Automatic Parameter Tuning of Algorithms using Optimization

Joo Pedro Santos Azevedo
joaopsazevedo@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

October 2017

Abstract

The Adaptive Monte Carlo Localization algorithm is a well-known approach for performing the localization of a robot using a particle filter. The Adaptive Monte Carlo Localization have some parameters which boost the performance but at the expense of an increase in computational resources. In this thesis, the multi-objective optimization of the Adaptive Monte Carlo Localization algorithm parameters was performed, using a state of the art algorithm, the Predictive Entropy Search for Multi-objective Bayesian Optimization. In order to do so, a framework for single objective optimization was extended. The added output of this framework is a Pareto front where the user can select the desired resources and get the parameter set that is optimal for that particular setup. One of the objective functions used in the multi-objective optimization is the position error of the robot. Since its computation normally requires expensive tracking hardware, a method for estimating the robot's two dimensional pose error using the Iterative Closest Point algorithm, is proposed and validated.
Keywords: Multi-objective optimization robotics, Adaptive Monte Monte Carlo Localization parameter optimization, Predictive Entropy Search, Iterative Closest Point

1. Introduction

Domestic robots will take part of our daily lives in a near future. From the extraordinary amount of developed algorithms, most of them contain configurable parameters, that in most cases, are manually tuned by the hands of an expert technician. The problem occurs when these algorithms contain a significant amount of configurable parameters, or when the output of these algorithms is not consistent (meaning that repeated evaluations with the same parameters do not yield similar results), or even when its optimization requires expensive hardware not easily accessible to to general public of robotics enthusiasts. Multi objective optimization (MOO) algorithms answer to this problem, by generating a set of optimal solutions that allow the decision maker to analyze the trade offs between objectives and select a final preferred solution. In this case, each one of these optimal solutions will be a set of parameters that will optimize the robot's functionality. It is interesting to study how these multi objective optimizers can be applied to address the parameter optimization problem.

In this thesis, the main goal is to optimize the parameter configuration of algorithms exploring its trade offs in the objective functions, by using MOO algorithms. These algorithms optimize problems involving multiple incommensurable and conflicting objectives. The chosen algorithm to be optimized is Adaptive Monte Carlo Localization (AMCL) algorithm, a stochastic nature algorithm, where to perform a reliable evaluation, the time needed is in the order of minutes. It is also not possible to perform more than one evaluation at one time. Since that the implementation of the AMCL algorithm we want to optimize has 47 parameters, 22 of them configurable, the search space of the chosen algorithm is extremely high. Due to this feature, it is expected that the landscape might be highly multi modal and rough. The majority of the MOO algorithms has not been tested under this situation, so it is also this thesis aim to analyze their performance under this stressful problem.

The objective functions were two: the position error given by the AMCL algorithm and the CPU time consumed by the AMCL process. These criteria have an opposite relationship, meaning that a higher precision of the AMCL estimated position will result in a higher CPU time consumption. This thesis contributes by extending a framework for optimizing robotic algorithms developed in [17], now taking into account more than one evaluation criteria. The framework can be easily be used to optimize other robotic algorithms with any other optimization algorithm. In this thesis, as output of this framework, it is desired a trustful Pareto front.

In the second aim of this work, the important aspect is to remove the need of a ground truth sensor

in order to optimize the AMCL algorithm. In this thesis is proposed a method that uses the recorded laser scan data and the map of the environment to estimate the two dimensional pose error of the robot, allowing users that, for instance, have home made robots using for localization the AMCL algorithm, to optimize their location precision without a try and error process and without the use of expensive and many times unavailable tracking hardware. This can be achieved with an Iterative Closest Point (ICP) approach where the laser sensor data is compared to the map data to get an estimate of the pose error. This thesis contributes with the development of the proposed method, as well as experimental results for its validation.

1.1. Related work

For more than three decades, MOO algorithms have been used in many fields of study. In the aerospace industry, [16] used a Multi Objective Derandomized Evolutionary Strategy (MODES) to optimize an aircraft wing design. An Aircraft Control System Design was also optimized using a multi-objective evolutionary algorithm, presented in [1]. In [4], is developed an hybrid algorithm, combining Nondominated Sorting-based Genetic Algorithm (NSGA)[20] with the Solar Electric Propulsion Trajectory Optimization Program (SEPTOP) in order to optimize low-thrust spacecraft trajectory profiles. In control theory, [15] used NSGA-II [5] to perform a tri-objective optimization of an industrial greenhouse climate control system. In the electromagnetic and antennas field of study, [14] using a Multi objective Particle Swarm Optimizer, designing a time-delay equalizer metasurface for an EBG resonator Antenna. Borg, CMAES [11] and MOEA/D [22], are all three also used to optimize an electromagneitc band gape structure. Other algorithms as ParEGO [13], ϵ-PAL and PESMO [9] were also successfully used in test functions specifically designed to test MOO algorithms.

This thesis multi objective algorithm of choice was PESMO. This choice was made based in the following criteria:

- In contrast to most of the previously mentioned algorithms, PESMO is a model based approach and not an evolutionary algorithm, having a stronger statistical approach.
- PESMO is designed to perform a small amount of evaluation functions, specially adequate for cases where the evaluation is expensive. In contrast, evolutionary algorithms require a much greater amount of function evaluations to yield good results.
- PESMO is tested and compared against ParEGO, SMSego [19], EHI [21] and SUR [18] outperforming all in every study case.
- The computational cost of PESMO grows linearly with the number of objectives, having a lower computational cost than other model based related methods from the literature.

