A constraint based motion optimization system for quality inspection process improvement

Nicolò Boscolo[1], Elisa Tosello[2], Stefano Tonello[1], Matteo Finotto[1], Roberto Bortoletto[2], and Emanuele Menegatti[2]

[1] IT+Robotics Srl, Strada Prima 11, 35129 Padova, Italy *
nicolo.boscolo@it-robotics.it, stefano.tonello@it-robotics.it, matteo.finotto@it-robotics.it
[2] Intelligent Autonomous Systems Laboratory, University of Padova, via Gradenigo 6a, 35131 Padova, Italy elisa.tosello@dei.unipd.it, emg@dei.unipd.it, bortolet@dei.unipd.it

Abstract. This paper presents a motion optimization system for an industrial quality inspection process where a vision device coupled with a manipulator robot arm is able to perform quality and completeness inspection on a complex solid part. In order to be deployed in an industrial production plant, the proposed system has been engineered and integrated as a module of an offline simulator, called WorkCellSimulator [11], conceived to simulate robot tasks in industrial environments. The novelty of the paper concerns the introduction of time constraints into the motion planning algorithms. Then, these algorithms have been deeply integrated with artificial intelligence techniques in order to optimize the inspection cycle time. This integration makes the application suitable for time-constrained processes like, e.g., autonomous industrial painting or autonomous thermo-graphic detection of cracks in metallic and composite materials.

1 Introduction

A manufactured product must adhere to a defined set of quality criteria. It must be designed and built according to safety standards specified in the purchasing documentation, and it has to be free of defects and non-conformities. Quality control (QC) [8] is a procedure intended to ensure the observation of these requirements. In order to implement an effective QC program, an enterprise must first decide which specific standards the product must meet; then the sequence/type of QC actions must be determined, for instance the visual inspection to detect defects, such as cracks or surface blemishes. Typically, companies that engage in quality inspection have a team of workers who focus on inspecting products. Some studies show that manual inspection faces numerous problems [2]: human experts require intensive training, and even between well-trained individuals, results tend to be observer-dependent. [6], [10], and [12] state that the accuracy of human visual inspection declines with dull, endlessly routine jobs, that means a 100% quality assurance is often unfeasible. Inspection tasks can be dangerous other

*This research has been funded by the European Unions 7th Framework (FP7/2007-2013) under grant agreement No. 284607, *Thermobot*[1] project.

than difficult: workers may be required to handle materials hazardous. For example, detection of cracks in metallic and composite parts in the automotive and aircraft industry requires magnetic particle inspection: an ecologically undesirable and injurious to human health. Automated visual inspection is an alternative. Automation leads to several advantages [5], including:

- Freeing humans from a dull and routine;
- Saving human labor costs;
- Performing inspection in unfavorable environments;
- Reducing demand for highly skilled human inspectors;
- Matching high-speed production with high-speed inspection.

In order to maximize the aforementioned benefits, experts have to select the automated strategy that better affect the performance of production processes in terms of production cost, cycle time, and production quality. The choice can be complicated because these impacts vary from one inspection strategy to another. Thus, simulation can be used to compare different strategies and select the most appropriate one [7].

In this paper, a motion optimization system for an industrial quality inspection process is presented. A robot with a vision system on its end-effector has to fully examine a complex solid part, e.g., for detecting cracks. Employing an automated inspection system is useful if the robot correctly inspects the part performing a time-saving coverage path. The novelty of the paper is the use of a minimum path covering algorithm able to comply with time constraints.

In this way, even time limited industrial quality inspection processes can be analyzed; e.g., in the autonomous thermo-graphic detection of cracks, a manipulator robot coupled with a thermo-camera inspects a 3D component.

In order to save time during the inspection strategy setup, the authors decided to simulate the process. With this aim, the proposed system has been engineered and integrated as a module of an offline simulator, called WorkCellSimulator [11]. The software, developed by IT+Robotics, is able to simulate robots tasks in industrial environments allowing the definition of specific automated cells for customized production processes, the robot off-line programming, the examination of robots and work cell machineries proper functioning, and the plan transfer into the real world. For a correct product inspection, WorkCellSimulator enables the user to reproduce the scene and to select the inspection points over the product. The system checks the points validity: every point must be collision free and reachable by the robot. Points not satisfying the requirements are removed and the remaining ones are ordered to form the minimum coverage path the robot will follow during the inspection.

