# VISUAL GUIDANCE OF ROBOTS INTEGRATED IN INTELLIGENT MANUFACTURING

Octavian STOCKLOSA[1], Theodor BORANGIU[2], Silviu RĂILEANU[3], Octavian MORARIU[4], Cristina MORARIU[5]

*Guidance vision is applied as an advanced motion control method, which provides flexibility when integrating robots in intelligent manufacturing cells with unstructured environment. The paper develops a methodology for on-line implementing vision-based robot control strategies that use robot-object models a priori learned, and are on-line checked for collision-free grasping based on the models of the gripper's fingerprints. Experiments have been carried out on a development platform using a Cobra s850 SCARA robot with compact Adept controller and vision extension.*

**Keywords**: Motion control, visual robot guidance, intelligent manufacturing, grasping model, collision avoidance, gripper fingerprint

## 1. Introduction

Tasks in *visual servoing* consist into controlling the motion of the robot in its environment using vision, as opposed to just observing the environment, like in active vision from motion. Visual servoing of robots uses structural features extracted from images as *form*- and *contour image features* for object recognition and locating or collision avoidance. *Context features* may be added to this data to simplify object search at run time. The form- and contour image features refer to the projection of a *body*- or *hole* physical feature of an object (e.g. the part to be grasped, the gripper's fingers or the robot tool) onto the camera image plane [1]. Typical image features are: *edges* and *corners* for contours, respectively the *shape*, *centre of mass*, *orientation* of bodies and holes or *contrived patterns* for form descriptors. Image features must be unambiguously located in different views of the robot scene by different virtual cameras [2]. Visual servo systems typically use one of the following camera configurations [3, 4]: (a) *Stationary*

[1] East Electric Romania, e-mail: octavian.stocklosa@cimr.pub.ro
[2] Dept. of Automation and Applied Informatics, University POLITEHNICA of Bucharest, Romania, e-mail: theodor.borangiu@cimr.pub.ro
[3] Dept. of Automation and Applied Informatics, University POLITEHNICA of Bucharest, Romania, e-mail: silviu.raileanu@cimr.pub.ro
[4] Dept. of Automation and Applied Informatics, University POLITEHNICA of Bucharest, Romania, e-mail: octavian.morariu@cimr.pub.ro
[5] Eng., Cloud Research Department, Cloud Troopers Intl., Cluj-Napoca, Romania

(fixed outside the robot workspace): their location is time-invariant relative to the robot base frame ($x_0$, $y_0$, $z_0$); (b) *Mobile* (arm-mounted or hand-held): their location relative to ($x_0$, $y_0$, $z_0$) is time variant as the robot link on which the camera is mounted moves relative to the world frame.

A systemic view of the robot motion planning and tracking for robot grasping stationary and moving parts is presented in this paper. This view allows the task-oriented management of a scene's foreground and provides part qualifying, robustness to variation in workspace lighting and adaptation to unstructured material flows to be accessed by the robot.

Random scene foregrounds, as the conveyor belt, may need to be faced in robotic tasks. Depending on the parts shape and on their dimension along $z^+$, grasping models $\mathcal{G}s\_m$ are off line trained for object classes. If there is the slightest uncertainty about the risk of collision between the gripper and parts in the scene – touching or very close one to the other –, then *extended grasping models* $\mathcal{EG}\_m = \{\mathcal{G}s\_m, \mathcal{FGP}\_m\}$ must be created by adding the gripper's *fingerprint model* $\mathcal{FGP}\_m$ to authorize part access only after *clear grip tests* at run time.

## 2. Frames, features and servoing taxonomy

Robotic tasks are typically described with respect to more than one frame [3, 4]. The fixed or moving locations of these coordinate frames can be linked via relative transformations, or poses. A robot motion, guided by vision to grasp an object or to interact with it, will be planned from visual data created at run time and mapped into the operational space to provide a pose in which the end-effector accesses the object or interacts with it in a desired manner, imposed by the application task [10].

An analysis of the components of this transformation and of the coordinate frames they interconnect results from Fig. 1 which shows a SCARA robot manipulator and a stationary, down looking camera [5, 6, 7].

The visually planned and tracked motion will guide the robot towards the grasping location of an object of interest identified and located in the image plane ($x_{vis}$, $y_{vis}$). There are 4 components used to compute the end-effector's destination transformation $\mathbf{x}_n^0 = \texttt{part.loc}$ which is relative to the base frame ($x_0$, $y_0$, $z_0$) of the robot:

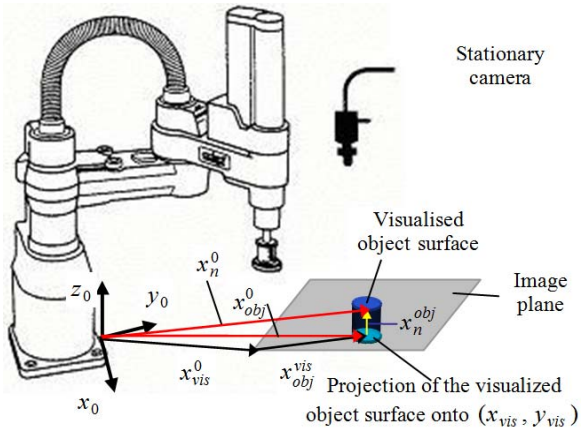


Fig. 1. Stationary camera configuration and related camera – robot relative transformations $x_{obj}^0$, $x_n^0$ respectively for *feature tracking*- and *feature tracking for object grasping*.