The algorithm we want to optimize is AMCL, implement in the ROS infrastructure[1] and has previously been optimized for a single objective case in [17]. In this case, SMAC [10], a single objective optimization method was used to optimize the position estimate of the AMCL algorithm. This thesis is in fact inspired in [17], aiming to be its extension. Besides the previously mentioned work, in [3], a single objective parameter optimization algorithm, the Particle Swarm Optimization (PSO) [12], was also applied to Monte Carlo Localization.

2. Background

2.1. Adaptive Monte Carlo Localization

MCL is a localization algorithm that estimates the pose (position and orientation) of the robot through a particle filter, where each particle represents a likely state, i.e, a possible robot pose. For 2D localization cases, the state $\mathbf{x}$ of the robot corresponds to a Cartesian reference frame position $[x, y]$ and an angular orientation θ, i.e., $\mathbf{x} = [x, y, \theta]^\intercal$. Let $\mathbf{x_k}$ be the state of the robot at time-step k and $Z^k = \{z_k, i = 1, \ldots, k\}$ the sensor measurements up to the current time-step k. MCL algorithm uses a samling-based method, that consists in, at each time-step k, *represent the posterior density* $p\left(\mathbf{x}_k|Z^k\right)$ *by a set of particles* $S_k = \{s_k^i, i = 1, ..., N\}$ *that are randomly drawn from it*[6][8]. Each particle represents a possible state of the robot.

The efficiency of the MCL algorithm can be greatly improved with an Adaptive (or KLDSampling) approach. The main idea of KLD-Sampling is to adapt the number of samples over time. When the robot location is uncertain, a large number of particles is necessary to cover all the possible states. However, when the robot position is known with low uncertainty, a such large number of particles is unnecessary and inefficient. A detailed explanation on how KLD-Sampling is applied to particle filter can be found in [7].

2.2. Iterative Closest Point

The Iterative Closest Point (ICP) algorithm is used to geometrically match and align two 3D data structures and was first introduced in [2]. This alignment process is also denoted as registration. Basically, each point of the data set that we want to align, also called source data set, is paired with the closest point of the other data set that we wish to align

[1] http://wiki.ros.org/amcl

to, denoted as the model or target data set. An error metric is used to determine how good the alignment is. The process is iteratively repeated until converging and stopping at the desired stop criteria, outputting an optimal rotation and translation that minimizes the error metric between the model data set and the registered data set. For finding the optimal rotation, the singular value decomposition (SVD) method is suggested.

2.3. Predictive Entropy Search for Multi Objective Optimization

The purpose of a MOO is to find the right balance among several objectives. The right balance is defined by the Pareto points, points that provide an optimal design in terms of more than one objective. Another common aim of many MOO is to achieve the first purpose with as few evaluations of the design space as possible.

Formally, an MOO algorithm seeks to simultaneously optimize n objective functions $f_1, \ldots, f_n : \mathcal{X} \to \mathbb{R}$ over a finite design space $\mathcal{X}$. The finite design space is a set of all possible inputs $\mathbf{x}$.

We say that a point $\mathbf{x}$ dominates other point $\mathbf{x}'$ if $f_i(\mathbf{x}) \leq f_i(\mathbf{x}') \, \forall i \in 1, \ldots, n$ and $\exists i \in 1, \ldots, n$ for which $f_i(\mathbf{x}) < f_i(\mathbf{x}')$.

For $\mathbf{x} \in \mathcal{X}$, $\mathbf{f}(\mathbf{x})$ is all the evaluations of the input $\mathbf{x}$ that belongs to the design space, i.e., $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_n(\mathbf{x}))$.

The objective space is then the image $\mathbf{f}(\mathcal{X}) \subset \mathbb{R}^n$. In words, is a subset of $\mathbb{R}^n$, consisting in all the evaluations of all the design space.

The Pareto set $\mathcal{X}^*$ is a subset of the design space $\mathcal{X}$, containing all non-dominated points of $\mathcal{X}$, i.e., the set such that $\forall \mathbf{x}^* \in \mathcal{X}^*, \forall \mathbf{x} \in \mathcal{X}, \exists i \in 1, \ldots n$ for which $f_i(\mathbf{x}^*) < f_i(\mathbf{x})$, i.e., $\mathbf{x}^*$ dominates $\mathbf{x}$.

The Pareto frontier is then the image $\mathbf{f}(\mathcal{X}^*) \subset \mathbf{f}(\chi)$, in words, is a subset of the objective space, consisting in all the evaluations of all the Pareto set.

PESMO is a MOO based on *predictive entropy search*. The idea is to, at each iteration of the algorithm, choose the new evaluation point $\mathbf{x}$ of the design space $\mathcal{X}$, that maximizes the information gained about the Pareto set $\mathcal{X}^*$. Each of the n objective functions f_i for $i \in 1, \ldots, n$ is an unknown black box function so they are probabilistically modeled as a Gaussian process (GP) prior, with i.i.d. zero mean Gaussian observation noise.

