

# Laser Tracker Placement Optimization for highly flexible manufacturing systems

Jan Baumgärtner<sup>1\*</sup>, Max Goebels<sup>1\*</sup>, Alexander Puchta<sup>1</sup> and Jürgen Fleischer<sup>1\*\*\*</sup>

No Institute Given

## Abstract

In recent years, modular and reconfigurable manufacturing systems have gained attention for their adaptability. While system design is now highly automated, designing the verification setup remains a manual design problem involving tasks like placing external sensors to measure performance. This paper argues that design optimization approaches for manufacturing systems can also automate the design of the verification setup. This is demonstrated using the placement of a laser tracker system with markers that due to their line-of-sight complexity and simultaneous measurement of multiple systems can be challenging to design. We present a straightforward optimization approach to demonstrate potential automation in this area.

## 1 Introduction

## 2 State of the art

The work presented in this paper lies at the intersection of two research fields: optimization of manufacturing systems and sensor placement optimization. The question this work first has to answer is what do we actually verify in a manufacturing system. The first thing we have to clarify here is that we are not looking at product based verification but system based verification. This means we don't want to look at the resulting quality of the product but at the performance of the manufacturing system itself. The main reason for this is that the reconfigurable manufacturing systems we are looking at are not varying in processes but in how the machine achieves the same process. This means we don't want to measure the process influence but the machine influence. In the framework of **[reconfigurable\_production]** this means we are mainly dealing with the reconfiguration design issue. Examples of such systems include the robot cell optimization problem proposed in **[previous\_work]** or **[stiffness\_placement]** but also robot design optimization problems such as **[task\_synthesis]**, **[ad-hoc\_manipulator]** or **[multi\_objective]**. These works are very robotics specific for a more manufacturing based perspective we can look at works such as **[johannes\_1]** or **[johannes\_2]**.

---

\* \*These authors contributed equally to this work.

\*\* <sup>1</sup>Jan Baumgärtner, Max Goebels, Alexander Puchta, and Jürgen Fleischer are with the wbk Institute of Production Science, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany  
max.goebels@kit.edu

The main considerations in these works where the possible payload, accuracy, stiffness, force, weight and possible motion of the systems. The important point is that in order to verify that the system configuration that was optimized is indeed capable of achieving the desired performance we need to measure the performance of the system. Since most configurable manufacturing systems are not capable of directly measuring their accuracy, force or stiffness we need to use external sensors to measure these values. This requires a verification setup that is capable of measuring the performance of the manufacturing system. If we want to automate this process we are dealing with the domain of sensor placement optimization. The main goal of sensor placement optimization is to find the optimal placement of sensors in a system such that the performance of the system can be measured as accurately as possible. This is a well studied field in the context of structural health monitoring where a comprehensive overview can be found in [shms]. Important algorithms in this field are the genetic algorithm, particle swarm optimization and simulated annealing [shms]. Borrowing from this field we will also use a particle swarm optimization algorithm to solve our optimization problem. The paper also explores a host of different measurement types such as Velocity, displacement, acceleration, strain, temperature or pressure. Howeverwhile it is possible to borrow from this field additional challenges are introduced by the fact that manufacturing systems are typically moving and that the sensor placement thus has to take into account the total trajectory of the manufacturing system. Only recently has this challenge been started to be addressed in the context of manufacturing systems [1]. Allthough this early work focuses on process control and not system verification its methodologies can be applied to our problem. Mainly that instead of optimizing a single objective function we need to optimize a objective integrated over the total process time. However since we are not looking at cameras such as the previous paper we need to derive our own visibility model reflecting the measurement characteristics of a laser tracker.

