# The Graph Matching Optimization Methodology for Thin Object Recognition in Pick and Place Tasks

Pierre Willaume*†‡§, Pierre Parrend*†§, Etienne Gancel‡
and Aline Deruyver*§
*ICube Laboratory, Université de Strasbourg, France
4 Rue Blaise Pascal, 67081 Strasbourg, France
†ECAM Strasbourg-Europe, Schiltigheim, France
2 Rue de Madrid, 67300 Schiltigheim
‡Hager Group, Obernai, France
132 Boulevard de l'Europe, 67215 Obernai
§Complex System Digital Campus (UNESCO Unitwin) http://unitwin-cs.org
pierre.willaume@etu.unistra.fr

***Abstract*—Bin-picking emerges as a major interest in the industry. The aim is to replace current 'pick and place' systems, where one must place mechanical components in dedicated distribution devices such as bowl feeders for picking them up with a robot arm. A large number of image processing methods are available for recognizing these components. For instance, the stereovision approach provides fine results by comparing several images of the objects taken from different angles. However, when several types of components are available or for thin components, the identification remains a delicate task. We propose the Graph Matching Optimization methodology, which uses graph comparison with evolutionary algorithms between stereoscopic images and a model, in order to identify thin pieces in a constrained time frame. First, we extract characteristic component information by binarization and skeletonization of the images. Then, we retrieve the position of the objects in a 3 three-dimensional space through an evolutionary algorithm derived from Harmony Search Optimisation (HSO). Lastly, we extract and validate optimal parameter ranges for which the devised algorithm shows a high efficiency for representative component positions of randomly arranged thin objects.**

## I. Introduction

A topic of research for many years, 'pick and place' still generates a great interest especially in the industrialization domain [1]. It should enable robot arms to grasp one or several randomly placed objects and put them at a desired location. Many machines exist for simplifying the process, but currently no method permits to grasp any kind of components placed randomly in a given environment in three dimensions, *ie.* to support actual bin-picking. For example, bowl feeder requires a different device for each kind of objects, which makes the process expensive when assembling several components. Another method consists in taking the objects on a conveyor belt, but this system is cumbersome because the objects need to be preset on the conveyor. One of the current methods for recognizing and locating objects randomly arranged on top of each other is comparing the image of one or several cameras with a model of the object. The principal difficulty is the pick part because the system need to recognize and locate the pattern of the objects thanks to input images in order to calculate the position in the space of the object (output). This system saves space, is cheaper and is adaptable to various object types. However, it requires a costly image pre-processing in order to recover the exact three dimensional position of the objects which can be hidden by others. As shown in figure 1, the aim is to recognize and locate a part of the desired object in order to pick it up. However, the objects can be entangled and do not necessarily stands out on the picture.

Fig. 1. Randomly placed objects

The purpose of this paper is to propose an algorithm for detecting randomly arranged thin objects and to optimize the time using an evolutionary algorithm. First, binarization of the image is performed on two stereoscopic images in order to highlight the target objects. Second, in order to highlight the curve of the components, a thinning algorithm is applied to the binary image. Third, the search of junction points and extremum point by graph theory is applied. Finally, an evolutionary algorithm of the Harmony Search family is defined for optimizing point matching between both images, and evaluated. This method allows to retrieve randomly arranged objects without a greedy pretreatment. Calibrated cameras allow to identify the position in the 3D space of the desired object and to calculate the length of the object.

This paper is organized as follows: Section II details the existing state-of-the-art of 'pick and place' methodologies. Section III presents Evolutionary Algorithms and introduces the Harmony Search algorithm family. Section IV presents the Graph Matching Optimization methodology. Section V describes experimental setup and results. Section VI discusses the advantages and the limitations of the proposed method. Section VII concludes this work.

## II. Image processing in Bin-Picking

Bin-picking is a method of choice for recovering disorganized objects [1] and continues to be the subject of significant research breakthroughs, for efficient storing of information [2] or for better discrimination of the component form using structure descriptors [3]. Different scientific fields such as vision, control technologies, information reduction [4] are required to support this process. The performance of the algorithms is usually calculated according to different criteria such as the setup time, which corresponds to the time to build the model, or the detection time, for actual location of the objects, according to the structure of the target components. The identification of an object having a particular structure requires a comparison model. Such a model can be represented by a CAD file, by an image, or by a set of descriptors or characteristics.

### A. Detecting objects using key points

The Scale-Invariant Feature Transform (SIFT) method supports the extraction of objects in images [5] and can be leveraged for real-time location of these objects. It then looks for key points in the component images, and compare them with the different key points of the reference model to find similarities [6], as shown on figure 2. The key points are thus gathered using clustering in order to find the objects. Alternative solutions use derivation of this method such as Speeded Up Robust Features (SURF) [7] or PCA-SIFT [8] to improve computation time and the algorithm capability to find object with distortions. They allow to retrieve common graphic features that stand out, such as marking. The advantage of this approach is the flexibility to find hidden objects, but unfortunately it does not find thin objects because the key points do not stand out.

