Unified cooperative localization and tracking in micro aerial vehicles

Pedro Miguel Vieira de Faria
pedro.m.faria@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal
November 2020

Abstract

In this work, it is proposed a solution to solve the full Cooperative Localization and Tracking (CLT) problem. The full CLT problem consists in simultaneously self-localizing multiple robots and tracking multiple targets. The solution proposed uses an online, decentralized and unified approach based on particle filters and is an improvement of an already existing algorithm named PF-UCLT. In the developed algorithm, PF-UCLT is generalized to the case of multiple micro aerial vehicles - instead of ground-robots - tracking multiple objects - instead of a single object. The developed algorithm is scalable with both the number of robots and number of objects being tracked. This method does not need an increase in the number of particles with respect to the number of robots and targets, only of the number of robot and target subparticles. The space and time complexity requirements growth is then reduced from exponential to linear with respect to both the number of robots and targets in order to maintain a given level of accuracy in the full state estimation. Through a set of experiments on a large number of randomized datasets it is demonstrated the correctness and efficiency of the developed algorithm, showing that it can be a potential algorithm for everyday use in multi-robot scenarios.

Keywords: cooperative localization, cooperative tracking, micro aerial vehicles, particle filter, sensor fusion

1. Introduction

In the past few years, multi-robot systems have become an active research area [30]. Compared to single robot approaches [33, 24], multi-robot systems have greater performance and versatility [26, 38, 12]. This type of systems has, therefore, been being used in robot self-localization and object tracking [12, 22, 2]. The idea behind this approach of Cooperative Localization and Tracking (CLT) is to share information among robots - of their odometry readings and their observations - and enhance their individual beliefs. In this work this approach will be used.

A recent online and decentralized method that tackles the CLT problem in a unified fashion is reviewed and improved theoretically. This method, Particle Filter for Unified Cooperative Localization and Tracking (PF-UCLT) [2], performs with the assistance of a known map of landmarks the localization and object tracking processes in a single step, preventing the recursive propagation of estimation error between them.

The PF-UCLT algorithm scales well with the number of robots in the team, however, it does not with the number of objects being tracked. This fact makes it unusable to track online any number of relevant objects. Hence, in this work, it is proposed a solution to solve this problem, and allow the online tracking of multiple objects. The improved developed algorithm will be named PF-UCLT+ (Particle Filter for Unified Cooperative Localization and Tracking Extended).

The developed algorithm will be implemented in multiple Micro Aerial Vehicles (MAVs). The MAVs will track one or more objects moving randomly using sensors to localize themselves and the targets in their field of view. This is another extension of the PF-UCLT algorithm, since the aforementioned one was implemented on ground robots (movement in 2-D space).

2. Literature Review

The target tracking problem is now a mature field of research [37, 16, 14]. Over the years different methods, coming from a variety of fields of study, were developed to tackle this problem: from single to multiple robots tracking one or more objects [26], to approaches that use known maps of the environment to improve the tracking performance [18].

Particle filters [26, 34, 15] are one the most interesting solutions to the tracking problem due to both its multi-modal and non-parametric form - they are suitable for scenarios where the target’s motion model is unpredictable or changes to a different model over time - and its tractability in high-dimensional problems.

For the single-robot single-target problem many efficient solutions using particle filters were developed [15, 18, 19]. However, a single-robot approach has some limitations when compared with a multi-robot/multi-sensor one, namely the limited range of the sensors and occlusions to that single-robot.

To bypass some of those problems, recently, multi-
robot particle filter approaches have been gaining attention. One of these approaches is a decentralized particle filter-based technique [26, 27], in which inherent issues with multi-robot cooperative object tracking, namely, limited communication bandwidth, are addressed. Other explored approaches include sharing target-related world-frame constraint relations among the robots [12], or sharing reduced sets of particles among the robots [25]. The problem of multiple moving targets using a single moving platform was handled in [6]. However, in the developed work a more general situation is taken care of: multiple moving objects tracked by multiple moving robots, which at the same time, also localize themselves.

The cooperative multi-robot localization problem is also a wide field of research [10, 13, 20, 23, 29]. This problem has been addressed from several viewpoints and using numerous techniques. One of them is using inter-robot measurements, i.e., communicating the relative positions between robots and treating them as observation measurements in a filtering algorithm where the states include the poses of all robots [10, 11, 32]. This problem was attacked using a variety of methods, for example, Roumeliotis and Bekey [31] used a Kalman filter, whereas Rekleitis et al. [29] used a particle filter, and Bailey et al. [4] presented a solution which overcomes the recursive propagation of errors by centralizing the cooperative estimation. However, all these approaches that use inter-robot measurements, can be troublesome, as, in order to obtain inter-robot measurements, more sensors may be needed to not only obtain the distance/angle to other robots, but also their IDs. Most measurements also became strongly correlated which leads to a different class of cooperative localization problem, and exploitations of conditional and mutual independence proprieties that are used in this work could not be done.