- The location offset $\mathbf{x}_{vis}^0 = \texttt{to.cam[cam]}$ of the image frame ($x_{vis}$, $y_{vis}$) with respect to the robot's world frame ($x_0$, $y_0$, $z_0$). This coordinate transformation is created off line by an interactive *camera – robot calibration* program for all virtual cameras *cam* in use.
- The pose $\mathbf{x}_{obj}^{vis} = \texttt{vis.loc}$ of the object-attached frame ($x_{obj}$, $y_{obj}$) relative to the vision frame ($x_{vis}$, $y_{vis}$). This coordinate transformation is computed at run time by the vision processor and describes, in most cases, the location of the frame placed in the centre of mass of the currently identified object, with an orientation expressed by the angle between the minimum inertia axis of the object's 2D image and $x_{vis}$;
- The pose $\mathbf{x}_T^{obj} = \mathbf{x}_n^{obj} = \texttt{grip.trans}$ of the frame $(x_T, y_T, z_T)$ attached to the end-effector with respect to the object's locating frame ($x_{obj}$, $y_{obj}$). This relative transformation is always considered when visually controlling object grasping, to recuperate the third dimension (the depth of view) which was lost during image acquisition by the 2D projection;
- The pose $\mathbf{x}_T^F = \mathbf{x}_n^F = \texttt{TOOL}$ of the end-effector frame $(x_T, y_T, z_T)$ with respect to the coordinate frame ($x_F$, $y_F$, $z_F$) which is attached to the tool mounting flange of the manipulator.

*Image feature parameters* represent real-valued quantities that are computed from one or several image features. Typical feature parameters used in visual servo control are: the image plane coordinates $u, v$ of visualised object points $\mathbf{P}^{vis}$; the distance between two image points, $\text{dist}(\mathbf{P}^{vis}, \mathbf{Q}^{vis})$ and the orientation of the line connecting these two points, $\sphericalangle(\mathbf{P}^{vis}\mathbf{Q}^{vis}, x_{vis})$; the parameters

of the gripper's fingerprints projection on the image plane: shape, area, location, etc.

Hence, a real-valued mathematical or logical expression having as arguments one or more image feature parameters may be used to dynamically update the description of an object (part to be tracked / grasped) or robot tool. Image feature parameters can be stored in numerical form $f_i$ (possibly bounded); they map the *object space* into the *feature parameter space* $\mathcal{F}$, by means of feature parameter vectors $\mathbf{f} = [f_1 \quad f_2 \quad \dots \quad f_k]^{\mathrm{T}} \in \mathcal{F} \subseteq \mathbf{R}^k$. Any feature parameter $f_i$ takes values in $\mathcal{F}_i$. The set of all $k$ chosen feature parameters defines a function which maps the object space into the Cartesian product $\mathcal{F}_1 \times \mathcal{F}_2 \times \cdots \times \mathcal{F}_k$. Thus, a certain object to be visually tracked and grasped, or a gripper (tool) to be visually positioned will appear as a point in the $k$-dimensional feature space.

The mapping from the location of the end-effector to the set of corresponding image feature parameters, computed according to the projective geometry of the camera will be denoted by $\mathbf{F}: \mathcal{T} \rightarrow \mathcal{F}$. Considering $\mathcal{F} \subseteq \mathbf{R}^2$ the feature parameter space of $u, v$ coordinates of the projection of some point $\mathbf{P}^{vis}$ onto the image plane ($x_{vis}$, $y_{vis}$), then, assuming perspective projection, $\mathbf{f} = [u \quad v]^{\mathrm{T}}$.

The pair "physical camera-virtual camera (a data set describing the task - driven context in which the physical camera's information will be interpreted)" is related to the base coordinate system of the robot by the time-invariant pose $x_{vis}^0$ evaluated a single time during an interactive off line camera-robot calibration session, and to the object in the scene by $x_{obj}^{vis}$.

The camera image of the object $x_{obj}^{vis}$ is independent of the robot motion (unless the target is the end-effector itself, described for example by image feature of the gripper's fingerprints projected onto the image plane). The pose $x_{obj}^{vis}$ is variable in time for parts travelling on conveyor belts; the computation of this information is done at run time, and involves the *search*, *recognition* and *locating* of the image features(s) on the object of interest [2, 3].

For object grasping, the image features must unambiguously describe the entire object for its successful identification and locating at run time. In addition, the pose $x_n^{obj}$ of the gripper, relative to the frame attached to the object in its current location, is required.

For a *stationary camera*, the relationship between these poses is:

$$\mathbf{x}_{obj}^0 = \mathbf{x}_{vis}^0 : \mathbf{x}_{obj}^{vis}, \text{ for (a) feature tracking}$$

$$\mathbf{x}_n^0 = \mathbf{x}_{vis}^0 : \mathbf{x}_{obj}^{vis} : \mathbf{x}_n^{obj}, \text{ for (b) feature tracking}$$

for object grasping

The taxonomy of visual servo architectures is defined by classifying robot-vision systems according to the *type of structure* of the closed-loop motion controller and to its *type of control law*:

*Hierarchical motion control structure* [8, 9], with the vision processor providing set-points as reference input to the robot's joint-level controller – thus using joint data feedback to internally stabilise the robot. This structure corresponds to the *interlaced look-and-move* control schemes, where motion tracking and image processing are *pipelined* (Fig. 2).

