Visual Robot Guidance in Conveyor Tracking with Belt Variables

Th. Borangiu[1], F.D. Anton[1], Anamaria Dogar[1]
[1]University Politehnica of Bucharest, {borangiu, anton, dogar}@cimr.pub.ro

Abstract-**The paper presents a method and related algorithm for visual robot guidance in tracking objects moving on conveyor belts; the instantaneous location of the moving object is evaluated by a vision system consisting from a stationary, down looking monochrome video camera, a controller-integrated image processor and a vision extension of the structured V+ robot programming environment. The algorithm for visual tracking of the conveyor belt for "on–the-fly" object grasping is partitioned in two stages: (i) visual planning of the instantaneous destination of the robot, (ii) dynamic re-planning of the robot's destination while tracking the object moving on the conveyor belt. The control method uses in addition the concept of Conveyor Belt Window – CBW. The paper discusses in detail the motion control algorithm for visually tracking the conveyor belt on which objects are detected, recognised and located by the vision part of the system. Experiments are finally reported in what concerns the statistics of object locating errors and motion planning errors function of the size of the objects and of the belt speed. These tests have been carried out on a 4-d.o.f. SCARA robot with artificial vision and the AVI AdeptVision software extension of the V+ high-level, structured programming environment.**

I. Modelling Conveyor Belts By Means Of Belt Variables

The principal mechanism which allows specifying robot motions relative to a conveyor belt consists into modelling the belt by a special type of location data, defined as *belt variables*. A belt variable is considered as a relative transformation (having a component variable in time) which defines the location of a reference frame attached to the moving belt conveyor [1].

Using such a belt variable, it becomes possible to describe in time the relationship between a belt encoder and the location and speed of a reference frame which maintains a fixed position and orientation relative to the belt. The definition of a belt variable is given by:

`DEFBELT` *%belt_variable = nominal_trans, scale_factor*

where: (i) *%belt_variable* is the name of the belt variable to be defined, (ii) *nominal_trans* represents the value in $\mathbf{R}^6$ of the (simple or composed) relative transformation which defines the position and the orientation of the conveyor belt, and (iii) *scale_factor* is a calibrating constant specifying the ratio between the elementary displacement of the belt and one pulse of the encoder. The X axis of *nominal_trans* indicates the direction of motion of the belt, the XY plane defined by this transformation is parallel to the conveyor's belt surface, and the position (X,Y,Z) specified by the nominal transformation points to the approximate centre of the belt relative to the base frame of the robot. It has been decided that the origin of *nominal_trans* be always chosen in the middle of the robot's working displacement over the conveyor.

The current orientation in a belt variable is invariant in time, equal to that expressed by *nominal_trans*. In order to evaluate the current location updated by the same belt variable, the following real-time computation has to be performed: multiplication of a unit vector in the direction of $\mathrm{X}|_{\text{nominal\_trans}}$ by the distance *belt_distance* derived from the encoder's contents (periodically read by the system), and then addition of the result to the position vector of *nominal_trans*. In a symbolic representation, on has:

$$\text{XYZ\_instantaneous} =$$
$$\text{XYZ\_nominal} + belt\_distance * vers(\mathrm{X}|_{nominal\_trans})$$

with

$$belt\_distance =$$
$$(encoder\_count - encoder\_offset) * scale\_factor$$

where *encoder_count* is the encoder's read contents and *encoder_offset* will be used in order to establish the instantaneous location of the belt's reference frame (x_i, y_i) relative to its nominal location (x_n, y_n). In particular, the belt's offset can be used to nullify a displacement executed by the conveyor (by setting the value of the offset to the current value of the encoder's counter). Te designed algorithm for visual robot tracking of the conveyor accounts for variable belt offsets which are frequently changed by software operations

`SETBELT` *%belt_variable = expression*

The dynamic robot-vision synchronisation according to the "Look–and–Move" cooperation principle will be triggered, for *picking-on-the-fly* robot tasks, by a fast digital-input interrupt line detecting the occurrence of an external event of the type: "an object has completely entered the Conveyor Belt Window" (and hence is completely visible). This event is detected by a photocell, and will determine an image acquisition of the moving part [2], [3].

II. The Logical Mechanism Conveyor Belt Window

There has been designed a logical mechanism called Conveyor Belt Window (CBW) which allows the user to check the area of the belt in which the robot will operate. A CBW defines a segment of the belt delimitated by two planes perpendicular to the direction of motion of the belt (this window is restricted only in the direction of motion of the conveyor's belt).