Beyond inter-robot measurements, other techniques in cooperative localization were developed. For instance, communicating environment static features (landmarks) information to robots that might not be able to observe those themselves due to occlusions/distance [9, 17], or sharing information among the robots regarding a common observable object by all robots [21]. In this last case, the object is also being tracked simultaneously by all robots and, therefore, both the cooperative localization problem and cooperative tracking problem can benefit from each other in a unified fashion.

Simultaneous robot Localization and Object Tracking (SLOT) for single-robot applications has attained a widespread attention [24, 33]. However, in multi-robot scenarios, this problem has proven to be difficult to solve, due to the substantial increase in computational complexity when the state space dimensionality increases [5, 8]. This dilemma was addressed by Zhou and Roumeliotis [38], who succeed in reducing the computational complexity from exponential to linear with respect to (w.r.t.) the number of robots by using the iterative algorithm Gauss-Seidel relaxation. Still, their approach only tackled the case of cooperative tracking of a target (they didn’t address the robot localization problem). In contrast, PF-UCLT, the method that this work will extend, cooperatively estimates both the localization of the robots and the localization of a single target in a unified and efficient fashion (linear computational complexity w.r.t. the number of robots). In the developed algorithm, PF-UCLT+, the computational complexity w.r.t. the number of objects tracked is also reduced from exponential to linear. Other methods, not based on particle filters, were developed to tackle the CLT problem. For example, Chang et al. proposed an Extended Kalman Filter (EKF)-based approach [7]; nevertheless it was shown in [2] that the PF-UCLT algorithm consistently outperformed the EKF method, leveraging the non-linearity of particle filters. An offline method was also proposed by Ahmad et al. in [3]; their approach is based on non-linear least square minimization. It was also shown in [2] that the PF-UCLT algorithm has a comparable performance with the previous method, while having the considerable advantage of being online. More recently, an online method based on moving-horizon nonlinear least squares was also developed [1]. In this method the SLOT problem is addressed, but only for single object tracking.

3. Background

3.1. Particle Filter

A particle filter, also known as sequential Monte Carlo method, is a non-parametric implementation of the Bayes filter. The Bayes filter is a recursive algorithm used to calculate beliefs. A belief reflects the robot’s internal knowledge about the state of the environment. A belief distribution assigns a probability to each possible hypothesis with regards to the true state. Belief distributions are posterior probabilities over state variables conditioned on the available data [36].

A standard particle filter approximates the posterior by a finite number of parameters. The key idea is to represent the posterior belief by a set of $M$ state samples (particles) drawn from this posterior [36]. Because the distribution is represented in a non-parametric form, this technique can represent a broader space of distributions than, for example, Gaussians.

The most basic variant of the particle filter algorithm is stated in Algorithm 1 [36]. The inputs of the algorithm are the particle set of the previous iteration $X_{t-1}$, along with the most recent control $u_t$ and the most recent measurement $z_t$. In line 3, a hypothetical state $x^{[m]}_t$ for time $t$ is generated based on the particle $x^{[m]}_{t-1}$ and the control $u_t$ (prediction step). In line 4 it is calculated the importance weight $w^{[m]}_t$ of each
particle \( x_t^{[m]} \). The importance weight is the probability of the measurement \( z_t \) under the particle \( x_t^{[m]} \) (update step). Lines 7-10 are the resample step of the algorithm. The algorithm draws with replacement \( M \) particles from the temporary set \( \tilde{X}_t \). The probability of drawing each particle is given by its importance weight. Resampling transforms a particle set of \( M \) particles into another particle set of the same size, and the particles with lower importance weights tend to be eliminated, surviving the particles with higher weight.