### 3 Formulation of the optimization problem

To solve the joint optimization problem of marker and laser tracker pose we first need to define the objective function. Our goal is ofcourse to maximize the visibility of the marker to the laser tracker. Similar to approaches such as [1] we divide the visibility into two components. Line of sight visibility and field of view visibility. Both are depicted in figure ???. Mathematically we can define line of sight visibility (los visibility) as the following binary function:

$$f(p_m, p_l) = \begin{cases} -1 & \text{if marker is in line of sight of the laser tracker} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where  $p_m$  is the position of the marker,  $p_l$  is the position of the laser tracker. Note that  $p_m$  is typically dependent on time as it is attached to a moving manufacturing system such as a robot. The field of view objective can similarly be defined as a binary function:

$$g(p_m, p_l) = \begin{cases} -1 & \text{if marker is in the field of view of the laser tracker} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

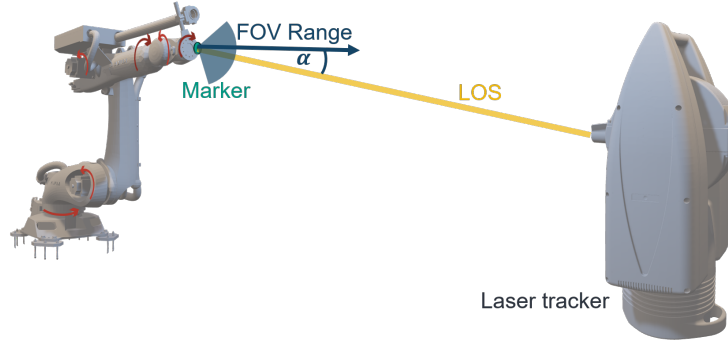


Fig. 1: Field of view (VOW) angle range and line of sight (LOS) components of marker visibility

However in practice it might make sense to not be close to the boundary of the field of view. Instead one can define a continuous function that is 1 in the center of the field of view and 0 at the boundary. The simplest way to encode this using the angle  $\alpha$  between the normal of the marker and the line of sight of the laser tracker.

$$g(p_m, p_l) = -\cos(\alpha) \quad (3)$$

Apart from line of sight we also want to optimize the measurement performance of the laser tracker. The accuracy of the laser tracker is mainly dependent on the distance between the marker and the laser tracker. The further away the marker is the less accurate the measurement will be. However there is also a minimum measurement distance that the laser tracker can handle. To encode this we can define a penalty function that penalizes markers that are too far away from the laser tracker.

$$p(p_m, p_l) = \begin{cases} d & \text{if marker is in measurement range of the laser tracker} \\ \rho & \text{otherwise} \end{cases} \quad (4)$$

where  $d$  is the distance between the marker and the laser tracker, and  $\rho$  is a large penalty factor. The final objective function is then a weighted sum of the two components:

$$h(p_m, p_l) = \int f(p_m, p_l) + g(p_m, p_l) + p(p_m, p_l) dt \quad (5)$$

The integral is taken over the entire trajectory of the manufacturing system that needs to be observed by the laser tracker. Having defined the objective function, we can proceed to define the design variables and constraints. The design variables are the pose of the laser tracker ( $p_l$ ) and the pose of the marker ( $p_m$ ).

First, we assume the marker is attached to the flange of the manufacturing system, meaning we only need to define a transformation matrix  $T_m$  that describes the marker's pose relative to the flange. The final pose of the marker ( $p_m$ ) is then given by combining the flange transformation  $T_f(q)$ , where  $q$  is the robot's joint configuration, with the transformation  $T_m$ :

$$p_m = T_f(q)T_m \quad (6)$$

While  $T_m$  is a 4x4 matrix, it can be parameterized using the marker's position ( $p_m$ ) and orientation ( $p_{om}$ ), resulting in a 6-dimensional design space for the marker pose. To avoid unrealistic positions, we constrain this space to keep the marker close to the manufacturing system's surface. This constraint can be represented cylindrically:

$$\begin{aligned} \sqrt{p_{m,x}^2 + p_{m,y}^2} &\leq r \\ 0 &\leq p_{m,z} \leq h \end{aligned} \quad (7)$$

I

where  $r$  is the cylinder's radius and  $h$  is its height. Additionally, to ensure the marker remains near the surface, a penalty function can be added to the objective function to penalize markers positioned too far from the flange.