Because the conveyor is modelled by a belt variable (e.g. *%belt_variable*), in order to define a CBW it is necessary to refer the same belt variable and to specify two composed transformations which, together with the direction of motion of the belt, restrict the motion of the robot along a desired area of the conveyor [4]:

`WINDOW` *%belt_variable* = *downstr_lim*, *upstr_lim*, *program_name*, *priority*,

where *downstr_lim* and *upstr_lim* are the two relative transformations defining respectively the downstream and upstream edges of an invariant window positioned along the belt within the working space of the robot and the image field of the camera. Supposing that, while the robot tracks the conveyor belt, a window violation occurs, it would be necessary that the system reacts correspondingly; for such purpose a reaction routine *program_name* and its *priority* are included in the CBW definition. Normally, the priority of the reaction program must be greater than that of the conveyor tracking program, so that the motion of the robot can be immediately interrupted in case of window violation.

The CBW defined as above will be used not only in the stage of robot motion planning, but also in real time during motion execution and tracking control in order to check if the motion reference (the destination) is within the two imposed limits:

- when a movement of the robot is *planned*, the destination of the movement is checked against the operating CBW; if a window violation is detected, the reaction program is ignored and an error message will be issued;
- when a movement of the robot relative to the conveyor belt is *executed*, the destination is compared every 16 milliseconds with the window's limits; if a window violation is detected, the reaction program is automatically invoked according to its priority level and the robot will be instructed to stop tracking the belt [5], [6].

There have been designed two useful CBW functions which allow the dynamic reconfiguring of programs, decisions, branching and loops during the execution of robot – vision conveyor tracking programs, function of the current value of the part-picking transformation relative to the belt, and of the current status of the belt tracking process:

`WINTEST` (*robot_transformation*, *time*, *mode*)

`BELTSTATUS`

The `WINTEST` function returns a value in millimetres indicating where is situated the location specified by the belt-relative composed transformation *robot_transformation*, with respect to the predefined (fixed) belt window limits *downstr_lim* and *upstr_lim* at *time* seconds in the future, computed according to its current position and belt speed. For robots tracking conveyor belts in order to pick-on-the-fly visually recognised and located objects, the transformation of interest is *%belt*: *part.lock* (variable in time), where *%belt* models the conveyor belt, the character ":" holds for the composition of two homogenous transformations having a minimal representation of the orientation, and *part.loc* is the composed, time invariant transformation expressing the gripper frame (x_g, y_g, z_g) with respect to the base frame of the robot (x_0, y_0, z_0). Finally, the argument *mode* is a real-valued expression which specifies whether the result of the `WINTEST` function represents a distance inside or outside the predefined conveyor belt window.

For example, if *mode* is positive, the value *returned* by `WINTEST` will be interpreted as:

- returned distance = 0, if the location is inside the CBW;
- returned distance<0, if the location is upstream of *upstr_lim* of the CBW;
- returned distance>0, if the location is downstream of the *dwnstr_lim* of the CBW.

For example, the `WINTEST(%belt:part.loc,4,1)` distance is positive if, in 4 seconds from the time being, the belt-relative part picking location will be outside the window defined for the conveyor modelled by *%belt*. If the robot tries to move towards a belt-relative location that has not yet appeared inside the belt window (it is still upstream relative to the CBW), the motion control algorithm has been designed with two options:

- *temporarily stops the robot*, delaying thus the motion planning, until the time-variable destination enters the belt window;
- *definitively stops the robot* and generates immediately an error message.

Also, the control algorithm generates a condition of window violation anytime the vision-based robot motion planner computes a destination which is downstream the CBW, or will exit the CBW at the predicted time the robot will reach it.

The real valued function `BELTSTATUS` returns a result of the type field-of-bits indicating the current status of the belt tracking process: *robot tracking the belt*; *destination upstream*; *destination downstream*; *window violation*, which can be used to dynamically reconfigure the robot-vision task [7].

III. Belt-Relative Robot Movements Planned By Vision

To command a belt-relative motion of the robot with linear interpolation in the Cartesian space, i.e. to define a transformation relative to an instantaneous location of a moving frame (x_i, y_i) on the conveyor belt, one has to use a belt variable (which models the conveyor belt) in a composed object – picking transformation variable in time, instead of a regular one.

The research was directed to develop a motion control algorithm and related software implementation techniques which visually plan the motion of the robot as a result of object detection, recognition and locating on a moving conveyor belt and than track the object in order to grasp it inside a conveniently defined belt window. The main idea relies on dynamically changing the visually computed destination of the robot end point by composing it with a belt-related transformation updated every 8 milliseconds from the encoder data [8], [9], [10].