A MMO algorithm that has a model based approach needs an acquisition function, i.e., a function that serves as the criteria for choose the next evaluation point of $\mathcal{X}$. Before defining the acquisition function of PESMO, let $D = \{(\mathbf{x}_n, \mathbf{f}(\mathbf{x}_n))\}_{n=1}^{N}$ be a set, with the evaluation point $\mathbf{x}_n$ from $\mathcal{X}$ chosen at the step n and its objective functions evaluation $\mathbf{f}(\mathbf{x}_n)$, at every iteration until iteration N. For PESMO, the acquisition function, known as *entropy search*, is the expected reduction in entropy $H(\cdot)$ of the posterior distribution over the Pareto set $\mathcal{X}$, i.e., $p(\mathcal{X}^*|D)$:

$$\alpha(\mathbf{x}) = H(\mathcal{X}^*|D) - E_{\mathbf{y}}\left[H(\mathcal{X}^*|D \cup \{(\mathbf{x}, \mathbf{y})\})\right] \quad (1)$$

where $\mathbf{y}$ is the value of the a priori independent GP models of the objective functions at $\mathbf{x}$. The expectation $E_{\mathbf{y}}[\cdot]$ is taken with respect to the posterior distribution for $\mathbf{y}$ given the GP models, $p(\mathbf{y}|D, \mathbf{x}) = \prod_{i=1}^{n} p(y_i|D, \mathbf{x})$. The next acquisition point $\mathbf{x}_{N+1}$ is the one that maximizes the acquisition function, i.e., $\mathbf{x}_{N+1} = \text{argmax}_{x \in \mathcal{X}} \, \alpha(\mathbf{x})$.

The acquisition function is however, unfeasible because it is difficult to evaluate the entropy of the Pareto set $\mathcal{X}^*$ given D, i.e., $H(\mathcal{X}^*|D)$ in equation 1. So it is approximated to the *predictive entropy search*:

$$\alpha(\mathbf{x}) = H(\mathbf{y}|D, \mathbf{x}) - E_{\mathcal{X}^*}\left[H(\mathbf{y}|D, \mathbf{x}, \mathcal{X}^*)\right] \quad (2)$$

This new acquisition function is formulated from equation 1 due to the principle of mutual information. For this case, the information known about $\mathbf{y}$ is used to gain information about $\mathcal{X}^*$, so in equation 1, the roles of $\mathcal{X}^*$ and $\mathbf{y}$ are exchanged, leading to the equation 2. Now, the first term of equation 2 is the entropy of the predictive distribution $p(\mathbf{y}|D, \mathbf{x})$ and the expectation $E_{\mathcal{X}^*}(\cdot)$ in the second term is taken with respect to the posterior distribution for the Pareto set $\mathcal{X}^*$ given the observed data D. $H(\mathbf{y}|D, \mathbf{x}, \mathcal{X}^*)$ is the entropy of $P(\mathbf{y}|D, \mathbf{x}, \mathcal{X}^*)$ which is the predictive distribution of the objective function at $\mathbf{x}$ given the collected data D conditioned to the Pareto set $\mathcal{X}^*$. The deduction of how to evaluate both terms of equation 2 can be found in detail in [9]

3. System Architecture

A valuable aspect of the proposed system architecture is that all modules were designed to be generic and completely independent from each other, so the user would only have to construct the link between the proposed architecture and an optimizer of his choice. The constructed modules were implemented as ROS packages and can easily be used or adapted for other study scenarios. In figure 1 is shown a diagram with a global picture of the developed system architecture. This architecture was inspired in a more generic automatic parameter optimization pseudo code, that can be found in [17].

The chosen third party MOO software was PESMO which is developed in Python. PESMO is fed with a configuration file where is discriminated the objective functions, the parameters of the algorithm we want to optimize, the parameters domain and the maximum number of the objective functions evaluations. The objective functions

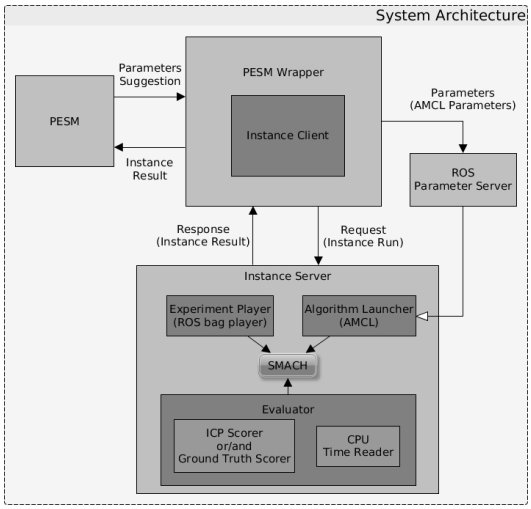

Figure 1: Diagram of the developed system architecture

are the CPU time consumption of the AMCL algorithm (measured in the number of ticks the CPU was used for processing the AMCL algorithm process) and the mean euclidean distance error between robot's pose given by AMCL and the ground truth pose given by the mo-cap system. PESMO starts by suggesting a set of AMCL parameters to evaluate. These parameters are then loaded into the ROS parameter server by the PESMO wrapper. The wrapper also triggers the instance server by requesting an experiment run with the suggested parameters, and waits for a response containing the objective functions results which are then forwarded to PESMO. The experiment run consists in playing the recorded robot sensor data and evaluate its AMCL position error as well as the CPU time consumption by AMCL. PESMO analysis the results, beginning a new iteration with a new promising parameter set suggestion. The optimization stops when the maximum number of experiment evaluation is reached. The output of the system will be all evaluated parameter sets with their correspondent scores. From this, one can easily plot the achieved Pareto frontier and choose a parameter set that best fits our needs.

