each generation.

$$\mathtt{PopSize} = \mathtt{ATP} \Rightarrow \mathtt{Gen.} = \frac{\mathtt{TotAnts}}{\mathtt{ATP}} \tag{6}$$

where $\mathtt{ATP}$ is the period of a pheromone amplification cycle, seen in section 3.3. Equation (6), fixes the GA parameters linking the number of generations directly with

 DOI: 10.2420/AF09.2014.59

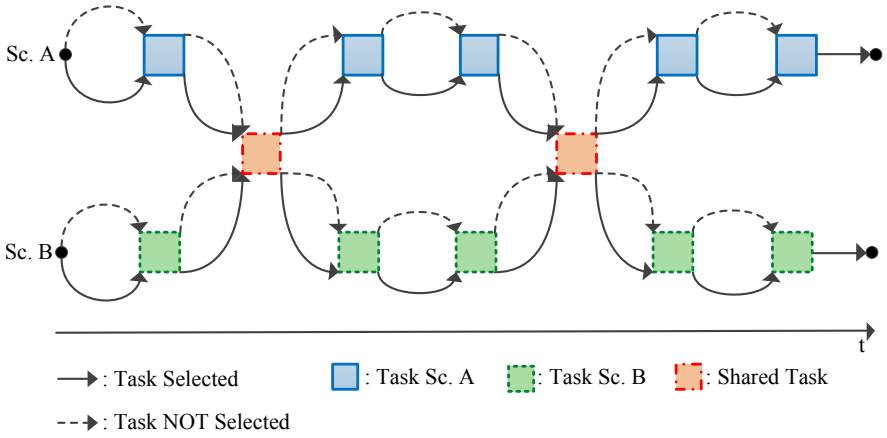


Figure 2: Problem representation in case of multiple spacecraft with shared tasks.

the total number of ants. This means linking the number of generations to the runtime itself. This is the most intuitive approach and it works fine with the dynamic test scenario where we have a variable runtime. The population size is fixed to the ATP. This parameter indicates the time of the cycle exploration/exploitation of the AC system. Thanks to this rule, after each test, we can compare groups of solutions of similar size because the GA generations are equal to the number of time the AC system converges. Table 1 shows the settings of the other parameters characterizing the GA. These values are the best tuning we found for the GA and are in agreements with the literature [8].

4.2 Dynamic Scenario

The scenario considered here regards dynamic problems for single spacecraft. This scenario is useful to understand how the systems' solutions adapt to a changing problem.

Table 2 shows the experimental setup. The top part of the table describes the algorithm setup while the lower part of the table concerns the test dimensions. Each test case is formed by 50 problems of size 20 tasks and, for statistical reasons, we run the system against each of them 50 times. The problems are automatically generated keeping the resource constraints constant. The size 20 tasks is a compromise between real problems and problems that can be solved using complete algorithms to obtain the theoretical optimum. In general, the graph chain's length depends on the number of user requests and on the planning horizon considered. In this case, these numbers represent a typical planning horizon of one day. At this stage, the computational power required is negligible as each run takes few seconds in a desktop pc.

Table 1: GA parameters settings

Crossover Type:	one-point
Crossover Prob.:	0.9
Mutation Prob.:	0.005
Selection Method:	Binary Tournament

In the context of dynamic problems, we are interested in observing how the systems respond to such changes. New events, i.e. changes in the problem are translated to changes in the environment, i.e. in the graph. We consider two typologies of events, which define different type of changes:

- **Weather updates**, the weather information is a key factor for the image requests. Update weather information need to be taken in account to realize an efficient plan. On the graph, this information affects the tasks' *quality*, which translates in the amount of pheromone deposited on the relative path.

- **Disaster management**, new images at high priority can be requested at any time. In this case, the im-

ages are translated to new tasks, which need to be inserted in the graph. It is important to note that the impact on the graph structure is minimal.

These changes are not random but are chosen in order to change the theoretical optimum.

Our evaluation aims at analysing the systems along the following test dimensions:

- **Change Frequency**, it indicates how fast the problem changes. It is given by the number of changes performed on the problem during the run, keeping constant the run time.
- **Change Severity**, it indicates the impact of a change on the problem. It is given by the number of simultaneous changes. For each event, n changes are applied at the same time.

The setup of the experiment sees the systems running for a specific time frame. The time is measured in logical time-steps that are equal to the number of evaluations of the objective function. This means that for the AC system the time-steps are equal to the number of ants while for the GA system the time-steps determine the number of generations by using eq. 6. As explained in Section 3, independent of any change, the AC system continuously searches for new solution and updates the current plan. Given a specific time window, in which the problem can be considered static, the AC system will provide a set of solutions; one set every time the problem changes. Similarly, for the "GA system", we can consider a set of solutions for each time window where the system is static. This set is formed by the best individual of each generation. That being defined, to compare the two systems we are going to compare these sets of solutions. We have defined a number of metrics to properly characterize these sets, however, for sake of brevity, the following charts regard only the quality of the set's best solution. We found this the most meaningful metric because normally we are interested on the best solutions that the system can give us in a specific time window.

Table 2: Setup for the Dynamic Scenario

Problem size:	20 tasks
Problem set:	50 problems
Runs per problem:	50
Run Time:	5000 time-steps
Change Frequency:	[0-9] ch./run
Change Severity:	[1-5] simult. ch.

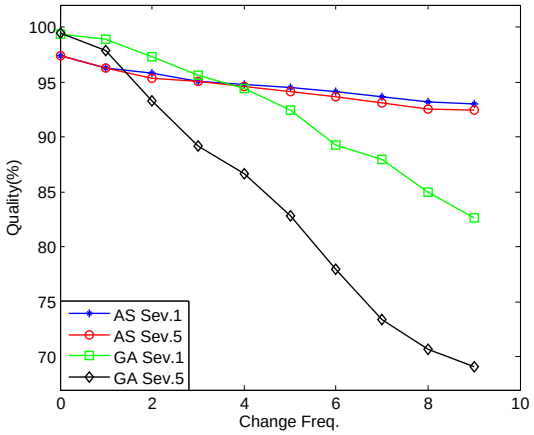


Figure 3: Quality of the set's best solution varying the change frequency and the change severity. The GA system never resets its population during the run.

Figure 3 shows the trend of the quality as the change frequency increases (x-axis). The y-axis expresses the quality in percentage respect the theoretical optimum. In this figure, we plot four different trends: two related to the AC system and two related to the GA system. These trends show how the systems perform when facing problems with different change severities, 1 simultaneous change or 5 simultaneous changes. This is equal to a problem variation of 5% for each change and to a variation of 25% respectively. Table 3 presents the same data in numerical format. "AC Sev.1" is the AC system facing problems with change severy 1 while "GA Sev.1" is the GA system facing problemes with change severity 1.

Starting with the AC system, first of all we observe performance never below the 10% from the theoretical optimum. Moreover, we observe a small decrease in performance, about 4%, as the change frequency increases. This result is expected because the system has less time to explore and find better solutions. Worth noting is that the AC system seems to be very robust with regards to the increasing severity of the problems. Regarding the GA system, in this experiment every time a problem change occurs we use the last GA population as initial conditions for the next generation. Looking at the performance, figure 3 shows slightly better performance of the GA system respect the AC system for low levels of change frequency. This means that the GA sys-

 DOI: 10.2420/AF09.2014.59

Table 3: Quality of the set's best solution varying the change frequency and the change severity. The GA system never resets its population during the run.

Ch. Freq.	AC Sev.1	GA Sev.1	AC Sev.5	GA Sev.5
0	97.42	99.34	97.41	99.40
1	96.30	98.90	96.25	97.88
2	95.77	97.27	95.31	93.25
3	95.01	95.65	95.03	89.17
4	94.72	94.43	94.59	86.60
5	94.53	92.44	94.12	82.86
6	94.11	89.22	93.68	77.98
7	93.61	87.94	93.09	73.39.
8	93.21	84.99	92.51	70.66
9	93.01	82.64	92.47	69.10

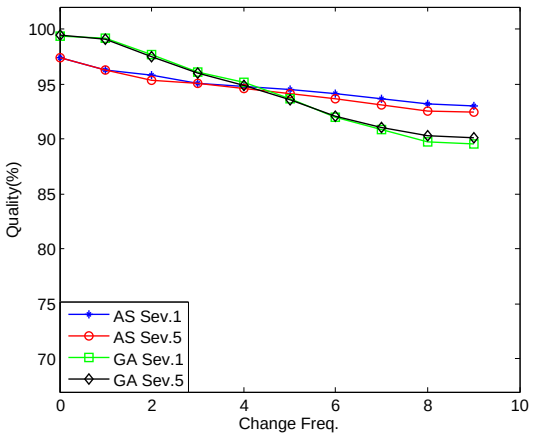


Figure 4: Quality of the set's best solution varying the change frequency and the change severity. The GA system resets its population after every problem change occurs.

tem when facing static problems is able to give solutions in average 2% better than the AC system. However, as soon as the change frequency increases the drop in performance of the GA system is quite remarkable. In case of low change severity the GA system shows a decrease of 15% while in case of high change severity we can see a drop of 28%. This behaviour can be justified considering the level of diversity in the GA population. The GA system does not have the capability to maintain constant the population diversity during the evolution. It is able to converge quickly to good solutions but as soon as the problem changes the GA system is not able to explore efficiently the solution space. This is even more evident for high change severity.

Figure 4 is the results of a similar experiment where we change only the GA system's workflow. In this case every time a problem change occurs we reset the GA population to random initial conditions. Comparing these results with the previous experiment, we can observe a similar behaviour of the GA system for low change frequency. However in case of high change frequency, the GA system presents a decrease in performance of only 8%. This improvement confirms the previous argument. Resetting the population to random initial conditions allows the system to start every time a new exploration and it makes the system robust with regards to variations of the change severity. Table 4 presents the same data in numerical format.

These results show that the GA system is not designed to solve efficiently dynamic problems because it is not able to adapt when the problem changes. For the GA system, as well as for any standard optimization algorithm, it is better to decompose the dynamic problem as a series of static problems and to solve them separately. As said in the previous sections, the purpose of the comparison between the GA system and the AC system is not to demonstrate which one is better but to put in evidence the characteristics of the AC system. With this test, we can conclude that the AC system shows do handle efficiently dynamic problems. Moreover, further tests, not described in this paper, show that the AC system offers always similar solutions. It is important that the current solution does not change completely at every problem change. This desirable property is given by the exploration technique performed by the AC system; the exploration always starts from the previous solution and considers first its neighbours.

4.3 Constellation Scenario

The scenario considered here regards static problems for multiple spacecraft, such as constellations. This scenario is useful to understand how the systems cope with high-complex problems, where coordination between the satellites is needed to provide efficient solutions. As explained in the previous sections, given a set of user requests, a pre-processing phase mission/problem specific defines which image opportunities need to be rep-

Table 4: Quality of the set's best solution varying the change frequency and the change severity. The GA system resets its population after every problem change occurs.

Ch. Freq.	AC Sev.1	GA Sev.1	AC Sev.5	GA Sev.5
0	97.42	99.35	97.41	99.45
1	96.30	99.12	96.25	99.08
2	95.77	97.65	95.31	97.51
3	95.01	96.09	95.03	95.97
4	94.72	95.10	94.59	94.85
5	94.53	93.64	94.12	93.56
6	94.11	91.99	93.68	92.06
7	93.61	90.80	93.09	91.07
8	93.21	89.74	92.51	90.26
9	93.01	89.58	92.47	90.05

resented as shared tasks among the satellites. At this stage, we are not interested in operational details and orbital models of the spacecraft because there are situations when duplications are even wanted. This phase therefore is subjected to the operators' needs. In the rest of the section, we are considering a test case where the operators already defined the shared tasks.

Table 5 shows the experimental setup. Each test case is formed by 50 static problems of 35 tasks and for each problem we perform 50 runs, for statistical reasons. The problems are automatically generated using random task distributions and resource availability. The lower part of Table 5 shows the test dimensions of our analysis. From one side we are interested in how the performance changes increasing the number of spacecraft in the constellation. From the other side we want to see the impact of the level of shared tasks. This is an indicator of the problem difficulty for the coordination system.

As part of the test case, we need to define the performance metrics relevant to the entire constellation:

- **Constellation Quality**, it is given by the sum of the mean of the solution quality of each spacecraft caring of possible duplications between them.
- **Memory Utilization**, the mean of the onboard storage memory utilization of each spacecraft. This is an important metric because in case of duplications a certain amount of data becomes useless causing a decrease of efficiency. We are interested therefore in maximizing it as well.

Table 6 shows the evolution of the constellation quality varying the number of spacecraft in the constellation [2-10] and the number of shared tasks [50%-100%]. "AC Sh50%" refers to the AC system facing problems with a number of shared tasks of 50% while "GA Sh50%" is the GA system facing problemes with a number of shared tasks of 50% . It is clear that the constellation quality grows with the spacecraft number because more tasks are performed. However this positive trend decreases as the amount of shared tasks increases. This happens because the satellites are required to share more. For high level of shared task, such as 100%, the constellation can saturate the problem, not having free tasks to perform. This can be easily observed in Figure 5 that shows only the case of 100% of shared tasks. Depending on the characteristics of the problem therefore, increasing the number of spacecraft might not always provide a benefit.

The GA system shows lower performance respect the AC system. The GA system does not coordinate the solutions among the spacecraft because it solves the planning of each spacecraft as a separate problem. For this test, we cannot use the theoretical optimum as performance reference because the problems are exponentially bigger than in the previous scenario and very difficult to solve in a reasonable time. Thanks to this test we can understand the benefits of having a coordination mechanism such as the one implemented for the AC system. It is important to note that during the entire test the AC system has never proposed any duplicated task. Moreover, the computational power required and the memory used for the computation grow linearly with the number of spacecraft considered, confirming the scalability property of the self-organizing mechanism.