### B. Detecting objects using feature extraction

Feature extraction enables to classify objects based on learnt characteristics of the images Kumar [9]. The process starts by converting the pixels into greyscale and then transforming the image into binary image. Edge detection is performed in order to highlight the objects shape. In order to preserve high value pixels, a double thresholding is performed. Edge tracking is also performed with hysteresis and followed by filling the dilated image. A blob analysis ends the algorithm, which allows to retrieve the components with a high contrast. However, and as in SIFT, the system is not suitable for the detection of thin objects because the threshold would not stand out the object to take.

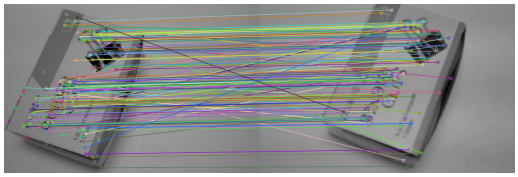

Fig. 2. Sift Algorithm

### C. Detecting objects using geometric models

Reference models are typically available in industrial environments as 3D CAO models. Such models are thus logically a standard reference for 'pick and place' tasks [10]. The 3D model of the object is generated for several orientations in order to retrieve the position of the object in the scene. The generated positions are classified according to the characteristics of the dominant visible faces of the object. This data is classified into an interpretation tree in order to optimize the search time, for instance using relationships between the neighbours. It is then compared to a set of CAD-model templates for retrieving the matching shapes and performing object identification. Recognition of the orientation and position of a 3D part is done using one [11] or several cameras, and a set of CAD files used for generating a hierarchical model of the object structure. These methods prove their efficiency in a large number of applications and are still used by companies. The advantage of the system is to have an easy setup which only requires a CAD file and a calibrated camera. However, the generation of the view of the CAD file in the preprocessing step is time-consuming because the system has to save a set of possible positions of the model.

Efficiency can be greatly improved by not using full images of CAD files as reference models, but abstracted models. Graph representation of object structure enable efficient processing, storage, and pre-treatment of the component image. We therefore propose to apply the graph matching approach to bin-picking for introducing a radical breakthrough in the performance of the object recognition process.

## III. Evolutionary algorithms

### A. Principles

Evolutionary algorithms [12] aim at creating an autonomous intelligence and to adapt its behaviour according to the circumstances [13]. It is inspired by natural selection, first exposed by Darwin [14]. These algorithms use metaphorical concepts inspired by natural evolution. It describes an environment where individuals:

1) Evolve
2) Survive
3) Reproduce

This metaphor implies an environment (*ie.* problem) where individuals (*ie.* candidate solutions) evolve until they build a satisfactory solution. A fitness threshold represents candidate's adaptation to the environment.

Evolutionary Algorithms [15] have been used successfully in different domains like system optimization or machine

learning. Their global advantage is that there is a simple conceptualization of the evolutionary process by the equation:

$$x[t+1] = s(v(x[t])) \tag{1}$$

where x[t] is the population at time t, v is the random variation operator and s is the selection operator. It is interesting to mix different algorithms in order to improve the quality of solutions. The limitations are the lack of guarantee of finding the optimum solution, the expensive computing cost and the usual need for an expensive setting.

All types of Evolutionary Algorithms [16] entail several specific phases:

- Step 1: Randomly Initialize the Population
- Step 2: Evaluate all individuals from the population
- Step 3: Choose the best individuals from population to generate the next generation
- Step 4: Check termination criteria. Either the program stops or it continues to the step 2.

### *B. Harmony Search*

Harmony Search (HS) [17] is a heuristic algorithm inspired by music whose aim is to find the best solution among value generated by an improvisation-like process. In the analogy, a solution is expressed as a harmony, and the value as the note played by the musicians. Its advantages are to be quick, easy to implement and with a fast convergence to quickly find a solution. These advantages are important in the industrial world. The results obtained by the Harmony search algorithm [18] applied to various optimisation problems makes it in any case a serious candidate for obtaining robust and efficient solutions. The algorithm 1 presents the structure of Harmony Search and is separated in five phases which are:

- Step 1: Formulate the problem and set the parameters
- Step 2: Randomly initialize the population
- Step 3: Generate a new solution
- Step 4: Generate the next generation
- Step 5: Check Stopping Criterion

*1) Step 1: Formulate the problem and set the parameters:* The Harmony Search algorithm [19] begins with problem formulation. It allows to define parameters such as the elements of satisfaction (fitness function) and time execution. To execute the algorithm, one has to set following parameters:

- Harmony Memory Size (HMS[1-X]) : Number of solutions handled simultaneously in the algorithm
- Harmony Memory Considering Rate (HMCR[0-1]) : Rate at which the algorithm picks one value randomly from the Harmony Memory
- Pitch Adjusting Rate (PAR[0-1]) : Rate at which the algorithm tweaks the value originally picked from memory

*2) Step 2: Randomly initialize the population:* It consists on randomly generated solutions. The number of solutions generated depends on the Harmony Memory Size (HMS). An example of six solutions is presented on Table I where each solution is scored and saved in the Harmony Memory HM.