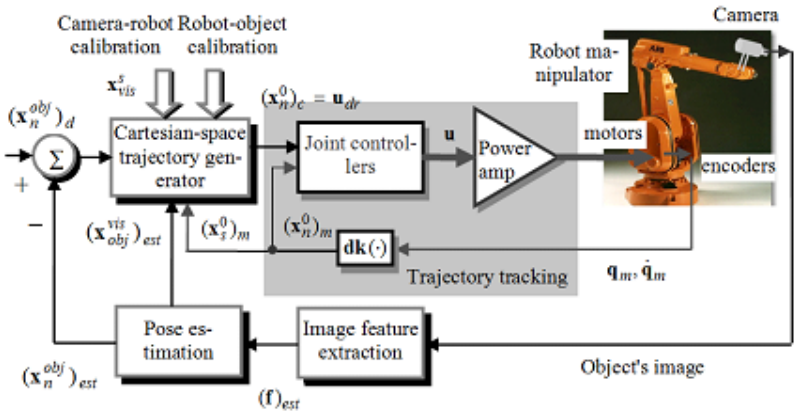


Fig. 2. Position-based look-and-move servoing scheme for object tracking

This is a *position-based control structures*, using error signals defined in task space coordinates. Features are extracted from images and their associated parameters are computed and used in conjunction with a geometric model of the visualised object and of the known camera-robot ensemble, to estimate first the pose of the object with respect to the camera. Feedback is then computed by reducing errors in estimated pose space, and applied to the task-space trajectory generator.

*Position-based look-and-move control* will be used, according to the system structure in Fig. 2. Features are extracted from the image and used to estimate the pose $\hat{\mathbf{x}}_{obj}^{vis} = (\mathbf{x}_{obj}^{vis})_{est}$ of the target (object, point) relative to the camera. Using these values, an error between the current estimated and the desired pose of the robot $(\mathbf{x}_{obj}^{vis})_d$, is defined in the task space $\mathcal{T}$. Thus, position-based control neatly separates the control actions, i.e. the computation of the feedback signal $(\mathbf{x}_s^0)_m = \mathbf{dk}(\mathbf{q}_m, s),\ n \ \ 3 \le s \le n$ using the direct kinematics model $\mathbf{dk}(\ )$ of the manipulator, from the estimation problem involved in computing position or pose $\hat{\mathbf{x}}_{obj}^{vis}$ from visual data $(\mathbf{f})_{est}$.

Definition. *A visual positioning task* is expressed by an error function $\mathbf{E}: \mathcal{T} \rightarrow \mathbf{R}^m$. This function is referred to as *virtual kinematic error function* VKE. A positioning task is fulfilled when the end-effector has been moved in pose $\mathbf{x}_n = \mathbf{x}_n^0$ if $\mathbf{E}(\mathbf{x}_n) = \mathbf{0}$.

Once a suitable VKE function is defined and its parameters are instantiated from visual data, a compensator can be designed that reduces the value of the VKE function to zero. This compensator computes at every sampling time instant the necessary end-effector position $(\mathbf{x}_n)_c$ that is sent as dynamic reference $\mathbf{u}_{dr}$ to the joint-space (operational - space) motion tracking controller. Since the VKE functions are defined usually in the Cartesian space, it is common sense to develop the compensator's control law through geometric insight.

## 3. Motion control based on pose-feature extraction from fixed camera

In industrial applications of position-based dynamic look-and-move control structures, the robot-vision system works with offline learned objects which can be visually recognised and located at runtime [10]. It becomes thus possible:

- To recover the object's pose, $\hat{\mathbf{x}}_{obj}$, relative to the base frame of the robot, from the direct estimate $\hat{\mathbf{x}}_{obj}^{vis}$ of the object's pose in the vision frame and by composing it with the camera-robot calibration estimate $\hat{\mathbf{x}}_{vis}$;
- To define stationing points $\mathbf{S}^{obj}$ on the object's image, relative to a suitable object-attached frame ($x_{obj}$, $y_{obj}$).

Assuming a random part presentation in the robot workstation, the object's pose relative to a (unique) camera frame $\hat{\mathbf{x}}_{obj\_1}^{vis}$, will be estimated at run time, in a *first stage* in terms of the following image feature parameters:

- $x_C, y_C$: coordinates of the centre of mass C of the 2D projection of the object's visualised surface onto the image plane $(x_{vis}, y_{vis})$;
- $orient = \angle(\text{MIA}, x_{vis})$: orientation angle of the object.

The object-attached frame $(x_{obj\_1}, y_{obj\_1})$ will have the origin in **C** and the abscissa $x_{obj\_1} \equiv \text{MIA}$ (Fig. 3).

To grasp objects of a certain type always in the same way, irrespective of their location in the robot scene, the desired (unique) pose of the gripper, $\mathbf{x}_{n^*}^{obj}$, relative to the object-attached frame must be a priori learned.

Let us denote by **G** the projection of the end-tip point **T**, the origin of the gripper frame $(x_n, y_n, z_n)$, onto the image plane: $\mathbf{G} = \text{proj}|_{(x_{vis}, y_{vis})} \{\mathbf{T}\}$.

For a desired grasping style [11], $\mathbf{G}^{obj\_1}$ is a stationing point in the object's coordinates $(x_{obj\_1}, y_{obj\_1})$, irrespective of the current position and orientation of the object. Its coordinates are $x_G = d_{CG} * \cos(alpha)$; $y_G = d_{CG} * \sin(alpha)$,

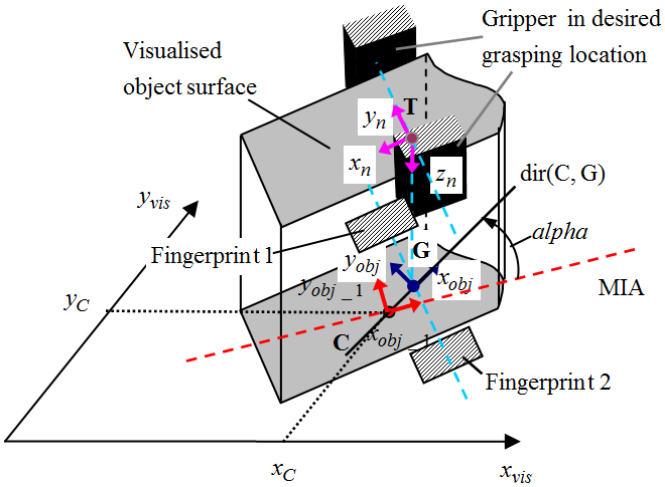


Fig. 3. The 2-stage definition of the object-attached frame

where $d_{CG} = \text{dist}(\mathbf{C}, \mathbf{G})$, $alpha = \angle(\text{dir}(\mathbf{C}, \mathbf{G}), \text{MIA})$ measured CCW from MIA to the direction CG, i.e. $\text{dir}(\mathbf{C}, \mathbf{G})$.