The laser tracker's pose is defined by a transformation matrix  $T_l$ , but due to its spherical measurement volume, only its position ( $p_l$ ) needs to be specified. The laser tracker is mounted on a stand with a height range constraint:

$$h_{min} \leq p_{l,z} \leq h_{max} \quad (8)$$

Where  $h_{min}$  and  $h_{max}$  are the minimum and maximum heights of the stand, respectively. Using the design variables and constraints of the marker and laser tracker, the optimization problem can be formulated as follows:

$$\begin{aligned} \max_{p_m, p_l} \quad & h(p_m, p_l) \\ \text{s.t.} \quad & \sqrt{p_{m,x}^2 + p_{m,y}^2} \leq r \\ & 0 \leq p_{m,z} \leq h \\ & h_{min} \leq p_{l,z} \leq h_{max} \end{aligned} \quad (9)$$

## 4 Optimization algorithm

There are a large number of possible optimization algorithms that can be used to solve this problem. Following the no free lunch theorem [**no\_free\_lunch\_theorem**] we want to pick the algorithm that makes the most use of the underlying structure of the problem. In this case the problem is a continuous optimization problem with a relatively low number of design variables. On the other hand it is a gradient free optimization problem due to the line of sight visibility function. These two properties make the problem well suited for the particle swarm optimization (PSO) algorithm [**pso**]. Particle swarm optimization is a computational method for finding the optimal solution to a problem. It is inspired by the behaviour of swarms in nature, such as flocks of birds or schools of fish, which exhibit emergent behaviour that is intelligent and efficient. In PSO, a group

of particles (also called agents or individuals) move through a search space, according to the following update rules:

$$\begin{aligned} x_{i,j} &= x_{i,j} + v_{i,j} \\ v_{i,j} &= wv_{i,j} + c_1r_1(p_{i,j} - x_{i,j}) + c_2r_2(p_{g,j} - x_{i,j}) \end{aligned} \quad (10)$$

where  $x_{i,j}$  is the position of particle  $i$  in dimension  $j$ ,  $v_{i,j}$  is the velocity of particle  $i$  in dimension  $j$ ,  $w$  is the inertia weight,  $c_1$  and  $c_2$  are the cognitive and social coefficients,  $r_1$  and  $r_2$  are random numbers between 0 and 1,  $p_{i,j}$  is the best position of particle  $i$  in dimension  $j$  and  $p_{g,j}$  is the best position of the swarm in dimension  $j$ . In our case the particles are a 9 dimensional vector based on the position of the laser tracker and the pose of the marker. The best particle is the one that maximizes the objective function defined in equation ???. The hyperparameters we need to tune are collected in table ??. Note that the pso algorithm has no direct way to encode the constraints we defined earlier. Instead we can use a penalty function that penalizes any particles that violate the constraints. This is a common approach in optimization and is also used in the implementation of the pso algorithm in the scipy library. The penalty function can be defined as:

$$p(x) = \begin{cases} 0 & \text{if } x \text{ satisfies all constraints} \\ \rho & \text{otherwise} \end{cases} \quad (11)$$

where  $\rho$  is a large penalty factor.

Hyperparameter	Value
Number of particles	500
Number of iterations	100
Inertia weight	0.8
Cognitive coefficient	0.1
Social coefficient	0.1

Table 1: Hyperparameters of the PSO algorithm

## 5 Simulation Environment

Evaluating the objective function requires a simulation of the manufacturing system. The tricky part here is that since the manufacturing is actively manufacturing parts we not only need to simulate the behavior of the system itself but the changing shape of any parts that are being manufactured. For this purpose we are using the pybullet-industrial package [2]. This extension of the pybullet physics engine allows us to simulate not only the robot but also the manufacturing process. For the purposes of this paper the markers are implemented as a pybullet industrial endeffector object that is attached to the robot. A image of the simulation environment is shown in figure ??.