TABLE I
RANDOMLY INITIALIZE THE POPULATION

| Number | HM | Score |
|---|---|---|
| 1 | Solution 1 | 10 |
| 2 | Solution 2 | 2 |
| 3 | Solution 3 | 53 |
| 4 | Solution 4 | 21 |
| 5 | Solution 5 | 77 |
| 6 | Solution 6 | 73 |

TABLE II
GENERATE A NEW SOLUTION

| Solution | Score |
|---|---|
| S{X1,...Xij} | 17 |

TABLE III
GENERATE THE NEXT GENERATION

| Number | HM | Score |
|---|---|---|
| 1 | Solution 1 | 10 |
| 2 | Solution 2 | 2 |
| 3 | Solution 3 | 53 |
| 4 | Solution 4 | 21 |
| ~~5~~ | ~~Solution 5~~ | ~~77~~ |
| 6 | Solution 6 | 73 |
| 5 | Solution 5 | 17 |

*3) Step 3: Generate a new solution:* In this part, and as shown on table II, a new solution is improvised. Each elements of the solution Xij can be improved either randomly or thanks to the Harmony Memory (HM). A random value is generated and is compared with the parameter Harmony Memory Considering Rate (HMCR: $0 \leq$ HMCR $\leq 1$) in order to determine whether the value is selected randomly or among the elements of the Harmony Memory (HM). If the value is selected among the Harmony Memory (HM), another random value is compared with the parameter named Pitch Adjusting Rate (PAR: $0 \leq$ PAR $\leq 1$) in order to determine whether the value is mutated by selecting a value close (called neighbour) to that selected. The final result is a set of values which corresponds to a new solution.

*4) Step 4: Generate the next generation:* If the solution is better than the worst solution among the population, replace the element by the new one. As shown in table III, the fifth element in the Harmony Memory (HM) has the worst score with 77. This component is replaced by the new one which has a score inferior of 17.

*5) Step 5: Check Stopping Criterion:* Represented as $stop\_alg()$ on the algorithm 1, if one of the stopping criteria is matched (number of iterations, solution found), the algorithm returns the best solution among the element stored in the Harmony Memory. Otherwise the algorithm goes back to the third step.

**Algorithm 1** Standard Harmony Search algorithm

**Require:** $HMS \in \mathbb{N}^+$, population Size
$HMCR \in$ [0,1), Rate for choosing HM value
$PAR \in$ [0,1), Rate for choosing neighbour value
**Ensure:** $s \in S$, $S$ set of possible solutions
#INITIALIZATION#
HM $\leftarrow \{s_0, s_1, ..., s_{HMS-1}\}$
**for** stop_alg() $\neq$ True **do**
#CREATE A NEW ELEMENT#
$s_{new}$ = IMPROVISATION(HM)
$s_{worst} \leftarrow s_i \in HM | f(s_i) < f(s_j) \forall s_j \in HM, s_j \neq s_i$
#MEMORY UPDATE#
**if** $f(s_{new}) < f(s_{worst})$ **then**
REMOVE $s_{worst}$ from HM
REMOVE $s_{worst}$ to HM
**end if**
**end for**
$s \leftarrow s_i \in HM | f(s_i) > f(s_j) \forall s_j \in HM, s_j \neq s_i$
**return** $s$

### C. Variants of the Method

Different way for choosing parameter ($HMS$, $HMCR$, $PAR$...) values has been proposed. Moreover, some methods have been proposed [20] [21] [22] for improving the results. An overview [23] of the proposed algorithms has been realised. A very important proposition has been done by Mahdavi [24] which propose to include a dynamic Pitch Adjusting Rate (PAR) by the equations:

$$PAR(gen) = PARmin + \frac{(PARmax - PARmin)}{NI} * gn \quad (2)$$

$$bw(gn) = bw_{max}.e^{(c*gn)} \quad (3)$$

$$c = \frac{ln(\frac{Bwmin}{Bwmax})}{NI} \quad (4)$$

Where :

- $PARmin/PARmax$ are the minimum and maximum Pitch Adjusting Rate ($PAR$)
- $NI$ is the number of solutions
- $gn$ is the number of generation
- $Bwmin/Bwmax$ are the minimum and maximum bandwidth

