Università
Ca'Foscari
Venezia

Computer in a Room Challenge

Final Report
Team 4

Authors: *Ciccone Mattia, Geretto Nicola, Geretto Thomas, Nicoletti Gabriele, Sale Riccardo*

Department: Computer Science
Date: November 21, 2022

Contents

1 Programming language

We decided to use python for the following reasons:

- Availability of various libraries specific to the project requirements, some already included in the standard library.
- Ease of use that allows focusing on program logic/reasoning instead of implementation compared to low-level languages.
- Ability to translate various python scripts into executables using C++ API calls through various libraries, this solution improves efficiency.
- A specific cross-platform library (APScheduler) was used, which allows the management of timed and simultaneous jobs.

2 World mapping

Concerning the complete map acquisition, it was decided to focus on the implementation of a fuel-saving algorithm whose focal points are:

- The only circumstance in which the mapping is interrupted is the individuation of a target whose deadline is less than 1.5 hours or the detection of two or more simultaneous targets.
- Keeping the speed of the x constant at a value of 10 (maximum speed of the narrow lens).
- Descent speed y constant at 1 to minimise the delta between speeds resulting in a consumption of 0.02 litres per descent.
- The descent is carried out only after completion of the row we are in.
- To ensure complete map acquisition each adjacent photo has an overlap of 15px in x and 15px in y.
- A specific data structure has been set up to store already mapped areas so that no photos are taken again on the same coordinates. The structure is a dictionary that for each key (coordinate y) contains an array of intervals, representing the zones in x already visited, in case of overlap on the x intervals these are merged (until the final situation [0.21600] where we have covered the entire row).
  All this has been implemented with an eye to efficiency in terms of the computational complexity to be observed within the code (file utility.py, functions: see_if_takephoto, cover_zone, merge, e.g. the array of intervals in x is not really repeated for every single co-ordinate).
- With this structure, it is possible to make two choices, the first is to wait for the mapping to descend at speed 1 and then go and fill in the holes and the second uses more fuel but is faster, which treats the few remaining holes in the final stage of mapping as if they were targets.
- In the configuration we used in the evaluation phase, the complete mapping time net of deviations for targets and any photos not taken is about 13 hours, with a total consumption of 0.3 litres.
- During mapping, the battery always returns to 100 per cent, as the only times the melvin is in active mode are when taking the photo or changing speed.
- Once 100 per cent coverage of the mapping is reached, this job will be terminated and no longer scheduled to avoid wasting fuel.

3 Objective scheduling and acquisition process

A job was set up that every 60 seconds evaluates how many and what are the active objectives and then provides them to a function that verifies and decides whether to start acquiring them. The entire process follows the listed strategies:

- If there is only one target with more than one and a half hours to go then it continues with the mapping.
- If it is at a distance of less than 300px in y it suspends the mapping and begins the acquisition.

These first two points make it possible to limit the waste of fuel as if an objective has a lot of time it is likely to be reached by the mapping.

- In the event that there is only one active target and a time of less than one and a half hours or several simultaneously active targets, it immediately starts the process of travelling and subsequent acquisition.
- A special function has been developed which given two coordinates on the map and the current speed of the melvin processes all compatible speed pairs (x, y) to reach the destination in the desired time limit also providing parameters for the total time needed for the trip, acceleration time, deceleration time, cruising time and fuel needed.(file calculations.py: functions(calculator,calculate_distance_time_fuel)
- Once the target or set of targets to be reached has been identified, a sorting of the targets is carried out based on the distance in y relative to the current position of the MELVIN in order to create a route. Appropriate speeds will be calculated for each segment of the path.

- Once an objective has been completed, if new objectives have been activated it recalculates the route including them otherwise it maintains the previous one.
- Objectives have been categorised as follows, SINGLE-LAYER an objective whose required lens covers the entire top-bottom distance, MULTI-LAYER an objective for which more than one y is required to cover the entire area.
  The SINGLE-LAYER type has the median between the top and bottom as the y and the left shifted appropriately for the x, from there on the photos will be timed according to the size of the lens.
  In the case of MULTI-LAYER objectives, on the other hand, all the y's on which you will have to scroll to complete the area are first identified, the number of photos required for each line and then once the destination is reached at set intervals a photo is taken.
- The descent between layers in MULTI-LAYER targets involves a two-point movement to allow the reversal of the speed in x from 10 to -10. Initially, the speed in x is set to 0 and the speed y to 8 (best solution for time required and fuel consumed) after which once the x is stabilised at 0 we proceed with a cruise phase which ends when the acceleration of x to the value -10 is required and after a further time the y is decelerated to 0. This is repeated several times if the number of layers should be greater than 2.
- Whenever a target is completed, stitching is performed if necessary and consequently, the total image of the target is compressed to speed up its subsequent download and minimise the data exchanged.

4 Unknown objectives

In the evaluation phase, it would have been possible to complete unknown objectives manually by identifying them from the global map and then creating custom objectives. The solution developed but not made operational due to the limited time for in-depth testing involves a high degree of automation. The acquisition of unknown targets was divided into four macro phases: initial global wide mapping, global remapping, target zone recognition, and custom target creation.

- At a time with no active targets, a global map acquired with a wide lens was created in 1.20 hours consuming about 3 litres of fuel.
- At the activation of an unknown target, it is checked whether there is time for further wide mapping.
- Using OpenCV to lighten the spatial cost, the two mapping images are vertically segmented into four zones.
- For each old and new pair, the following moves are repeated :
  - A new image is created with a white background, the differences between the two images are highlighted with black pixels and then calculate the area of the single or multiple black pixel zones.
  - Each zone will be assigned to a new custom target if not included in known objectives.
  - The custom target will be scheduled together with the normal objectives.
- The developed, functioning and documented functions concerning computer vision can be found in the file unknonws.py, the mapping wide can be found in mappingwide.py.

5 Console

Since most of MELVIN's operation is automated, the main task of the operator console is booking sessions and downloading stitched images of the map and completed objectives.

Some parts of the console itself can be automated with the auto-booker script. The auto-booker can be used to book slots automatically every specified amount of hours and download objectives and map progress. Every time the auto-booker reaches a booked session it tries to connect to MELVIN and books the next available slot if it fails. Once connected, the compressed files of the new objectives and map progress are downloaded and automatically converted to png format. If connection and download are successful a new session is booked after the selected time.

To reduce the amount of bytes transmitted, the map and the objectives are stitched and compressed before being downloaded, so only the required images are downloaded. We use the lzma algorithm which has a good ratio of efficiency to compression ratio.