Analogous results can be observed analysing the system's efficiency expressed in terms of onboard storage

Table 5: Setup for the Constellation Scenario

Problem size:	35 tasks
Problem set:	50 problems
Runs per problem:	50
Run Time:	10000 time-steps
Shared Tasks:	[50%-100%]
Spacecraft Number:	[2-10]

 DOI: 10.2420/AF09.2014.59

Table 6: Constellation quality varying the number of spacecraft in the constellation [2-10] and the shared task percentage[50%-100%]

N. SC	AC Sh50%	GA Sh50%	AC Sh70%	GA Sh70%	AC Sh90%	GA Sh90%	AC Sh100%	GA Sh100%
2	50.2	45.6	47.5	41.0	44.8	35.4	42.8	33.3
3	74.2	61.5	65.2	52.4	56.8	41.2	52.6	35.7
4	93.1	76.6	84.0	60.1	69.1	46.1	57.4	37.2
5	117.6	91.1	101.8	71.1	78.1	51.8	59.6	38.3
6	135.2	108.0	116.3	82.8	86.8	55.1	60.7	39.2
7	159.3	122.4	133.4	91.3	95.9	57.1	61.3	39.9
8	178.3	136.0	150.3	99.4	102.2	59.8	61.8	40.5
9	201.1	153.2	168.1	107.2	108.3	62.1	62.2	41.0
10	218.5	170.3	179.9	118.0	114.6	64.4	62.4	41.5

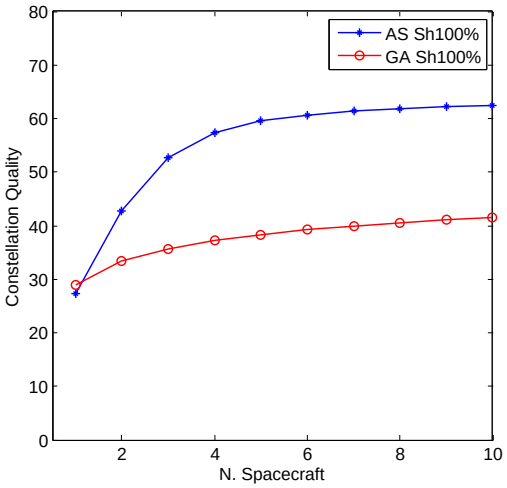


Figure 5: Constellation quality varying the number of spacecraft in the constellation [1-10], shared tasks=100%.

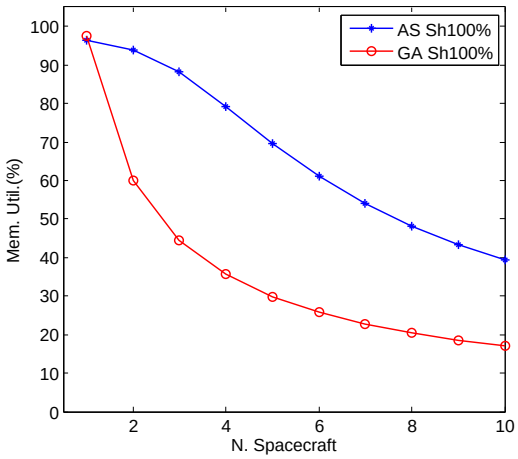


Figure 6: Onboard storage memory utilization varying the number of spacecraft in the constellation [1-10], shared tasks=100%.

memory utilization. Table 7 shows the evolution of the onboard storage memory utilization varying the number of spacecraft and the number of shared tasks. We can observe that the system presents always high-level of memory utilization except when we consider more than 90% of shared tasks. This can be easily observed in Figure 6 that shows only the case of 100% of shared tasks. As explained above, as soon as the satellites do not have more free tasks to perform, their memory utilization decreases drastically. The GA system shows a trend in agreement with the lower constellation quality presented in Table 6.

The analysis presented in this section clearly shows the benefits of the AC system respect the GA system that focuses only on the best solution for a single satellite, not taking advantage of the collaboration among the spacecraft.

DOI: 10.2420/AF09.2014.59

Table 7: Onboard storage memory utilization varying the number of spacecraft in the constellation [2-10] and the shared task percentage[50%-100%]

N. SC	AC Sh50%	GA Sh50%	AC Sh70%	GA Sh70%	AC Sh90%	GA Sh90%	AC Sh100%	GA Sh100%
2	95.4	78.8	94.7	71.6	94.3	63.3	93.9	60.1
3	94.9	70.5	93.1	59.9	89.2	51.8	88.3	44.4
4	94.4	66.6	92.3	52.8	86.4	42.2	79.2	35.6
5	94.2	64.4	90.9	49.9	80.1	38.1	69.7	29.9
6	94.1	62.9	89.8	48.7	76.8	34.4	61.1	25.9
7	93.9	61.1	89.4	46.1	72.1	31.2	53.9	22.8
8	93.7	60.6	89.0	44.6	69.7	28.1	48.1	20.4
9	93.3	59.5	87.9	43.1	65.4	26.7	43.4	18.5
10	92.7	58.6	87.2	42.0	63.9	25.3	39.5	17.0

5 Conclusions

Distributed missions present new challenges for automated systems. They need to be highly responsive, adaptable to face dynamic environments and scalable in terms of number of spacecraft and image requests considered. Today a number of new advanced technologies are available for meeting these requirements such as self-organizing multi agent architectures and natural-inspired collective algorithms. In this paper, we presented a system based on these technologies to face distributed missions' scenarios. In this paragraph, we want to summarize the main benefits and limitations of our approach. The main benefits are:

- **Efficiency**, the system developed exploits the optimization capabilities of the ant colony paradigm, a technique able to achieve high performance in a number of contexts, in particular in assignment and scheduling problems.

- **Adaptability**, self-organizing multi agent architectures are by definition more flexible and adaptable then monolithic systems. The system developed is highly adaptable thanks to the integration of the problem dynamics in the solution construction phase of the ant colony paradigm.

- **Scalability**, classic multi agent architectures suffer of scalability due to the strong responsibility schema. The self-organizing coordination system implemented satisfy this requirement.

Despite these benefits, a number of limitations need to be taken in account:

- **Problem modelling**, the planning & scheduling problem need to be translated to a graph-like structure. This formalism cannot represent all the types of constraints but the agents' logic can incorporate most of them. A pre-processing phase mission/problem specific is necessary for this translation.

- **Black box**, all the *soft-computing* techniques such as neural networks, genetic algorithms and ant colony algorithms offer solutions without providing the relative reasoning chain. This could be a critical issue for the human operators. However, increasing the autonomy in the ground segment, the operators need to move to a supervision level and our system represents a powerful tool for this task.

- **Cost & Operational feasibility**, we are proposing a different operational workflow. As explained in Section 3, our system generates continuously plans of equivalent quality which the operators are called to evaluate. Such a system offers more flexibility but it requires a different manpower management.

- **Stochastic nature**, a further challenge is the mindset shift. The issue is to shift from deterministic to stochastic systems. This is a mandatory step if we want to build more complex and adaptable systems.

 DOI: 10.2420/AF09.2014.59

Acknowledgment

This work is co-funded by the Surrey Space Centre (SSC) of the University of Surrey, the Surrey Satellite Technology Ltd (SSTL) and the Operations Centre of the European Space Agency (ESOC/ESA).

References

[1] J. C. Al Globus, J. Lohn, and A. Pryor. Scheduling earth observing satellites with evolutionary algorithms. In *Conference on Space Mission Challenges for Information Technology (SMC-IT)*, 2003.

[2] S. Brueckner. *Return from the ant. synthetic ecosystems for manufacturing control*. PhD thesis, Humboldt-Universitat, Berlin, 2000.

[3] A. R. Carrel and P. L. Palmer. An evolutionary algorithm for near-optimal autonomous resource management. In *'i-SAIRAS 2005'-The 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, volume 603, page 25, 2005.

[4] W. Chen, J. Zhang, H. Chung, R. Huang, and O. Liu. Optimizing discounted cash flows in project scheduling - an ant colony optimization approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(1):64–77, 2010.

[5] S. A. Chien, J. Doubleday, D. McLaren, D. Tran, V. Tanpipat, R. Chitradon, S. Boonya-aroonnet, P. Thanapakpawin, C. Khunboa, W. Leelapatra, V. Plermkamon, C. Raghavendra, and D. J. Mandl. Combining space-based and in-situ measurements to track flooding in thailand. In *IGARSS'11*, pages 3935–3938, 2011.

[6] T. De Wolf and T. Holvoet. Designing Self-Organising emergent systems based on information flows and feedback-loops. In *Self-Adaptive and Self-Organizing Systems, 2007. SASO'07. First International Conference on*, pages 295–298, 2007.

[7] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26(1):29–41, 1996.

[8] C. Fernandes, V. Ramos, and A. C. Rosa. Stigmergic optimization in dynamic binary landscapes. In *Proceedings of the 2007 ACM symposium on Applied computing*, pages 747–748. ACM, NY, USA, 2007.

[9] R. Grasset-Bourdel, G. Verfaillie, and A. Flipo. Building a really executable plan for a constellation of agile earth observation satellites. In *IWPSS - International Workshop on Planning & Scheduling for Space*, ESOC, Darmstadt, 2011.

[10] M. Gravel, W. L. Price, and C. Gagne. Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristic. *European Journal of Operational Research*, 143(1):218–229, 2002.

[11] W. J. Gutjahr. On the finite-time dynamics of ant colony optimization. *Methodology and Computing in Applied Probability*, 8:105–133, 2006.

[12] K. Hadeli, P. Valckenaers, C. Zamfirescu, H. Brussel, B. Germain, T. Hoelvoet, and E. Steegmans. Self-organising in multi-agent coordination and control using stigmergy. In G. Marzo Serugendo, A. Karageorgos, O. Rana, and F. Zambonelli, editors, *Engineering Self-Organising Systems*, volume 2977 of *Lecture Notes in Computer Science*, pages 105–123. Springer Berlin Heidelberg, 2004.

[13] S. J. Huang. Enhancement of hydroelectric generation scheduling using ant colony system based optimization approaches. *Energy Conversion, IEEE Transactions on*, 16(3):296–301, 2001.

[14] C. Iacopino and P. Palmer. The dynamics of ant colony optimization algorithms applied to binary chains. *Swarm Intelligence*, 6(4):343–377, 2012.

[15] C. Iacopino, P. Palmer, A. Brewer, N. Policella, and A. Donati. A novel ACO algorithm for dynamic binary chains based on changes in the system's stability. In *2013 IEEE Swarm Intelligence Symposium (SIS-13)*. IEEE Press, Piscataway, NJ, 2013.

[16] M. Kong and P. Tian. A binary ant colony optimization for the unconstrained function optimization problem. In Y. Hao, J. Liu, Y. Wang, Y.-m. Cheung, H. Yin, L. Jiao, J. Ma, and Y.-C.

Jiao, editors, *Computational Intelligence and Security*, volume 3801 of *Lecture Notes in Computer Science*, pages 682–687. Springer Berlin Heidelberg, 2005.

[17] D. Merkle, M. Middendorf, and H. Schmeck. Ant colony optimization for Resource-Constrained project scheduling. *IEEE Transactions on Evolutionary Computation*, 6:893–900, 2000.

[18] M. Mitchell. *An introduction to genetic algorithms*. MIT Press, Cambridge, Mass., 1998.

[19] J. Ocon, E. Rivero, A. Sanchez Montero, A. Cesta, and R. Rasconi. Multi-agent frameworks for space applications. In *SpaceOps 2010*, Huntsville, Alabama, 2008.

[20] C. Pralet, G. Verfaillie, and X. Olive. Planning for an ocean global surveillance mission. In *IWPSS - International Workshop on Planning and Scheduling for Space*, ESOC, Darmstadt, 2011.

[21] D. A. Raghava Murthy, V. Kesava Raju, M. Srikanth, and T. Ramanujappa. Small satellite constellation planning for disaster management. In *International Astronautical Federation*, Prague, Czech Republic, 2010.

[22] H. Tripp and P. Palmer. Distribution replacement for improved genetic algorithm performance on a dynamic spacecraft autonomy problem. *Engineering Optimization*, 42(5):403–430, 2010.

[23] H. Tripp and P. Palmer. Stigmergy based behavioural coordination for satellite clusters. *Acta Astronautica*, 66(7-8):1052–1071, 2010.

[24] P. Valckenaers, M. Kollingbaum, and H. Van Brussel. Multi-agent coordination and control using stigmergy. *Computers in Industry*, 53(1):75–96, 2004.

[25] P. Wang and Y. Tan. Joint scheduling of heterogeneous earth observing satellites for different stakeholders. In *SpaceOps 2008*, Heidelberg, Germany, 2008.

[26] K. Wei, H. Tuo, and Z. Jing. Improving binary ant colony optimization by adaptive pheromone and commutative solution update. In *2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)*, pages 565–569. IEEE, 2010.

[27] W. J. Wolfe and S. E. Sorensen. Three scheduling algorithms applied to the earth observing systems domain. *Management Science*, 46:148–166, 2000. ACM ID: 970349.

 DOI: 10.2420/AF09.2014.59

Acta Futura 9 (2014) 73-82
DOI: 10.2420/AF09.2014.73

Acta Futura

Pattern-Based Modeling for Timeline Planning in Space Domains

Simone Fratini*, Nicola Policella and Alessandro Donati
European Space Agency, ESA/ESOC, Robert-Bosch-Str. 5, Darmstadt, Germany

Abstract. This paper describes the design principles of K2E (Knowledge to Ease), a new Knowledge Engineering Environment (KEE) based on modeling patterns currently under development at the European Space Agency (ESA). This environment aims at assisting the operational engineers in designing, maintaining, and updating the models during the whole life cycle of current and future AI Planning and Scheduling systems. The work is motivated by the distance between primitives usually available for modeling domains and problems in AI planning systems and the real needs of the final users of these technologies, the operational engineers. Despite the efforts devoted to the adoption of expressive modeling languages, this distance is still quite big, leading to the need of modeling experts to assist continuously the user during the whole life-cycle of the application. Our goal is to introduce a KEE to bridge the gap between the planning languages currently used by AI technologies, based on constraints, resources and timed automata, and the current practice in the operational context, based on procedures, imperative primitives, and temporal synchronizations among processes.

Keywords: Timeline-Based P&S, Knowledge Engineering for P&S, Languages for P&S.

*Corresponding author. E-mail address: simone.fratini@esa.int

1 Introduction

Space has been often a fertile field for the introduction of novel AI based planning and scheduling technologies. In fact, the AI model-based approach allows reusing of software modules across different missions because of the great flexibility introduced by the symbolic representation of goals, constraints, logic, parameters to be optimized and so on. The great advantage of model-based, domain independent planning technologies is the possibility of using a software not designed to achieve (possibly parameterized) goals in a given domain but for manipulating symbolic entities. This makes the software deployment and test substantially independent from the specific mission. As a drawback, when proper symbolic constructs are not available for modeling, a great effort and amount of time are in general necessary to understand domains and problems, to capture all the characteristics, and to eventually create the model. Another issue is the performance of domain independent planners. Infact the performances of P&S systems, at a large extent, depend on how problems and domains are formulated. These systems often show poor performances iwhen compared with ad-hoc, domain specific applications. The mix of modeling difficulties and performance issues constitutes a not trivial barrier for the practical adoption of AI model-based P&S technologies, seriously harming the great advantage that, in theory, the approach could bring.