This method allows to find the optimum solution more quickly because the parameter $PAR$ dynamically varies according to the number of iterations. Wang and Juang have proposed [25] another solution which replaces and updates the parameters $PAR$ and $bw$ (distance bandwidth) according to the maximal and the minimal values in $HM$. The equations are:

$$trial^i + [max(HM^i) - trial^i] \times ran[0,1) \quad (5)$$

$$trial^i - [trial^i - min(HM^i)] \times ran[0,1) \quad (6)$$

Where:

- $ran[0,1) \Leftarrow$ random number $\in$ [0, 1)
- $min(HM^i) \Leftarrow$ lowest values of the $i^{th}$ variable in Harmony Memory (HM)
- $max(HM^i) \Leftarrow$ highest values of the $i^{th}$ variable in Harmony Memory (HM)
- $trial \Leftarrow$ selected pitch from Harmony Memory (HM)

Thanks to this method, the search of solution is faster because the parameters $bw$ and $PAR$ are increasing and decreasing according to the worst and the best candidates of the population.

## IV. The Graph Matching Optimization Methodology

This paper proposes an approach based on 3D graph matching using a Harmony Search algorithm in order to find an object from a set of randomly placed overlapping mechanical pieces. Graphs are extracted from the skeletons of objects in the stereo images of the scene and matched with the characteristic graph of the object model. They are used to recover the position of the object in 3 dimensions. The block diagram shown in figure 3 presents the different steps for recovering the object. The first step is the binarization of the image to highlight the desired object. The second step is the application of the thinning algorithm to retrieve the curves of the skeleton of the binarized images. The third step is the graph extraction in order to retrieve the end points and the junction points to match between stereo images. The search of an object is applied in the step four thanks to the evolutionary algorithm in order to reduce the duration of graph matching. This method will provide an optimal solution for the search of objects without matching all nodes. Calibrated cameras are taken into account [26]. This section is organized as followed: Part 1 presents the binarization of the image. Part 2 details the thinning algorithm. Part 3 describes the graph extraction method. Part 4 explains the objects retrieval method by using Harmony Search algorithm.

### A. Binarization

Binarization (bi-level or two-level) is the process for converting a pixel image into a binary image, *ie.* an image with only two possible values (colours) for each pixel. The two colors used are typically the white and the black. The aim is to highlight the objects and to remove the background of the scene. The pixel can be stored in a single bit which is 0 or 1. Algorithm 2 presents the process of binarization. The color of all pixels of the image are compared with a model. A color is represented and is encoded by a system called RGB corresponding to the three primary colors which are Red, Green and Blue. The values are usually between 0 and 255. If the RGB present in the pixel is outside the range transmitted by the model, then the pixel is set to 0. Otherwise, the pixel is set at 1. As shown in figure 5, the binarization of the figure 4 with a threshold set between 100 and 255 brings out the desired object by setting all these pixels in black and all the background in white.

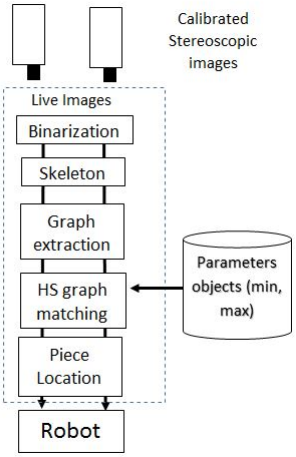


Fig. 3. Diagram of the different steps to find the objects

**Algorithm 2** Binarization of the image

**Require:** $Image$ **and** $Color\_min$ **and** $Color\_max$
  #ITERATE THROUGH THE PIXELS#
  **for** int $x$=0 **to** x=$Image.weight$ **do**
    **for** int $y$=0 **to** x=$Image.height$ **do**
      #COMPARE WITH THE THRESHOLD#
      **if** $Color\_min$ <Image.Pixel.color(x,y) > $Color\_max$ **then**
        #ASSIGN NEW VALUE#
        Image.Pixel.color(x,y) = white
      **else**
        Image.Pixel.color(x,y) = black
      **end if**
    **end for**
  **end for**

### B. Skeletonization

The algorithm of skeletonization [27] aims at representing the structural shape of a plane region by reducing the pattern to a set of curves named skeleton which are centred on the original shape. Topologies properties kept from the original shape are an advantage. Applying the thinning algorithm permits to let thin objects stand out as shown on figure 6. Performance improvement can be achieved by deleting all the points of the image edge with the exception of the skeleton points [28]. The iterations are divided in subiteration to preserve the connectivity of the skeleton. This part which is the foundation of the work is the most important because a poor skeletonization does not allow to find the links between different parts of the right image and the left image.