Algorithm 1 Particle_filter \((X_{t-1}, u_t, z_t)\)

1: \( \tilde{X}_t = X_t = \emptyset \)
2: for \( m = 1 \) to \( M \) do
3: Sample \( x_t^{[m]} \sim p(z_t|u_t, x_{t-1}^{[m]}) \)
4: \( w_t^{[m]} = p(z_t|x_t^{[m]}) \)
5: \( \tilde{X}_t = \tilde{X}_t + (x_t^{[m]}, w_t^{[m]}) \)
6: end for
7: for \( m = 1 \) to \( M \) do
8: Draw \( i \) with probability \( \propto w_t^{[i]} \)
9: Add \( x_t^{[i]} \) to \( \tilde{X}_t \)
10: end for
11: Return \( \tilde{X}_t \)

The performance of a standard particle filter is heavily dependent on the number of particles [8, 28]. In order to achieve a good approximation of the posterior belief, the number of particles must increase exponentially with the dimension of the state space represented by a particle [35]. Otherwise, the particle filter tends to fall into the particle deprivation problem [36] (situation where none of the particles are in the vicinity of the correct state).

Considering the case of a single robot localization, the required number of particles to achieve a given accuracy level in the localization estimates by a particle filter-based method is \( M \) (usually \( M \) in the order of thousands gives results with acceptable accuracy and computational speed). However, when the state space consists of poses of \( N \) robots tracking \( K \) objects, the number of particles required to achieve the same level of accuracy as the one obtained by \( M \) particles in the case of a single-robot localization must be increased to \( M^{(N+K)} \). This makes the usage of a standard particle filter to solve the CLT problem impractical. Even with \( N = 2 \) and \( K = 1 \), the most simple case for CLT, the required number of particles will be in the order of millions.

3.2. Cooperative Localization and Tracking (CLT) Problem Formulation

Consider a team of \( N \) robots \( r_1, ..., r_N \) tracking \( K \) objects \( o_1, ..., o_K \) in an environment with \( L \) static landmarks represented as a set \( L_{\text{map}} \). The IDs and positions of the landmarks in the world frame are known. The state of the robot \( r_n \) is given by \( x_t^{o_n} = [x_t^{o_n}, y_t^{o_n}, z_t^{o_n}, \phi_t^{o_n}, \theta_t^{o_n}, \psi_t^{o_n}]^\top \) and the state of the tracked moving object \( o_k \) is given by \( x_t^{o_k} = [x_t^{o_k}, y_t^{o_k}, z_t^{o_k}]^\top \) at the \( t^{th} \) time stamp. The 2D-position of the \( t^{th} \) known static landmark is given by \( l^i = [x^i, y^i]^\top \). The landmarks are assumed to be static and on the ground plane.

The odometry measurement made by the robot \( r_n \) at the \( t^{th} \) time-step is given by the vector \( u_t^{r_n} \) and an associated noise with zero mean and covariance matrix \( R_t^{r_n} \). The observation measurement of the landmark \( l \) made by the robot \( r_n \) in its local frame at the \( t^{th} \) time-step is given by the vector \( z_t^{o_n, l} \) and an associated noise with zero mean and covariance matrix \( Q_t^{o_n,l} \). The observation measurement of the object \( o_k \) made by the robot \( r_n \) in its local frame at the \( t^{th} \) time-step is given by the vector \( z_t^{o_n, o_k} \) and an associated noise with zero mean and covariance matrix \( P_t^{o_n,o_k} \).

The state vector being estimated at the \( t^{th} \) time-step is defined as \( x_t = [x_t^{r_1}, ..., x_t^{r_N}, x_t^{o_1}, ..., x_t^{o_K}]^\top \). The control vector at the \( t^{th} \) time-step is obtained by stacking all odometry measurements: \( u_t = [u_t^{r_1}, ..., u_t^{r_N}]^\top \). The observations vector at the \( t^{th} \) time-step is obtained by stacking all the observation measurements: \( z_t = [z_t^{r_1}, ..., z_t^{r_N}, z_t^{o_1}, ..., z_t^{o_K}, z_t^{o_1,o_k}, ..., z_t^{o_N,o_K}]^\top \).

3.3. Particle Filter for Unified Cooperative Localization and object Tracking (PF-UCLT) Algorithm

PF-UCLT is a computationally decentralized and online algorithm, based on particle filters, that recursively estimates the self-localization of each robot in a team, and tracks the position of a single object. The estimation is done cooperatively and in a unified fashion, that is, the self-localization of the robots and the object’s position are estimated in a single step, with no recursive error propagation.

PF-UCLT overcomes the particle deprivation problem by using a rearranging technique of subparticles, only valid under some conditional and mutual independence assumptions between some of the variables involved. Using this technique, it is possible to maintain a similar level of accuracy when the number of robots \( N \) increases, while maintaining the number of particles constant. Additionally, the space and time complexity grow only linearly w.r.t. the same variable, instead of exponentially as in a standard particle filter.

