

Collision-free motion planning and scheduling

A.G. Gonzalez-Rodriguez^{a,*}, A.Gonzalez-Rodriguez^b

^a*Department of Electronic Engineering and Automation, University of Jaen, Edif. A3, 23071, Jaen, Spain*

^b*Department of Applied Mechanics, University of Castilla-La Mancha, Avda. Camilo Jose Cela s/n, Ciudad Real, Spain*

Abstract

The positioning, motion coordination and test ordering procedures of a new testing equipment for printed circuit boards is presented. The equipment structure consists of four mobile probes and two CCD cameras mounted on two of them to correct the misalignments and for optical inspections. Compared to the inspection systems based on beds of nails, the mobility of the probes entails a decisive advantage for prototypes and short throughputs where maximum flexibility is demanded. The paper explains the algorithms used to move the probes in a general space with obstacles, including the coordination to avoid collisions between probes. It also includes several techniques used for test ordering.

Key words: flying probes, printed circuit testing, Minkowski sum, visibility graph, path planning, motion coordination, traveling salesman problem

2000 MSC: 94C12, 90C35, 68U10, 68Q25

1. Introduction

Quality verification has become one of the essential stages both in the development and the manufacturing process because of total quality management or being required by quality-certified industries as a mandatory task [1]. The identification of defects on Printed Circuit Boards (PCBs) is performed by two classes of procedures: contact/electrical methods and non-contact/non-electrical methods [2][3][4]. Considering contact tests, three main types of tests for the defect detection exist: In-Circuit tests (ICT), Flying Probe tests (FPT) and Boundary Scan tests (BST). The main non-contact test method is the Automated Optical Inspection (AOI).

Considering the electrical tests, the ICT bed-of-nails has traditionally been the preferred equipment to inspect

the electrical parameters of large throughputs due to its quite lower test times [3][5]. However flying probe test systems are the platform of choice for prototype testings, and even in the production process, they are replacing or complementing the in-circuit testers. Many factors contribute to this expansion:

- Fixtureless design, eliminating the expense and time lag of ICT fixture changes.
- Flexibility to introduce changes in the test program, thus keeping pace with product revisions.
- Additional test options may be integrated such as AOI, boundary scan, power up tests...
- Test access in tighter places can be accomplished.
- Flying probe testers can contact pads smaller than ICT testers.

On modern, varied PCBs, implementing a single test or inspection system may offer a reduced amount of test

*Corresponding author

Email addresses: agaspar@ujaen.es

(A.G. Gonzalez-Rodriguez), antonio.gonzalez@uclm.es

(A.Gonzalez-Rodriguez)

coverage. The solution is developing a single platform that implements some combination of test-and-inspection techniques, such as FPT and AOI. Different systems, such as the *SPEA 4040*, the *Flying Scorpion*, the *Seica Pilot* and also the *Flying Probe 3T* from TwoToTest are based on this approach. The computer vision system is also used to assure probe/board alignment.

The electrical probes as well as the cameras for the optical inspection must be mounted in several cartesian arms. They locate the probes at the desired board testing points. The calculation of the probe trajectories demands robust and optimal algorithms that minimize the displacement times eliminating the possibility for collision between themselves and with high devices (transformers, big electrolytical capacitors, dissipators...), that are considered as obstacles.

However, papers dealing with flying probe testing are referred to Multi-Chip Modules (MCM) [6][7][3][8], where no obstacles are present and any point can be accessed by any probe. The flatness of the modules allows a direct path calculation as well as very simple coordination procedures in case of detecting interferences between the movement of the probes [9].

On the other hand, the capabilities of the flying probes appearing in these papers are limited to test opens and shorts in the nets, and therefore they propose heuristic solutions to order the tests according to this functionality. In most of these papers a proposed solution is based on moving only one of the probes while the other probe stays in a fixed test point of the net under verification, thus avoiding coordinating the probe movements.

However no bibliography have been found tackling the complete problem of routing, coordinating and scheduling the flying probes in a system aimed to test integrated circuits or discrete devices in an assembled PCBs. In this case, the estimated time calculated only as a function of the maximum distance travelled by the probes (the L_∞ norm is generally accepted) should be corrected by adding

an additional quantity due to:

- the limited acceleration and deceleration values and
- the time consumed in the command transmission and status reception between the motor control unit and the central computer.

This correction is specially important when the system is not testing MCM substrates but PCBs with assembled devices where obstacles must be taken into account and therefore the probe trajectories can consist of multiple segments.

Furthermore, a coordinated movement of the probes is required in order to avoid collisions. This coordination is very simple in the case of linear direct routes, but entails a more sophisticated planning in the case of general multi-segment routes.

This paper will describe the main features of the system named *Flying Probe 3T*, designed by the firm *TwoToTest*, with regard to the accomplishment of electrical tests, gathering all the procedures required to accomplish the performing of a sequence of electrical tests with the maximum accuracy and speed.

Section 2 describes the overall testing system and its mechanical structure.

Section 3 introduces the procedure to detect the misalignment of a printed circuit board when introduced in the testing system.

Test speed has always been a critical parameter in the production and prototyping of integrated circuits. Different algorithms to reduce the time required to test the board are presented in the next two sections.

In connection with this, section 4 introduces the task of assigning the probes to the test points, shows the algorithm for the calculation of the trajectory that each probe must follow to travel to the next test point avoiding the obstacles, and explains the algorithm to coordinate the starts and stops of each linear movement such that the probes do not collide between themselves.