### C. Graph transform: Extraction of the reference points

In image processing, a graph is a tool to represent an image by a component assembly with a set of relationships [29]. Graph are defined by a two-tuple G=(V,E) where:

- V (nodes) $\rightarrow$ finite set of vertices $V = \{v_1, v_2, v_3...v_n\}$ (n=Card(V))

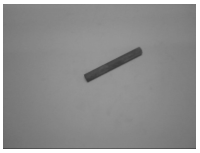

Fig. 4. Input image

Fig. 5. Binarized image ($Color\_min$: 100 and $Color\_max$: 255)

Fig. 6. Thinned image

- E $\rightarrow$ set of Edges $E = \{e_1, e_2, e_3...e_m\}$ (m=Card(E))

In our case, the graph represents the information of the set of shapes of the skeleton image where:

- Nodes $\rightarrow$ endpoints and connection points of each part of the skeleton
- Edges $\rightarrow$ link between the nodes

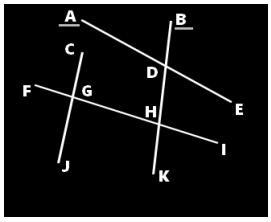


Fig. 7. Graph transform of left image: A and B are selected for object retrieval

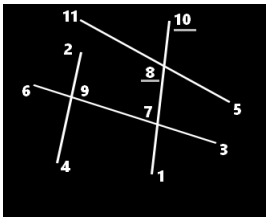


Fig. 8. Graph transform of right image: 8 and 10 are selected for object retrieval

The figures 7 and 8 show the nodes (endpoints, connection points). The recovery of the points is performed thanks to an algorithm that goes through each pixel and compares all pixel and neighbour with a model. The mapping will return whether or not the pixel belongs to a connection point, an endpoint or standard points in the image. This step allows to retrieve the nodes of the skeleton in order to do the graph matching and to search for an object in the next section.

### D. Harmony Search for object retrieval

The extraction step of the object is applied by retrieving the known characteristics of the object. A link between nodes on the right and left image allows to calculate the 3D coordinates of the object and to allow the robot to pick the object. Harmony Search algorithm is applied to calculate the function fitness in order to find the object by matching nodes and by computing the distance between them. The fitness function is calculated according to decision variables which are:

- the equations of the euclidian distance in 2D and 3D :

$$3DS = \sqrt{(Xi - Xj)^2 + (Yi - Yj)^2 + (Zi - Zj)^2} \quad (7)$$

$$2DS = \sqrt{(Xi - Xj)^2 + (Yi - Yj)^2} \quad (8)$$

- the percentage of pixels of $3DS$ belonging to the setted binarized part

The algorithm 3 presents the different steps to follow to find the object. The algorithm needs the data input of $HMS$, $HMCR$, $PAR$ and the distances of the different nodes

to match. The initialization is established according to the parameter $HMS$. For each iteration, a number of nodes is selected for the left and right image. A random matching of the node is performed in order to process the distance in two and three dimension. The figures 7 and 8 shows a random selection of the nodes to match (A and B for the left image and 8 and 10 for the right image). If the points between the right image and the left image does not match, then a penalty is added to the fitness function. It is defined according to the presence of the object between the nodes and according to the difference between experimental and theoretical distance in two and three dimensions. After the initialization and while the stop criteria are not met, a new solution is created. The selection of the nodes depends on the $HMCR$ and $PAR$ parameters. For each node to match between both images, either both nodes are selected randomly or by selecting two nodes matched of the population $HM$. The $PAR$ variable permits to interchange nodes already match in $HM$ by those selected. All the nodes matched between the right image and the left create a new solution. If the new solution is better than the worst in the Harmony Memory ($HM$), then the worst object proposition is replaced by the new one. The stop criterion corresponds either to an object found according to the fitness function or to no object found in the required time.

## V. Experiment

The experimentation has been realised with an Intel(R) Core(TM) i7-5700HQ, CPU 2.70GHz. The OpenCV library has been used to calibrate both webcams of 640x480 pixels. Several cases were studied for the search of objects. The use of the 3D position of the points to compute the distance between the nodes also allows the determination of the coordinates to provide to the robot so that it can grasp the object. The figure 9 and the figure 10 show the representation of the solution of Harmony Search. The detected object allows to retrieve the information of location and tilt in 3D space to give to the robot in order to take the object. The figure 9 shows the search of several thin objects, separated from each other on the same plane. The segment corresponds to the segments between both nodes. The text corresponds to the fitness function retrieved by the algorithm. This value is set according to the kind of skeleton and its environment needs to be under a threshold in order to be identified as an object. The figure 10 represents the search of a piece in context of cluttering and overlapping. In spite of the brightness of the pieces which tangle the ones on the others, the components are properly recovered. This image shows the advantage of the method to find objects that are not fixed in a same three dimensional space as could be bin-picking. The figure 11 represents the time needed by the algorithm to retrieve the object according to the number of nodes. The proposed method has been compare with a standard algorithm where each node is selected randomly. The higher the number of nodes, the longer the processing time. A minimum of 8 nodes was considered to avoid easy search test cases so that the object is not found in the initialization part. The algorithm was applied