In a second stage, the object-attached frame will be shifted to origin **G**, by a translation of distance $d_{CG}$ along $\text{dir(MIA)}$ followed by a rotation of angle *alpha* about the normal in **C** to the image plane, as represented in Fig. 3.

Given an object pose, $\mathbf{x}_{obj}^{vis}$, estimated visually at run time, and assuming that the object was recognized as a member of that class for which a relative grasping pose $\mathbf{x}_{n^*}^{obj}$ was a priori learned using a stationary camera calibrated to the robot base frame by $\mathbf{x}_{vis}$, then the positioning error can be defined by the VKE function:

$$\mathbf{E}(\mathbf{x}_n; \tilde{\mathbf{x}}_{n^*}^{obj}, \hat{\mathbf{x}}_{obj}^{vis}, \hat{\mathbf{x}}_{vis}) = \mathbf{x}_{n^*}^{n} = \hat{\mathbf{x}}_0^n : \hat{\mathbf{x}}_{vis} : \hat{\mathbf{x}}_{obj}^{vis} : \tilde{\mathbf{x}}_{n^*}^{obj}$$

where:

$$\tilde{\mathbf{x}}_{n^*}^{obj} = \begin{cases} \mathbf{x}_{n^*}^{obj} \text{ a priori known from learning, } \textit{particular} \text{ "grasping style"} \\ \hat{\mathbf{x}}_{n^*}^{obj} \text{ visually updated at run time, } \textit{general} \text{ "grasping style"} \end{cases}$$

With an EOL system, $\hat{\mathbf{x}}_0^n = \text{inverse}(\hat{\mathbf{x}}_n^0)$ will be dynamically updated by the trajectory generator to bring to zero the positioning error $\mathbf{x}_{n^*}^n$. This can be simply done applying for an IK-based Resolved Motion Rate Control algorithm.

The closed-loop servo control uses the visually estimated pose of the object, $\hat{\mathbf{x}}_{obj}^{vis}$, the estimated camera-robot calibration pose, $\hat{\mathbf{x}}_{vis}$ and assumes that reduced-error direct kinematics ($\hat{\mathbf{x}}_n^0$) – and inverse kinematics ($\hat{\mathbf{x}}_0^n$) models are available.

As for the imposed grasping pose, for a priori unknown object location in the scene, some components in $\hat{\mathbf{x}}_{n^*}^{obj}$ must be estimated at run time whenever the "grasping style" is *general*, i.e. such that $\mathbf{G} \not\equiv \mathbf{C}$ and $\mathbf{G}$ does not lie on MIA.

With an ECL system capable to observe both the object ($\hat{\mathbf{x}}_{obj}^{vis}$) and the end-effector ($\hat{\mathbf{x}}_n^{vis}$) and to estimate their poses, the error equation (4) becomes:

$$\mathbf{x}_{n^*}^n = \hat{\mathbf{x}}_{vis}^n : \hat{\mathbf{x}}_0^{vis} : \hat{\mathbf{x}}_{vis}^0 : \hat{\mathbf{x}}_{obj}^{vis} : \tilde{\mathbf{x}}_{n^*}^{obj} = \hat{\mathbf{x}}_{vis}^n : \hat{\mathbf{x}}_{obj}^{vis} : \tilde{\mathbf{x}}_{n^*}^{obj}$$

For an ECL system the uncertainties in both the robot's kinematics models and in the camera-robot calibration model do not affect the positioning accuracy of the global system, since the corresponding terms dropped out of the error equation.

## 4. Learning procedure for robot-object model parameters

To calculate the set of robot scene parameters for object grasping, an off line training scheme will use interactively pose- and point-*image features* (centre of mass and orientation in the object's image) and *robot points* learned in Cartesian space.

The following four robot scene parameters completely specify the relative gripper-object grasp pose [12]: (1) the grasping "height", *ht*; (2) the gripper's opening, *openg*; (3) the *xy* offset, *d*, of the grasping point $G=proj|_{(xvis,yvis)}\{T\}$ relative to the mass centre *C* of the object's image (*T* - end tip point); (4) the roll angle of the gripper, φ, relative to an orientation axis of the object (in the sequel the minimum inertia axis (MIA) of the object's image. These 4 grasping parameters *(ht, openg, d, φ)* are particular for each class of objects and "grasping style" and will be therefore learned and stored in related class records.

The parameters of a grasping transformation $\mathbf{x}_n^{obj}$ for a given class of objects are hence the components of the grasping style model, off line trained. They map at run time the shape of an object recognised and located in the image scene into an object-related grasping description, part of the robot scene.