Section 5 compares the programmed algorithms used to solve the problem of obtaining a sequence of tests that minimizes the total time required to complete all of them. It can be seen as a version of the travelling salesman problem.

Finally, in sections 6 and 7 the work conclusions are discussed, contributing ideas to increase the test speed.

2. Overall description

The exposed system can perform electrical tests and optical tests. The testing possibilities of the presented system determines the required components of this testing machine. It consists of the following elements (see Fig. 1):

- A mechanical structure, where the rest of elements are installed. A damping study is necessary to reduce the vibrations occasioned by the high accelerations and decelerations of the probes in their movements.
- Four Cartesian robotics arm. A probe is mounted at the endpoint of every arm.
- An electrical third-party testing system provided by Checksum.
- A computer vision system for the optical inspection. It is also used for the calibration of the system and to correct the misalignment of the electronic boards when introduced in the testing system. Two different micro CCD cameras (12mm and 24mm) capture the digital images. They are also mounted at the endpoint of two arms.
- An industrial PC for the robot control and the optical tests. Some of its tasks are to calculate and execute the algorithms for the probe movements and to process the images captured by the cameras.

2.1. Cartesian Robotic System

The mobile part of the machine is composed of four Cartesian robotic arms actuated through eight stepping

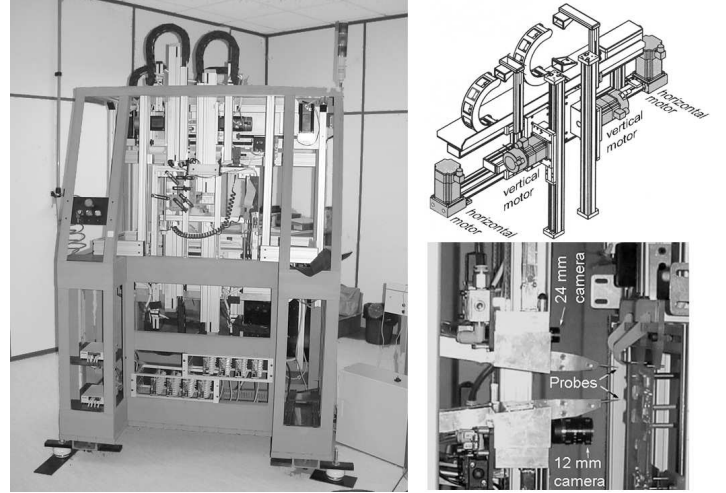


Figure 1: Flying Probe from TwoToTest: overall view (left hand graph), CAD of the positioning mechanism (upper graph) and electrical probes and microcameras (lower graph).

motors, four for horizontal displacements and four for vertical ones. Upper right graph of Fig. 1 is a CAD scheme for the two upper arms, where their four corresponding motors and mechanical guides are shown.

For the control of robot motions two PCI-7334 National Instruments electronic boards have been used. They provide a set of programming primitives to simplify and to speed up the implementation of all the interpolation functions needed for coordinated collision-free robot motions. These interpolations allow accurate motions in diagonal directions.

Each of the four robotic arms has its own testing probe (lower right graph of Fig. 1) for the execution of the electrical tests. Each probe is composed of a pneumatic double effect cylinder, commanded by a two-position five-way solenoid valve, and spring return. A rail guides the movement of the rod to avoid transversal displacements, and a reed sensor is installed at the end of each probe rail. The sensor is activated when the probe reaches the PCB. An improvement in the performance of the overall system can be achieved by substituting the pneumatic system into small servo motors.

Therefore, the structure holding the probe can be seen

as an extensible vertical bar and a rectangular two-piece head where the probe is mounted (see Fig 3, 4). The lengths of the bars are defined by the vertical motors, and the horizontal positions of the structures are defined by the horizontal motors.

On the other hand, each probe is electrically connected to a electronic equipment, responsible for carrying out each electrical test in around thirty milliseconds.

3. Correction of the values provided by the database

All the geometrical information needed for the Flying Probe to operate (pads coordinates and types, boards dimensions, position of obstacles, etc) are provided by a CAD database. Prior to using these values, a correspondence must be calculated between the coordinates of a point such as given by the database and the physical position to be accessed by the probes and the cameras. This involves two transformations: orientation detection and offset estimation.

3.1. Detection of the board orientation

A procedure must be executed in order to detect if the board has been placed in the system by its component side or its weld side. In addition, it is also necessary to determine if the board is rotated with respect to the position defined in the database. To that respect, the user is provided the representation of a cloud of points drawn from the database as well as a visualization of the real board composed by sweeping it with the 12mm camera, the one with the greater field of vision. Then, the user must turn around and rotate this representation until both images match (Fig. 2).

This procedure must be performed only once for the whole throughput.