As mentioned before, the evaluation functions are the CPU time consumption of the AMCL algorithm and the euclidean distance between the robot's AMCL pose and the ground truth pose. Since the ground truth pose is acquired by a mo-cap system, it has also been developed a new approach on how to estimate this error using only the recorded data. This approach is discrebed by the next three subsections.

3.1. Occupancy Grid to Point Cloud Conversion

The primary content of this package is a method to convert an occupancy grid (a map in this case) into a point cloud. To transform each cell of the occupancy grid into a map coordinate, firstly we need to know the grid height and width. Since the grid data is organized in row-major order, its width and height will tell us how to organize the point cloud in terms of rows and columns. The grid resolution will tells us how much a cell measures in the real world coordinates and the grid origin position will tell us where in the real world coordinates, the map is centered. This method can be described by the following algorithm:

Algorithm 1 Occupancy Grid to Point Cloud Conversion Algorithm

procedure CELLSTOCOORDINATES
  $w \leftarrow$ grid width ▷ In *cells*
  $h \leftarrow$ grid height ▷ In *cells*
  $x \leftarrow$ grid origin x coordinate ▷ In m
  $y \leftarrow$ grid origin y coordinate ▷ In m
  $r \leftarrow$ grid resolution ▷ In $m/cell$
  $c \leftarrow$ occupancy value threshold
  $d \leftarrow$ grid data array ▷ Contains the occupancy probability of each cell
  for $j \leftarrow 1, h$ **do** ▷ Iterate through the number of rows
    for $i \leftarrow 1, w$ **do** ▷ Iterate through the number of columns
      if $d[w * j + i] \geq c$ **then**
        $p_x \leftarrow x + i * r + r/2$ ▷ World map x coordinate
        $p_y \leftarrow y + j * r + r/2$ ▷ World map y coordinate
        $p_z \leftarrow 0$
        $PCL \leftarrow (p_x, p_y, p_z)$ ▷ Add p_x, p_y and p_z to the point cloud PCL
      end if
    end for
  end for
  return PCL
end procedure

3.2. Laser Scan to Point Cloud Conversion

In this module, one or multiple laser scans are converted into a point cloud. In order to do so, the ROS package *laser_geometry*[2], a package already developed, documented and publicly available in ROS repository is used. The *laser_geometry* package makes use of the *tf*[3] package that lets the user keep track of multiple robot coordinate frames over time. With this, each individual laser ray is appropriately converted, which is fundamental for a precise localization specially in tilting laser scanners

[2] http://wiki.ros.org/laser_geometry
[3] http://wiki.ros.org/tf

and moving robots. The main difference between *laser_geometry* and the current work implementation is the extension for receiving and publishing trigger events, essential for controlling when this modules start and stops doing its purpose.

3.3. Point Cloud ICP Scorer

In this module, two points clouds are received and a spatial transformation that best align them is found. In the alignment process, using the ICP algorithm, is given a fitness score. If this score is lower than the user defined threshold, an estimation of the pose error is computed, so this process can be used as objective function in the MOO. The ICP algorithm is implement using he Point Cloud Library, a standalone, large scale, open project for 2D/3D image and point cloud processing [4]. The output of this package is therefore, the alignment fitness score, the estimated pose error both as an euclidean distance error and an angular error, the registered point cloud and the transformation that aligned both clouds (and transforms the source cloud into the registered cloud). In figure 2 is shown a diagram representing the proposed evaluation method.

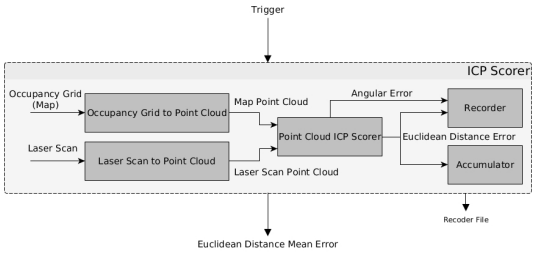

Figure 2: Diagram of the proposed ICP scorer method

4. Experiment Setup

The experiment consists in teleoperating the robot around an apartment-like test bed that was previously mapped. The robot movement consists in both linear and angular movements in order to simulate realistic robot movements and to have a more reliable error evaluation of the daily tasks these robots are being developed for. Firstly, an estimate of the ground truth pose of the robot provided by the mo-cap system is given to the AMCL in order to initialize the filter. Then, while the robot is being teleoperated around the test bed, the real robot pose given by the mo-cap system, the estimated AMCL pose and the laser and odometry readings are recorded into a data set. Since this data set will later be used for offline multi objective optimization of the AMCL parameters, additional data required to correctly offline rerun the AMCL algorithm was also recorded. The recorded data is then used to optimize the AMCL algorithm with the developed system. This recorded data has enough duration for being cropped into multiple parts. The idea is to have multiple data sets with different movements so one can tune AMCL in one training data set, and evaluate the tune robustness in another data set different from the training set.