If a stationary matrix video camera is used, down looking at the conveyor belt, and supposing that its field of view covers completely a conveyor belt window defined inside the working area of the robot (after execution of a camera – robot calibration session), then the image plane can be referred by the time – invariant frame (x_{vis}, y_{vis}) as represented in Fig. 1. It is also assumed that the X axes of the reference frame of the robot (x_0), of the conveyor's direction of motion (x_n) and of the image plane (x_{vis}) are parallel between them. The conveyor belt is modelled by the special belt variable defined as *%belt*. Objects are circulating on the belt randomly; their succession (the current object type entering the CBW), their distance from the central axis of the conveyor and their orientation are unknown. The "Look-and-Move" interlaced operating principle of the image processing section and motion control section is used.

According to this principle, while an image of the CBW is acquired and processed for object identification and locating, no motion command is issued and the camera will not snap an image while the robot tracks a previously located part in order to pick it "on-the-fly" [11].

The robot motion control algorithm for tracking the conveyor belt in order to pick "on-the-fly" one by one objects recognised and located by vision computing consists of the following basic steps:

1. *Triggering the strobe light* (synchronously or asynchronously with respect to the read cycle of the video camera) when the *image acquisition* is requested from a fast digital-input interrupt line connected to a photocell mounted at the upstream limit of the CBW. The interrupt line drops from 1 to 0 signalling that an object has completely entered the belt window.

2. *Recognising a single object* that has just completely entered the belt window. Recognition of objects is exclusively based on the match with previously learned prototypes (or models of all classes of objects of interest).

3. *Locating the object* which was recognised, by computing the coordinates of its centre of mass and the angle between its minimum inertia axis (MIA) and the x_{vis} axis. As can be seen in Fig. 1, the object-attached frame (x_{obj}, y_{obj}) has the abscissa aligned with MIA, and the current location of the object in the image plane is computed by the vision section and returned in *vis.loc* [4].

4. *Planning the instantaneous destination of the robot.* Once the object recognised as an instance of a prototype, and successfully located, the corresponding relative grasping transformation *grip.trans* is called. Assuming that the grasping style is such that the projection of the gripper's centre on the image plane coincides with the object's centre of mass, the gripper-attached coordinate frame (x_g, y_g) will be offset relative to the object-attached frame along the z_0 axis by *z_off* millimetres and turned with *r_off* degrees about the same z_0 axis. Now, using the relative camera-robot calibration transformation to.cam[ ] (as an output of the existing camera-robot calibration session) relating the vision frame (x_{vis}, y_{vis}) to the base frame of the robot (x_0, y_0, z_0), the current destination of the robot (for a frozen conveyor belt) is computed from the accessed and processed in real time vision data as a composed transformation (called for instance part.loc), expressing the gripper's frame location relative to the robot base frame:

part.loc = to.cam[cam]:vis.loc:grip.trans

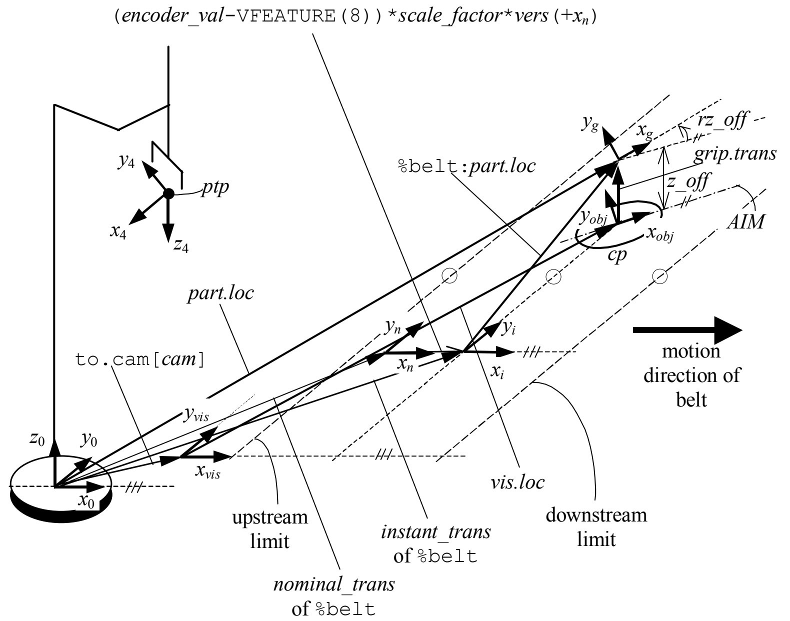

Figure 1. Robot-Vision and belt-relative transformations for conveyor tracking.