4. Particle Filter for Unified Cooperative Localization and object Tracking Extended (PF-UCLT+) 

4.1. Algorithm description

A particle is defined as a \( (N+K+1) \)-tuple as follows (see Figure 1 for a visual representation):
where the first $N$ elements of the tuple are the robot subparticles, the next $K$ elements of the tuple are the target subparticles, and the last element is the weight of the particle.

![Diagram](image.png)

Figure 1: Structure of particles, subparticles, particle weights and the associated notations.

The algorithm developed (algorithm 2) is a recursive predict-update loop. Each robot in the team should run an instance of the algorithm. In the description of the algorithm below, it is assumed that the algorithm is running on the robot $r_i$. The algorithm’s inputs are $\mathcal{X}_{t-1}$, $u_i$, $z_i$, and $L_{\text{map}}$.

In lines 1 and 2, the robot transmits its control and observation measurements to all other robots in the team, and receives the same information from the available robots. Note that, even though, it was stated before that the algorithm is decentralized, it is only from a computational viewpoint, not from an information viewpoint. Saying this, it does not mean that all robots have the exact same measurements at every time-step, since there could be communications delays or failures between robots.

In lines 3 and 4 the temporary sets $\tilde{\mathcal{X}}_t$ and $\tilde{W}_t$ are both initialized as empty sets. $\tilde{\mathcal{X}}_t$ is a temporary particle set, in which each particle contains the states of all robots and objects tracked, as well as the particle weight. $\tilde{W}_t$ holds all the individual temporary weights of the subparticles corresponding to the robots and tracked objects in the set $\tilde{\mathcal{X}}_t$.

Lines 5 to 12 entail the hypothesis prediction step, similarly to a standard particle filter. The odometry measurements of each robot are incorporated on each robot subparticle, $\mathbf{x}_{t-1}^{[m],r_{ni}}$, and a user-defined motion model is applied on each target subparticle, $\mathbf{x}_{t-1}^{[m],o_{nk}}$. These predicted values are then stored in $\tilde{\mathcal{X}}_t$. The hypothesis prediction step can be done separately for each robot subparticle, $\mathbf{x}_{t-1}^{[m],r_{ni}}$, and each object subparticle, $\mathbf{x}_{t-1}^{[m],o_{nk}}$, due to the following two assumptions:

(i) All control measurements concern only the internal measurements of each individual robot and are independent from each other.

(ii) The pose of each robot and the position of each tracked object are independent of the poses of the other robots and positions of the other tracked objects.

From lines 13 to 30, the weight update step of the algorithm is performed. In this step, the fusion of all observation measurements is performed. This step can also be performed separately for each robot subparticle and each object subparticle, because of the following three assumptions:

1. The first $N$ elements of the tuple are the robot subparticles, the next $K$ elements of the tuple are the target subparticles, and the last element is the weight of the particle.
2. A robot subparticle, $\mathbf{x}_{t-1}^{[m],r_{ni}}$, and an object subparticle, $\mathbf{x}_{t-1}^{[m],o_{nk}}$, are not supposed to have the same location.
3. Each robot subparticle, $\mathbf{x}_{t-1}^{[m],r_{ni}}$, and each object subparticle, $\mathbf{x}_{t-1}^{[m],o_{nk}}$, have distinct weight.

Algorithm 2: PF-UCLT+ ($\mathcal{X}_{t-1}, u_i, z_i, L_{\text{map}}, r_i$)