**Algorithm 3** Harmony Search algorithm for object retrieval

**Require:** $HMS \in \mathbb{N}^+$, population Size
$HMCR \in$ [0,1), Rate for choosing $HM$ value
$PAR \in$ [0,1), Rate for choosing neighbour value
$V(G1), V(G2) \leftarrow$ Set of nodes of the graph G1 and G2 corresponding to the left and the right images
**Ensure:** $s \leftarrow \{V_0, d_1, ..., d_n\} \in S$, $S$ set of possible solutions and $n$ is a node belonging to an image
**node** $v[m] \in V(G1), w[m] \in V(G2)$, where $m$ is the number of nodes to match between stereoscopic images
#INITIALIZATION#
**for** $i = 0$ **to** $HMS$ **do**
$v[m] \leftarrow$ random nodes $\in V(G1)$
$w[m] \leftarrow$ random nodes $\in V(G2)$
$HM[i] \leftarrow M[v[m], w[m]]$, Population $HM$ where $M$ is the set of nodes
**end for**
#IMPROVISATION#
**while** stop_alg() $\neq$ True **do**
$s_{new} \leftarrow M[v[m], w[m]]$, Population $HM$ where $M$ is the set of nodes
$x_i \leftarrow M[v[0], w[0]]$, Set of two nodes $v$ and $w$
**for** $i = 0$ **to** $m$ **do**
**if** $random[0, 1) \leq HMCR$ **then**
$x_i \leftarrow$ Random $M[v[i], w[i]]$ from $HM$
**if** $random[0, 1) \leq PAR$ **then**
#Inversion of a node compared to a previous#
$tempon \leftarrow v[i-1]$
$v[i-1] \leftarrow v[i]$
$v[i] \leftarrow tempon$
$x_i \leftarrow M[v[i], w[i]]$
**end if**
**else**
$v[i] \leftarrow$ random node $\in V(G1)$
$w[i] \leftarrow$ random node $\in V(G2)$
$x_i \leftarrow M[v[i], w[i]]$
**end if**
**end for**
$s_{new} \leftarrow X\{0, 1..., i\}$
$s_{worst} \leftarrow s_i \in HM | f(s_i) < f(s_j) \forall s_j \in HM, s_j \neq s_i$
#MEMORY UPDATE#
**if** $f(s_{new}) < f(s_{worst})$ **then**
REMOVE $s_{worst}$ from $HM$
REMOVE $s_{worst}$ to $HM$
**end if**
**end while**
$s \leftarrow s_i \in HM | f(s_i) > f(s_j) \forall s_j \in HM, s_j \neq s_i$

100 times for each parameter change in order to recover a consistent average. An iteration takes about 1 millisecond. The pretreatment part (binarization, skeletonization) is not taken into account in the calculation time to avoid disrupting the time processing of harmony search part. The treatment time depends on several factors, in particular the parametrization of the search algorithm. In this case, the $HMS$ parameter has been set to 20, $HMCR$ has been set at 0.7 and $PAR$ at 0.3. As shown in figure 12, the $HMS$ parameter plays an important role in the search of objects. Figure 12 represents the number of iterations as a function of the $HMS$ parameter and the number of nodes present in the image. In our case, $HMS$ has to be set in the range of 20 to 70. If the number is lower, the number of iterations increases dramatically and the algorithm behaves randomly. The figure also shows that after the value 70, the curve of the number of iteration increases ever more. As demonstrated with the time curve (figure 11), the number of nodes impacts on the required number of iterations. $HMCR$ and $PAR$ settings play also an important role in improvisation of the search of the object. As shows the figure 13, the parameter $HMCR$ need to be set between 0.5 and 0.8 and $PAR$ between 0.1 and 0.3 to be in the optimal search configuration. The test has been applied in order to retrieve a component of two nodes in an image of 14 nodes. The $HMS$ parameters has been set to 20.

Fig. 9. Search of objects separated the ones from the others. Blue: Segment between the nodes. Red: Fitness function of the object.

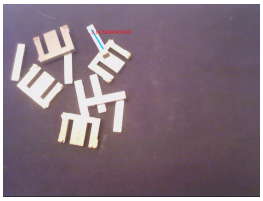

Fig. 10. Search of objects placed randomly with overlapping. Blue: Segment between the nodes. Red: Fitness function of the object.