The remainder of the paper is organized as follows: Section 2 contains an overview of the WorkCellSimulator planning procedure, exposing the adopted approaches for the path and motion planning, the collision avoidance and the ATSP[3] algorithm used to sort the inspection zones and to obtain the minimum coverage path. Section 3 reports the experiments performed on a practical case: the thermographic detection of cracks, and Section 4 concludes the paper and outlines the future work.

[3]Asymmetric Travelling Salesman Problem

2 Proposed Architecture

Given a 3D model of the inspected product, WorkCellSimulator guides the user from the definition of the workcell (robot and other components) to the selection of set of the visually inspected zones over the product. Then, the simulation core computes a sub-optimal valid path which takes in account the time constraints extracted from the defined zones. In a second stage, the simulator solves the motion planning problem related to the path generated in the previous step. Concluding, the motions are translated in the robot controller language and sent to the machine.

2.1 Data workflow

When the 3D model of the product is imported, an oriented point cloud is created. This redundant set of points will represent the feasible robot positions where the robot is able to acquire images which will be used for cover the desired inspection with the selected camera device. The points set is redundant in order to take more acquisitions of the same surface. Each point is composed of three translations (X,Y,Z) (in millimetres) and three rotations (R_x,R_y,R_z) (in radians) representing the coordinates of the robot end-effector taking the product as the reference system. The simulator core verifies each point: points can be valid (i.e. reachable and not colliding), not reachable (robot joints limits do not let the robot reaching the point) or colliding (if the robot reaches the point, it will collide with the surrounding world). In the last two cases, they are discarded from the trajectory and only the remaining points are exploited to plan the path.

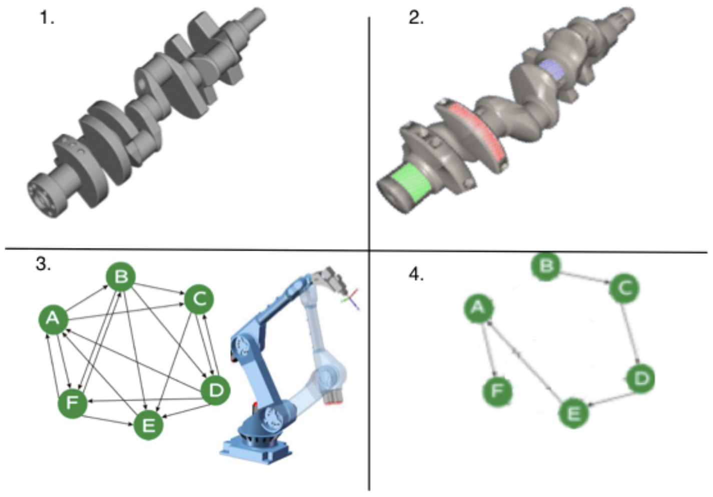
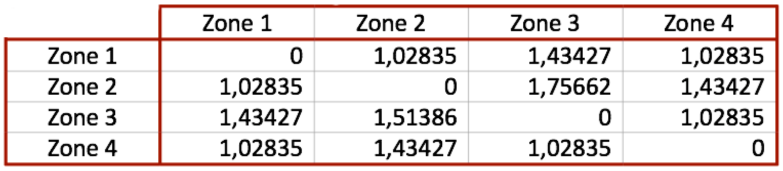


Fig. 1. System workflow overview. In (1) the product is imported and divided in zones (2). Then in (3-4), the motion planner coupled with the ATSP solver will compute the path.

Usually the product is not completely inspected, only the most relevant and sensible parts are checked. For this reason the system permits to the user to select and group in zones the points over the product. Each zone represents a space region to inspect. As in the case of crankshaft, usually the zones are not placed one next to another. Then, the robot needs to compute some extra movements to reach one zone from another. These transitional movements increase the time of the inspection critically if not well managed. The times required for moving from one zone to another can change if the zones ordering change. Subsequently, the simulator attempts to reduce these times modeling them as a feature to be optimized. Inside the simulator the problem is treated as a complete graph. Formalizing, the nodes represent the zones and the edges are weighted with the time required by the robot to drive between a couple of zones. Using the motion planning module, the simulator will compute the time cost to go from one zone to another, the edge cost. The graph is used as input for the ATSP solver, which will generate a time-saving sequence to cover all the inspection zones.

2.2 ATSP Solver and heuristics

The solver used in this work is the Concorde TSP solver (Applegate et al. 2001), one of the most advanced and fastest TSP solvers using branch-and-cut where the Chained Lin-Kernighan [3] heuristic has been implemented and used in the described application. The Concorde TSP Solver is written in the ANSI C programming language and has been used to obtain the optimal solutions to the full set of 110 TSPLIB [9] instances, the largest having 85,900 cities.