The $\mathcal{G}s\_m(O)$ model, for an object *O* and desired grasping style, is function of the object-related image data [C, MIA(*O*)]:

*Gs_m = {x_off, y_off, rz_off,z_off}*, where:

- *x_off, y_off* are the position offsets of the user-defined object frame ($x_{obj}$, $y_{obj}$) which will be automatically attached by vision at run time to a recognised instance of *O*– relative to the object's centre of mass;
- *rz_off* is the roll offset angle $\angle(x_{vis}, x_{obj}) = \angle(x_{vis}, x_n)$ of the user-defined object frame relative to $x_{vis}$, which coincides with the roll orientation of the gripper in the object grasping location;
- *z_off* is the value offsetting the gripper along $z^+$ from the image plane; if the object coordinate system's orientation is trained with *z_off* such that $x_{obj} \parallel x_n$ for a desired grasping style, and if *x_off*, *y_off* and *rz_off* are automatically applied at run time to the recognized instance of class object *O*, located from image data [C, MIA], it results that *z_off* is the only component (always non-zero) of the grasp transformation `grip.trans` to be composed at run time with the camera-robot calibration transformation and the current object pose (returned by vision) for effective robot access to the part to be grasped.

Learning the robot-object scene parameters *x_off*, *y_off*, *rz_off* the *x*, *y*, *roll* offsets of the object frame relative to its centre of mass and MIA) and *z_off* (the *z* offset along $z^+$ of the grasping transformation) of the $Gs\_m$ model was done in an interactive session which uses the robot itself as a measuring device (encoder data and features extracted from image data) and the user-defined numeric computation. An object was modelled with the name "LA"; the learning procedure is exemplified in Fig. 4 for the $V^+$ robot programming environment:

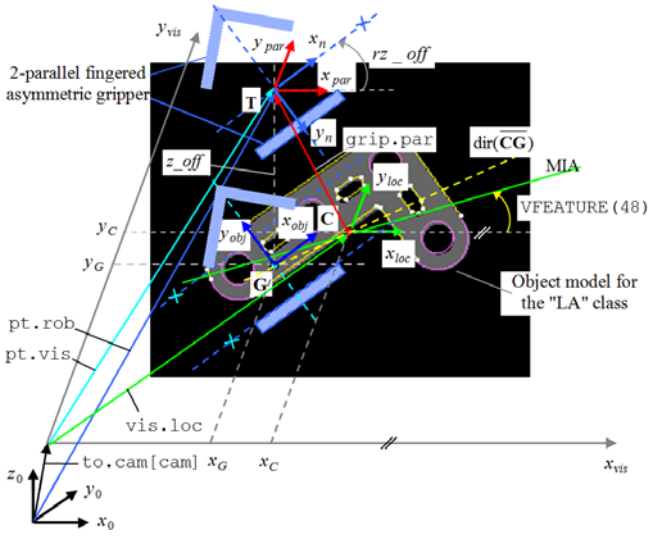


Fig. 4. Learning in $V^+$ the location offsets *x_off, y_off, rz_off* of the object-attached frame and the component *rz_off* of the grasping transformation on the "LA" model

1. Place the model in the camera's field of view. Because the object class "LA" is not yet modelled, it is a *blob*.
2. Use the teach box, the robot is manually moved with open gripper in the desired grasping location of the blob.
3. `DO CLOSEI`
   `DO OPENI`
   `DO CLOSEI` (eliminate part dragging).
4. `HERE pt.rob`: The current robot location is stored in `pt.rob`, which expresses the position and orientation offsets of the gripper-attached frame ($x_4$, $y_4$, $z_4$) with respect to the robot base frame ($x_0$, $y_0$, $z_0$).
5. The gripper is opened, and the robot is moved outside the camera's field of view in an a priori learned point, `safe`:
   ```
   DO OPENI
   DO DEPARTS 100
   DO MOVE   safe
   ```
6. Compute the location of the gripper's frame ($x_4$, $y_4$, $z_4$) relative to the vision frame ($x_{vis}$, $y_{vis}$). The camera with the ID `cam` is used; the result is assigned to the transformation variable pt.vis :
   ```
   DO SET to.cam[cam]:pt.vis = pt.rob
   ```
7. Accessing the components of pt.vis :
   ```
   DO DECOMPOSE arr.pt[] = pt.vis
   ```
8. Visualising the blob and locating it in the image plane, without having moved it in the scene meanwhile:
   ```
   VPICTURE (1, -1) -1,1
   DO VLOCATE (1, 2) "?", vis.loc
   ```
   Because a blob was detected and located, its vision location returned in the variable `vis.loc` has the *x, y* coordinates of the blob's centre of mass $\mathbf{C}(x_C, y_C)$ and a null orientation, $rz = 0$, i.e. $\angle(x_{loc}, x_{vis}) = 0$.
9. Accessing the components of vis.loc:
   ```
   DO DECOMPOSE arr.c[] = vis.loc
   ```
10. Computing the $xy$ offsets of the user-defined model reference frame relative to the default reference frame of the model, based on its centre of mass :
    ```
    DO x_off = arr.pt[0]-arr.c[0]
    DO y_off = arr.pt[1]-arr.c[1]
    ```
11. Compute the $z^+$ offset as non zero parameter of the grasp transformation to be added at run time for recognised model instances:
    ```
    DO z_off = arr.pt[2]
    ```

12. Compute the *roll* offset angle *rz_off* of the user-defined model reference frame. A relative transformation `grip.par` is defined for the actual location of the blob, expressing the offsets of the frame $(x_{par}, y_{par})$ attached in the gripper's terminal point **T**, and having its axes parallel to those of the vision frame $(x_{vis}, y_{vis})$, i.e. $x_{par} \parallel x_{vis}, y_{par} \parallel y_{vis}$:

```
DO SET grip.par =
    TRANS(x_off,y_off,z_off,0,0,0)
```

The roll offset *rz_off* is evaluated from:

```
DO SET to.cam[cam]:vis.loc:grip.par
    :rot_rz = pt.rob
DO DECOMPOSE arr.roll[] = rot_rz
DO rz_off = arr.roll[5]
```

It can be observed that x_*off* = $x_G$ - $x_C$, *y_off* = $x_G$ - $x_C$ and $rz\_off = \measuredangle(x_4, x_{vis})$; they will be input as "*xy-roll*" offsets in the mode 5 `VTRAIN.FINDER` training operation of V$^+$, and stored. At run time the system will use them to automatically shift and rotate the instance-attached frame from the default model frame (in C and with zero rotation).