5. *Synchronising the encoder belt with the motion of the object* recognised in the belt window. This operation consists into setting the offset of the conveyor belt at a correct value. The operation `SETBELT`

```
%belt = encoder_val(strobe)
```

establishes the point of interest of the conveyor belt modelled with *%belt* as the point corresponding to the current value of the encoder counter *encoder_val*(*strobe*) at the time the strobe light was triggered. This value is available immediately after the object locating operation was successfully completed. Thus, as soon as one object is recognised and located, the instantaneous position of the belt, identified by the frame (x_i, y_i), will be reset since

$$\begin{aligned} & xyz_i = xyz_n \\ & + (encoder\_val - encoder\_val(strobe)) \\ & + scale\_factor * vers(x_n) = xyz_n \end{aligned}$$

6. *Tracking the object moving on the belt and picking it*. This requires issuing a linear motion command in the Cartesian space, relative to the conveyor belt. A composed relative transformation *%belt* : *part.loc*, expressing the current computed location of the gripper relative to the instantaneous moving frame (x_i, y_i) is then defined. Practically, the tracking procedure begins immediately after the instantaneous position of the conveyor belt – expressed by the frame (x_i, y_i) has been initialised by the `SETBELT` operation, and consists into periodically updating the time-variable destination of the gripper by shifting it along the x_n axis with encoder counts (pulses of variable frequency) accumulated during the successive sampling periods $\ldots, t_{k-1}, t_k, t_{k+1}, \ldots,$ where: $\Delta t = t_{k+1} - t_k = const$

$$\begin{aligned} & \%belt : part.loc|_{t_{k+1}} = \\ & \texttt{SHIFT}\ (\%belt : part.loc|_{t_k}\ \texttt{BY}\ldots \\ & \ldots\ encoder\_count(t_{k+1} - t_k) * scale\_factor, 0, 0) \end{aligned}$$

```
MOVES %belt:part.loc
CLOSEI
```

7. Once the robot commanded towards a destination relative to the belt, the gripper will continuously track the belt until a new command will be issued to approach a location which is not relative to the belt.

In the case of belt-relative motions, the destination changes continuously and, depending on the magnitude and the variations of the conveyor speed it is possible that the robot is not able to attain the final positions within the default error tolerance. In such cases, the error tolerance must be augmented [12], [13].

Fig. 2 presents the robot motion control algorithm for tracking the conveyor belt in order to pick "on-the-fly" one object recognised and located by vision computing inside the belt window.

The reaction mechanism REACT continuously monitors the fast digital-input interrupt line which, when changing from *on* to *off* signals that an object has completely entered the belt window.