	Zone 1	Zone 2	Zone 3	Zone 4
Zone 1	0	1,02835	1,43427	1,02835
Zone 2	1,02835	0	1,75662	1,43427
Zone 3	1,43427	1,51386	0	1,02835
Zone 4	1,02835	1,43427	1,02835	0

Fig. 2. Costs matrix representing the robot time needed to travel from one zone to another.

As described in Subsection 2.1 the graph represents what the robot trajectory time is in moving from one zone to another. The manipulator dynamics are stressed by different and not manageable mechanic actors, like the friction. Then, the time of moving from zone A to zone B is different from performing the symmetric action (see Figure 2). For this reason our problem has been modeled as the Asymmetric TSP where each pair of nodes has two edges connecting each other where edge costs are different. Despite this, the Concorde is coded for the symmetric traveling salesman problem (TSP) and some related network optimization problems. To reformulate the ATSP as a TSP, for each zone a dummy zone (e.g, for Zone A a dummy zone Zone A*) is added. Between each zone and its corresponding dummy zone a negative or very small distance with cheap value is used. This ensures that each zone always occurs in the solution together with its dummy zone. The original distances are used between the zones and the dummy zones, where each zone is responsible for the distance going to the zone and the dummy zone is responsible for the distance coming from the zone. The distances between all zones and the distances between all dummy zones are set to infeasible, it means that a very large value is selected for these distances in order to make them unelectable from the solver.

2.3 Motion Planning

The aim of the offline motion planning architecture is find the correct robot arm joint values in order to avoid collisions between from one task and the following one. The planning and the optimization are performed over a configuration space of the robot kinematic (joint positions) and dynamic variables (joints velocity and acceleration) in order to plan a motion. A valid robot joints configuration is called key-point and it corresponds to a collision free placement of the robot in its physical workspace. The temporal sequence of these key-points is called the trajectory. The planners will add other configurations among each couple of key-points with the aim of avoiding collisions. The collision checking between two configurations is performed using a linear interpolation of the robot motions. Each algorithm performs a backward configuration search: from the last key-point to the first one. The software architecture core can be split in three steps as shown in the Figure 3.

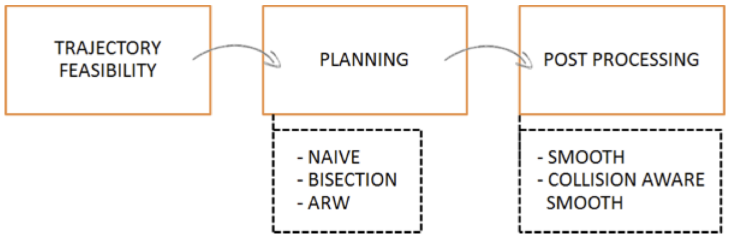


Fig. 3. Motion planning architecture overview.

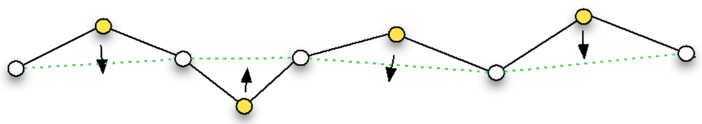


Fig. 4. Representation of smooth perturbation action. The middle points are moved towards the straight line.

In the first step, the trajectory feasibility is checked. The algorithm checks if each trajectory keypoint corresponds to a configuration reachable by the robot and it is not colliding. If the first step ends in a good way, the motion planners are executed in a hierarchical order, from the most simple and fast to the most effective but slower. A planner is executed only if the previous one fails to find a solution following this order:

1. *Naive*. Given the configuration sequence, the algorithm will check if each couple of points is connectible[4] otherwise it will try to add a point between them. Typically the middle point of three points, may include a collision or the three points are not connectible, then the algorithm will attempt to reposition this point editing each single point dimension with various correction factors.
2. *Bisection*. This algorithm uses a logic that is similar to the previous one with the following two main different features:
   (a) A middle point dimension is moved using perturbation of random points over a normal distribution.
   (b) When a collision free middle point is found, the connectible test is performed. But in this case each segment (first-middle and middle-second key-points) is treated recursively using a divide and conquer technique.
3. *ARW*. The Adaptive Random Walks algorithm is part of the Probabilistic Roadmap Methods. The proposed algorithm turns out to be simple to implement, and the solution it produces can be easily and efficiently optimized. Furthermore, the algorithm can incorporate adaptive components in order to be adapted to particular problems. Proofs of the theoretical soundness of the algorithm are provided in [4].