To cope with modeling issues, the cognitive distance between the modeling primitives of the AI system and the objects to be modeled has to be as small as possible. For this reason the *timeline based paradigm* has proved to be particularly suitable for space applications, being very close to the way problems and constraints are naturally represented in space applications. In timeline based planning the analogy with problems commonly handled in space applications is more obvious than for other AI planning paradigms (like PDDL [14] for instance). However, the distance between primitives usually available for modeling and the real needs of potential users of these technologies is still quite big, leading to the need of modeling experts to assist the user during the whole life-cycle of the application.

This paper presents the initial results of a research activity currently ongoing at ESA-ESOC.The approach is based on *modeling patterns* derived from languages for defining procedures. Instead of modeling the problem in terms of generic timed automata, temporal synchronizations and constraints among timelines, the user composes the model by means of a set of pre-defined automata (the patterns) and connect them with a reduced set of temporal and logical primitives. The patterns allow modeling concepts like tasks, procedures, imperative rules, cycles, conditional branches and resource allocations. The output of the process is a specification in timeline-based planning language that can be used to feed a domain independent timeline planner.

2 Languages for AI planning in space

The principles behind AI planning and scheduling in space applications inherit from control theory. In fact a common background among all the AI planning and scheduling systems currently in use in the major space agencies is an underlying assumption that the world is modeled as a set of entities whose properties can vary in time (such as one or more physical subsystems) according with some internal logic or as a consequence of external inputs. The intrinsic property of these entities, represented by means of *timelines*, is that they evolve over time concurrently, and that their behaviors can be affected by external inputs. The analogy with control theory can be extended to conceive the problem solving with timelines as a problem of controlling components with external inputs in order to achieve a desired behavior. Hence different types of problem (e.g., planning, scheduling, execution or more specific tasks) can be modeled by identifying a set of inputs and relations among them that, once applied to the components modeling a domain with a given initial set of possible temporal evolutions, will lead to a set of final behaviors which satisfy the requested properties (for instance, feasible sequences of states, or feasible resource consumption, as well as more complex properties[1]).

The analogy with problems commonly handled in space applications is obvious. In fact there are already software and platforms in use at NASA and ESA (Europa[13], Aspen[12], Apsi[16], Goac[9]) based on timelines. Unfortunately these systems do not use a standardized language to model problems and domains, with a consequent objective difficulty in spreading and re-using information, models and software solutions. Nevertheless these platforms provide a set of similar services to implement planning and scheduling algorithms as well as complete end-to-end applications [11].

Moreover, the languages currently in use for defining planning and scheduling problems (not only for space applications) are based on different assumptions. Some languages are based on the notion of action and state, like the Planning Domain Definition Language (PDDL [14]), others are based on the notion of time intervals and values, like the Europa's New Domain Definition Language (NDDL[5]) or the Apsi's Domain Definition Language (DDL[19]), some are based on the notion of activities decomposition and resource usage, like the Aspen Modeling Language (AML[24]).

These languages are actually in use, i.e., there are solvers able to find plan/schedules for domains and problems represented in these languages. This is due to the fact that these languages have been either deployed together with (or on purpose for) the related software platforms (as in the case of NDDL/Europa, AML/Aspen and DDL/Apsi) or they have been designed targeting a specific community of experts to entail planners comparison and compatibility (as in the case of PDDL). No or little attention has been devoted to the usability or suitability of these languages out of the specific community/environment for which they have been originally designed. In other words, these languages have been designed having in mind the planner and the planner experts more than the modeler and the domain experts, resulting in the need of an expert of the language (and often an expert of the planner to be

[1] A detailed description of the timeline based approach, state of the art of the technologies in use and basic concepts like timeline, state variable and resource is out of the scope of this paper. More information can be found for example in [22, 15, 19] among the others.

 DOI: 10.2420/AF09.2014.73

used as well) to practically use them.

Recently the interest in the usability of AI planning and scheduling tools is growing up and the research in the areas lying between planning & scheduling technology on the one side, and practical applications and problems on the other is gaining more and more interest. In particular the focus is on the area covering the acquisition, formalization, design, validation and maintenance of domain models, and the selection and optimization of appropriate machineries to work on them.

Recent works aimed at putting together the most useful features (for applicative domains) of the different proposals, with the ambitious goal of defining languages able to represent all the different aspects of domain and problems. The Action Notation Modeling Language (ANML[4]) is an example of such an effort. ANML derives features from PDDL , NDDL and AML to represent actions, conditions and effects, rich temporal constraints, activities, resource usages and HTN decompositions. This is certainly a prominent direction to bridge the gap between the problem and the technology, but does not solve the core problem yet: the need of an expert of the language to model the problem.

On the other side, an alternative solution could be to use the languages designed for domain experts and operators (see for instance the Spacecraft Command Language SCL[8], the Procedure Representation Language PRL[20] and the Procedure Language for Users in Test and Operations, PLUTO[1]). This approach simplifies the problem from the user's perspective, and in fact thre are both translations of procedural languages into planning languages (see for instance [7]) as well as attempts of direct using procedural languages in mission planning systems (see [6] for instance). The main problem, at the current stage of deployment of AI solvers, is that it is very difficult to use directly languages not based on strong theoretical assumptions like (temporal) logic or constraint satisfaction problems. In fact the use of these languages for AI planning poses problems with the underlying semantic of the procedures[23].

A possible different approach is to investigate how to translate typical modeling primitives used in the operational context into fragments of domains represented in languages used for timeline based planning. The advantage is to make available these primitives without giving away all the advantages of the languages close to the planning technology. In other words, we are not proposing a translation of a language into another one, but a conceptual mapping of some primitives not usually available in languages for planning and scheduling into

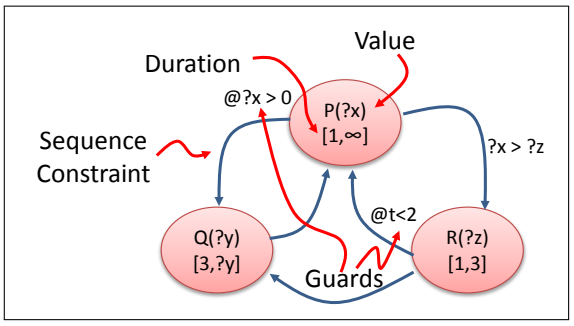


Figure 1: State Variable

fragments of domains specified in a language for planning and scheduling. This of course does not solve the modeling problem *per se*, in fact there is no direct translation of a language into another one, but it can provide modeling primitives which are semantically closer to user's knowledge.

In the following section we first focus on timeline-based planning briefly discussing the modeling primitives they provide, then we introduce the concepts behind the K2E knowledge engineering environment under deployment.

3 Modeling P&S Problems with Timelines

State-of-the-art timeline-based planning languages are based on a few modeling primitives: state variables, resources, temporal/value synchronizations among different timelines and hierarchical decomposition primitives.

State variables represent components that can take sequences of symbolic states subject to various (possibly temporal) transition constraints. This primitive permits the definition of *timed automata* as the one represented in Figure 1; here the automaton represents the constraints that specify the logical and temporal allowed transitions of a timeline. A timeline for a state variable is valid if it represents a *timed word* accepted by the automaton.

The automaton models: (1) the values that the timeline can take, possibly as function of numeric or enumerated parameters; (2) transition constraints on these values, possibly with additional constraints that restrict the transition to a subset of the possible values that the parameters can take (in the example in Figure 1 the transition from `P(?x)` to `R(?z)` imposes that `?x>?z`); (3) temporal constraints that state the minimal and maximal temporal duration of a value and (4) guards that re-

strict the applicability of a transition, either based on the value of a parameter (in the example the transition from `P(?x)` to `Q(?y)` is allowed only if `?x>0`) or on the relative timing of the transition (in the example the transition from `R(?z)` to `P(?x)` is allowed only if `R(?z)` has been maintained for less than 2 time units).

The timed automaton (i.e., state variable) is a very powerful modeling primitive, widely studied both at theoretical level (see [3] for instance) and for which exist implemented algorithms to find valid timelines. In fact all the planning architectures mentioned above are able to calculate valid timelines for timed automata, with some restrictions on the type of values and constraints specified for the parameters, and under certain assumptions on the number of states and structure of the transitions.

A resource is any physical or virtual entity of limited availability, such that its timeline (or profile) represents its availability over time, a decision represents a quantitative use/production of the resource over a time interval. A *reusable* resource (as in [10]) abstracts any real subsystem with a limited capacity c_{max}, an *activity* uses a quantity of resource during the limited interval. For example, an electric generator has a maximal available power P_{max} (its capacity). An activity uses power during an interval of time and as soon as the activity ends, the amount of resource can be reused by other activities. A set of activities are feasible when for each time t the aggregate demand $p(t)$ (or profile) is below or equal to the resource capacity c_{max}. A *consumable* resource (as in [21]) abstracts any subsystem with a minimum capacity c_{min} and a maximum capacity c_{max}, *consumptions* and *productions* consume and restore a quantity of the resource in specific time instants. For example, a battery has a minimum amount of charge that has to be guaranteed (can be more than 0 for operational or security reasons) and a maximum capacity. Operations can either consume (e.g., by using payloads) or recharge (e.g., by using solar arrays) the battery. A set of productions and consumptions are feasible when for each time t the aggregate use $u(t)$ (or profile) is in between the resource minimum and maximum capacity (i.e., $c_{min} \leqslant u(t) \leqslant c_{max}$). All the architectures mentioned above have the capability of representing and solving scheduling problem with resources. This is a crucial capability for modeling and solving planning and scheduling problems in space, since the need of representing and reasoning efficiently on time and resources is a must in this field.

In timeline-based modeling the physical and technical constraints that influence the interaction of the sub-systems (modeled either as state variables or resources) are represented by means of temporal and logical synchronizations among the values taken by the automata and/or resource allocations on the timelines. Languages for timeline-based planning have constructs (as the synchronization in DDL or the compatibility in NDDL, see [22, 19] for formal definitions) to represent the interaction among the different timelines that model the domain. Conceptually these constructs define valid schema of values allowed on timelines and link the values of the timelines with resource allocations. Despite the syntactic differences, they allow the definition of Allen's[2] like quantitative temporal relations among time points and time intervals as well as constraints on the parameters of the related values. As an example to clarify the concept, let us present a synchronization in a robotic domain.

Let's consider a rover equipped with a stereo camera, an on-board memory and a communication facility. The rover is able to autonomously navigate the environment, to take pictures (storing them in the on-board memory) and to send files to a remote orbiter. In order to model the rover domain, the following subsystems are considered: a mobility system MS, a camera CAM, a communication system COMM and a memory MEM[2]. In the model, we also assume the rover able to move between two points in space. Hence the mobility system can be modeled as a state variable which can assume the following values: AT(?x, ?y) when the rover is standing in $\langle x, y \rangle$ and GOTO(?x, ?y) when the rover is moving toward $\langle x, y \rangle$. A transition GOTO(?x, ?y) $\rightarrow$ AT(?x, ?y) denotes a successful move to $\langle x, y \rangle$ and a transition AT(?x, ?y) $\rightarrow$ GOTO(?x′, ?y′) denotes the rover starting to move from a point $\langle x, y \rangle$ to a point $\langle x', y' \rangle$.

The rover camera can take pictures (in the current position of the rover) and store each picture on an on-board memory with a given file id. Hence the camera can be modeled as a state variable which can take the following values: CAMIDLE(), when the camera is not taking pictures and TAKEPIC(?file_id) when the unit is taking a picture that will be stored in a file with id = ?file_id.

The communication system can dump a file with a given id. Hence it can be modeled as a state variable which can take the following values: COMMIDLE(), the idle status, and DUMP(?file_id) when the unit is dumping a picture stored in a file with id = ?file_id. We

[2]This is just an excerpt of a real domain. See [17] for an extensive description of the domain and the model.

 DOI: 10.2420/AF09.2014.73

assume a timeline VW modeling the availability of a visibility window between the rover and the orbiter.

The memory has a fix amount of cells available to store pictures. A cell is occupied when a picture is stored and freed when a picture is transmitted to the orbiter. Hence it can be modeled as a consumable resource with unary consumption and production events.

A goal takePicture(?x, ?y, ?file_id) can be achieved by the rover by: (a) taking a picture with id = ?file_id, with the rover in ⟨?x, ?y⟩ and (b) dumping the picture. The sub-goal (a) of taking a picture in a given position can be achieved by synchronizing the value cam.takePic(?file_id) of the camera timeline with the value ms.at(?x, ?y) of the mobility system timeline. The sub-goal (b) of dumping a picture with a given id can be achieved by synchronizing the value comm.dump(?file_id) of the communication timeline with a value vw.available() of the visibility window timeline. The memory management is modeled by synchronizing a memory consumption at the start of the task of taking a picture and a memory production at the end of the dumping task.

This mix of causal and temporal relationships among the operations can be stated with the following synchronizations in DDL[3]:

```
synchronize MissionTimeline {
value takePicture(?x, ?y, ?file_id) {
op1 cam.takePic(?file_id);
op2 comm.dump(?file_id);
op3 ms.at(?x, ?y);
ref contains [0,+INF] [0,+INF] op1;
ref contains [0,+INF] [0,+INF] op2;
op1 before [0, +INF] op2;
op1 during [0, +INF] [0, +INF] op3; }}

synchronize comm {
value dump(?file_id) {
op1 vw.available();
ref during [0, +INF] [0, +INF] op1; }}

synchronize comm {
value dump(?file_id) {
op1 mem.produce(1);
ref end-at op1; }}

synchronize cam {
cam.takePic(?file_id) {
op1 mem.consume(1);
ref start-at op1; }}
```

HTN is one of the most used planning techniques in real world applications, in part because it allows to effectively encode knowledge into domain-independent planners. It is based on the differentiation between primitive tasks that can be directly executed and compound tasks, which must be decomposed in primitive tasks. The organization of tasks in hierarchies helps to simplify the modeling, which is one of the major problems that engineers need to face to deploy automated tools and at the same time allows for more understandable plans for the human expert. Some of the languages currently in use provide primitives for decomposing tasks into sub-tasks. Such a primitive differ conceptually from the synchronization because the reference activity is not justified by the sub-goals but is actually meant to be substituted by the sub-goals.

The modeling primitives introduced so far, although simple and intuitive, proved to be not so straightforward for users without a specific knowledge in advanced planning. In fact it is not obvious how to express certain constraints or imperative primitives very common in operational contexts. More in general even though there are common structures in practical problems that can be modeled with timed automata, resources, synchronizations and decompositions, it is not obvious for not experts how to do it. And the resulting models are difficult to be specified and managed if constructs at the proper level of abstraction are not made available.

Last but not least, even for experts, it is difficult to manage domains when the number of states, synchronizations and decomposition grows up, often leading to a situation where even experts have problem in understanding the logic of the domains if they have to be updated or fixed after the initial design. This is a typical problem in programming, and modeling for AI systems can be considered conceptually similar to programming. For this reason, the assumption is that techniques widely used in traditional programming can be adapted and applied to the design and maintenance of models in AI planning.