3.2. Estimation of the board offset

Due to the tolerances in the board contour, the rated dimensions of the board cannot be used to accurately calculate the physical positions of the test points starting

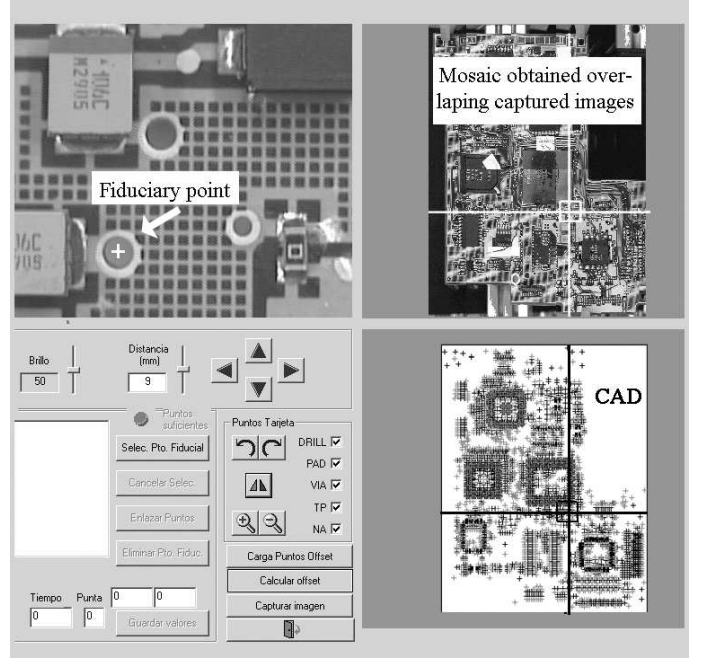


Figure 2: Interface to calculate the orientation and offset of the board.

from their coordinates extracted from the database. In order to assure the maximum accuracy it is compulsory to detect the offset between the actual positions of the test points to be reached for every board and their positions such as provided by the CAD database. Expressing the point coordinates as complex numbers, this offset consists in a displacement (in the form $\mathbf{p}^{\text{off}} = x^{\text{off}} + jy^{\text{off}}$) and a rotation θ , according to

$$\mathbf{p}^e = \mathbf{p}^{\text{off}} + e^{j\theta} \mathbf{p}^{\text{db}} \quad (1)$$

where $\mathbf{p}^{\text{db}}$ defines the point coordinates appearing in the database, $e^{j\theta}$ performs the rotation of the previous vector and $\mathbf{p}^e$ is the theoretical (or expected) position of the point after being affected by the displacement and rotation.

The offset estimation procedure consists of:

- a training operation that must be performed only once for every board model (and preferentially together with the orientation detection), and
- an automatic procedure for every board of the same

throughput that will calculate $\mathbf{p}^{\text{off}}$ and θ thus establishing the relation between the database frame and the probes frame for every particular board to be tested.

This way, the operator must first select a set of n fiduciary points that can be easily identifiable from the represented database and also physically with the 24mm camera (smaller field of vision with a higher accuracy). They should be at least 4 points, preferentially holes or circular elements. The interface of Fig. 2 facilitates this task.

In a second step that must be repeated for every board, the offset due to misalignment in the placement of every board is calculated by automatically moving the 24mm camera to the coordinates where the fiduciary point is expected to be located ($\mathbf{p}_i^e$ of (1) for the fiduciary point i) and measuring its actual (or seen) position $\mathbf{p}_i^s$. Reproducing this measurement for at least $n = 4$ points (theoretically only 2 are necessary), the displacement and rotation of the actual board with respect to the reference system can be calculated minimizing the sum of squares of the differences between both arrays of n expected and actual points. The displacement and rotation can be therefore obtained by means of:

$$\theta = \arg \sum_{i=1}^n (\mathbf{p}_i^s \cdot \mathbf{p}_i^{e*} - (\mathbf{p}_{\text{cg}}^s \cdot \mathbf{p}_{\text{cg}}^e)^*) \quad (2)$$

$$\mathbf{p}^{\text{off}} = \mathbf{p}_{\text{cg}}^s - \mathbf{p}_{\text{cg}}^e \cdot e^{j\theta} \quad (3)$$

where

$$\begin{aligned} \mathbf{p}_{\text{cg}}^e &= \frac{1}{n} \sum_{i=1}^n \mathbf{p}_i^e \\ \mathbf{p}_{\text{cg}}^s &= \frac{1}{n} \sum_{i=1}^n \mathbf{p}_i^s \end{aligned} \quad (4)$$

which, together with (1) allows the control system to access the real the coordinates starting from the database ones.

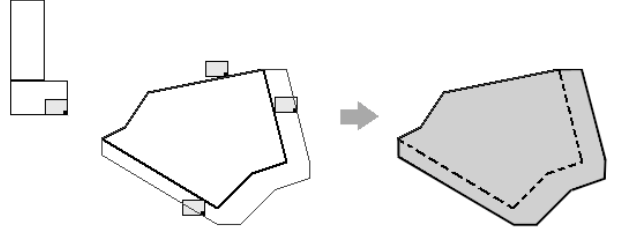


Figure 3: The shape of the obstacles must be enlarged such that the head holding the probe does not touch the obstacle.

4. Positioning algorithms

The task of positioning the probes must be solved in a fast and robust way. The probes must move from the coordinates of the previous test to the coordinates of the next test, avoiding fixed obstacles and the collision with the other probes.

The coordinated movements of the probes has been divided into two steps: calculation of the shorter path to the final point for every probe; and coordination of the movements to avoid collision between probes.

Prior to that, an adaptation of the obstacles must be done. The obstacles are forbidden zones where the probes cannot pass through. High capacitors, transformers or dissipators are considered as obstacles.

4.1. Ampliation of obstacles

From a geometrical point of view, the probe can be assumed to be composed of a extensile bar, an horizontal rectangle separated enough from the PCB, and a small rectangle with the electro-valve housing the probe. This small rectangle is the part of the probe structure that should be avoided to collide with the obstacles. It is grayed in Fig. 3.