After the kinematic evaluation, the optimization algorithms will edit the key-points in order to make a smoother robot trajectory. All the algorithms that optimize the trajectory will be computed:

1. *Smooth*. When the feasible motion path has been computed, it is not rare that the final path has been filled with redundant points that makes the robot motions jerky. In order to avoid motion that stress robot mechanics and increase the motion time, several actions are performed:
   - If two points are connectible and the are some middle points, the lasts will be deleted.
   - Given each consecutive 3-tuple points, the middle one is moved using a perturbation method. The perturbation pushes the middle point straight to the line connecting the first and the third point (Figure 4).
2. *Collision aware smooth*. At this point, the motion plan is smooth, collision free but discrete. The resulted trajectory is smooth and time saving, but it does not consider the distance from the objects. This smooth planner keeps key-points to a minimum distance without loosing the trajectory smooth property built from the previous plan step.

3 Results and case study

The system has been evaluated on the Thermobot [1] test product shown in Figure 5. The first test focuses on the reliability of the system, where the zones over

[4]Two key-points are connectible if the robot motions needed to reach the second point starting from the first one are collision free.

the product are labeled with consecutive numbers (see Figure 5). After the system computation, the output sequence is a reasonable one, 4-3-2-1, where the robot should spend 4.599 seconds to transitional movements. The worst case is the sequence 3-1-2-4, with a transitional trajectories time of 5.654 seconds. On average, this will save half second for each inspected product. This appears to be a good result considering the large number of pieces made in a factory.

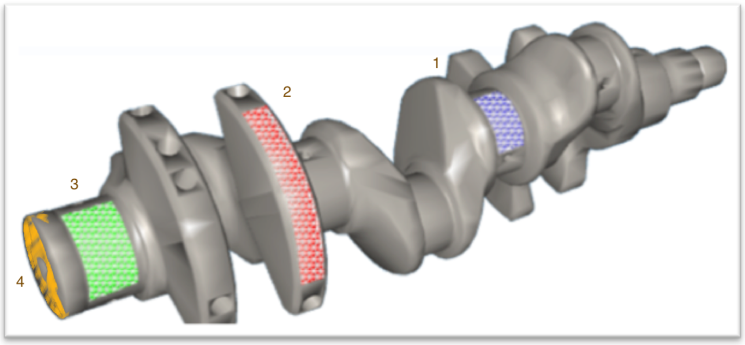
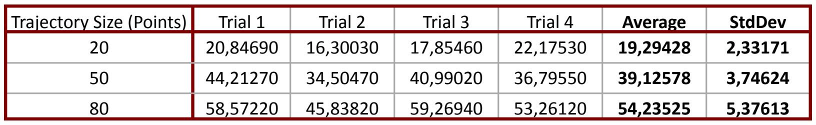


Fig. 5. Zones over a mechanical engine part: the crankshaft.

The second set of tests focus on the performance of the motion planning architecture provided by the WorkCellSimulator. A good performance of the motion planning module is a must in order to compute the cost matrix for the ATSP solver (see Figure 2). The architecture is compared with a successful single shot algorithm, the ARW. Several trials with different trajectory sizes (Figures 6, 7) have been performed. Further, Figure 8 shows that the WorkCellSimulator motion planning architecture (hierarchical set of algorithms and heuristics) has higher performance than a single algorithm. That is because the mixed policy built inside the simulator is more flexible than a general purpose algorithm like ARW. Looking deeply, the standard deviation of the computational times is straightforward that the architecture is more stable in terms of response times. Which is a strong requirement for applications used in industrial processes.

Trajectory Size (Points)	Trial 1	Trial 2	Trial 3	Trial 4	**Average**	**StdDev**
20	4,71620	5,06383	4,99058	4,94046	**4,92777**	**0,12979**
50	11,24590	11,71840	11,73470	12,11530	**11,70358**	**0,30829**
80	16,06670	16,65910	16,56760	16,52070	**16,45353**	**0,22881**

Fig. 6. WorkCellSimulator planning architecture performance in different size trajectories.