4 Pattern-Based Modeling

A well established methodology for simplifying the problem of writing and maintaining code is the use of *patterns*. In software engineering, a design pattern is "a general reusable solution to a commonly occurring problem within a given context in software design. A design pattern is not a finished design that can be transformed directly into source or machine code. It is a description or template for how to solve a problem that can be used in many different situations" (from Wikipedia).

[3]We use here the APSI platform modeling language as a reference because this is the syntax we are more familiar with. This choice is only for convenience, since we are using standard primitives available in any language for timeline based P&S.

The same principle can be applied to the problem of designing and maintaining models of AI planning and scheduling technologies. Instead of modeling the problem in terms of generic timed automata, temporal synchronizations and constraints among timelines, the user composes the model by means of a set of pre-defined automata (the patterns) and connect them with a reduced set of temporal and logical primitives. The K2E environment allows modeling concepts like processes, tasks, imperative constructs, cycles, conditional branches, concurrency and resource usages. The output of the process is a specification in timeline-based planning language that can be used to feed a domain independent timeline planner.

From the point of view of the methodological approach, to use patterns with AI models, we need to: (a) identify "common occurring problems" in P&S and Execution, (b) design and test (with the planner) modeling patterns for these problems, (c) classify an instance of a real problem to be modeled as one of the "common occurring problems" identified at point a) and (d) instantiate the pattern into a fragment of the real model specified in the target language.

Let's think for instance to processes that cyclically need to interrupt the sequence of states they're going trough to get a specific status before re-starting the sequence (like processes that cyclically need to shut down an instrument to avoid overheating for instance). Conceptually such a process can be defined as: (1) a set of operational states s_i cyclically taken by the process and the growth of temperature Δ_i after the execution of a task s_i; (2) the condition on the temperature to detect the need for interrupting the sequence, like for instance the maximum temperature t_{max} allowed for operating the instrument and (3) the special status R that restore the safety conditions when the sequence is interrupted. From the modeling point of view these processes when modeled with a state variable need to have a transition from any operational status to the shut down status (to entail the interruption of the activities from any operational status) and a transition from the shut down status to all of the operational states (to entail the re-start of the operation after the shut down). In addition to that, proper constraints need to be added to the transitions to entail the correct logic of interrupt and re-start.

Such an automaton is difficult to be specified manually in the model because of the combinatorial explosion of the number of the transitions and constraints to be added in order to enforce the correct logic, and because there are elements that requires careful modeling.

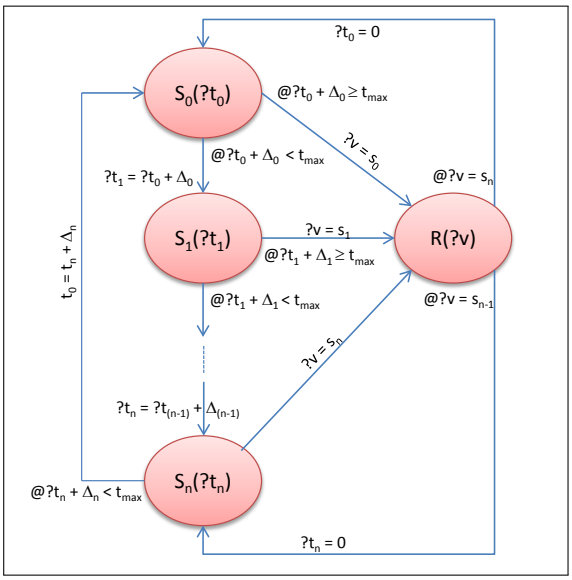


Figure 2: Interrupt Pattern

In this case for instance, since the process interrupted has to re-start from different states depending on when it has been interrupted, it is necessary to add a purely syntactical parameter to the idle status to keep track of the break point (see Figure 2).

This is a good candidate for a pattern: it is general enough because it is parametric with respect to the states and the conditions that interrupt the sequence; it is significant because this is a common behavior of a wide range of processes; can be translated automatically into a timed automata and specified with the planner language and, last but not least, it adds semantic value to the model allowing the modeler to immediately identify the logic behind all the states and constraints of the automata.

One more example of pattern could be related to the need of synchronizing a process with contingent values known at execution time, to obtain for instance the execution of specific tasks to recover from a error conditions driven by the actual status of the platform. In this case we assume that the plan will be executed, and during the execution the values of the telemetry of the platform that is executing the process are made available to the executor. We want to model a process that is able to react to some values of the telemetry, i.e., a process that by default executes a specific task, and can execute some other tasks in response to contingent situations. This scenario is more complex than the one presented above, because

 DOI: 10.2420/AF09.2014.73

the planner does not have, at planning time, all the information needed to choose the right set of tasks to be executed. For this reason, a *nominal plan* has to be calculated for this scenario, but the model has to be robust enough to enforce the proper reaction (by re-planning) if, at execution time, the value of the telemetry is different from the one supposed at planning time.

Hence we need to model at least two different scenarios: a "nominal" one, i.e. what is the configuration $\langle T_D, v_D \rangle$ of task T_D and telemetry value v_D expected in default conditions and (2) one (or more) not nominal scenarios $v_i \rightarrow T_i$, i.e., what are the telemetry values that require a reaction and what task has to be executed to react. There is a number of requirements for the plans that can be generated from the model to guarantee the correctness of the behavior that can be obtained executing these plans. First of all a nominal plan has to fail if the telemetry configuration changes during the execution of the nominal task T_D into a status that require to execute a different task (this is to enforce the reaction, if the envelope for the nominal plan accidentally contains configurations of the telemetry that needs a reaction, the platform might not react when needed). Secondly, a plan to react to a given configuration of the telemetry must fail with any other configuration (included the nominal one), to ensure the right reaction of the platform and to avoid an "over-reaction" (i.e. a reaction in nominal conditions). Finally, any plan for the not nominal conditions has to interrupt the nominal task and restore the status after its execution (to handle multiple reactions).

To translate this pattern into timelines, we need to add to the model also synchronizations among the timeline TL_P of the automata that models the process and the timeline TL_T that models the telemetry. To TL_P we associate an automata with at least one status T_D. To react to a generic value v_i we conceptually add to the automata the paths $T_D \rightarrow I \rightarrow T_i \rightarrow R \rightarrow T_D$, where the states I (for *Interrupt*) and R (for *Restore*) aim a modeling the procedures to interrupt the current activities and to restore the default status of the platform(see Figure 3, top).
Besides that, we synchronize the values of T_D and T_i with the values of the timeline TL_T that triggers the reactions. In order to guarantee at the modeling level that nominal plan must fail if the telemetry configuration changes during the execution into a status that require a reaction, we force the nominal task T_D to occur during the value v_D of TL_T. In order to ensure that a plan for a reaction T_i will fail with any other configuration of telemetry than v_i, T_i will be synchronized to occur during v_i. Finally, to guarantee that the platform will be able to perform T_D after T_i, R will be synchronized to restore the value v_D at its end (see Figure 3, bottom)[4].

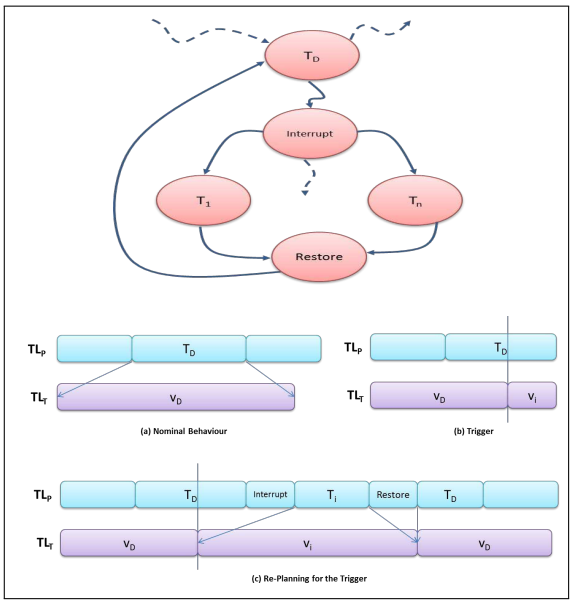


Figure 3: React Pattern

This is also a good candidate for a pattern. This is a typical behavior of systems that can get into not nominal states that require reactions. The pattern can be described with a reduced set of information, basically the pair $\langle T_D, v_D \rangle$ that model the nominal conditions and the list of reactions $v_i \rightarrow T_i$. Having that, the automata and the synchronizations of the model can be derived automatically.

5 The *k2e* Knowledge Engineering Environment

The *k2e* kee under deployment is based on the notion of *process* and *task*. A domain is made of a set of processes, supposed in execution concurrently (for each process there is one timeline in the model). Each process can be decomposed into sub-processes and tasks (which in turn can be again decomposed into sub-tasks) by applying *decomposition patterns*. At each process correspond a timed automaton. For each task there is a status in the

[4]An application of this pattern in a real robotic domain can be found in [18].

DOI: 10.2420/AF09.2014.73

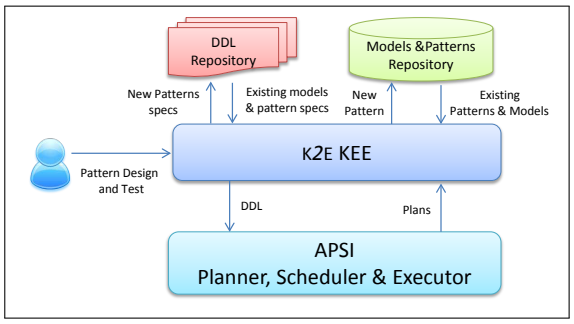


Figure 4: Use case: P&S Expert

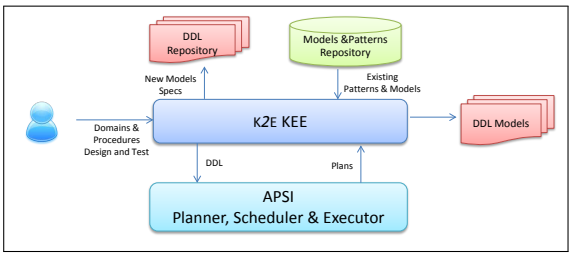


Figure 5: Use case: Domain Expert

automaton plus some other states that depends on the decomposition pattern applied to generate the automaton.

In addition to that there are symbolic and numeric variables. Variables can be of two types: *controllable* and *uncontrollable*. Controllable variables are associated to one or more processes. The actual value of controllable variables is given by the tasks of the processes to which the variables are associated. Uncontrollable variables model the telemetry of the physical system being modeled as well as any other exogenous event pertinent to the model. The values of uncontrollable variables can be either unknown at planning time (like the telemetry for instance) or provided with the specific instance of a problem (like orbital events for instance).

The structure of the automaton associated to a process can be designed by decomposing it into tasks applying *decomposition patterns* or *structural patterns*. These patterns can be applied recursively to the tasks to further decompose them into sub-tasks or to the whole process to decompose it into concurrent processes. Processes can be synchronized by applying *synchronization patterns*. Resource usages are modeled by applying *resource allocation patterns* to the tasks of one or more processes.

Decomposition patterns allow the definition of sequences of tasks, alternative tasks, if-then-else branches, for and while loops. Regarding the decomposition of a process into concurrent sub-processes, the patterns allow the definition of fork/join structures. At the branching points, conditions on both controllable and uncontrollable variables can be specified. In the first case, branches are translated directly into constraints on the timed automata that represents the process (an example of this type of branching is the variable t in the automaton in Figure 2, which is controllable because its value depends on the planned tasks), in the second case synchronizations are added to the model to enforce the right logic (an example is the value of the telemetry variable in the pattern in Figure 3).

Structural patterns allow the definition of entire blocks (like the twos described in Section 4). Combined with decomposition patters, structural patterns can be used to enforce a structure to the whole process or to the sub-tasks of a process. The structural pattern can be designed and tested directly into the environment, making available an evolving repository of patterns that can be used to compose increasingly complex domains.

Resource allocation patterns allows the definition of set of tasks that concurrently requires amounts of a given resource or sets of tasks that produce or consume a given resource. The translation of these patterns is done by synchronizing in the model the tasks with activities to be allocated on the resource (the first case) or production/consumption events for consumable resources (the second case).

The K2E environment uses the services provided by the ESA's APSI framework to compile fragments of DDL into APSI domains. The P&S solvers and the execution platform of the APSI framework can be used to test the patterns and the domains.

Possible K2E's use-cases depend on user's class. When the user is a planning and scheduling expert, K2Euses the environment to design and test patterns (see Figure 4). A new pattern can be designed in the environment by analyzing the code of existing patterns and DDL specifications in the repository and building the new pattern either as a composition of existing ones or by direct manipulation of the DDL code (see Figure 4). This is the only type of user that directly manipulate the DDL language (if needed). The modeling expert uses the planner to test the code associated to the pattern. The output of the process is a new pattern in the repository as well as a fragment of DDL for that pattern.

 DOI: 10.2420/AF09.2014.73

When the user is a domain and procedure expert, K2Euses the environment to design models by composing patterns and/or analyzing/updating existing models in the repository (see Figure 5). The planner and executor in this case is used to test the domain model on testing problems. The output of the process is a DDL specification of the domain that can be used with any APSI based application.

6 Conclusions

This paper presents the initial design of K2E, an ESA Knowledge Engineering Environment to deal with the acquisition, design and maintenance of domain models for current and future AI application for mission planning. The aim of K2E is to simplify modeling for the users, without loosing or harming the modeling power or the properties of the modeling language directly used by the planner.

The concept of K2E is to make the domain expert independent from the actual language used by the underlying AI technology. For this purpose, the environment is based on patterns that the modeler composes to define the domain. The output of the process is a machine generated specification in timeline-based planning language that can be used to feed the domain independent timeline planner. The set of proposed patterns is still initial, nevertheless is already possible to manage a subset of interesting features of real domains in mission planning. Moreover, K2E is designed also as a support for P&S experts for designing new patterns.

An environment to generate automatically domains inevitably induces an increased level of complexity in the problems managed by the AI technology. As a side effect, such a tool induces the need for increasingly efficient solvers, resulting not only in a new technology (hopefully) useful in operational contexts, but also as a leverage to foster research in domain independent planning and scheduling. In our case, since one of the main advantages of the pattern-based design is the strong semantical connotation of domain fragments, the possibility of using the additional information (induced by the structure of the patterns) to speed the search of the domain independent planner is also currently being investigated.

References

[1] Space engineering: Test and operations procedure language. European Cooperation for Space Standardization (ECSS), ECSS-E-ST-70-32C, ESTEC, The Netherlands, July 2008.

[2] J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[3] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[4] ANML. ANML Software Distribution Web Site. http://code.google.com/p/anml/, 2009.