The outer corner of this small rectangle defines the reference position for the probe. In Fig. 3, it is the black dot at the lower right corner of the depicted probe.

Depending on which probe is considered, there will be zones close to the obstacle that the previously defined reference position cannot reach without colliding the obstacle

and the grayed rectangle. An expansion of obstacles for every probe allows the probe to become a single point when moving to the next test point.

The final expanded obstacle is calculated by means of the *Minkowski difference*, that for any two sets $S, T \subset \mathbb{R}^n$ ($n = 2$ in the planar case), is defined as

$$S \ominus T = \{s - t \in \mathbb{R}^n \mid s \in S \text{ and } t \in T\}. \quad (5)$$

In our case, S is the obstacle and T is the small grayed rectangle representing the head of the probe. An algorithm for computing it can be found in [10].

4.2. Assignment of a probe to every test point

Since the system includes two micro-CCD cameras mounted on two arms, it will be advisable not to move these arms when a two-point test is to be executed. When a third probe is needed, the system will preferentially use only one of the arms with camera.

Furthermore, the system will dispose the probes in the appropriate positions to avoid collisions or interferences in their movement. As a consequence, the system must analyze all possible assignment combinations between a test point and a probe, in order to:

- assure that a feasible solution is given; the probes does not collide with obstacle or between themselves
- the probes without cameras will be preferentially used,
- the upper left probe will be assigned preferentially the upper left test point, and analogously for the remaining probes and test points.

Fig. 4 shows two different situations where an assignment between probes and test points has been calculated. For the sake of clarity, the test points have been artificially separated. At the left hand graph, a situation with two obstacles is represented. Each probe is assigned the test point closest to its region. At the right hand graph, the inclusion of a third obstacles make the previous solution

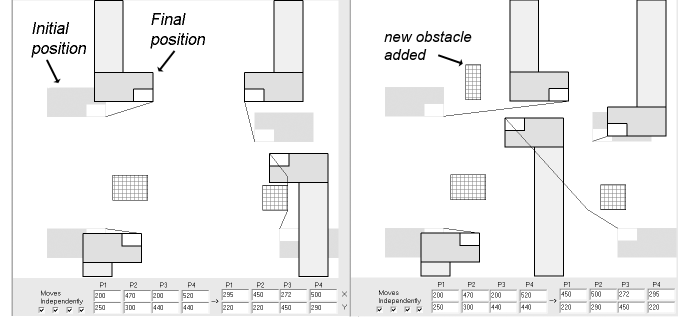


Figure 4: The new obstacle introduced in right hand graph influences the assignment of probes to the test points. Previous positions are drawn without bars and not outlined.

not feasible, and a different combination must be assigned, thus complicating the planning and/or increasing the overall travelling time.

4.3. Individual paths

In the case of PCBs without obstacles, the optimum path from an initial (start) point to a final (goal) one is simply the line joining both points. When obstacles must be considered in the movement of the probe, the calculation of the paths entails a more complicated algorithm. This calculation is based on the condition that the minimum trajectory is composed of straight line segments connecting a subset of the vertices of the polygonal obstacles, together with the start point and the goal one [11][12]¹. The graph composed of all the vertices (and the start and goal points) and the links joining them that do not cross any obstacle is called the *Visibility graph*. A solution to the search of the shortest path in the *Visibility graph* is explained by Hart et al. in [13]. A compilation of different methods for this search can be found in [10].

However a recursive tailored algorithm for the searching of the shortest path has been programmed that allows the control system to include some specific guidelines:

¹This is not strictly true if the increase in time due to accelerations and decelerations when stopping at every vertex is considered.

- The algorithm will preferentially select the vertices located in the forward direction.
- Some features have been included in the algorithm in order to disregard non-feasible and long paths. For example, at every step the algorithm reviews if two or more partial paths lead to the same vertex, disregarding the paths with the higher lengths. This also avoids running forever by cycling through the same vertices.

An objective function has been defined that must be minimized by the search algorithm. It is function of:

- The number as well as the length of the segments must be minimized. The metric used for the calculation of the length is the Chebyshev distance (L_∞ norm), that defines the length of a vector as the maximum of its components. For a 2-dimensional space, the length of segment with end points s and t is defined as:
- $$\begin{aligned} D(s \rightarrow t) &= \lim_{k \rightarrow \infty} \left(|s_1 - t_1|^k + |s_2 - t_2|^k \right)^{1/k} \\ &= \max(|s_1 - t_1|, |s_2 - t_2|) \end{aligned} \quad (6)$$
- The upper left probe will preferentially move in the upper left area. Analogously for the remaining probes. Therefore, the probes will avoid interferences by visiting preferentially the vertices closer to the mother board contour.

4.4. Coordination between movements

Once the individual paths for every probe have been calculated, the trajectory coordination must be solved. This problem has been widely studied as a case of the *multi-robot path planning problem*, where a number of robots move following independently planned paths in a workspace with static obstacles. There are two basic approaches to solve it: *centralized planning* that plans a free path for a composite robot that is composed of the mobile objects;

and *decoupled planning* where all the paths are computed a priori, and in a second stage the movements are coordinated to avoid collisions. Decoupled schemes significantly reduces the amount of computation, at the expense of loss of completeness. This drawback has been evaded in three ways:

- The interferences will be avoided or reduced by a proper assignment of the probes to the test points as pointed out in subsection 4.2 and giving priority to paths closer to the mother board contour (subsection 4.3).
- In the transitions between two tests where a coordination of the paths were not possible, a solution as a last resort consists in taking the probes to their corresponding safe-corner before moving them to their goal positions.
- The test scheduling algorithm in section 5 is responsible of avoiding the high-cost transitions where the obtained coordination is far from the optimum, sequencing the tests in a more efficient way.