Trajectory Size (Points)	Trial 1	Trial 2	Trial 3	Trial 4	**Average**	**StdDev**
20	20,84690	16,30030	17,85460	22,17530	**19,29428**	**2,33171**
50	44,21270	34,50470	40,99020	36,79550	**39,12578**	**3,74624**
80	58,57220	45,83820	59,26940	53,26120	**54,23525**	**5,37613**

Fig. 7. ARW performance in different size trajectories.

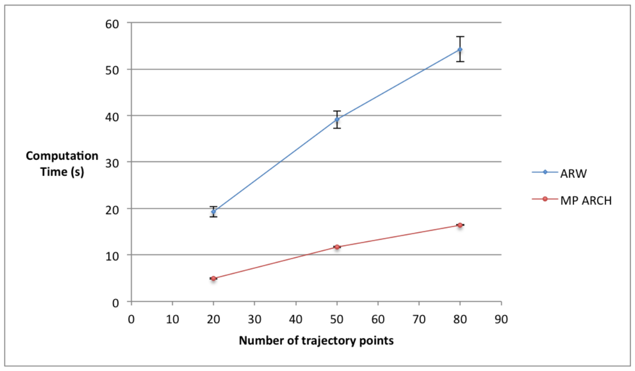


Fig. 8. Comparison between a classic single planner (ARW) and WorkCellSimulator motion planning architecture.

4 Conclusions and future works

The paper presented the development and validation of a motion optimization system for the manufacturing quality inspection process. A robot equipped with a vision system has to cover a complex solid part. To correctly simulate the task execution, WorkCellSimulator was integrated with a module able to plan the minimum coverage of the path according to predefined temporal constraints. As a result, given a set of inspection points and a time within which they must be inspected, the system checks the points validity (i.e. every point must be reachable by the robot, and not cause collisions) and deletes those which do not satisfy such requirements. The ATSP algorithm orders the remaining points and forms the minimum coverage path observing the time constraints. The computed path is the trajectory the robot has to follow to inspect the product.
The use case proves that the system is able to select the most suitable inspection strategy according to the process cycle time.
As future work, we envision to be able to implement dynamic planning. It will include a dynamic path planning, collision avoidance and on-line simulation. In detail, an on-line work-cell simulation will be implemented in order to simulate the path of the robot in real time and to make continuously available information about potential collisions so that a dynamic adaptation of the planned motion will be possible. When the work cell is equipped with a system able to forward dynamic feedback to the simulator, the dynamic planning will adapt the motion of

the robot according to this feedback; in the detection of cracks, for example, if the vision system provides feedback to the simulator, the latter will dynamically and locally adapt the collision-free path of the robot in order to optimize the image quality.

References

1. Thermobot project. `http://thermobot.eu/`.
2. M. Govindaraju A. Mital and B. Subramani. A comparison between manual and hybrid methods in parts inspection. *Integrated Manufacturing Systems*, 9:344–349, 1998.
3. David Applegate, William Cook, and André Rohe. Chained lin-kernighan for large traveling salesman problems. *INFORMS Journal on Computing*, 15(1):82–92, 2003.
4. Stefano Carpin and Gianluigi Pillonetto. Motion planning using adaptive random walks. *Robotics, IEEE Transactions on*, 21(1):129–136, 2005.
5. Roland T. Chin and Charles. A. Harlow. Automated visual inspection: A survey. *Pattern Analysis and Machine Intelligence, IEEE Transaction on*, PAMI-4(Issue: 6):557–573, Nov. 1982.
6. S. Coren and J. S. Girgus. Visual spatial illusions: Many explanations. *Science*, 179:503–504, Feb. 1973.
7. Liangsiri J Corstack H.-A., Hfling M. Simulation in quality management an approach to improve inspection planning. Schottland, 5-8 September 2004.
8. George S. Radford. *The Control of Quality in Manufacturing*. OCLC 1701274. Ronald Press Co., New York, retrieved 2013-11-16.
9. Gerhard Reinelt. Tspliba traveling salesman problem library. *ORSA journal on computing*, 3(4):376–384, 1991.
10. J. W. Schoonard and J. D. Gould. Field of view and target uncertainty in visual search and inspection. *Human Factors*, Feb. 1973.
11. Stefano Tonello, Guido Piero Zanetti, Matteo Finotto, Roberto Bortoletto, Elisa Tosello, and Emanuele Menegatti. Workcellsimulator: a 3d simulator for intelligent manufacturing. In *Simulation, Modeling, and Programming for Autonomous Robots*, pages 311–322. Springer, 2012.
12. J S. C. Wang. Human reliability in visual inspection. *Quality*, Sept. 1974.