[5] T. Bedrax-Weiss, C. McGann, A. Bachmann, W. Edgington, and M. Iatauro. Europa2: User and contributor guide. *NASA AMES RESEARCH CENTER, TECH. REP*, 2005.

[6] S. Bell and D. Kortenkamp. Embedding procedure assistance into mission control tools. In *Proceedings of the International Joint Conference on Artificial Intelligence Workshop on AI in Space*, 2011.

[7] M. Boddy and R. Bonasso. Planning for human execution of procedures using ANML. In *Proceedings of the Scheduling and Planning Applications woRKshop at ICAPS (SPARK-10)*, 2010.

[8] B. Buckley and J. Vangaasbeck. Scl: An off-the-shelf system for spacecraft control. In *Proceedings of the Third Int'l Symp. on Space Mission Operations and Ground Data Systems*, 1994.

[9] A. Ceballos, S. Bensalem, A. Cesta, L. De Silva, S. Fratini, F. Ingrand, J. Ocon, A. Orlandini, F. Py, K. Rajan, R. Rasconi, and M. Van Winnendael. A goal-oriented autonomous controller for space exploration. In *ASTRA 2011. 11th Symposium on Advanced Space Technologies in Robotics and Automation*, 2011.

[10] C.-C. Cheng and S. F. Smith. Generating feasible schedules under complex metric constraints. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, volume 2, pages 1086–1091, Seattle, Washington, USA, Aug. 1994. AAAI Press/MIT Press.

[11] S. Chien, M. Johnston, J. Frank, M. Giuliano, A. Kavelaars, C. Lenzen, and N. Policella. A Generalized Timeline Representation, Services, and Interface for Automating Space Mission Operations. In *Proceedings of SpaceOps 2012*, 2012.

[12] S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, and D. Tran. ASPEN - Automated Planning and Scheduling for Space Mission Operations. In *Proceedings of SpaceOps'00*, 2000.

[13] Europa. Europa Software Distribution Web Site. http://code.google.com/p/europa-pso/wiki/EuropaWiki, 2008.

[14] M. Fox and D. Long. PDDL 2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.

[15] J. Frank and A. Jonsson. Constraint based attribute and interval planning. *Journal of Constraints*, 8(4):339–364, 2003.

[16] S. Fratini and A. Cesta. The APSI Framework: A Platform for Timeline Synthesis. In *Proceedings of the 1st Workshops on Planning and Scheduling with Timelines at ICAPS-12, Atibaia, Brazil*, 2012.

[17] S. Fratini, A. Cesta, R. De Benedictis, A. Orlandini, and R. Rasconi. Apsi-based deliberation in goal oriented autonomous controllers. In *ASTRA 2011. 11th Symposium on Advanced Space Technologies in Robotics and Automation*, 2011.

[18] S. Fratini, S. Martin, N. Policella, and A. Donati. Planning-based controllers for increased levels of autonomous operations. In *ASTRA 2013. 12th Symposium on Advanced Space Technologies in Robotics and Automation*, 2013.

[19] S. Fratini, F. Pecora, and A. Cesta. Unifying Planning and Scheduling as Timelines in a Component-Based Perspective. *Archives of Control Sciences*, 18(2):231–271, 2008.

[20] D. Kortenkamp, R. P. Bonasso, D. Schreckenghost, and K. M. Dalal. A procedure representation language for human spaceflight operations. In *Proceedings of the 9th International Symposium on Artificial Intelligence Robotics and Automation in Space (i-SAIRAS 08)*, 2008.

[21] P. Laborie. Algorithms for Propagating Resource Constraints in AI Planning and Scheduling: Existing Approaches and new Results. *Artificial Intelligence*, 143:151–188, 2003.

[22] N. Muscettola. HSTS: Integrating Planning and Scheduling. In Zweben, M. and Fox, M.S., editor, *Intelligent Scheduling*. Morgan Kauffmann, 1994.

[23] D. J. Musliner, M. J. S. Pelican, and P. J. Schlette. Verifying equivalence of procedures in different languages: Preliminary results. In *Proceedings of the ICAPS 2009 Workshop on Verification and Validation of Planning and Scheduling Systems (VV&PS 2009)*, 2009.

[24] R. Sherwood, B. Engelhardt, G. Rabideau, S. Chien, and R. Knight. ASPEN, Automatic Scheduling and Planning Environment. Technical Report D-15482, JPL, 2000.

 DOI: 10.2420/AF09.2014.73

Acta Futura 9 (2014) 83-91
DOI: 10.2420/AF09.2014.83

Acta Futura

Procedural Onboard Science Autonomy for Primitive Bodies Exploration

Steve Chien*, Gregg Rabideau, Dero Gharibian, David Thompson,
Kiri Wagstaff, Brian Bue and Julie Castillo-Rogez
Jet Propulsion Laboratory, California Institute of Technology, 91109-8099, Pasadena, CA (USA)

Abstract. Future missions to primitive bodies will have limited time to explore these unknown bodies. Because of long round trip light times, if a mission commands the spacecraft at a detailed, time-sequenced level, there will be no opportunity to dynamically change the mission to respond to science opportunities. In order to address this issue, we are developing flight software to enable onboard science target detection and onboard response technologies to enable closed loop autonomous response for primitive bodies missions. These response methods must be able to predict the future opportunities to view the newly detected target using predicted spacecraft trajectory, target position and rotation, and future illumination conditions. These types of geometric reasoning for observation planning have traditionally been performed on the ground by highly skilled operations personnel. We describe the software under development and its application to future primitive bodies missions.

1 Introduction

Primitive bodies offer a unique view into the early solar system. Many of these objects (e.g. asteroids, comets) are unevolved since the early solar system and therefore present a unique view of the early solar system. Current space missions are exploring primitive bodies - most notably Dawn which is exploring Vesta and Ceres and Rosetta which is visiting the comet Churyumov-Gerasimenko. Additionally, future missions are under study to further explore these unknown primitive bodies.

Many primitive body missions have limited durations at a target. Because primitive bodies have a lesser gravitational field and there are a large number of asteroid primitve bodies (in the asteroid belt) it is possible to visit multiple bodies in a single mission. Therefore missions with multiple flybys and orbits are quite feasible. During a flyby the approach velocity is likely to be quite high so that the entire encounter might be of short duration (hours) - not enough time for the ground to be in the loop to modify the plan based on observed science.

We have adapted and extended a science event detection and procedural response capability to enable onboard autonomous mission response for primitive body exploration. This procedural response capability uses a resource-aware reasoning system [9] developed as an extension to the Virtual Machine Language (VML) flight executive [5] flying on numerous spacecraft. Future plans include adapting the CASPER model-based onboard planning system [1, 2] as well as compari-

*Corresponding author. E-mail: steve.chien@jpl.nasa.gov

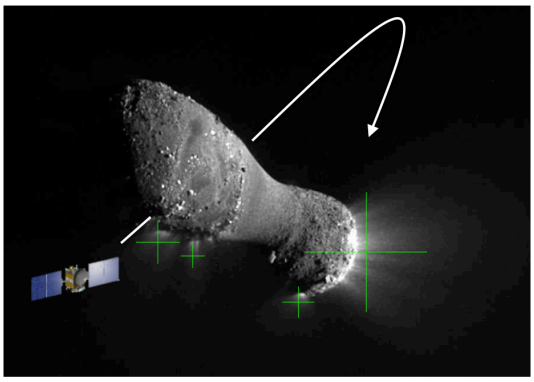


Figure 1: An Agile Science Spacecraft investigating high albedo areas on an asteroid.

son and evaluation of the two approaches. Our current VML adaptation runs in a software simulation of an embedded platform.

This prototype interfaces with onboard science detection software to enable rapid detection of science phenomena, and with navigation and geometric reasoning libraries to enable accurate planning and re-planning of followup observations. In the implemented scenario, the executive has a default observation plan of mapping the asteroid as it part of a pre-planned flyby. However early imagery is processed onboard to detect targets of interest, such as an outgassing event, an area of compositional interest, or detection of a satellite. Predeveloped science priorities indicate that such an even is higher priority than the pre-defined observation campaign, spawning new science goals. The onboard navigation/geometry software calculates potential new slews to support possible followup imaging activities to enhance science. The resource-aware VML accepts these new observation requests, and incorporates them as possible within the science priorities and operations constraints. The new plan is then executed and the spacecraft is able to acquire the preferred science imagery. This scenario is depicted in the graphic shown in Figure 1 in which a spacecraft investigates high albedo areas on an asteroid.

The software prototype is currently running with ad hoc interfaces (e.g. file) and a software simulation. As part of our continuing effort this software will be deployed to a hardware embedded platform to further mature it and ready it for future mission use. Future efforts also include expanding the range of operations scenarios.

Autonomous science provides the spacecraft with the capability to:

- detect science phenomena onboard, and
- respond by altering the original mission plan to take key observations to increase science.

Many operational scenarios exist where this onboard capability could enhance science. For example, onboard software could enable detection of an outgassing event at an asteroid or comet using an imaging instrument. Additional onboard software might then respond by commanding the spacecraft to acquire additional imagery, thereby exploiting a brief science opportunity otherwise missed.

In the remainder of this paper we first describe some of the unique challenges for primitive body missions. We then describe the ways in which agile science tehnologies can address these challenges through adaptive, onboard autonomy. Next we describe the overall concept of operations of agile science. We then provide examples of onboard science event detectors under study. We then describe the onboard procedural response component of the onboard autonomy system. Finally we discuss status, related work, future work, and conclusions.

2 Primitive Bodies Exploration - Challenges and Opportunities

Exploring primitive bodies present a number of challenges. First, the science features and events being detected include varied and subtle signatures:

- Plumes and outgassing events can be quite faint and may present in orientations that challenge detection (e.g. a plume erupting towards the spacecraft).
- The relative position of the Sun (illumination) with respect to the target and observer may not be ideal (e.g lighting behind the target).
- The morphology of the target may also present illumination challenges. If the target body has a very irregular shape, the exact illumination and observer viewing geometry may not be easily predictable.
- The target body may have unkown geology, making estimation of reflectance and other parameters more challenging.

 DOI: 10.2420/AF09.2014.83

		Missions: Asteroid / inert				Comet / active					
		Dawn-style mapping	Hayabusa II	OSIRIS-Rex	Trojan Tour	Rosetta Chiron Orbiter	Hopper	CSSR Comet	CNSR/CCSR	Coma Sampler	
Mission and science unknowns	Morphological units	x	x	x	x	x	x	x	x	x	x
	Surface composition, mineralogy	x	x	x	x	x	x	x	x	x	x
	Localized targets (boulders, crater walls, etc)	x	x	x	x	x	x	x	x	x	x
	Satellites	x									
	Plume activity, distribution over space and time						x	x	x	x	x
	Gravity field	x									
	Location of site for sampling/landing		x	x			x	x	x	x	
	Surface conditions at sample site		x	x			x	x	x	x	
	Rotation rate and pole location	x	x	x			x	x	x	x	x
	Spacecraft performance / faults	x	x	x	x	x	x	x	x	x	x
Applicable ground ops technologies	Single-cycle trajectory/observation selection	x	x	x	x		x	x	x	x	x
	Fast instrument data processing	x	x	x	x		x	x	x	x	x
	Fast instrument data interpretation	x	x	x	x		x	x	x	x	x
Applicable onboard technologies	Trajectory replan (fault or hazard recovery)		x	x			x	x	x	x	x
	Observation replan (opportunistic targeting)	x	x	x	x		x	x	x	x	x
	Morphological pattern recognition	x	x	x	x		x	x	x	x	x
	Spectral pattern recognition	x	x	x	x		x	x	x	x	x
	Plume/change detection						x	x	x	x	x
	Satellite detection	x									
	TRN / optical navigation for prox. ops		x	x			x	x	x	x	
	Onboard planning / execution for prox. ops		x	x			x	x	x	x	

Figure 2: Table highlighting (top) unknowns for primitive bodies missions and (bottom) applications of agile science technologies to these primitive bodies missions. Table from [12] reproduced with author permission.

In addition, primitive bodies exploration often present challenging timescales. Target bodies in the asteroid belt imply round trip light times to the Earth of approximately 1 hour. Given flyby durations of approximately 1 hour ground analysis and response to downlinked science data by a ground team is not possible.

Exploring primitive bodies is challenged by many unknowns.

- The gravity field of the target body is typcially now known except with very poor estimation. This poor gravity model will add uncertainty to any projected trajectory.
- Gas fields (e.g. for a comet) and outgassing events for comets and asteroids are unpredictable and can change the science environment as well as trajectory (due to changes in drag) with little or no warning.
- Unkown satellites. It may not be known if there are satellites prior to arrival. Satellites are both science targets and spacecraft safety hazards.

Finally, there are a limited number of close-up datasets for primitive bodes. This is particularly important in the context of the diversity of primitive body objects.

Agile science tchniques are applicable across a wide range of primitive body missions/concept either flying or under study. Figure 2 (top) shows the many unknowns that primitive bodies missions encounter. Figure 2 (bottom) shows the many relevant agile science technologies for each of the target primitive bodies missions.

3 Agile Science Scenario: Flyby

Our driving scenario for onboard autonomy aka "Agile Science" is a primitive body flyby scenario. Consider the 2010 Rosetta Orbiter flyby of the Lutetia asteroid. The timeline of the flyby is shown in Figure 3. With a relatie velocity of approximately 15 km per second, the flyby lasts less than an hour, far too short to involve the ground in the loop to command the spacecraft with

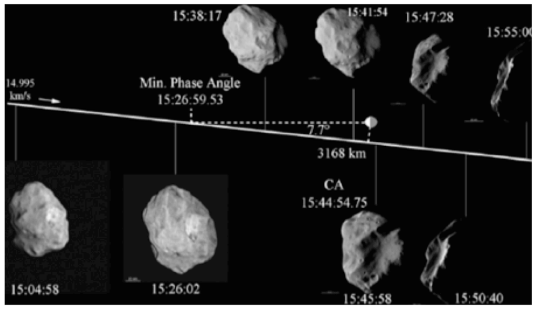


Figure 3: Timeline of the Rosetta spacecraft flyby of the asteroid Lutetia in 2010.

round trip light times being around an hour.

Traditionally, flybys wold be painstakingly planned by the ground using best estimated locations of expected targets of highest science interest. These ground planned observation sequences would be time-based sequences and would be executed open loop. Specifically, early acquisitions of science data would not be able to inform later observations.

In the Agile Science paradigm, the spacecraft and flight software enable onboard analysis of acquired science data to inform the subsequent actions of the spacecraft. Specifically, the Agile Science paradigm of execution would be as follows.

- Acquire science data
- Analyze science data
- Generate new data acquisition/target requests with priorities as pre-specified by the science team
- Assimilate new target requests into operational plan as appropriate based on prioritization.