The AMCL algorithm implemented in ROS has a total of 47 parameters that can be customized. Some of these 47 parameters are not necessary to tune since they are related to, for instance, data for visualization purposes or the initial position of the robot. We end up with a subset of 22 configurable parameters shown in table 1. In this table is also shown the default parameter set and minimum and maximum value of each parameter.

Table 1: AMCL parameters configuration

	default	min	max
laser_max_beams	30	5	100
min_particles	100	5	980
max_particles	5000	1000	10000
kld_err	0.01	0.005	0.5
kld_z	0.99	0.5	0.999
odom_alpha1	0.005	0.001	1
odom_alpha2	0.005	0.001	1
odom_alpha3	0.010	0.001	1
odom_alpha4	0.005	0.001	1
odom_alpha5	0.003	0.001	1
laser_z_hit	0.95	0.1	5
laser_z_short	0.1	0.05	1
laser_z_max	0.05	0.005	1
laser_z_rand	0.05	0.005	1
laser_sigma_hit	0.2	0.05	1
laser_lambda_short	0.1	0.005	1
laser_likelihood_max_dist	2.0	0.1	20
update_min_d	0.2	0.05	0.5
update_min_a	0.5236	0.1	1
resample_interval	2	1	3
recovery_alpha_slow	0.01	0.001	0.01
recovery_alpha_fast	0.1	0.05	1

A complete description of all AMCL parameters can be found in the AMCL ROS wiki web page[5].

In table 2 are show the chosen ICP parameters, based on a repeated visual analyses of the alignment of two point clouds, done before the MOO.

Table 2: ICP parameters configuration

	type	value
max_correspondance_distance	float	2.5
max_iterations	integer	200
transformation_epsilon	float	0.00000001
euclidean_fitness_epsilon	float	1

[4] http://pointclouds.org/

[5] http://wiki.ros.org/amcl

Finally, in table 3 it is shown the chosen PESMO parameters.

Table 3: PESMO parameters configuration

	type	value
grid_size	integer	1000
max_finished_jobs	integer	100
likelihood	string	GAUSSIAN

5. Results

5.1. Ground Truth Pareto Frontier

The results of the MOO of AMCL are shown in figure 3, already zoomed, showing only the region of interest, the Pareto frontier. For this solution, the position error was computed based on the ground truth mo-cap data.

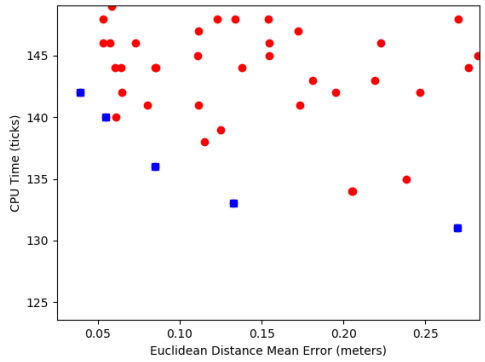

Figure 3: Pareto frontier Zoomed: Ground truth position error - CPU time

The Pareto frontier points and its associated objective functions evaluations are listed in table 4.

Table 4: Pareto frontier points. CPU Time vs Euclidean Distance Ground Truth Error

Point	**1**	**2**	**3**	**4**	**5**
Error GT	0.039	0.055	0.085	0.133	0.270
CPU Time	142	140	136	133	131

As table 4 indicates, a Pareto frontier with 5 points was found. The position error ranges from 0.039 m (3.9 cm) to 0.270 m (27.0 cm) and the CPU time consumption ranges from 131 ticks to 142 ticks. This results are acceptable since the first three points have an error below 9 cm, which are in fact usable parameter configurations. It is notable that the CPU time consumption of the Pareto front are very similar form one another. This rises the question on how conservative were the the limits applied to the design space listed under table 1, preventing the evaluations of certain regions of space.

In order to validate and understand how consistent these parameter sets from the Pareto frontier are, each Pareto point is reevaluated 25 times with a different recorded data. With this, we can verify how these parameter sets behave in situations different from the training data. It is also convenient to verify, how the *Point Cloud ICP Scorer* would predict the position error for this Pareto points. Table 5 list the mean, standard deviation and RMSE results of evaluation the previous Pareto points 25 times with the test data. At bold are the previously shown Pareto points obtained with PESMO.

Table 5: Ground truth Pareto frontier validation results

Point	**1**	**2**	**3**	**4**	**5**
Error GT	**0.039**	**0.055**	**0.085**	**0.133**	**0.270**
Mean GT	0.039	0.054	0.084	0.130	0.269
Mean ICP	0.061	0.072	0.081	0.165	0.246
Std GT	0.000	0.001	0.001	0.004	0.015
Std ICP	0.001	0.002	0.002	0.004	0.012
RMSE GT	0.000	0.002	0.002	0.005	0.015
RMSE ICP	0.022	0.017	0.004	0.033	0.026
CPU Time	**142**	**140**	**136**	**133**	**131**
Mean	146	146.96	146.08	136	133.32
Std	2.236	4.521	3.224	2.905	2.328
RMSE	5.441	8.299	10.583	4.205	3.286

Analyzing the previous table, firstly, the ground truth position error is very consistent, with a maximum difference of 0.003 m (3 mm) in point 4. With this results, one can be confident in saying that using the mo-cap system for evaluating the pose error, and using this evaluation as an objective function in single or MOO algorithms yields good results. Secondly, comparing the ground truth results to the *Point Cloud ICP Score* results, we see that the estimated position error had a maximum difference of 0.032 m (3.2 cm) in point 4 and a minimum difference of 0.003 m (3 mm) in point 3. The standard deviation and the RMSE of both previous evaluation methods are in the order of millimeters, suggesting a consistency of the values along the 25 evaluations.