Decoupled planning schemes has also been widely studied. The method presented by Kant and Zucker in [11] first calculates the paths among the stationary obstacles and then tunes the velocities to avoid collisions. Its limitations are pointed out by O'Donnell and Lozano-Perez in [14], where they propose a *task-completion diagram*, but it is focused to the motion of two robots. In the *prioritized-planning* of Erdmann and Lozano-Perez [15], the paths of the objects are planned one object at a time, according to predefined priorities. The disadvantage is the loss of completeness, as indicated in the same paper or in [16]. Simeon in [17] proposes a method to coordinate a high number of robots. The search corresponds to Manhattan paths where only one robot moves at once, with a postprocessing step to combine simple robot motions into simultaneous ones, which does not assure the optimum coordination.

The proposed algorithm also belongs to the *decoupled planning* techniques but assures the optimum coordination of the preplanned paths. This is accomplished by creating and exploring a tree composed of sequences of valid complex movements or *stages*, where more than one probe can move simultaneously (see fig. 5 and 6) without risk of collision. The execution of a complete sequence must eventually perform all the elementary movements that take all the probes to their goal positions.

A cost function is defined for every sequence, and the search algorithm must travel the tree searching for the sequence with the minimum cost function. The procedure and confection of the tree are explained below.

For np probes, the complex movements or *stages* are represented as $h \in H = \{1, 0\} \times \dots \times \{1, 0\}$, indicating the elementary movements that are intended to be performed simultaneously. For example, $h_j = (1, 0, 1, 1)$ expresses that at stage j , the probes 0, 2 and 3 will move linearly to the next vertices or positions of their paths.

A *complete sequence* s is defined as the succession of ns stages $h_j \in H$ satisfying that for every probe p

$$\sum_{j=0}^{ns} h_j^p = nm_p \quad \forall p \leq np \quad (7)$$

where nm_p defines the number of segments that the path (for the probe p) consists of. This constrain imposes that all the elementary movements required by every probe to reach the goal positions are included.

Denoting as $v_j^p(s)$ the number of segments travelled at a certain stage j , it can be calculated from:

$$v_j^p(s) = \sum_{k=0}^j h_k^p(s) \quad (8)$$

and can be also seen as the index of vertex (along the path) that the probe is located at, and logically differs from one sequence to another.

Assuming two probes p and q located respectively at vertices m and n of their paths, let us introduce a feasibility mapping f_{qn}^{pm} that is defined to equal 0 or 1 if respectively there is interference in the movement of the probes from

the present point to the next one or not ². A probe p is considered independent in their movement if

$$f_{qn}^{pm} = 1 \quad \forall q \neq p \quad \forall m \leq nm_p \quad \forall q \leq nm_q. \quad (9)$$

and can be excluded from the coordination procedure thus reducing its complexity.

A *valid sequence* is that complete sequence s satisfying that at each stage j

$$h_j^p(s) = h_j^q(s) = 1 \Rightarrow f_{q v_j^q(s)}^p v_j^p(s) = 1 \quad (10)$$

which means that both probes p and q can move from their present vertices $v_j^p(s)$ and $v_j^q(s)$ to the next ones without risk of colliding.

A tree is then formed with the merging of all valid sequences, and a cost function is defined for every sequence

$$c(s) = \sum_{j=1}^{ns} \max_{p=1}^{np} (h_j^p(s) \cdot \text{distance}(v_j^p(s), v_j^p(s) + 1)) \quad (11)$$

where the *distance* is calculated as the maximum difference between the coordinates of the positions of probe p at vertices with indexes $v_j^p(s)$ and $v_j^p(s) + 1$.

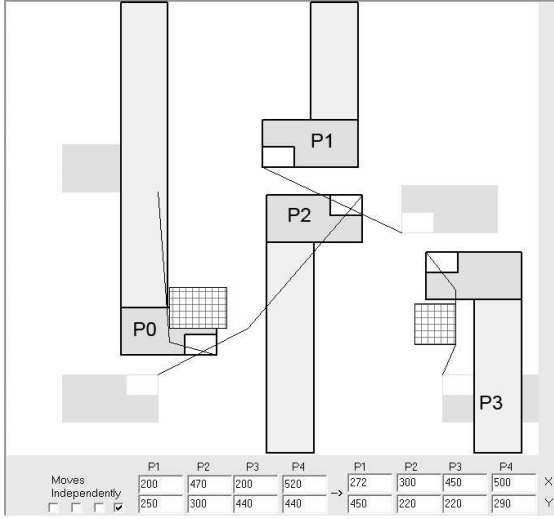
The resulting tree is then travelled searching for the sequence with the minimum cost function. The computation time can be drastically reduced by introducing some few heuristic rules to detect and disregard high-cost sequences.

4.4.1. Example 1

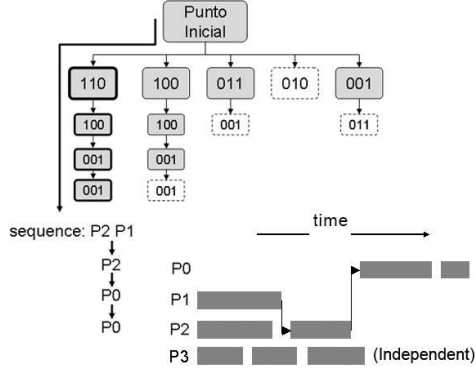
The situation is presented in Fig. 5a) showing the individual paths for the transition from one test to the following one. It corresponds to a forced situation since the assignment routines presented previously should assign the probes to the test points according to a more intelligent criterium (similar to the one shown in Fig. 4).