This paradigm is highlighted by the operations scenario shown in Figure 4 which provides further detail on the autonomous target detection, prioritization and response.

4 Automated Target Identification

An important aspect of the Agile Science methodology is the ability to analyze data autonomously onboard the spacecraft to detect high priority science targets. While the general concept of Agile Science applies to a wide range of instruments initially we have focused on imaging instruments because they are central to most space missions. For primitive bodies exploration there are a wide range of science phenomena that can be discovered upon arrival at the target that warrant followup observations. A number of these we discuss below.

- Satellite search
- Outgassing/plume detection
- Volatiles search, materials search

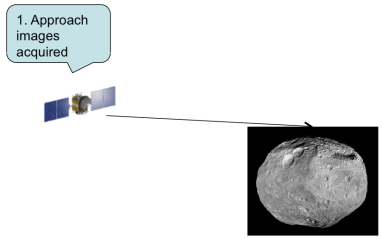
Figure 5 shows two promising onboard processing analysis products. At left is shown a High Albedo detection in imagery of the Hartley asteroid as acquired by the Deep Impact spacecraft. In this algorithm bright areas on the target body are extracted as these are areas of high science interest because of possible presence of volatile substances. At right is shown a morphology-based detection of a plume in imagery of Enceladus acquired by the Cassini spacecraft. In this algorithm the body of the target (The moon Enceladus) is fitted to an ellipse and the algorithm is searching for an area of brightness outside of the estimated target body (ellipse). Such an area if found may be a plume which is of high scientific interest.

The onboard detection algorithms produce images with a (possibly empty) set of detections. Next, based on pre-specified science priorities defined by the science team, targets are determined from these detections. For example, a target may be generated only if a plume appears in N consecutive frames of imagery. Or a bright albedo algorithm may only produce a target if the area of bright albedo exceeds a given threshold of brightness and exceeds an area (size) threshold. The output of the target detection algorithm is a set of prioritized targets in the acquired imagery.

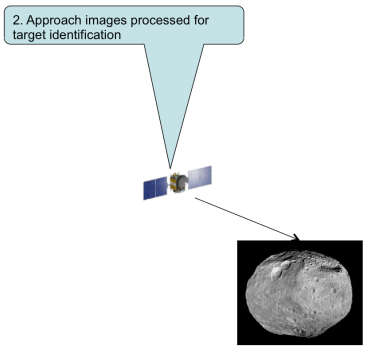
5 Geometric Computation

The target identification produces a set of targets (with associated priorities) in the imagery (e.g. specified in the image space, e.g. line, sample). This image space coordinate must be transformed into a target space coordinate (e.g. lat/lon, altitude on target body). Next calculations based on the spacecraft trajectory must be combined to determine legal viewing times (in effect accounting for solar position, rotation of target body, etc.). This will produce a set of possible re-imaging opportunities. Each of these can be considered a tuple of (opportunity-type-ID, priority, start time, end-time)

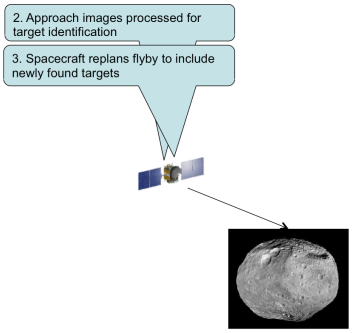
 DOI: 10.2420/AF09.2014.83



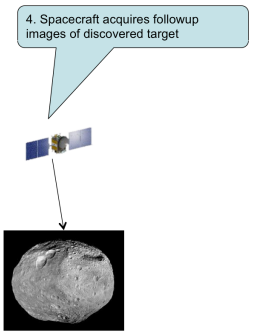
(a) Initial images acquired on approach.



(b) Approach images processed onboard to determine followup targets.



(c) Remaining flyby replanned in light of new detected targets.



(d) Followup images acquired if science priorities warrant.

Figure 4: Agile Science Flyby Scenario

which implies a required spacecraft pointing (via the opportunity type and associated observation and target location. These observation opportunities are then passed onto the onboard response system as new requested science goals with appropriate prioritization. An important point is that this geometric reasoning involving the relative positioning and trajectories of the target body, spacecraft, sun, and other bodies is typically done in a time and knowledge intensive ground-based observation planning process. One of the unique aspects of this work is to migrate this functionality onboard the spacecraft.

For many of these geometric calculations the SPICE library [7] is the common standard used for spacecraft operations. In our implementation we have used a combination of libraries from SPICE as well as some custom code. One element of future work will be to ensure that these calculations can fit within limited flight software computing resources.

Once the timing of the re -observation opportunities has been computed they can be passed to the procedural response system which can then attempt to (schedule followup observations as warranted by the science priorities.

6 Procedural Onboard Response

For the current Agile Science software prototype we utilize a procedural response system. Specifically we have implemented our response system on top of the Virtual Machine Language (VML) flight executive [5]. VML is flying aboard numerous missions including Mars Odyssey, MRO, Genesis, Pheonix, Grail, Spitzer, Juno, Dawn, and others. VML enables layered, modular autonomous response organized by Virtual Machines (VM's). We utilize a goal and resource manager [9] layered on top of base VML that enables efficient reasoning about prioritized goals and resource conflicts. Central

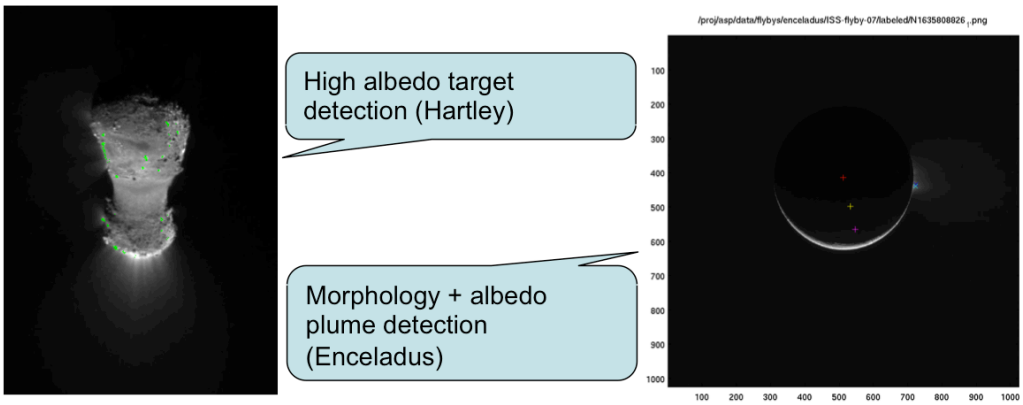


Figure 5: Example target detection and identification as shown by [11]. Left: High Albedo Target detection on Deep Impact/Hartley data Right: Morphology-based plume detection using Cassini/Enceladus data.

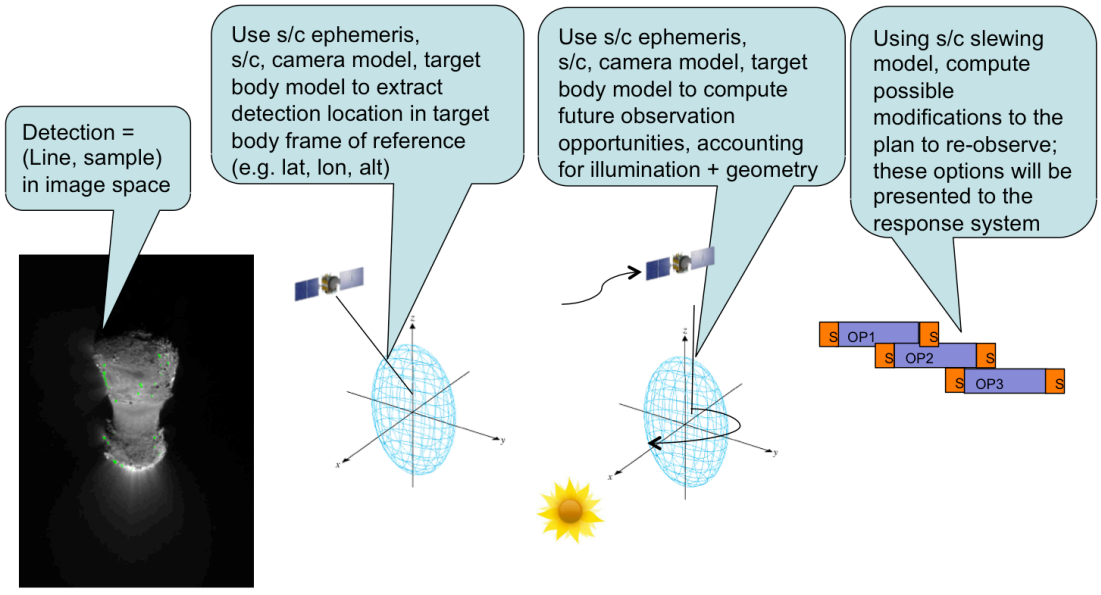


Figure 6: The process of determining when a target can be re-observed requires mapping the image space target into the target frame of reference coordinate system and then acounting for spacecraft position and pointing, target position and rotation an dother relevant features (such as the position of the sun).

 DOI: 10.2420/AF09.2014.83

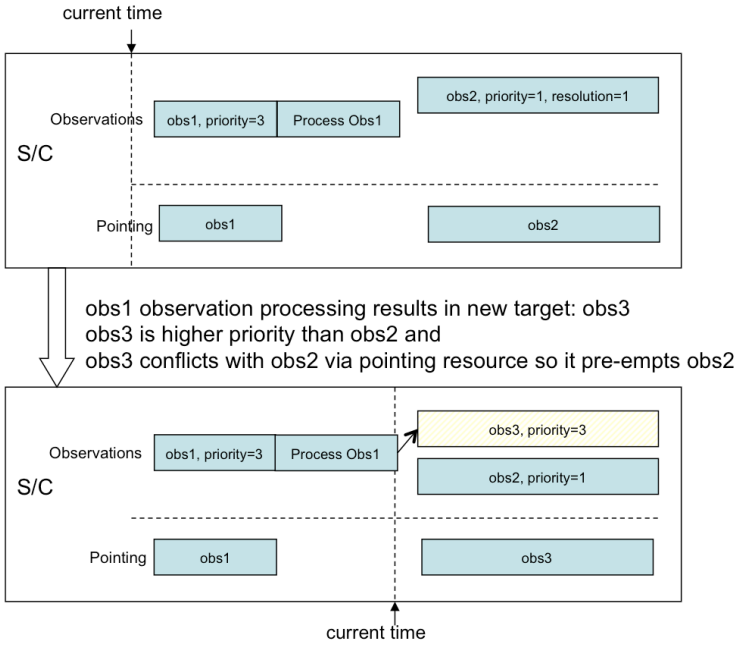


Figure 7: Dynamic goal selection in response to a newly generated high priority observation goal.

to the goal and resource manager is the concept of a flexible goal set. In our goal manager, goals represent a fixed set of activities to accomplish some purpose. For example, the set of activities required to take an image of the comet may represent a single goal even though many subparts are required to point the spacecraft, prepare the camera, acquire the image, write the image data to mass memory, repoint the spacecraft, and shutdown the camera. The goal manager also understands state and resource needs of the goals and uses this information to detect and track interactions (conflicts) between goals. At the core of the goal manager is an efficient computational algorithm to select the highest priority goals that do not conflict for inclusion into the current baseline plan at any point in time.

Figure 7 illustrates the operation of flexible goals sets. At first, the plan includes Observation 1, processing the data from observation 1, then taking observation 2. However, processing the data from observation 1, results in the creation of a new goal observation 3. Observation 2 and Observation 3 conflict, as they occur at the same time and require different pointings. The goal manager therefore inserts Observation 3 into the baseline plan and removes Observation 2 from the baseline plan. The baseline plan is then executed. A key assumption of the goal selection algorithm is that goals do not have temporal flexibility (i.e. their start and end times are fixed). This assumption is key to the computational tractability of the goal selection algorithm. The goal selection algorithm is an incremental algorithm. It must be run to re-select goals whenever goals are added or removed from the goal set. A goal is changed by removing it and adding the updated version. Goals are maintained in sorted sets. Shared resource and state interactions are maintained. When adding or removing a goal, only goals of equal or lower-priority to the added/removed goals need to be re-selected. Re-selection is worst-case $O(N^2 lgN)$ and average case $\Theta(NlgN)$. The worst case is when each goal has a constraint on every available resource (i.e. maximum interaction) and all goals are selected (i.e. no conflicts). In the average (typical) case each goal has a constant number of interactions via state/resource. The goal selection algorithm is described in greater detail in [9].

DOI: 10.2420/AF09.2014.83

7 Discussion and Conclusions

7.1 Related Work

Considerable prior work has investigated spacecraft autonomy using procedural methods.

- The Autonomous Sciencecraft (ASE) [1] utilized a planner (CASPER) in concert with the Spacecraft Command Language (SCL) executive and has been used for primary operations of Earth Observing One 2004 to the present (2013).
- V AMOS [13] is an onboard executive in development by DLR that validates branching plans on the ground and then selects execution branches onboard for operational flexibility.
- GOAC [3] is a goal-oriented architecture developed by ESA for future onboard use.
- The Remote Agent [6] utilized the batch planner RAX-PS and the procedural executive ESL [4] to control the Deep Space One mission for 48 hours in 1999.
- T-Rex [10] – is a planning and execution architecture that has been deployed for control of autonomous underwater vehicles.
- SCL [8] in addition to ASE has been used on the Tacsat-2 mission to offer onboard procedural rule-based automation.

7.2 Conclusions

Onboard autonomy can enable dynamic science for primitive body missions. Many science events can be detected via instrument processing techniques that are amenable to onboard computation.

We have demonstrated a capability to perform:

- Target Detection
- Target extraction and geometric computation required for re-observation opportunity analysis
- Modification of the existing observation plan to incorporate the new observation if warranted by science priorities
- Execution of the new plan

This capability is currently implemened in a software testbed and is being matured for future NASA missions.

Acknowledgements

This work was perfomed by the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration.

References

[1] S. Chien, R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davis, D. Mandl, B. Trout, S. Shulman, et al. Using autonomy flight software to improve science return on earth observing one. *Journal of Aerospace Computing, Information, and Communication*, 2(4):196–216, 2005.

[2] S. A. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau. Using iterative repair to improve the responsiveness of planning and scheduling. In *AIPS*, pages 300–307, Beckenridge, CO, April 2000. AAAI.

[3] S. Fratini, S. Martin, N. Policella, and A. Donati. Planning-based controllers for increased levels of autonomous operations. In *Proc 12th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA 2013)*, Noordwijk, The Netherlands, May 2013.

[4] E. Gat. Esl: A language for supporting robust plan execution in embedded autonomous agents. In *Proceedings of the IEEE Aerospace Conference*, volume 1, pages 319–324, Snowmass, CO, 1997. IEEE.