1: Transmit $\{\mathbf{z}_{t-1}^{[m,r_{ni}],P_t^{[m,r_{ni}]},u_t^{[m,r_{ni}],R_t^{[m,r_{ni}]},z_{t-1}^{[m,r_{ni}]}},Q_t^{[m,r_{ni}]}\}$ for all $l = 1, ..., L$; $k = 1, ..., K$ to all robots $r_n$; $n = 1, ..., N$; $n \neq i$
2: Receive $\{\mathbf{z}_{t}^{[m,r_{ni}],P_t^{[m,r_{ni}]},u_t^{[m,r_{ni}],R_t^{[m,r_{ni}]},z_{t}^{[m,r_{ni}]}},Q_t^{[m,r_{ni}]}\}$ for all $l = 1, ..., L$; $k = 1, ..., K$ such that $n \in \{1, ..., N\}$; $n \neq i$
3: $\tilde{\mathcal{X}} = \{(\tilde{\mathbf{x}}_{t}^{[m,i]}, w_t^{[m,i]})\}_{m=1}^{M} = \{(\tilde{\mathbf{x}}_{t}^{[m,i]}, ..., \tilde{\mathbf{x}}_{t}^{[m,i]},z_{t}^{[m,i]}, \tilde{\mathbf{x}}_{t}^{[m,i]}, ..., \tilde{\mathbf{x}}_{t}^{[m,i]},o_{iK}, w_t^{[m,i]}\}_{m=1}^{M} = \emptyset$
4: $\tilde{W}_t = \{(w_t^{[m,i]}, ..., w_t^{[m,i]},z_{t}^{[m,i]}, w_t^{[m,i]}, ..., w_t^{[m,i]},o_{iK})\}_{m=1}^{M} = \emptyset$
5: for $m = 1$ to $M$ do
6: for $n = 1$ to $N$ do
7: $\tilde{x}_t^{[m,n]} = \text{robot\_motion\_model}(\mathbf{x}_{t-1}^{[m,n]}, u_t^{[m]})$
8: end for
9: for $k = 1$ to $K$ do
10: $\tilde{x}_t^{[m,k]} = \text{object\_motion\_model}(\mathbf{x}_{t-1}^{[m,k]})$
11: end for
12: end for
13: for $m = 1$ to $M$ do
14: for $n = 1$ to $N$ do
15: $w_t^{[m,n]} \propto \prod_{i=1}^{K} p(\mathbf{z}_t^{[m,n]}|\mathbf{z}_{t}^{[m,n]}, L_{\text{map}})$
16: end for
17: end for
18: for $n = 1$ to $N$ do
19: $\{(\tilde{\mathbf{x}}_{t}^{[m,n]}, w_t^{[m,n]})\}_{m=1}^{M} \leftarrow \text{sort\_descend}(\{(\tilde{\mathbf{x}}_{t}^{[m,n]}, w_t^{[m,n]})\}_{m=1}^{M})$
20: end for
21: for $k = 1$ to $K$ do
22: for $m = 1$ to $M$ do
23: $m' = \arg\max_{m \in [m,M]} \prod_{n=1}^{N} p(\mathbf{z}_t^{[m,n]}, \mathbf{z}_t^{[m,n]}, o_{nk})$
24: Swap $\tilde{x}_t^{[m,k]}, o_{nk}$
25: $w_t^{[m,k]} \propto \prod_{n=1}^{N} p(\mathbf{z}_t^{[m,n]}, \mathbf{z}_t^{[m,n]}, o_{nk})$
26: end for
27: end for
28: for $m = 1$ to $M$ do
29: $w_t^{[m]} = \sum_{k=1}^{K} w_t^{[m,k]}$
30: end for
31: Normalize $\{w_t^{[m]}\}_{m=1}^{M}$
32: $\tilde{\mathcal{X}} = \text{resample}(\tilde{\mathcal{X}})$
33: Return $\tilde{\mathcal{X}}$

4
(iii) The observation measurement of each static landmark and of each tracked object are independent of the observation measurements of the other static landmarks and of the other tracked objects.

(iv) The observation measurement of a given static landmark made by any robot \( r_n \) depends only on the predicted pose of the robot \( r_n \) and the fixed position of that static landmark. It is independent of the predicted poses of all the other robots, the fixed positions of all the other static landmarks and the predicted positions of the tracked objects.

(v) The observation measurement of each tracked object \( o_k \) made by any robot \( r_n \) depends only on the predicted pose of the robot \( r_n \) and the predicted position of the tracked object \( o_k \). It is independent of the predicted poses of all the other robots, the predicted positions of all the other objects, and the fixed positions of all the known static landmarks.

From lines 13 to 17, it is computed a weight, \( w_t^{[m], r_n} \), for each robot sub-particle, \( x_t^{[m], r_n} \), that depends on that robot observation measurements of the static landmarks.

In lines 18 to 20 each robot subparticle set, \( \{x_t^{[m], r_n}\}_{m=1}^M \), is sorted in descending order w.r.t. to its temporary weight set, \( \{w_t^{[m], r_n}\}_{m=1}^M \), computed on lines 13 to 17. By doing this rearranging of subparticles, separately for each robot, the highest weighted robot’s subparticles are indexed first. This means that all the different robot’s “good” subparticles (the subparticles that approximate better the correct posterior of that robot’s state have higher weight) stay coupled together, leading to an overall higher chance of that overall “good” particle surviving while the “bad” ones are eliminated in the resampling step of the particle filter. By doing this rearranging technique of each robot subparticle (and later of each target subparticle), the need to have an exponentially increasing number of particles to maintain a given level of accuracy in the state estimates is eliminated. This technique is only valid because any robot \( r_n \)’s subparticle set \( \{x_t^{[m], r_n}\}_{m=1}^M \) was predicted and weighted independently of all the other robots’ subparticle sets, and so, the distribution of the predicted particle set \( X_t \) does not change.