As previously explained, once the individual trajectories have been calculated, this algorithm analyzes all possibilities of completing all the trajectories and estimates

²For mathematical coherence it can be assumed that the point that follows the goal point is the goal point itself.



a)

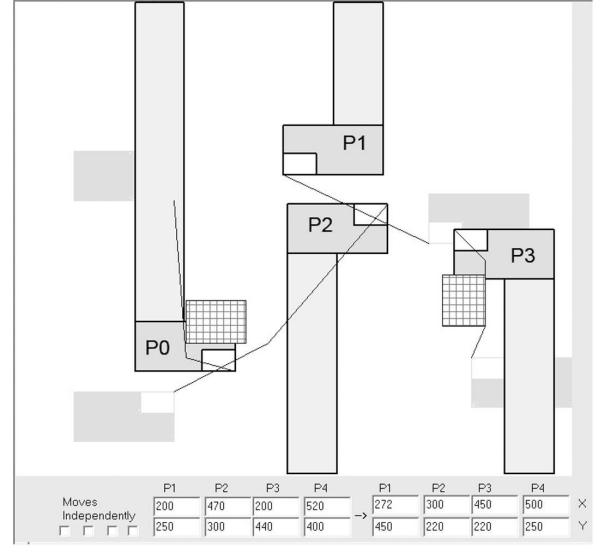


b)

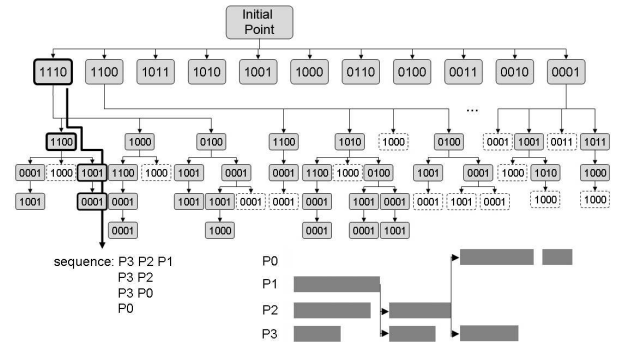
Figure 5: Example 1. Trajectories for every probe, tree and chronograph for the sequence of linear movements. At each node of the tree, the probes with '1' can move simultaneously. Optimum solution in thick line. Non feasible road maps in dashed line and white rectangles.

the invested time. Afterwards, the algorithm chooses the sequence that gives rise to the lowest time.

The elementary movements appear as sequenced nodes in the tree of Fig. 5 b). As can be seen, the algorithm operates in a binary way. The optimum solution yields the one where the probes $P1$ and $P2$ move first (0110), then only the probe $P2$ (0100), then the probe $P0$ (0001), and finally the probe $P0$ again (0001). The probe $P3$ moves independently. There are sequences of movements that cannot complete all the trajectories without colliding. They are represented with a white rectangle in a dashed-line contour at the step when the conflict appears.



a)



b)

Figure 6: Example 2. Change in the y coordinate of the test accessed by probe 3.

4.4.2. Example 2

The possibilities under consideration increase drastically if none of the probe trajectories is independent, and all four probes must be coordinated. This is the case of Fig. 6 a), identical to the previous case except for the position of probe 3.

The new tree and the chronograph is depicted in Fig. 6 b), where only part of the possibilities have been shown.

5. Ordering of the tests

Another algorithm must order the tests to be performed, with the aim of obtaining the optimum test sequence that minimizes the time needed to complete all the programmed tests (typically several thousands of electrical tests).

The task of optimizing the test sequence can be seen as a multi-dimensional Traveling Salesman Problem (TSP), where each test can be treated as a single multi-dimensional city. The objective is to obtain an ordering π_{opt} such that the total travel time is minimized.

$$\pi_{opt} = \min_j \left\{ \sum_{i=0}^N dt(\pi_j(i-1), \pi_j(i)) + \pi_j(N, 0) \right\} \quad (12)$$

where $dt(m, n)$ is the time invested for the probes in their movement from test m to test n , N is the number of tests and the starting position is denoted as test 0 .

For the case of a relatively large number of cities/tests, many papers can be found tackling the TSP providing a near-to-optimum solution. However the field of application of these algorithms is usually restricted to some specific situations and their performance can deteriorate when applying to scenarios with different characteristics.

For the case of multi-dimensional cities, in a space with obstacles, and where a collision-free movement is required for the probes, two different algorithms have been tested: band division (BD) algorithms and nearest-neighbor (NN) algorithm.

5.0.1. Band division algorithms

In this algorithm, the workspace (the electronic board) is divided into stripes, that can be horizontal, vertical or diagonal ones. The width of these stripes inversely depends on the number of tests. The tests are distributed among the bands according to the center of gravity of the locations of the test points (see Fig. 7). Afterwards, the tests of each band are sorted using an improved optimum-search algorithm.

A similar algorithm for substrate testing was used by Crowell et al. in [8] to detect opens and shorts in nets, but with the aim of determining the route of only one probe, while the other probe is located at the extreme end of the corresponding net.