[5] C. Grasso and P. Lock. Vml sequencing: Growing capabilities over multiple missions. In *Proceedings of SpaceOps*, volume 8, Heidelberg, Germany, 2008.

[6] N. Muscettola, P. P. Nayak, B. Pell, and B. C. Williams. Remote agent: To boldly go where no ai system has gone before. *Artificial Intelligence*, 103(1):5–47, 1998.

[7] NAIF. About the spice library, August 2013.

[8] B. L. Prumo, B. Buckley, M. Long, and C. Finley. Unmanned spacecraft operations. In *Proc AIAA Infotech@ Aerospace,*, Seattle, WA, 2009.

 DOI: 10.2420/AF09.2014.83

[9] G. Rabideau, S. Chien, and D. Mclaren. Tractable goal selection for embedded systems with oversubscribed resources. *Journal of Aerospace Computing, Information, and Communication*, 8(5):151–169, 2011.

[10] K. Rajan, F. Py, and J. Barreiro. Towards deliberative control in marine robotics. In *Marine Robot Autonomy*, pages 91–175. Springer, 2013.

[11] D. R. Thompson, M. Bunte, R. Castano, S. Chien, and R. Greeley. Image processing onboard spacecraft for autonomous plume detection. *Planetary and Space Science*, 62(1):153–159, 2012.

[12] D. R. Thompson, J. C. Castillo-Rogez, S. A. Chien, R. Doyle, T. Estlin, and D. McLaren. Agile science operations: A new approach for primitive bodies exploration. In *Proceedings of SpaceOps*, Stockholm, Sweden, June 2012.

[13] M. T. Wörle and C. Lenzen. V amos verification of autonomous mission planning onboard a spacecraft. In *Proc Intl Workshop on Planning and Scheduling for Space*, Moffett Field, CA, March 2013.

DOI: 10.2420/AF09.2014.83

Acta Futura 9 (2014) 93-100
DOI: 10.2420/AF09.2014.93

Acta Futura

Space Hopper: a Serious Game Crowdsourcing the Design of Interplanetary Trajectories

Wiktor Piotrowski, Marcus Märtens, Dario Izzo* and Daniel Hennes
Advanced Concepts Team, European Space Research and Technology Centre, Keplerlaan 1, Noordwijk, The Netherlands

Abstract. Humans have an innate ability to extract important information from data sets or images, but also to solve particular problems on which computers may struggle. Serious gaming, and the related crowdsourcing term, are new computational paradigms attempting to process enormous amounts of scientific data exploiting these innate humans capabilities. In this paper we present Space Hopper, a "serious gaming"experiment aimed at improving interplanetary spacecraft trajectories design techniques and, at the same time, at proving that part of an interplanetary trajectory design can be crowdsourced. Space Hopper exploits and collects data on the users' problem-solving skills and spatio-temporal reasoning to help formulate a "human-inspired" tree search algorithm allowing efficient traversal of vast trees.

Keywords: crowdsourcing, serious gaming, trajectory design, tree search algorithms, self-adaptive differential evolution

1 Introduction

The design of spacecrafts interplanetary trajectories has been the exclusive domain of mission analysis experts. However, recent advances in the field of trajectory optimization, coupled to the growth of computational power, makes it possible to automatize a good deal of the necessary design steps and thus decrease the knowledge necessary produce good designs. Sophisticated modern search strategies (e.g. based on evolutionary computations) can explore vast solution spaces and locate good interplanetary trajectories able to compete with human-competitive results [10] taking the domain expert almost completely out of the loop.

With the serious game Space Hopper, we want to experiment with a radically new approach to interplanetary trajectory design which neither solely relies on computational power nor on human expert knowledge. Instead, we make use of the human spatio-temporal reasoning, intuition, curiosity and general problem-solving skills, as recorded during a game session, to improve the search of available trajectory options. By providing an appealing and easy user interface, humans with little or no knowledge about orbital mechanics, mission analysis or space engineering are able to design real trajectories enabling specific scientific goals to be pursued. In this first game prototype, the exploration of the Jovian system is considered.

From this perspective, Space Hopper can be seen as a *serious game* in which players from all over the world compete to find the best-possible flyable interplanetary trajectories. Unlike several previous scientific crowd-

*Corresponding author. E-mail: dario.izzo@esa.int

sourcing games looking for explicit solutions to their specific problem, Space Hopper also extracts information about the individual actions of all players on their way to discovering their final solutions. By applying machine learning techniques to this sequential data we aim to model the human decision making progress for this particular task. Out of this model, we want to infer a human-inspired general search strategy that can be transferred to other complex interplanetary trajectory problems in order to improve already known fully-automated search algorithms.

The structure of this paper is as follows: in Section 2 we give a short survey about related work with respect to serious gaming and crowdsourcing applied to science. Section 3 deals with the domain of interplanetary trajectory optimization and introduces the problem to be solved by the non-expert human players. Section 4 will give details about the user interface, the gaming experience and the underlying technologies used to develop Space Hopper. In Section 5 we describe the underlying data model and outline how we intend to use the collected information. We conclude with discussing future work and how crowdsourcing experiments can help advance the field of space exploration in Section 6.

2 Related Work

Using games for other purposes than mere entertainment is widely known under the broad term *serious gaming* [24, 15], first introduced by Clark Abt [1] in 1970. While these games may still be entertaining, they traditionally attempt to educate, inform or train the player. The objective of a serious game is usually linked to a real world scenario like military defense [22], health care [18, 12], emergency management [13], urban planning [14], education [29] or others.

Luis von Ahn introduced the term *games with a purpose* [26] in conjunction with the term *human computation* in order to describe a certain type of game that makes use of human brains as a computational unit in order to solve expensive large-scale problems. This merges serious gaming with the concept of *Crowdsourcing* [30, 3], a modern principle of labour sharing, where the workload is distributed among a vast number of humans, creating a *swarm* or *crowd*.

One popular example for Crowdsourcing is the Recaptcha project [28] which took scans from books and magazines where optical character recognition (OCR) failed and used them for the generation of captchas. A captcha is a security mechanism for websites to prevent bots from accessing content dedicated only for humans. By deciphering two words (one with a known answer and one from the scanned book) millions of humans helped digitalizing the archives of the New York Times basically for free by passing captchas. Another example is *Crowdfunding* [1], where investments for projects are gathered in tiny amounts but by a large number of people.

James Surowiecki [23] argues about *the wisdom of crowds* in his same-titled book, by giving examples under which conditions a guess based on aggregated information from a large number of not necessarily well-informed people might surpass sophisticated guesses from experts, ranging from stock markets to finding sunken submarines. This intriguing effect is of high interest if applied to scientific problems and can be used among the other benefits of Crowdsourcing in *Scientific discovery games* [19, 7].

One of the biggest success stories in this field is Foldit, a game created by the Center for Game Science at University of Washington in collaboration with the Department of Biochemistry [6]. It allows the player to manipulate long protein strains in 3 dimensions (see Figure 1) to discover the folded spatial structure given a specific sequence of amino acids. While this problem is computationally expensive, the game makes use of the inherent reasoning, learning and pattern recognition skills of humans, allowing for a way faster computation. A major achievement of the players of Foldit was helping to fully model the Mason-Pfizer monkey virus enzyme, a virus which causes AIDS among monkeys. Bioengineers have been trying to decipher the structure of this virus for 15 years whereas the online player community required just 10 days to fully model it in 3D [11].

There are several other scientific discovery games to mention. In the citizen science project Galaxy Zoo Supernovae [21] players were shown the wield-field images to search for stars in order to classify super nova candidates. More projects involve humans to classify bat calls, galaxy types, cancer cells, discovering exoplanets, cyclones and many more [2]. In the online game EyeWire [3], the players help to trace the spatial structure of neural connections throughout the retina of mice.

Most of the time, scientific discovery games exploit

[1] See Kickstarter (`http://www.kickstarter.com` and Indiegogo (`http://www.indiegogo.com`) for two popular platforms.

[2] See `https://www.zooniverse.org`

[3] See `https://www.eyewire.org`

 DOI: 10.2420/AF09.2014.93

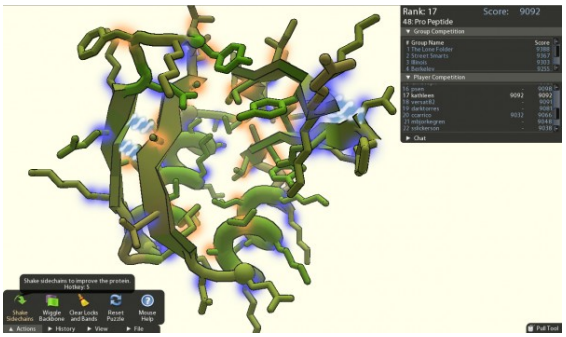


Figure 1: A screenshot of Foldit gameplay

the advanced image processing capabilities of humans, which are still by far superior than nowadays algorithms. Peekaboom [27] uses these abilities to let its players detect specific objects and relevant regions in pictures. By providing human annotations, there is hope that games with a purpose are able to create ontologies to help weaving the semantic web [20].

3 Background

Space Hopper is based on the problem proposed by NASA's Jet Propulsion Laboratory (JPL) as part of the sixth edition of the Global Trajectory Optimisation Competition (GTOC6). The participating team were tasked to globally map Jupiter's Galilean moons [17] (Io, Europa, Ganymede and Callisto) using the multiple gravity assist technique within a four-year window. The surface of each moon is divided into 32 areas (faces) where each face is either a pentagon or a hexagon. This division approximates the moons as truncated icosahedrons (classic soccer balls). Each face is assigned a score from 1 to 3 depending on the scientific interest of the area on a given moon (see Figure 2). Faces on Europa are worth double points due to higher scientific interest of this particular moon. Points are scored by successively mapping faces of each of the four moons. A face F_i is visited if the closest approach vector r_p of a flyby around a chosen moon is passing through that particular face. For a face to be visited, it has to be fully or partially within the "visitable band" an area on the surface of the given moon where the flyby parameters are within the allowable bounds (see Figure 3). A face F_i is visitable if at least one of its vertices lies within the band or its vertices lie on both sides of the band (yet none inside it). Points for visiting face F_i are scored only for the first flyby over that face, no points are added to the overall trajectory score for subsequent flybys over face F_i. The maximum cumulative score for a full tour of the Galilean moons is 324.

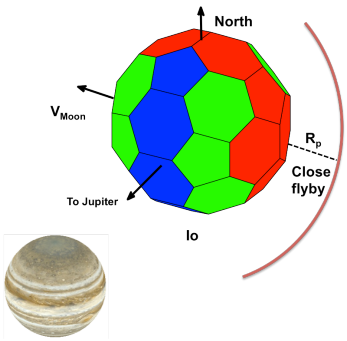


Figure 2: A model of a Galilean moon with face values represented through colours

At starting epoch t_0 the spacecraft mass is M_0 = 2000kg, half of which is the propellant, therefore the mass of the spacecraft cannot fall below 1000kg. With each flyby the overall spacecraft mass is reduced based on the thrust used during the Deep Space Manoeuvre (DSM) with a maximum continuous thrust of τ_{max} = 0.1N. Furthermore, there also exists a mass penalty for close approaches to Jupiter to account for additional shielding from Jupiter's strong magnetic field. The penalty is applied if, at any time during the mission, the spacecraft range to Jupiter goes below $2R_J$.

GTOC6 was won by a joint team of University of Rome and Turin Polytechnic with best score of 311 points. The team attempted to fully map one moon before moving onto the next one. However, after the competition a new solution scoring 316 points was found by Advanced Concepts Team (ACT) of European Space Agency. The trajectory is fundamentally different from the winning solution of GTOC6 as it exploits a "moon-hopping" technique allowing rapid transfers between moons rather than fully mapping one moon before moving to the next one. The found trajectory was evolved using PaGMO/PyGMO[4] and is up to date the best known valid solution for the GTOC6 problem [10].

[4]See `http://github.com/esa/pagmo`

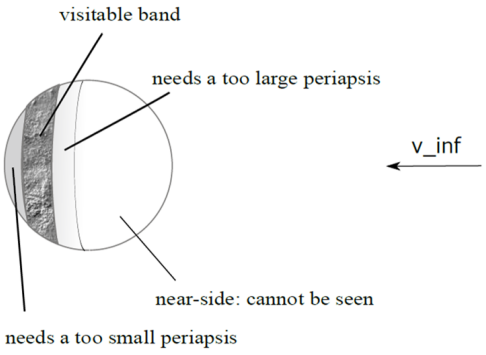


Figure 3: The visitable band shown on an example planet

3.1 Orbital mechanics

The problem was defined with great detail and accuracy thus it required an appropriate representation of the Jupiter system and spacecraft dynamics. Space Hopper uses state-of-the-art methods based on PyKEP, an open-source keplarian calculations toolbox developed by the ACT. The error margin for the calculations is at maximum 10^{-9}. The parameters of the Galilean moons as well as Jupiter were defined in JPL's problem statement (e.g. Keplerian orbit elements, gravitational parameter, radius, etc.). The time epochs are encoded in Modified Julian Date format (number of days elapsed since 17 November 1858). The moon ephemerides (cartesian position and velocity) are calculated according to values provided by JPL without any perturbations in the orbits of the satellites. The overall accuracy and realistic approach to the problem makes it feasible for a pre-phase A study.

3.2 Trajectory encoding and parameter optimisation

The Grand Tour starts at a point on a sphere of radius R_s=1000R_J with Jupiter at the centre. The first leg (called *incipit*) is unique as it does not include a DSM in the transfer to the first visited moon. The parameters of the first leg are encoded into a 4-dimensional array (called a chromosome in evolutionary computing) $x_0 = [t_0, u, v, T_0]$. t_0 is the epoch of the spacecraft launch date. u and v are used to calculate the coordinates of spacecraft at epoch t_0 on the aforementioned sphere around Jupiter:

$$r_0 = R_S(\cos\theta\cos\phi\hat{i} + \sin\theta\cos\phi\hat{j} + \sin\phi\hat{k})$$

where $\theta = 2\pi u$, $\phi = \cos^{-}1(2v-1) - \pi/2$, u and v were chosen over directly optimising the values of θ and ϕ to ensure a uniform distribution of sampled points on the sphere. T_0 is the duration of the spacecraft's journey to the first visited moon (in days) with its lower and upper bounds set at 180 and 220 days respectively.