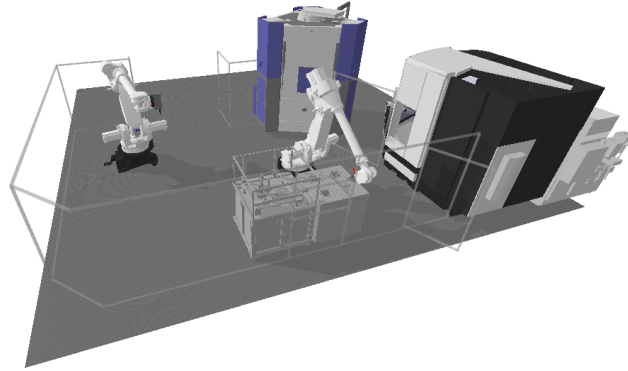


Fig. 2: Simulation environment used for the optimization

## 6 Sample Testcase

To test the performance of our optimizer, we need to define a testcase. Both robots are Comau NJ290 industrial robots with a reach of 2.9 meters. The right robot services the CNC machine and the additive manufacturing system, while the left robot performs a high-precision manufacturing task. For this test, the left robot follows a rectangular trajectory with dimensions of 0.4 meters in length and 0.6 meters in width, as depicted in Figure ???. The aim is now to find a laser tracker and marker placement that can track the left robot throughout its entire trajectory. We compare this to the performance of the initial population of marker and laser tracker placements used to start the optimization process. Figure ?? illustrates the performance of the algorithm over multiple iterations. The results show that the algorithm converges to a solution in roughly 40 iterations, which is relatively fast for an optimization approach. This speed is likely due to the existence of numerous optimal placements for the laser tracker and markers. The final solution is also depicted in Figure ??.

## 7 Acknowledgements

### References

- [1] Jan Baumgärtner et al. “Camera Placement Optimization for a Novel Modular Robot Tracking System”. In: *2023 IEEE SENSORS*. 2023, pp. 1–4. DOI: 10 . 1109/SENSORS56945.2023.10324941.
- [2] Jan Baumgärtner et al. “PyBullet Industrial: A process-aware robot simulation”. In: *Journal of Open Source Software* 8.85 (2023), p. 5174. DOI: 10 . 21105 / joss.05174. URL: <https://doi.org/10.21105/joss.05174>.

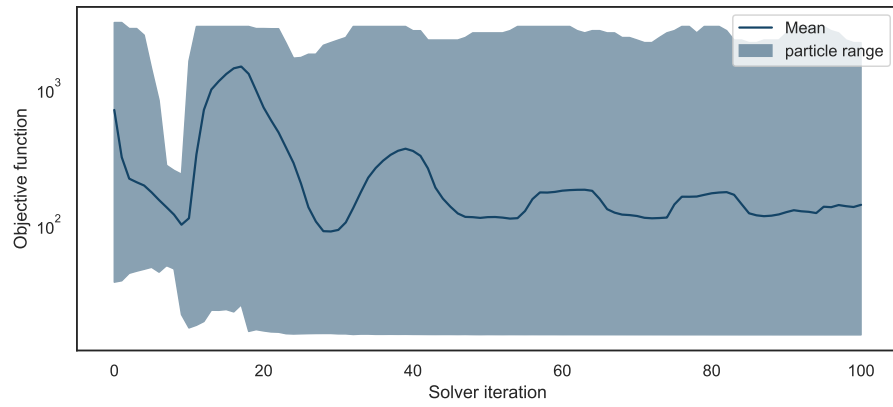


Fig. 3: Performance of the algorithm over the iterations.

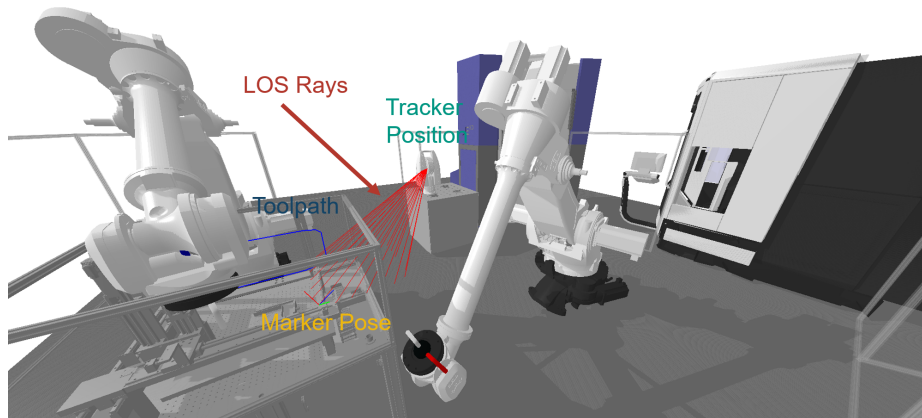


Fig. 4: Trajectory of the robot and final laser tracker and marker placements.