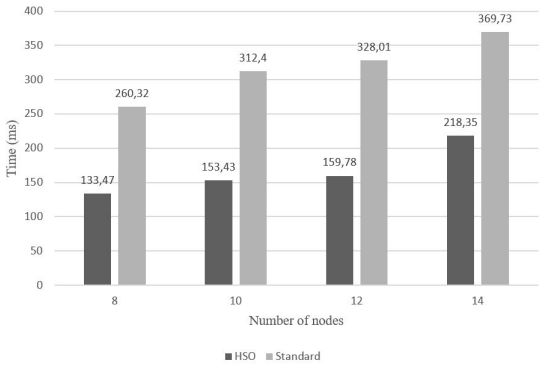


Fig. 11. Comparison of time algorithm between standard and HSO algorithm according to the number of nodes for objects with 2 nodes

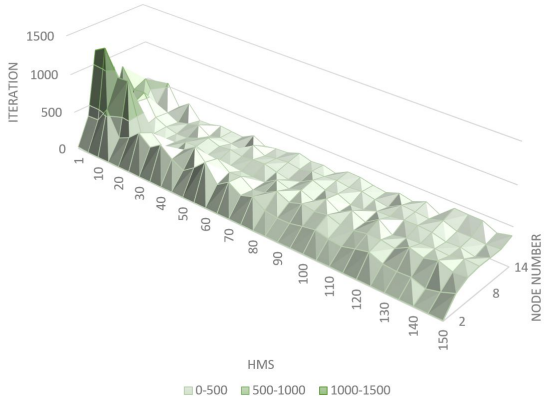


Fig. 12. Performance according to the number of nodes and the $HMS$ parameter

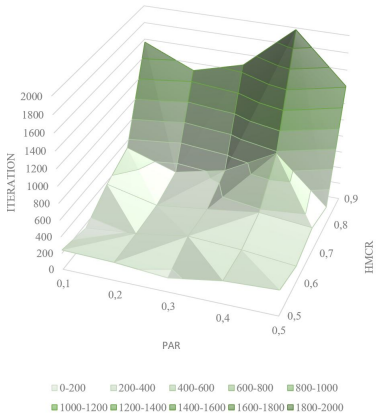


Fig. 13. Harmony Search performance according to the $PAR$ and $HMCR$ parameters

## VI. Discussion

The experimental part proves the efficiency of the proposed method to find thin components placed in a cluttered set. Binarization allows to stand out the objects in the scene. The thinning algorithm allows to retrieve the curve of each component. The graphs allow to retrieve the junction and extrema points. The Harmony Search allows to retrieve the object matching a reference object model. Parameterization plays an important role in optimizing the search for the object. $HMS$ setting should not be too low because research is then very close to a random search. A population too high makes the algorithm less efficient because the system takes time to optimize its search. The efficiency has been demonstrated by the capability to the method to find the object in a time cut off by at least 40% wrt. the process without HSO. The second advantage is the efficiency in setting up the system which is lightweight because only distance needs to be calibrated. The difficulty depends on the number on nodes on the object. The matching duration depends on the number of nodes. However, the actual search duration can be considered as acceptable for operational use for bin-picking in a production line. The method proves its efficiency in finding

thin objects with little nodes. In a future work, an evaluation of the performance on different situation will be studied. The next step of improvement is to adapt the algorithm for any kind of objects and to perform tests with objects with many nodes. The system can be adapted by comparing the shape of the skeleton as well as the shapes of the object using a polygonal approach, which defines the curves between the different nodes of the graph. Comparing the graphs would then allow to compute the fitness function between the model and the images collected by the cameras.

## VII. Conclusion and Perspectives

This paper proposes a method to recognize thin objects randomly disposed in a 3D space in a constrained time frame by using stereo images and an evolutionary algorithm. First a set of filters is applied to each two-dimensional images in order to retrieve the location of the object on the scene. This simplification allows to calculates the size of the objects based on their location in 3D space. For each object on the image, a skeleton is built and Harmony search algorithm is applied to match the junction points and the extremum points. Points are randomly matched in two and three dimensions. The three dimensions enable the graph matching between nodes and the use of Harmony Search algorithm significantly improves its performance. The process has been tested with different objects placed close to each other, with cluttering and overlapping. All objects have been found. The algorithm works with different kinds of noises like brightness and luminosity. The system is quickly initialized because the parameters of the objects are known by the operator. The perspectives of our work is to adapt the system by analysing the edges objects by using a polygonal approximation.

## Acknowledgment

We thank Icube laboratory for assistance with methodology and for comments that greatly improved the manuscript. This research was supported by Hager enterprise. We thank our colleagues who provided a great expertise and a huge insight that greatly assisted the research. We would also like to show our gratitude to the Ecam-Strasbourg Europe for sharing their pearls of wisdom with us during the course of this research.