Each subsequent leg of the tour is an interplanetary trajectory between two moons of Jupiter with a single Deep Space Manoeuvre. These legs are also encoded as a 4-dimensional chromosome $x_n = [\beta_n, r_{pn}, \eta_n, \tau_n]$ where β_n is the flyby angle at the periapsis, r_{pn} is the periapsis of the flyby, η_n is the timing of the DSM (ratio of days elapsed before the DSM to total leg duration τ_n). The bounds for β and r_p are calculated for each face of the currently visited moon based on the incoming velocity of the spacecraft but have to be inside $[-2\pi, 2\pi]$ and $[(R_m + 50000/R_m), (R_m + 2000000/R_m)]$ respectively where R_m is the radius of the currently visited moon. η values have to be between 0 and 1 exclusive.

For each leg its chromosome is optimised using a self-adaptive Differential Evolution algorithm (jDE) [4] to ensure the best results. jDE allows us to skip parameter tuning before solving each leg problem and still guarantee high performance of the optimisation. The flyby parameters for each leg of the trajectory are evolved 200 times over 60 generations with 12 individuals. At each step of the optimisation, an individual is evaluated using an objective function which calculates each legs change in velocity ΔV during the Deep Space Manoeuvre. The evolution attempts to minimise ΔV for each leg in order to make the trajectories as close to ballistic as possible.

The objective function of the first leg is computed as a Lambert Problem between r_0 and r_1 with transfer time T_0 where r_0 is the position of the spacecraft on the $1000R_J$ sphere (calculated using the u and v variables) at time t_0. r_1 is the position of the first visited moon m_0 at time $t_0 + T_0$. Each consecutive leg has an objective function defined as MGA-1DSM [9].

4 Game Design

This section describes the current design status and some future planned developments for Space Hopper. We present the details of the different technologies used to create the experiment. Furthermore, the gameplay and interface is explained in order to better understand the application. Lastly, we present the search tree that is used to represent the underlying problem and discuss its content and significance.

 DOI: 10.2420/AF09.2014.93

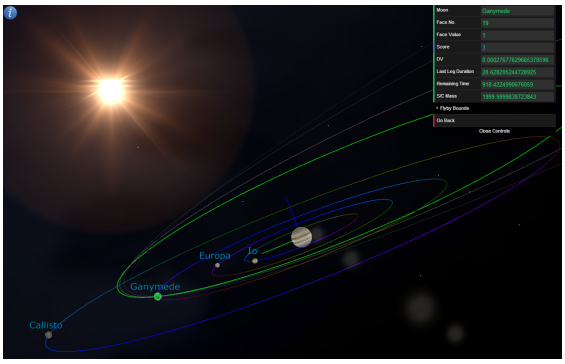


Figure 4: A screenshot of the Space Hopper experiment during play.

4.1 Development and design

Space Hopper was designed with adaptability and compatibility in mind. The chosen technologies and methods comply with the established and emerging standards of interactive web development.

HTML5 was chosen as to maximise the number of possible players/contributors as it is the current standard for dynamic websites and online games. Furthermore, it is currently being introduced on mobile devices allowing further expansion of the user base and development in a new direction through the use of touchscreens. HTML5 allows for a more engaging design through combined use with various other programming languages and technologies. An example of this is using inline CSS for secondary style definition of the webpage.

JavaScript is the most crucial element of interactive web pages and applications as it supports (among others) the object-oriented programming style. Such capabilities allow definition and manipulation of composite objects in Space Hopper as well as carrying out various complex astrodynamic calculations thousands of times a second. Furthermore, the differential evolution algorithm is also defined in JavaScript which is currently emerging as a programming language for scientific purposes and has proved to have high performance [2]. However, Space Hopper seamlessly incorporated the algorithm without sacrificing the fluidity or the experience of the experiment. This is mostly due to the implementation of equations being based on the aforementioned open-source keplerian toolbox, PyKEP.

Space Hopper graphics are also supported by JavaScript through the use of Three.js [5], a compact, yet powerful open-source 3D graphics library. It allows creating various visual objects such as trajectory and orbit curves, spheres and even truncated icosahedrons (moon models). Moreover, Three.js manages the crucial scene objects of the HTML canvas including camera settings, movement in three-dimensional space and object texturing.

MySQL is used to store data submitted by the players who choose to do so. The database contains the details of the players choices and the values of crucial variables at each step of trajectory design including backtracks. Each row in the database is a full interplanetary trajectory. PHP is used to transfer the gathered data to and from the MySQL database.

4.2 User interface and Game-flow

Space Hopper can start in two different ways, with or without the incipit phase. This choice affects the difficulty of the later gameplay. Although the two starting points of the game vary only by two legs of the trajectory, designing the rest of the tour is very different. Designing the full tour (including the incipit phase) is a much greater challenge because the first trajectory legs are quite eccentric and have a very large semi-major axis. The players have to adjust various variables such as leg duration and the perijove of the leg trajectories in order lower semi-major axis and achieve short-duration transfers between moons with $\Delta V \approx 0$. At the very beginning the player has to choose two moons to fly by, afterwards the incipit trajectory optimisation takes place. On the other hand, the second type of gameplay (post-incipit phase) sets up the initial conditions so that it is very easy for the players to achieve $\Delta V \approx 0$ trajectories without having to spend extensive time adjusting the flyby variables.

Following the initial phase of Space Hopper, the game continues with the spacecraft at a particular moon (either the second of the moons chosen during the incipit phase or Ganymede for the non-incipit level. The player has to select one of the faces on the surface of the currently visited moon. Depending on the velocity of the spacecraft and the moon ephemerides, for each flyby, on average 12 out of 32 of these faces are visitable. The player has to choose very carefully as each decision has a major impact on the following legs of the trajectory, even though the immediate gain from the face value might not be the highest.

After selecting the face to fly by, the player is presented again with the full view of the Jovian system

so that he can select the next moon to visit. Again, the automated leg optimisation follows. This optimisation depends on the previously selected face and the general user-adjustable parameters of the trajectory leg. The user can adjust the duration of the following leg (in days), the perijove of the flyby and the bounds for the face to ensure scoring of the chosen face (there is approximately 90% chance of scoring the selected face due to the non-rectangular shape of the faces).

The process of face and moon selecting is then repeated until either all of the faces are visited, the mission time has expired or the spacecraft mass has fallen below 1000kg.

When the game has finished, a prompt appears showing the details of the designed trajectory (total score, cumulative ΔV, remaining mass, etc.). The user can then submit the score to be saved in the ACT database and analysed later.

4.3 Search tree

The entire problem has been formulated as a tree-search. While usually tree traversal algorithms are very efficient, the search space grows exponentially in this case and thus becomes very quickly too vast to be fully explored in order to find an optimal solution. In Space Hopper, the average number of the visitable faces, at any point in the search, is 15 for each of the 4 moons. Therefore, the search tree has an average branching factor of 60 which means that at the n^{th} step the search tree is of size 60^n. Although there exist many efficient tree-search algorithms [31, 25, 16, 8], this problem is still too difficult for any of them to return a good solution in a reasonable time period due to sheer number of possible choices and necessary calculations at each step of the search to determine the overall fitness of the trajectory. In the tree, each of the nodes is a full interplanetary trajectory between two moons which contains crucial information about the position of the spacecraft, m (i.e. which moon of Jupiter it is visiting), at epoch t, its velocity (v_{in}) and the accumulative velocity change (ΔV) as well as a list of all the previously visited moon faces (**f**).

Humans are an integral part of Space Hopper's tree search. At each step, users provide crucial information by making informed decisions about which face and moon to visit and by manipulating the flyby parameters.

Space Hopper's approach to tree-search differs even more from the classical methods as each step can be re-optimised multiple times to achieve lower ΔV as the evolutionary algorithm responsible for finding the best flyby parameters can often return a non-optimal solution. Re-optimisation can also mean the players adjusting the flyby bounds to improve the optimisation of the trajectory through jDE.

5 Data Mining

In order to analyse the human players' reasoning, data needs to be collected. All of the decisions made by users are recorded to ensure maximum feasibility when formulating heuristics for the novel tree-search algorithm. The submitted solutions are the full Jupiter Grand Tour trajectories which, as mentioned before, are formulated as a search tree where each node contains details about the state of the spacecraft at that time (currently visited moon, velocity) and global values of the tour (epoch, previously visited faces and the cumulative change in velocity of the spacecraft).

We attempt to formulate a set of heuristics for a new "human-inspired" algorithm based on the players' understanding of the system and their decisions. We therefore need to track the players' gameplay and understand how their choices affect general outcomes in terms of score, ΔV, time, spacecraft mass, etc. As a result, backtracks in the search are also recorded. Thus the search tree also contains branches (parts of the tour) which were considered and explored but ultimately discarded.

5.1 Future Work

The second part of the project is the most important one as it will require very sophisticated machine learning algorithms for recognising patterns in the collected data submitted by the best online players. The resulting data should provide us with very interesting mix of Breadth-first (BFS) and Depth-first (DFS) search algorithms which comes naturally to humans and can prove beneficial to the advancement of efficient and powerful tree-search algorithms for trajectory design. Also, learning from re-optimisations of legs and user backtracks in the search should give us better techniques for assessing the feasibility of fitness scores for different types of trajectories.

The resulting "human-inspired" tree-search algorithm will then be compared and contrasted with various well-established and efficient algorithms (such as Breadth-first, Depth-first, Best-first, A* or Lazy-race).

 DOI: 10.2420/AF09.2014.93

This should give us an indication of how the human abilities fare against the best of decades of Artificial Intelligence research.

6 Conclusion

This paper introduced Space Hopper, a novel online, crowdsourcing experiment which attempts to improve automated trajectory design methods by analysing how humans try to tackle the complex Jupiter Grand Tour problem. The problem is currently too computationally expensive even for the state-of-the-art algorithms but humans have a set of certain problem-solving skills which could prove helpful in advancing search algorithms for vast search trees with a high branching factor. Space Hopper is released online and features a 3D game-like look & feel to maximise the user base. Current trajectory design methods have reached a level where non-experts can design complex yet feasible trajectories.

References

[1] C. C. Abt. Serious games: The art and science of games that simulate life. *New Yorks Viking*, 1970.

[2] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman. Julia: A fast dynamic language for technical computing. *CoRR*, abs/1209.5145, 2012.

[3] D. C. Brabham. Crowdsourcing as a model for problem solving an introduction and cases. *Convergence: the international journal of research into new media technologies*, 14(1):75–90, 2008.

[4] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *Evolutionary Computation, IEEE Transactions on*, 10(6):646–657, 2006.

[5] R. Cabello. Three.js. *URL: https://github.com/mrdoob/three.js*, 2010.

[6] S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, Z. Popović, et al. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756–760, 2010.

[7] S. Cooper, A. Treuille, J. Barbero, A. Leaver-Fay, K. Tuite, F. Khatib, A. C. Snyder, M. Beenen, D. Salesin, D. Baker, et al. The challenge of designing scientific discovery games. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, pages 40–47. ACM, 2010.

[8] R. Dechter and J. Pearl. Generalized best-first search strategies and the optimality of a*. *Journal of the ACM (JACM)*, 32(3):505–536, 1985.

[9] D. Izzo. Global optimization and space pruning for spacecraft trajectory design. *Spacecraft Trajectory Optimization*, 1:178–200, 2010.

[10] D. Izzo, L. F. Simões, M. Märtens, G. C. H. E. de Croon, A. Heritier, and C. H. Yam. Search for a grand tour of the jupiter galilean moons. In *Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013*, pages 1301–1308, 2013.

[11] F. Khatib, F. DiMaio, S. Cooper, M. Kazmierczyk, M. Gilski, S. Krzywda, H. Zabranska, I. Pichova, J. Thompson, Z. Popović, et al. Crystal structure of a monomeric retroviral protease solved by protein folding game players. *Nature structural & molecular biology*, 18(10):1175–1177, Dec. 2011.

[12] J. F. Knight, S. Carley, B. Tregunna, S. Jarvis, R. Smithies, S. de Freitas, I. Dunwell, and K. Mackway-Jones. Serious gaming technology in major incident triage training: A pragmatic controlled trial. *Resuscitation*, 81(9):1175–1179, 2010.

[13] M. Kobes, N. Oberijé, K. Groenewegen, and T. Morsche. Serious gaming for behavioural assessment and research in case of emergency. an evaluation of experiments in virtual reality. *Proceedings of SimTecT, Adelaide, Australia*, pages 15–18, 2009.

[14] K. Mallan, M. Foth, R. Greenaway, and G. T. Young. Serious playground: using second life to engage high school students in urban planning. *Learning, Media and Technology*, 35(2):203–225, 2010.

[15] D. R. Michael and S. L. Chen. *Serious games: Games that educate, train, and inform*. Muska & Lipman/Premier-Trade, 2005.

[16] J. Pearl. Heuristics: intelligent search strategies for computer problem solving. page 48, 1984.

[17] A. E. Petropoulos. Problem description for the 6th global trajectory optimisation competition, Sept. 2013. URL: `http://goo.gl/FmrOc`.

[18] P. Rego, P. M. Moreira, and L. P. Reis. Serious games for rehabilitation: A survey and a classification towards a taxonomy. In *Information Systems and Technologies (CISTI), 2010 5th Iberian Conference on*, pages 1–6. IEEE, 2010.

[19] N. Savage. Gaining wisdom from crowds. *Communications of the ACM*, 55(3):13–15, 2012.

[20] K. Siorpaes and M. Hepp. Games with a purpose for the semantic web. *Intelligent Systems, IEEE*, 23(3):50–60, 2008.

[21] A. Smith, S. Lynn, M. Sullivan, C. Lintott, P. Nugent, J. Botyanszki, M. Kasliwal, R. Quimby, S. Bamford, L. Fortson, et al. Galaxy zoo supernovae. *Monthly Notices of the Royal Astronomical Society*, 412(2):1309–1319, 2011.

[22] R. Smith. The long history of gaming in military training. *Simulation & Gaming*, 41(1):6–19, 2010.

[23] J. Surowiecki. *The wisdom of crowds*. Random House Digital, Inc., 2005.

[24] T. Susi, M. Johannesson, and P. Backlund. Serious games: An overview. 2007.

[25] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[26] L. Von Ahn. Games with a purpose. *Computer*, 39(6):92–94, 2006.

[27] L. Von Ahn, R. Liu, and M. Blum. Peekaboom: a game for locating objects in images. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 55–64. ACM, 2006.

[28] L. Von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.

[29] W. Westera, R. Nadolski, H. G. Hummel, and I. G. Wopereis. Serious games for higher education: a framework for reducing design complexity. *Journal of Computer Assisted Learning*, 24(5):420–432, 2008.

[30] A. Wiggins and K. Crowston. From conservation to crowdsourcing: A typology of citizen science. In *System Sciences (HICSS), 2011 44th Hawaii International Conference on*, pages 1–10. IEEE, 2011.

[31] R. Zhou and E. A. Hansen. Breadth-first heuristic search. *Artificial Intelligence*, 170(4):385–408, 2006.

 DOI: 10.2420/AF09.2014.93