As for the $z^+$ offset value *z_off*, it allows defining the grasping transformation to be applied at run time to access any recognised and located "LA" instance: `DO SET grip.la = TRANS(0,0,z_off,0,180,0)`.

## 5. Fingerprint modelling for collision-free grasping

If there is the slightest risk of collision between the gripper and parts on the belt - touching or close one relative to the other -, *extended* grasping models $\mathcal{EG}\_m = \{\mathcal{G}s\_m, \mathcal{FGP}\_m\}$ were created by adding the gripper's *fingerprint model* $\mathcal{FGP}\_m$ to authorize part access at run time only after *clear grip tests*.

Definition. $\mathcal{MFGP}\_m(\mathcal{G}, O) = \{\mathcal{FGP}\_m_1(\mathcal{G}, O), ..., \mathcal{FGP}\_m_k(\mathcal{G}, O)\}$ is defined as *multiple fingerprint model* for a $p$-fingered gripper $\mathcal{G}$ and a class of objects $O$, describing the shape, location and interpretation of $k$ sets of $p$ projections of the gripper's fingerprints onto the image plane $x_{vis}$, $y_{vis}$ for the corresponding *k* grasping styles $\mathcal{G}s\_m_i, i = 1,...,k$ of $O$-class instances. A $\mathcal{FGP}\_m_i(\mathcal{G}, O)$ model has the following parameter structure [13]:

- $finger\_shape(\mathcal{G}) = number, shape_i, size_i, i = 1,..., p$, expresses the shape of the gripper in terms of its number $p$ of fingers, the shape and dimensions of each finger. *Rectangular*-shaped fingers are considered; their size is given by "width" and "height";

- $fingers\_location(G,O) = \{x_{ci}(O), y_{ci}(O), rz_i(O)\}, i = 1,..., p$, indicates the relative location of each finger with respect to the object's *centre of mass* and *minimum inertia axis* (MIA). At training time, this description is created for the object's model, and its updating will be performed at run time by the vision system for any recognized instance of the prototype.

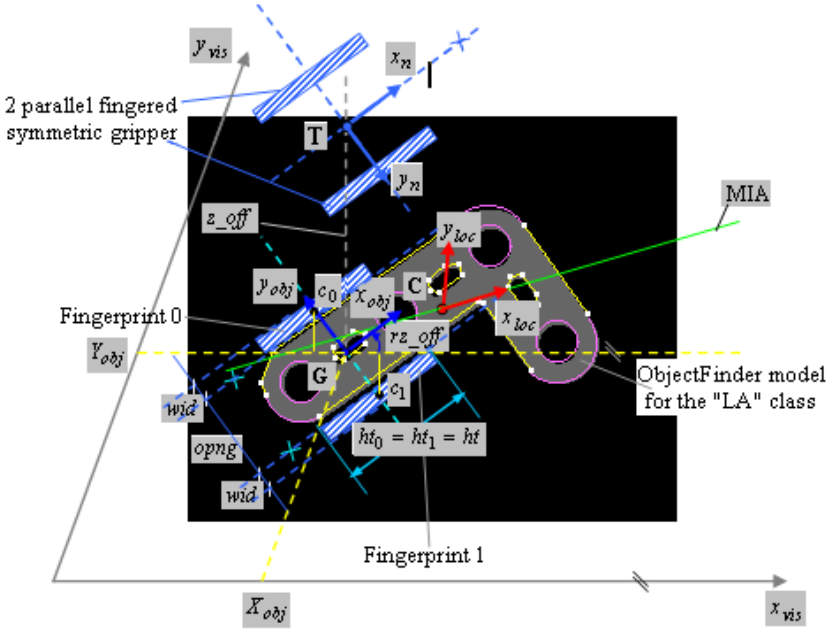


Fig. 5. Learning the fingerprint model *FGP_m location* data $c_{0x}$, $c_{0y}$, $rz_0$, $c_{1x}$, $c_{1y}$, $r_{z1}$ relative to the "LA" finder model for a parallel symmetric gripper with 2 rectangular-shaped fingers of size wd0, ht0, wd1, ht1. The current grip is for the grasp model *Gs_m={x_off, y_off, rz_off, z_off}*

- $fingers\_viewing(G, pose\_context_i, i = 1,...p)$, indicates how "invisible" fingers are to be treated; fingers are "invisible" if they are outside the field of view.
- $grip = 1,...,k$ are the $k$ gripper-object $Gs\_m(G,O)$ distinct grasping models a priori trained, as possible alternatives to face at run time foreground context situations.