5.2. Point Cloud ICP Scorer Pareto Frontier

The same analysis done in section 5.1 is now applied for the results of the MOO of AMCL now with the *Point Cloud ICP Scorer* method as evaluation function of the position error. The highlighting of the Pareto frontier is presented in figure 4.

The Pareto frontier points and the respective objective functions evaluations are listed in table 6.

Table 6: Pareto frontier points. CPU Time - Estimated Euclidean Distance Error

Point	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Error ICP	0.049	0.063	0.067	0.091	0.102	0.108	0.242
CPU Time	162	149	146	140	138	135	130

As table 6 indicates, PESMO was also able of finding a Pareto frontier with these objective func-

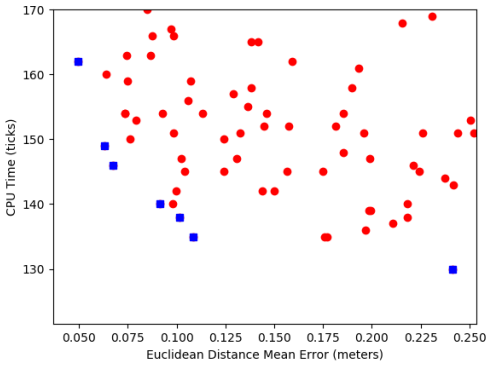

Figure 4: Pareto frontier zoomed: Estimated position error - CPU time

tions, now with 7 points. The position error ranges from 0.049 m (4.9 cm) to 0.242 m (24.2 cm) and the CPU time consumption ranges from 130 ticks to 162 ticks. Comparing with the results yielded in the previous Pareto frontier, the minimum euclidean distance error was 1 cm bigger (4.9 cm instead of 3.9 cm) and the maximum euclidean distance error was 2.8 cm smaller (24.2 cm instead of 27 cm). For the CPU time, the bigger difference was that the point with minimum position error now has a bigger CPU time. This difference was of 20 ticks (162 ticks instead of 142 ticks).

The goal now is to validate and understand how consistent these parameter sets from the Pareto frontier are. Again, each Pareto point is reevaluated 25 times with a different recorded data. The idea is also to verify how would the mo-cap system evaluate these points, and compare it to the developed method. This is exactly what was done in the previous section, but in an opposite order, resulting in a cross check validation. Table 7 lists the mean, standard deviation and root mean square error results of evaluation the previous Pareto points 25 times with the test data. At bold are the previously shown Pareto points obtained with PESMO.

Table 7: Estimated Pareto frontier validation results

Point	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Error ICP	0.049	0.063	0.067	0.091	0.102	0.108	0.242
Mean GT	0.094	0.096	0.092	0.077	0.072	0.136	0.258
Mean ICP	0.050	0.065	0.067	0.098	0.104	0.115	0.245
Std GT	0.002	0.002	0.002	0.003	0.007	0.004	0.02
Std ICP	0.001	0.002	0.001	0.004	0.004	0.003	0.019
RMSE GT	0.044	0.033	0.025	0.014	0.030	0.028	0.026
RMSE ICP	0.001	0.002	0.001	0.008	0.005	0.007	0.019
CPU Time	162	149	146	140	138	135	130
Mean	164	151.44	153.48	139.56	138.96	142.6	138.4
Std	3.572	4.167	4.215	2.547	2.891	2.993	5.838
RMSE	4.238	4.829	6.151	2.585	3.046	8.168	10.229

The estimated position error is again very consistent, with millimetric order in standard deviation and RMSE and with a maximum difference of 0.007 m (7 mm) in point 4 and 6. Comparing these results to the respective ground truth values, we see that there was a maximum difference of 0.045 m (4.5 cm) for point 1. Checking this difference for the remaining points, we conclude that the proposed method was not able to find a completely true Pareto frontier. *Point Cloud ICP Scorer* predicted that point 1 would have the lowest position error, while truly it was point 5 that had the lowest position error. It is also noted that the three estimations of the lower position error (point 1, 2 and 3), were all under estimations. As for points 4 and 5 (that generally would still be considered usable given the estimated error), they were overestimated. As expected, and even though that in this case the Pareto points had a maximum prediction error of 4.5 cm, we conclude that for the position error evaluation, a ground truth sensor data would be the best option, however, with the absent of it, the proposed *Point Cloud ICP Scorer* will give very similar results.

5.3. Stronger Pareto Frontier

The achieved Pareto frontiers are compromised by the very stochastic nature of the CPU time consumption by the AMCL algorithm. To surpass this step back, a new optimization was done, now reevaluating 10 times each suggested point by PESMO. Since each job takes at least 30 seconds to conclude, plus the time taken by PESMO to perform the optimization process, the number of reevaluations was 10 simply to avoid a extremely high optimization time. The goal of this approach is to suppress the wrong prediction of the CPU time consumption, achieving a more reliable Pareto frontier, both when the position is computed based on ground truth sensor or when is estimated by the *Point Cloud ICP Scorer*. Since it was already verified that the acquisition of the position error using the mo-cap system yields very good results, only the *Point Cloud ICP Scorer* method was used for computing this new stronger and more faithful Pareto front. The obtained results are shown in figure 5 with a highlight of the achieved Pareto frontier.