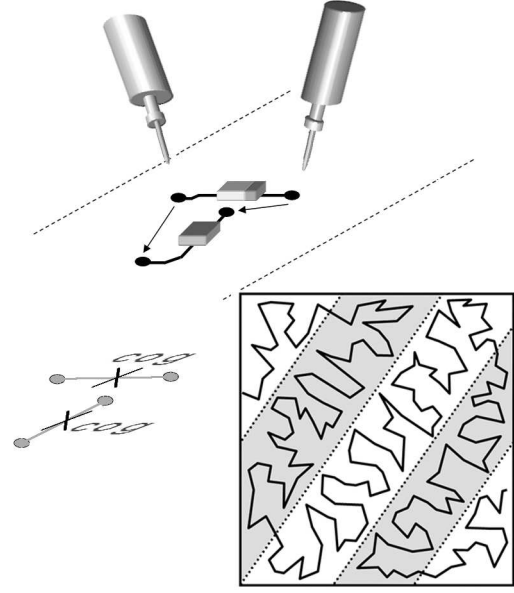


Figure 7: Distribution of the tests according to the center of gravity of the probe coordinates.

5.0.2. Nearest-neighbor algorithm

The idea when constructing the sequence of tests is to go next to the nearest unvisited location [18]. In the case under study, the NN algorithm constructs the sequence π_{NN} by adding the unvisited test which entails less cost (time) for the probes to be positioned at, starting from the previous position. Formally, the added test at the step i , denoted as $\pi_{NN}(i+1)$, is the test k that minimizes

$$\{dt(\pi_{NN}(i), k) : k \neq \pi(j), 1 \leq j \leq i\}. \quad (13)$$

Table 1 illustrates this procedure for a simplified test sequence.

The cost after the test ordering is:

$$Cost = 1224 + 502 + 1126 + 832 + 1200 + 7223 = 12007.$$

Eventually, a reallocating procedure of quasi-isolated tests is performed, searching for an overall cost reduction.

Therefore and referred to this example, the last path from test -6- to test -5- can be considered as a residual one because of its high cost. This suggests reallocating this last test -5-, inserting it between tests -1- and -3-.

Table 1: Test ordering $(0, 4) \rightarrow (4, 2) \rightarrow (2, 1) \rightarrow (1, 3) \rightarrow (3, 6) \rightarrow (6, 5)$

test	0	1	2	3	4	5	6
0		1351	2360	6558	1224	7854	2366
1	1351		1126	832	1665	2336	6552
2	2360	1126		3522	502	4874	3663
3	6558	832	3522		2633	3236	1200
4	1224	1665	502	2633		2555	1452
5	7854	2336	4874	3236	2555		7223
6	2366	6552	3663	1200	1452	7223	

The overall cost is now

$$Cost = 1224 + 502 + 1126 + 2336 + 3236 + 1200 = 9624.$$

Assuming that every test can be identified by a set of probe positions, the first task before executing the nearest-neighbor algorithm is to create the matrix with all the costs (required times) of taking the probes from one test to each of the other tests. For practical purposes, symmetry can be assumed for the distance between two tests. As a consequence $N(N-1)/2$ costs must be calculated, and for every cost, the planning of the trajectories and the probe coordination must be obtained.

5.1. Comparison of the algorithms

From the point of view of the off-line computational time required to execute the ordering algorithm, the band-division (BD) algorithm is the faster one, since there is no need to calculate the routes and distances for every possible transition from one arbitrary test to another. Indeed the distribution of N tests among n_b bands reduces the number of previous route calculations from $N(N-1)/2$ to approximately

$$n_b \frac{N/n_b(N/n_b - 1)}{2}. \quad (14)$$

which yields and approximate reduction of n_b in relation to the NN algorithm.

With regard to the on-line total travel time, the solution obtained from the BD algorithm for a set of 2-probe

tests is similar although a bit higher to the solution given by the NN algorithm. However, for tests involving three or four probes the solution given by the BD algorithm is rather worse, giving rise to a test scheduling with a higher total travel time of the probe heads.

Additionally a genetic algorithm has been compared to the nearest-neighbor one. The travelling salesman problem has become a typical scenario to test evolutionary algorithms. The first researcher to tackle the TSP with genetic algorithms was Brady in 1985 [19], and after him Grefenstette [20] and Goldberg [21], also in 1985.

The genetic algorithm was executed under different conditions, but none of them gave better results nor lower computational time than the nearest-neighbor algorithm. This was probably due to the fact that the genetic algorithm used for the comparison did not apply any local search to the solutions obtained at each generation. Johnson et al. [18] estate that, in the context of the original biological motivation for the genetic approach, the application of local optimization to the individual solutions embodies the discredited Lamarckian principle that learned traits can be inherited. Nevertheless, such local optimization appears to be essential if competitive results are to be obtained for the TSP [22][23].

The genetic algorithms also requires the calculation of the complete cost matrix (or half of it if symmetry is assumed).

In this system, the 2-probe tests have been restrained to the probes without camera whenever possible. Under this restriction, for medium or large throughputs the NN algorithm provides a satisfactory solution in a relatively short time. Unlike the genetic algorithms, this time can also be estimated and visualized during the execution of the ordering algorithm.