A *collision-free grasping transformation* $CF(Gs\_m_i, O)$ is selected at run time from one of the $k$ *grip* parameters, after checking that all pixels belonging to $FGP\_m_i$ (the projection of the gripper's fingerprints onto the image plane $x_{vis}$, $y_{vis}$, in the $O$-grasping location) cover only background-coloured pixels. To provide a secure, collision-free access to objects, the following robot-vision sequence must be executed:

1. *Training* $k$ sets of parameters of the multiple fingerprints model $\mathcal{MFGP}\_m(\mathcal{G},O)$ for $\mathcal{G}$ and object class $O$, relative to the $k$ learned grasping styles $\mathcal{G}s\_m_i(\mathcal{G},O), i=1,...,k$.
2. *Installing* the multiple fingerprint model $\mathcal{MFGP}\_m(\mathcal{G},O)$ defining the shape, position and interpretation (viewing) of the robot gripper for clear-grip tests, by including the model parameters in a data base available at run time. This must be done at the start of application programs prior to any image acquisition and object locating.
3. *Automatically performing the clear-grip test* whenever a prototype is recognized and located at run time, and grips $\mathcal{FGP}\_m_i, i=1,....k$ have been a priori defined for it.
4. On line *call of the grasping parameters* trained in the $\mathcal{G}s\_m_i(\mathcal{G},O)$ model, which corresponds to the first grip $\mathcal{FGP}\_m_i$ found to be clear.

Fig. 5 shows an "LA"-type object, trained and planned with the dual Object Finder model $\mathcal{DFG}\_m=(\mathcal{F}p\_m(O),\mathcal{G}s\_m(O))$ for the recognition and grasping style definition of an instance $O$.

Two frames are used:

- ($x_{loc}$, $y_{loc}$) is the default reference frame attached automatically by the vision system to the ObjectFinder model in its centre of mass $\mathbf{C}(x_C, y_C)$ and aligned with its minimum inertia axis, $x_{loc} \equiv$ MIA, at training time.
- ($x_{obj}$, $y_{obj}$) is the user-specified object coordinate system, which is shifted with $x\_off = x_G - x_C, y\_off = y_G - y_C$ mm and turned with $rz\_off = \angle(x_{obj}, x_{vis})$ deg, where $x_{obj} \parallel x_{4(\mathrm{SCARA})}$, and **G** is the projection of the gripper's end point **T** on the image plane, in the grip position.

Once trained and installed the robot-object model, any time an instance of the model "LA" is recognized and successfully located, the vision system returns the X, Y and RZ data of the instance's coordinate system.

The data $c_{0x}$, $c_{0y}$, $rz_0$, $c_{1x}$, $c_{1y}$, $rz_1$ of the fingerprint model was learned relative to the instance's location $X_{obj}$, $Y_{obj}$, $RZ_{obj}$, expressed in the vision frame ($x_{vis}$, $y_{vis}$). As can be seen from Fig. 5:

$$c_{0x} = X_{obj} - (opng/2 + wid/2) * \sin(rz\_off) \qquad c_{1x} = X_{obj} - (opng/2 + wid/2) * \sin(rz\_off)$$

$$c_{0y} = Y_{obj} + (opng/2 + wid/2) * \cos(rz\_off) \qquad c_{1y} = Y_{obj} + (opng/2 + wid/2) * \cos(rz\_off)$$

$$rz_0 = RZ_{obj} \qquad rz_1 = RZ_{obj}$$

where

$$X_{obj} = x_G \quad x_C, Y_{obj} = y_G \quad y_C,$$

$$RZ_{obj} = rz\_off_{|\mathcal{G}s\_m(\text{"LA"})}$$

Assuming that the fingerprint rectangles have identical size *wid, ht*, the clear-grip routine to be called and executed at run time, is based on generating two rectangular Windows Region of Interest WROI, having the exact shape, size, position and orientation of the gripper fingerprints. Statistics will be gathered from the two windows in what concerns the number of background pixels existing in each window, i.e. background pixels covered by each fingerprint projection.

## 6. Experimental results

Experiments have been carried out on a platform using a Cobra s850 Adept robot [14] equipped with vision system. The collision-free object access strategy was implemented for assembling MECCANO parts of 3.5 mm height on carburettor flanges (Fig. 6).

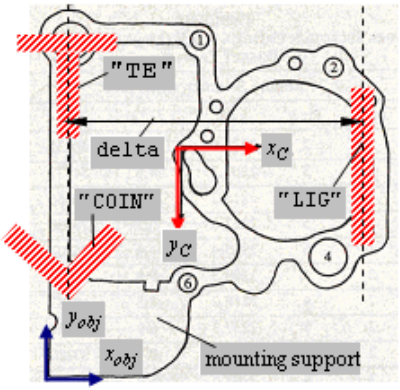


Fig. 6. Assembling "COIN", "TE" and "LIG" isolating components (MECCANO) on a carburettor flange

Table 1 indicates the offset values learned in an interactive training session for the models of the classes of thin MECCANO parts shown in Fig. 6. The parts, randomly arriving on a conveyor belt, are picked "on-the-fly" by the 2-fingered gripper of the robot, after checking the fingerprint models *FGP_m* against the foreground for parts closely placed or touching between them.

*Table 1*

**$Gs\_m$ parameters learned for "COIN", "TE", and "LIG" models of assembly components**

| *Gs_m* parameters<br>Class models | x_off [mm] | y_off [mm] | rz_off [deg] | z_off [mm] |
|---|---|---|---|---|
| "COIN" | –11.60 | –6.75 | –77.84 | 2.67 |
| "TE" | –19.07 | –11.02 | –32.89 | 1.81 |
| "LIG" | –11.38 | –6.38 | –77.06 | 2.26 |

A graphical user interface has been developed to assist the learning of robot-object and fingerprint models for a given gripper (opening, finger shape and

dimensions) and class of objects (Fig. 7). If, after placing the gripper in a desired position the clear grip test is ok, the $\mathcal{G}s\_m(\mathcal{G},O)$ parameters are automatically computed and displayed.