From lines 21 to 27, a weight for each tracked object, \( w_t^{[m], o_k} \), is computed based on the observation measurements of that tracked object by the robots. In this step of the algorithm, before computing the referred weight, the same rearranging technique is performed, now for each tracked object’s subparticle set \( \{x_t^{[m], o_k}\}_{m=1}^M \), which is valid when considering the same proprieties of conditional and mutual independence stated before, and it is ensured that the distribution represented by the predicted particle set \( X_t \) remains unaltered. The rearrangement is performed as follows: for each \( m^{th} \) previously rearranged robots’ subparticles, it is calculated the particle index \( m^* \) ranging from \( m \) to \( M \) at which the weight contribution by the object subparticle \( x_t^{[m^*], o_k} \) to the \( m^{th} \) particle’s weight is maximum. After finding the index, the object subparticle \( x_t^{[m^*], o_k} \) is swapped with the object subparticle \( x_t^{[m], o_k} \), which leads to the \( m^*^{th} \) object subparticle being grouped with the \( m^{th} \) set of robot subparticles, whereas the \( m^{th} \) object subparticle is saved at a later index. This way both the “good” robot subparticles and the “good” object subparticles will stay in the earliest particles, with a higher overall weight and higher chance of survival after resampling.

In lines 28 to 30 the total weight of each particle \( w_t^{[m]} \) is calculated using the individual weights of each robot subparticle, \( w_t^{[m], r_n} \), and each object subparticle, \( w_t^{[m], o_k} \), calculated before in lines 15 and 25.

Finally, in line 31 the particles’ weights are normalized so that in the following line the resample step (low variance resampling) can be performed. After resampling the particle set \( \chi_t \), the output of the algorithm, \( \chi_t \), is obtained and returned.

4.2. Space and Time Complexity Analysis

Assume that \( M \) is the number of particles required to obtain a given accuracy level by a particle filter-based method for a single robot localization and a single object tracked. The complexity of a standard particle filter scales exponentially w.r.t. the number of robots \( N \) and objects \( K \), both in space and time complexity. PF-UCLT, while considering a single object, reduces the dependency on the number of robots to linear. PF-UCLT+ further reduces the dependency on the number of objects to linear, as well.

- Space complexity: The worst-case scenario of space complexity for a standard particle filter-based method to solve the online CLT problem is \( O((N + K)M^{N + K}) \). Algorithm 2 limits the worst-case scenario to \( O((N + K)M) \) since only \( M \) particles, with \((N + K)\) subparticles each, are required to maintain the given accuracy level. This is linear in terms of the number of robots \( N \) and number of objects \( K \).

- Time complexity: The worst-case scenario of time complexity for a standard particle filter-based method is \( O(M^{N + K}) \). In algorithm 2, the worst-case scenario of the weight update step due to the observation measurements of the fixed landmarks is \( O(NM) \); for the sorting performed in lines 18-20 is \( O(NM \log M) \); for the weight update step due to the tracked object’s observation measurements is \( O(NKM^2) \). Considering only the highest order term, the worst-case scenario is \( O(NKM^2) \). This is linear in terms of the number of robots \( N \)
and number of objects $K$. This also means that the overall number of particles must not be too high for the algorithm to remain tractable online.

5. Implementation and Experimental Setup

The developed code\(^1\) uses features from the GNU-GCC compiler v9.2 to increase the parallelization rate. The algorithm has many expensive computations that are repeated, e.g., "for each robot" and "for each target". Furthermore, with careful considerations, these computations are completely independent from each other, preventing race conditions and allowing to completely parallelize the implementation. This property is desirable in modern algorithms, as the industry is moving further towards using processors with an increased number of less powerful logical cores.

The implementation was done within the Robot Operating System (ROS) framework. The version used was the ROS Melodic Morenia, released on May 23rd, 2018. The code was written in C++17.

All simulation experiments were run on a laptop Clevo 650RB with CPU Intel Core i5-6300HQ (2.30 GHz up to 3.20 GHz) and 8 GB RAM. The operating system used was 64-bit Ubuntu 18.04 LTS.