## References

[1] D. Buchholz, *Bin-Picking: New Approaches for a Classical Problem*. Cham: Springer International Publishing, 2016, ch. Bin-Picking—5 Decades of Research, pp. 3–12.

[2] ——, "Depth map based pose estimation," in *Bin-Picking*. Springer, 2016, pp. 39–56.

[3] C. M. Mateo, P. Gil, and F. Torres, "Visual perception for the 3d recognition of geometric pieces in robotic manipulation," *The International Journal of Advanced Manufacturing Technology*, vol. 83, no. 9, pp. 1999–2013, 2016.

[4] P. J. Sanz, A. P. Del Pobil, J. M. Inesta, and G. Recatala, "Vision-guided grasping of unknown objects for service robots," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 4. IEEE, 1998, pp. 3018–3025.

[5] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2. Ieee, 1999, pp. 1150–1157.

[6] P. Piccinini, A. Prati, and R. Cucchiara, "Real-time object detection and localization with sift-based clustering," *Image and Vision Computing*, vol. 30, no. 8, pp. 573–587, 2012.

[7] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer vision–ECCV 2006*. Springer, 2006, pp. 404–417.

[8] Y. Ke and R. Sukthankar, "Pca-sift: A more distinctive representation for local image descriptors," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2. IEEE, 2004, pp. II–506.

[9] R. Kumar, S. Kumar, S. Lal, and P. Chand, "Object detection and recognition for a pick and place robot," in *Computer Science and Engineering (APWC on CSE), 2014 Asia-Pacific World Congress on*. IEEE, 2014, pp. 1–7.

[10] K. Ikeuchi, "Generating an interpretation tree from a cad model for 3d-object recognition in bin-picking tasks," *International Journal of Computer Vision*, vol. 1, no. 2, pp. 145–165, 1987.

[11] M. Ulrich, C. Wiedemann, and C. Steger, "Combining scale-space and similarity-based aspect graphs for fast 3d object recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 10, pp. 1902–1914, 2012.

[12] P. J. Angeline, "A historical perspective on the evolution of executable structures," *Fundamenta Informaticae*, vol. 35, no. 1-4, pp. 179–195, 1998.

[13] R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *International Conference on Evolutionary Programming*. Springer, 1998, pp. 611–616.

[14] C. Darwin, *The origin of species*. Lulu. com, 1872.

[15] N. M. Al-salami, "Evolutionary algorithm definition," *American J. of Engineering and Applied Sciences*, vol. 2, no. 4, pp. 789–795, 2009.

[16] F. Streichert, "Introduction to evolutionary algorithms," *paper to be presented Apr*, vol. 4, 2002.

[17] Z. W. Geem, J. H. Kim, and G. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[18] Z. W. Geem, "Research commentary: Survival of the fittest algorithm or the novelest algorithm?" *International Journal of Applied Metaheuristic Computing (IJAMC)*, vol. 1, no. 4, pp. 75–79, 2010.

[19] ——, "State-of-the-art in the structure of harmony search algorithm," in *Recent Advances In Harmony Search Algorithm*. Springer, 2010, pp. 1–10.

[20] M. G. Omran and M. Mahdavi, "Global-best harmony search," *Applied mathematics and computation*, vol. 198, no. 2, pp. 643–656, 2008.

[21] N. Taherinejad, "Highly reliable harmony search algorithm," in *Circuit Theory and Design, 2009. ECCTD 2009. European Conference on*. IEEE, 2009, pp. 818–822.

[22] A. R. Yildiz and F. Öztürk, "Hybrid taguchi-harmony search approach for shape optimization," in *Recent Advances In Harmony Search Algorithm*. Springer, 2010, pp. 89–98.

[23] O. Moh'd Alia and R. Mandava, "The variants of the harmony search algorithm: an overview," *Artificial Intelligence Review*, vol. 36, no. 1, pp. 49–68, 2011.

[24] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied mathematics and computation*, vol. 188, no. 2, pp. 1567–1579, 2007.

[25] C.-M. Wang and Y.-F. Huang, "Self-adaptive harmony search algorithm for optimization," *Expert Systems with Applications*, vol. 37, no. 4, pp. 2826–2837, 2010.

[26] Z. Zhang, "A flexible new technique for camera calibration," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 11, pp. 1330–1334, 2000.

[27] R. C. Gonzales and R. E. Woods, "Digital image processing. 2002," *New Jersey: Prentice Hall*, vol. 6, p. 681, 2002.

[28] T. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, no. 3, pp. 236–239, 1984.

[29] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," *International journal of pattern recognition and artificial intelligence*, vol. 18, no. 03, pp. 265–298, 2004.