The Pareto points and respective objective functions evaluations are listed in table 8.

Table 8: Stronger Pareto frontier points. CPU time - Estimated Euclidean Distance Error

Point	**1**	**2**	**3**	**4**	**5**	**6**
Error ICP	0.050	0.076	0.077	0.094	0.106	0.248
CPU Time	138	135.9	124.6	122.9	122.2	117.7

Table 8 shows that 6 Pareto points were found. The estimated position error ranges between 0.050 m (5 cm) and 0.248 m (24.8 cm), while the CPU time consumption ranges from 117.7 to 138 ticks.

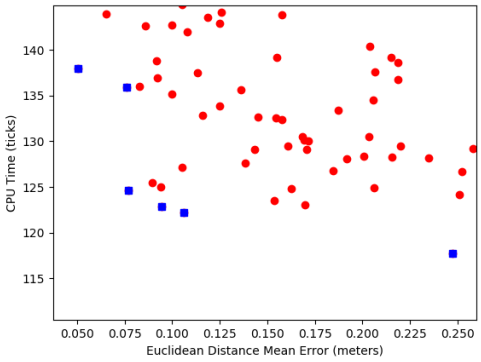

Figure 5: Stronger Pareto frontier zoomed: Estimated position error - CPU time

The range of values for the estimated euclidean distance error is very similar to the ones obtained in the previous Pareto frontier, which might evidence a lower and upper bound on the position error that the proposed method can compute. The lower bound is associated to the position error that will be always present in AMCL. The upper bound can be associated to the previously noticed limitation of the developed method, of not being able to compute a reliable position error estimation for high true position errors. When it comes to the range of values for the CPU time consumption, these were smaller than the one found in the previous Pareto frontier. This might be a result of reevaluating 10 times each suggested point by PESMO.

In order to check how more reliable this new Pareto frontier is, each Pareto point is again reevaluated 25 times with a testing data. Table 9 lists the mean, standard deviation and RMSE results of evaluation each Pareto point 25 times with the training data.

Table 9: Estimated stronger Pareto frontier validation results

Point	**1**	**2**	**3**	**4**	**5**	**6**
Error ICP	0.050	0.076	0.077	0.094	0.106	0.248
Mean GT	0.135	0.068	0.092	0.101	0.226	0.194
Mean ICP	0.15	0.057	0.105	0.089	0.183	0.232
Std GT	0.002	0.002	0.003	0.001	0.003	0.006
Std ICP	0.004	0.057	0.002	0.002	0.002	0.005
RMSE GT	0.085	0.008	0.015	0.006	0.120	0.053
RMSE ICP	0.100	0.019	0.028	0.005	0.076	0.016
CPU Time	138	135.9	124.6	122.84	122.2	117.7
Mean	152.08	152.96	138.92	139.84	136.00	133.40
Std	2.712	2.441	3.006	2.648	3.286	4.118
RMSE	14.339	17.234	14.632	17.146	14.186	16.231

As we seen in the previous sections, the values obtained for the estimated position error are very consistent between the 25 evaluation, with low standard deviations and RMSEs. Taking a look in the *Mean GT* and *Mean ICP* values, that are the mean values of reevaluating the position error 25 times using the ground truth sensor and the *Point Cloud ICP Scorer* package, we notice that, for the same point, they are very close to each other, with a minimum difference of 0.011 m (1.1 cm) in point 2 and a maximum difference of 0.043 m (4.3 cm) in point 5. It confirms once again, that the *Point Cloud ICP Scorer* method will give very similar results to the ground truth sensor. It is also noticeable however, that the first point of the Pareto frontier was extremely over fitted to the training data, since in training it yielded an estimated error of 0.050 m (5 cm), and in testing it yielded an estimated error of 0.15 m (15 cm) and a true error of 0.135 m (13.5 cm). This result, invalidated this Pareto point.

6. Conclusions

From the experimental results of this theses, the following conclusions are taken:

- The chosen MOO algorithm PESMO was able to find a Pareto front of the studied problem. The Pareto front presented a large range of values, specially in terms of the mean euclidean distance error which ranged from very usable values of 0.004 m (4 mm) to values larger than 0.30 m (30 cm). For the CPU time consumption, there was only a difference of up to 40 ticks between the Pareto points with the best a worst mean euclidean distance error. This low difference, might raise the question if the chosen design space was to limited, restricting PESMO of finding Pareto points with low CPU time consumption. It's noticeable however, that for this case, finding these points with lower CPU time consumption might lead to mean euclidean distance error values no longer usable, since in general, the user is interested in position error values below 10 cm.

- The methods used for computing the position error are both very consistent when repeated evaluations are performed. The CPU time consumption results are not as consistent due to the adaptive method of the AMCL, which changes the amount of particles used from one evaluation to another. Due to this aspect and in order to find a more reliable Pareto frontier, it is important to reevaluate each suggestion given by PESMO multiple times and average the results.