5.1. Randomized Dataset Generator

To test the developed algorithm, a simulator which generates a dataset for a provided scenario was created\(^2\) using the programming language Python. With this simulator, for a desired number of robots and targets, size of test field, number of landmarks, sensor range, etc., a randomized dataset can be created. The dataset simulates both the robot motions, including translation and rotations on the 3-D test field while avoiding collisions, and the arbitrary target motions in the 3-D test field.

5.2. PF-UCLT+

Even though the full pose of a MAV contains 6 states ($[x \ y \ z \ \phi \ \theta \ \psi]$) on the algorithm only 4 states ($[x \ y \ z \ \psi]$) are estimated. This simplification can be done, because in a MAV, the roll and pitch angles are available and accurate enough through the use of its on-board Inertial Measurement Unit (IMU).

In particle filters it is necessary to have an initial estimate. Both the initial robots particles and initial targets particles were initialized close to the true value for all simulations. The case study of random initialization of the target particles and robot particles initialized close to the true values was also tested, being verified that the algorithm would still converge rapidly, as soon as observations of the tracked objects start occurring. However, for the case of randomly initialized robot particles, it was verified that the algorithm did not converge.

For all simulations the number of particles used was 300.

6. Simulations

6.1. Cooperative Localization

The first set of simulations were designed to show that performing cooperative localization through mutually observed common objects improves both the localization estimates of the robots and of the common tracked objects, when compared with the case where each robot is localized independently from the other robots and the tracked objects positions are obtained by fusing their position estimates from each robot.

To simulate the independent particle filter single-robot localization approach\(^3\), algorithm 2 was run with $N = 1$ and $K = 3$, separately on each one of the 4 robots of the dataset. Algorithm 2 was also run with $N = 4$ and $K = 3$ on the same dataset\(^4\). In the generated dataset, the landmark sensor range was set to 3 m and the target sensor range to infinite. This was done to guarantee that the robots lose sight of the landmarks frequently, but always see the targets; this way it is verified whether or not the mutually observed tracked objects help in the robot pose estimates.

In figure 2, it is presented the results of all simulation runs in this experiment. For the localization of robots 1, 2 and 4 both the independent particle filters and the PF-UCLT+ algorithm had similar performances. However, the independent particle filter algorithm run on robot 3 showed poor localization accuracy. When compared with the PF-UCLT+ algorithm, it is verified that there is a significant increase in accuracy when the PF-UCLT+ algorithm is used. This can be explained by robot 3 observing few landmarks, or even no landmarks at certain times, and for short time periods. It is also verified that the PF-UCLT+ algorithm outperforms the independent particle filters algorithms on the targets estimations, showing that the cooperative nature of PF-UCLT+ in fact helps to estimate the tracked objects positions.

6.2. Time and Space Complexity

Firstly, it was evaluated the trend of computation time growth with the number of robots. For this experiment nine different configurations were ran (all with three targets). The number of robots varied from $N = 2$ to $N = 10$. The sensor range for both the observation of landmarks and targets was set to 4 m.

In figure 3 is the plot of the described experiment, showing the average iteration time for all the different configurations. It is verified that the computation time grows linearly w.r.t. the number of robots. This linear growth in computation time is due to the linear growth.

\(^{1}\)https://github.com/pmvfaria/pfuclt

\(^{2}\)https://github.com/pmvfaria/clt_random_datasets

\(^{3}\)Video link of a simulation with $N = 1$ and $K = 3$: https://vimeo.com/457732695

\(^{4}\)Video link of a simulation with $N = 4$ and $K = 3$: https://vimeo.com/457732892
in the number of landmark and target observations as the number of robots increases.

Figure 3: Computation time w.r.t. the number of robots.

In figure 4, it is shown the average memory occupied by the ROS messages in a full simulation for the nine different configurations. It is also verified that the grow is linear w.r.t. the number of robots.

The next experiment was made in order to evaluate the evolution of the iteration computation time with the increase in the number of targets. Ten different configurations, all with three robots, and increasing number of targets, from $K = 1$ to $K = 10$ were analysed. The sensor range for both the observation of landmarks and targets was also set to 4 m.

In figure 5, it can be verified that the average iteration computation time grows linearly with the increase of the number of targets. Comparing with the case of increase of robots, it is verified that the slopes of both straight lines are very similar to each other. It is expected a linear grow because of the increase in the number of target observations as the number of targets increases.