6. Results and limitations

After applying this ordering and the algorithms for the positioning of the probes, speeds of about 280 tests per

minute can be reached (1400 two-probe tests, 400 three-probe tests, 200 four-probe tests) . The bottleneck that impede a higher test speed is the bounded value for the acceleration and deceleration. On the one hand, the acceleration and deceleration must be bounded to prevent the stepping motors from loosing steps. This can be solved by using DC motors at the expense of a higher price. On the other hand, the higher deceleration the more marked vibrations in the structure and the probe, and therefore less accurate positioning at the end of the probe. This forces the mechanical engineer to design an efficient damping for the structure.

7. Conclusion

In this work, an advanced machine, that allows both electrical and visual test of electronic board, has been presented. This device is based on the integration of different technologies such as robotics (mechanical probes), vision and electrical testing. The main feature of the proposed system is its versatility, capability of performing electrical and optical tests and its low final cost, offering an important option to the traditional beds of nails.

The basic ideas of the algorithms used to estimate the offset in the placement of the testable board, and to assign and coordinately move the probes have been explained, as well as the algorithm to order the set of tests. This last problem can be seen as a case of the traveling salesman one, and three approaches for their solution have been programmed and compared.

References

- [1] L. Benetazzo, M. Bertocco, and C. Narduzzi, "Networking automatic test equipment environments," *IEEE Instrumentation & Measurement Magazine*, vol. 8, no. 1, pp. 16–21, March 2005.
- [2] M. Moganti and F. Ercal, "Automatic pcb inspection systems," *IEEE Potentials*, vol. 14, no. 3, pp. 6–10, Aug-Sep 1995.
- [3] O. Woodard, "High density interconnect verification of unpopulated multichip modules," in *Proc. 11th IEEE/CHMT Electronics Manufacturing Technology Symposium IEMT*.
- [4] J. M. Soden and R. E. Anderson, "Ic failure analysis: Techniques and tools for quality and reliability improvement," *Microelectronics and Reliability*, vol. 35, no. 3, pp. 429–453, March 1995.
- [5] Y. Zorian, "Fundamentals of mcm testing and design-for-testability," *Journal of Electronic Testing*, vol. 10, no. 1-2, pp. 7–14, Febr. 1997.
- [6] R. Pendurkar, C. Tovey, and A. Chatterjee, "Single-probe traversal optimization for testing of mcm substrate interconnections," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 18, no. 8, pp. 1178–1190, August 1999.
- [7] A. Kahng, "Optimal algorithms for substrate testing in multi-chip modules." [Online]. Available: cite-seer.ist.psu.edu/103660.html
- [8] J. C. Crowell, R. J. Keogh, and J. A. Conti, "Moving probe bare board tester offers unlimited testing flexibility," *Industrial Electronic Equipment Design*, pp. 473–484, Sept. 1984.
- [9] D. G. Manzer, J. P. Karidis, K. M. Wiley, D. C. Bruen, C. W. Cline, C. Hendricks, R. N. Wiggin, and Y.-Y. Yu, "High-speed electrical testing of multichip ceramic modules," *IBM J. Res. Dev.*, vol. 49, no. 4/5, pp. 687–697, 2005.
- [10] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at <http://planning.cs.uiuc.edu/>.
- [11] K. Kant and S. Zucker, "Planning collision-free trajectories in time-varying environments: a two-level hierarchy," in *IEEE International Conference on Robotics and Automation*, vol. 3, April 1988, pp. 1644–1649.
- [12] T. Lozano-Perez and M. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Comm. of the Assoc. of Computing Machinery*, vol. 22, no. 10, pp. 560–570, 1979.
- [13] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Science and Cybernetics*, vol. SSC-4, no. 2, pp. 100–107, 1968.
- [14] P. O'Donnell and T. Lozano-Perez, "Deadlock-free and collision-free coordination of two robotic manipulators," *IEEE International Conference on Robotics and Automation, Arizona*, pp. 484–489, 1989.
- [15] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 3, pp. 1419–1424, Apr 1986.
- [16] J. Latombe, *Robot motion planning*. Boston: Kluwer Academic Publishers, 1991.
- [17] T. Simeon, S. Leroy, and J.-P. Laumond, "Path coordination for multiple mobile robots: a resolution-complete algorithm," *Robotics and Automation, IEEE Transactions on*, vol. 18, no. 1,

pp. 42–49, Feb 2002.

- [18] D. S. Johnson and L. A. McGeoch, *Local Search in Combinatorial Optimization*. London: John Wiley and Sons, 2003, ch. The traveling salesman problem: a case study in local optimization, pp. 215–310.
- [19] R. M. Brady, “Optimization strategies gleaned from biological evolution,” *Nature*, vol. 317, pp. 804–806, 1985.
- [20] D. E. Gregentette, R. Gopal, B. Rosmaita, and D. V. Gucht, “Genetic algorithms for the travelling salesman problem,” *Proc. 1st Int. Conf. on Genetic Algorithms and their Applications*, pp. 160–168, 1985.
- [21] D. Goldberg and R. Lingle, “Alleles, loci and traveling salesman problem,” *Proc. 1st Int. Conf. on Genetic Algorithms and their Applications*, pp. 154–159, 1985.
- [22] T.G.Bui and B. Moon, “A new genetic approach for the traveling salesman problem,” in *Proc. 1st IEEE Conference on Evolutionary Computation*, 1994, pp. 7–12.
- [23] P. Merz and B. Freisleben, “Genetic local search for the TSP: New results,” in *Proc. IEEE Conference on Evolutionary Computation IEEECEP, IEEE World Congress on Computational Intelligence*, 1997, pp. 159–163.