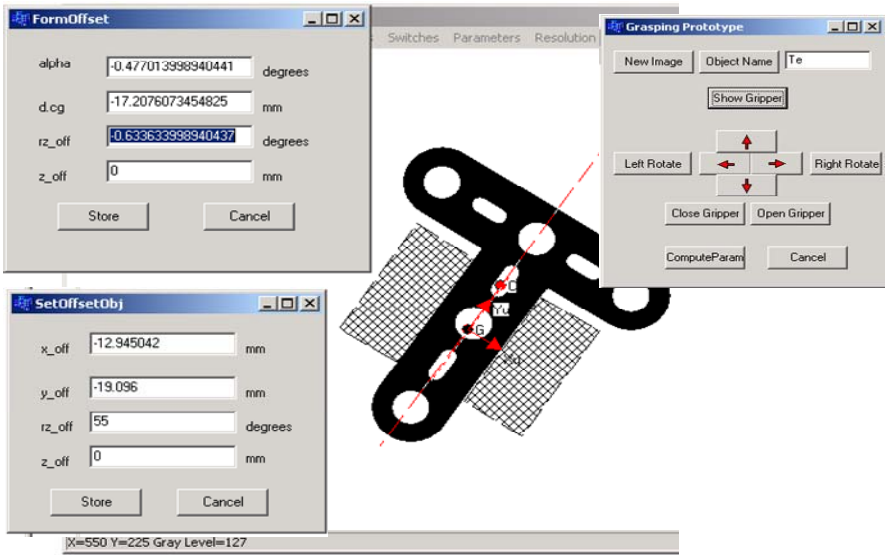


Fig. 7. Gripper parameters and control interface in GUI

## 6. Conclusions and perspectives

In this paper it has been presented a visual guidance technique used for advanced motion control. This technique provides flexibility when integrating robots in intelligent manufacturing cells with unstructured environment and in line quality inspection. The methodology for on-line vision-based robot control relies on robot-object models a priori learned which are on-line checked for collision-free grasping based on the gripper's fingerprints relative to the objects of interest.

Future research will be oriented towards adapting vision systems for automatic in line quality inspection and optimizing the energy consumption for industrial robots executing assembly operations.

## REFERENCES

[1]. *Morariu, C., Morariu, O., Borangiu, Th. and Y. Sallez*, Formalized Information Representation for Intelligent Products in Service-oriented Manufacturing, Proc. of the 11th IFAC Workshop on Intelligent Manufacturing Systems, Sao Paolo, Brazil, Vol. 11, Part 1, 2013, pp. 318–323

[2]. *Chaumette, F. and S. Hutchinson*, Visual servo control, Part 1: Basic approaches, IEEE Robotics and Automation Magazine, Vol. 13, No. 4, 2006, pp. 82-90

[3]. *Borangiu, Th.*, Intelligent Image Processing in Robotics and Manufacturing, Academy Press, Bucharest, 2004

[4]. *Ceccarelli, M.* (Ed), Service Robots and Robotics: Design and Applications, IGI Global, doi: 10.4018/978-1-4666-0921-5.ch017, 2012, pp. 338-356
[5]. *Borangiu, Th., Ciocan, M., Raileanu, S., Morariu, O. and O. Stocklosa*, 3D complex surface generation through procedural robot motion, Proc. of the IEEE Int. Conf. ICSTCC'14, Sinaia, 2014
[6]. *Siciliano, B., Sciavicco, L., Villani, L. and G. Oriolo*, Robotics, Modelling, Planning and Control, Springer, 2010
[7]. *Stocklosa, O., Borangiu, Th., Ivanescu, N.-A., Raileanu, S. and A. Rosu*, Vision-Guided Part Feeding in a Holonic Manufacturing System with Networked Robots, Proceedings of the Int. Workshop Robotics in Alpe-Adria-Danube Region, RAAD'08, Ancona, 2008
[8]. *Park, J. and Z. Bien*, Design of an advanced machine vision system for industrial inspection, Intelligent Automation and Soft Computing, Vol. 1, No. 2, 1995, pp. 209-219
[9]. *Borangiu, Th., Stocklosa, O., Răileanu, S., Trentesaux, D. and T. Berger*, Open Manufacturing Control with Agile Reconfiguring of Robot Services, Proceedings of the Int. Workshop on Robotics in Alpe-Adria-Danube Region, RAAD'10, Budapest, 2010
[10]. *Babiceanu, R. F., Chen, F.F. and R. H. Sturges*, Framework for the Control of Automated Material-handling Systems Using the Holonic Manufacturing Approach, International Journal of Production Research 42 (17), Francis and Taylor, 2004, pp. 3551–3564
[11]. *Barata, J.*, The Cobasa Architecture as an Answer to Shop Floor Agility, in Manufacturing the Future, Concepts, Technologies and Vision, V. Kordic, A. Lazinica, and M. Merdan (Eds.). Rijeka: InTech. ISBN 3-86611-198-3, 2006, pp. 31–76
[12]. *Stefanoiu, D. Borangiu, Th. and F. Ionescu*, Generating Planar Workspace Boundaries by Radius Parsimonious Increase, Proceedings of the IFAC Int. Workshop on Intelligent Assembly and Disassembly IAD'03, Bucharest, 2003, pp. 90-95
[13]. *Stocklosa, O., Borangiu, Th., Raileanu, S., Ivanescu, N.-A. and O. Morariu*, Multitasking Robot-Vision for Supply Holon Execution in Intelligent Manufacturing, Proc. of the Int. Workshop on Robotics in Alpe-Adria-Danube Region, RAAD'14, Smolenice, Slovakia, 2014
[14]. *** Adept Technology Inc., Adept Vision User's Guide, Version 14.0, Part Number 00964-03300, Rev. B, San Jose, CA, Technical Publication Adept, 2001