- PESMO was also able of finding Pareto points that dominate the default parameter set in CPU time consumption. Using the estimated position error as objective function, a Pareto point with 0.068 m (6.8 mm) of mean true position error and 135.9 ticks of CPU time consumption was found, dominating in terms

of CPU time the default parameter set by 24 ticks, but being dominated by 8 mm in terms of position error. Also all Pareto points found outperform the CPU time consumption of the default parameter set.

7. Achievements

The main purpose of this work was to design a system that would automatically tune an algorithm in respect to multiple objectives and provide the user freedom to choose the parameters that would best fit the his needs. An already developed framework was extended for also perform MOO. PESMO algorithm was implemented in this framework as is easily used to optimize other algorithms. An optimization of the AMCL algorithm with respect to the position error and CPU time consumption was executed, outperforming the default parameter set in one objective function.

For the second goal of the work, it was proposed and validated a method to estimate the AMCL position error without the use of external tracking hardware. This developed method presented results very similar to the ground truth values, being considered very usable in most situation. This method still has room for improvement.

8. Future Work

A proposal for future work would be the optimization of the ICP algorithm to get better results out of the *Point Cloud ICP Scorer*. This proposed package could be extended to not only estimate the pose error in terms of euclidean distance and orientation angle, but to also decompose the position error in x and y Cartesian components. This is useful to compute an estimation of the pose correction one needs to apply to the pose given by the AMCL. With this pose correction, one could implement a method to, when the position error is above certain threshold, send a pose estimation to AMCL, restarting the filter process. If this correction is good enough, AMCL would be able recover the robot's position when he is lost. One application could be, for instance, to solve the kidnap problem.

Lastly, in terms of MOO, it is suggested to study how, not only PESMO but also other MOO algorithms, behave in the optimization of algorithm parameters with more than two objective function. An idea would be to optimize the energy consumption of the robot. Since PESMO presented good results in this problem with a design space of high dimension, something that had never been tested before, it is also interesting to check if for lower dimensional design spaces the results would be better.

Acknowledgements

First and the most importantly, I would like thank my family including my girlfriend for the love and continuous support they have always offered until today. A special thanks to my mother, father and grandparents for always giving me the very best they can offer.

Also a special thanks to professor Rodrigo Ventura for guiding me through this thesis, and for the opportunity of being part of the SocRob team that I am very grateful for. I also have to thank Oscar Lima for the patience and formidable provided lessons that were very useful for this thesis.

References

[1] S. F. Adra, A. I. Hamody, I. Griffin, and P. J. Fleming. A hybrid multi-objective evolutionary algorithm using an inverse neural network for aircraft control system design. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 1–8 Vol.1, Sept 2005.

[2] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb 1992.

[3] A. Burchardt, T. Laue, and T. Röfer. *Optimizing Particle Filter Parameters for Self-localization*, pages 145–156. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[4] V. Coverstone, J. Hartmann, and W. Mason. Optimal multi-objective low-thrust spacecraft trajectories. 186:387–402, 06 2000.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, Apr 2002.

[6] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 2, pages 1322–1328 vol.2, 1999.

[7] D. Fox. Kld-sampling: Adaptive particle filters. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 713–720. 2002.

[8] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'99).*, July 1999.

[9] D. Hernández-Lobato, J. M. Hernández-Lobato, A. Shah, and R. P. Adams. Predictive entropy search for multi-objective bayesian optimization. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1492–1501. JMLR.org, 2016.

[10] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration (extended version). Technical Report TR-2010-10, University of British Columbia, Department of Computer Science, 2010. Available online: http://www.cs.ubc.ca/˜hutter/papers/10-TR-SMAC.pdf.

[11] C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evol. Comput.*, 15(1):1–28, Mar. 2007.

[12] J. Kennedy. *Particle Swarm Optimization*, pages 760–766. Springer US, Boston, MA, 2010.

[13] J. Knowles. Parego: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, Feb 2006.

[14] A. Lalbakhsh, M. U. Afzal, and K. P. Esselle. Multiobjective particle swarm optimization to design a time-delay equalizer metasurface for an electromagnetic band-gap resonator antenna. *IEEE Antennas and Wireless Propagation Letters*, 16:912–915, 2017.

[15] M. Mahdavian, S. Sudeng, and N. Wattanapongsakorn. Multi-objective optimization and decision making for greenhouse climate control system. In *2016 International Conference on Information Science and Security (ICISS)*, pages 1–5, Dec 2016.

[16] B. Naujoks, L. Willmes, T. Bäck, and W. Haase. *Evaluating Multi-criteria Evolutionary Algorithms for Airfoil Optimisation*, pages 841–850. Springer Berlin Heidelberg, 2002.

[17] R. V. Oscar Lima. A case study on automtic parameter optimization of a mobile robot localization algorithm. Instituto Superior Tcnico, Universidade de Lisboa, 2016.

[18] V. Picheny. Multiobjective optimization using gaussian process emulators via stepwise uncertainty reduction. *Statistics and Computing*, 25(6):1265–1280, Nov 2015.

[19] W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze. *Multiobjective Optimization on a Limited Budget of Evaluations Using Model-Assisted-Metric Selection*, pages 784–794. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[20] N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[21] T. Wagner, M. Emmerich, A. Deutz, and W. Ponweiser. *On Expected-Improvement Criteria for Model-based Multi-objective Optimization*, pages 718–727. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[22] Q. Zhang and H. Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, Dec 2007.