Figure 5: Computation time w.r.t. the number of targets.

Finally, in figure 6, it is shown the average memory occupied by the ROS messages in a full simulation for all different configurations. It is verified that the grow is linear w.r.t. the number of targets. Compared with the case of increasing robots, it is verified that the slope of the straight line in the case of increasing number of targets is smaller than the slope in the case of increasing number of robots. This is expected, because by increasing the number of targets only the number of target observations messages increases, while by increasing the number of robots both the number of land-
mark and target observations messages increase. Note that, as the number of either robots $N$ or targets $K$ increases, the memory required to run the algorithm also increases linearly, since a particle consists of $N + K$ subparticles.

Figure 6: ROS messages total size w.r.t. the number of targets

6.3. Estimation error with the number of robots

In this set of simulations\(^5\), the goal is to evaluate the accuracy of the estimates when the number of robots change (maintaining the same number of particles). For the experiment, a configuration with a constant number of targets, $K = 3$, and increasing number of robots, from $N = 2$ to $N = 10$ was performed. The sensor range for both the observation of landmarks and targets was set to 4 m.

In figure 7 is a summary of all simulations condensed into a single boxplot. In this boxplot, for each configuration, the robots estimation errors and targets estimation errors over all runs and datasets are presented in a single boxplot to facilitate the comparison between the different configurations. It is also shown the targets visibility ratio (ratio between the time the targets can be seen by at least one robot and the total dataset duration) for each configuration. For all configurations the robots localization estimation errors remain constant. However, for the targets estimation errors, it is also verified that the estimation errors remain approximately constant w.r.t. the number of targets.

Figure 7: State estimation accuracy w.r.t. the number of robots.

6.4. Estimation error with the number of targets

In this experiment\(^6\) a configuration with constant number of robots, $N = 3$, and increasing number of targets, from $K = 1$ to $K = 10$ is tested; the sensor range was also limited to 4 m.

In figure 8 is the result of the experiment summarized into a boxplot. For all configurations the robots localization estimation errors remain constant. As for the targets estimation errors, it is also verified that the estimation errors remain approximately constant w.r.t. the number of targets.

Figure 8: State estimation accuracy w.r.t. the number of targets.

With the last two case studies, it was experimentally verified that the PF-UCLT+ algorithm does not require an increase in the number of particles, and needs only a linear increase in the number of subparticles w.r.t. the number of robots and number of targets (linear increase in the memory requirements), in order to maintain an

---

\(^{5}\)Video link of one of the simulations for this case study ($N = 5$ and $K = 3$): https://vimeo.com/457726301

\(^{6}\)Video link of one of the simulations for this case study ($N = 3$ and $K = 5$): https://vimeo.com/457730041
approximately constant accuracy level in all the state estimates (robots and targets).

7. Conclusions and Future Work

In this work, it was presented the theoretical formulation of the algorithm PF-UCLT+, as well as, an implementation of it in C++. This algorithm deals with the problem of multi-robot self-localization and objects tracking. This relevant and challenging problem creates several difficulties to existing algorithms due to the high state space dimensionality. This solution uses a method of rearranging of the order of subparticles, possible through an exploitation of properties of mutual and conditional independence, to make a standard particle filter algorithm tractable w.r.t. both the number of robots self-localizing, and the objects to be tracked. The cooperative estimation through mutually observed objects increases the potential of PF-UCLT+ as a reference algorithm for everyday use in a multi-robot scenario.

The objectives of the present work were achieved. In the developed algorithm the generalization of the PF-UCLT algorithm [2] to track online multiple objects with robots moving in a 3-D environment was successful. Additionally, it was proved through experiment results that the spatial and time complexity grow only linearly w.r.t. the number of robots and tracked objects, while always maintaining the number of particles constant. It was also verified in the various experiments that the accuracy of the estimations of both the robots poses and targets positions are kept approximately constant w.r.t. the number of robots and targets.

There are some extensions to the currently proposed algorithm that can be pursued:

- The case of unknown correspondence in the object measurements;
- The case of a variable number of objects to be tracked;
- Adaptable number of particles, depending on available computational resources;
- A possible reduction in the quadratic dependency on the number of particles;
- Extending to SLAM (Simultaneous Localization And Mapping).

Acknowledgements

I would like to thank the researcher/PhD student Guilherme Lawless of the Institute for Systems and Robotics at Instituto Superior Técnico of the University of Lisbon. He helped me on the development of the programming code used in this dissertation. He also always helped me when I had questions about my research or writing.

References