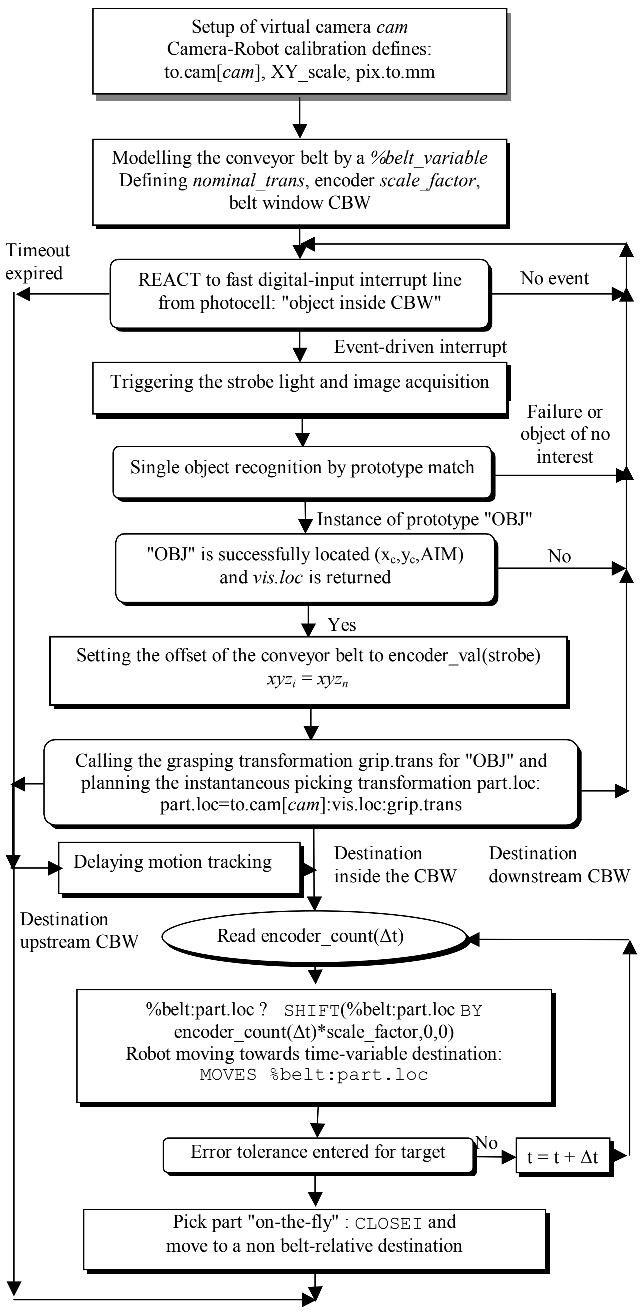
Setup of virtual camera cam
Camera-Robot calibration defines:
to.cam[cam], XY_scale, pix.to.mm
Modelling the conveyor belt by a %belt_variable
Defining nominal_trans, encoder scale_factor,
belt window CBW
Timeout expired
REACT to fast digital-input interrupt line from photocell: "object inside CBW"
No event
Event-driven interrupt
Triggering the strobe light and image acquisition
Failure or object of no interest
Single object recognition by prototype match
Instance of prototype "OBJ"
"OBJ" is successfully located (xc,yc,AIM) and vis.loc is returned
No
Yes
Setting the offset of the conveyor belt to encoder_val(strobe)
xyzi = xyzn
Calling the grasping transformation grip.trans for "OBJ" and planning the instantaneous picking transformation part.loc:
part.loc=to.cam[cam]:vis.loc:grip.trans
Delaying motion tracking
Destination inside the CBW
Destination downstream CBW
Destination upstream CBW
Read encoder_count(Δt)
%belt:part.loc ? SHIFT(%belt:part.loc BY
encoder_count(Δt)*scale_factor,0,0)
Robot moving towards time-variable destination:
MOVES %belt:part.loc
Error tolerance entered for target
No
t = t + Δt
Pick part "on-the-fly" : CLOSEI and
move to a non belt-relative destination

Figure 2. The robot motion control algorithm for visually tracking the conveyor belt.

When defining the Conveyor Belt Window, a special high-priority routine can be specified, which will be automatically invoked to correct any window violation during the process of tracking moving objects. In such situations the robot will be accelerated (if possible) and the downstream limit temporarily shifted in the direction of motion of the conveyor belt within a timeout interval depending on the belt's speed, in which the error tolerance must be reached [14], [15].

IV. Experimental Rezults And Conclusions

The robot motion control algorithm for tracking moving objects visually recognised and located on conveyor belts, modelled by means of belt variables and windows, has been tested on a robot-vision system containing a SCARA-type AdeptOne 4- *d.o.f.* manipulator with parallel fingered gripper, a VME-based robot controller including a four-channel frame grabber for a Sony medium resolution (480x640) CCD down looking camera, and a slave processor reading the pulses of the belt encoder and stepwise controlling the belt speed. The OS 9 operating system allowed the multitasking management of the "motion axes", "vision" and "conveyor" resources. The described motion control algorithm (based on dynamical re-planning of the robot destination along the conveyor belt, starting from an instantaneous location computed by the vision part) and related robot – vision cooperation tasks according to the "Look-and-Move" principle have been implemented in the robot language V+ with AVI vision software extension.

The experiment consisted in a batch production task for the robot, i.e. picking two types of parts travelling on a closed-loop cell conveyor and palletising them in two dedicated stacks; two models have been previously learned, respectively with the names "TDRB" and "TDCB". Other types of parts, prototyped or not, might also be transported by the conveyor belt which links several machine tools in a manufacturing cell. A single, stationary down looking camera was used to inspect the belt (Fig. 3). Light strobe lamps have been used to avoid blurred images from the camera; the strobe light is triggered by a photocell placed at the upstream limit *upstr_lim* of the belt window of the conveyor modelled by *%belt_variable*.

The external event transferred as an on-off signal to the fast digital input interrupt line signifies that "a part has completely entered the CBW", and occurs in the most unfavourable case after the part has travelled a distance

$$dist = \max(long\_max.tdrb, long\_max.tdcb)$$

inside the CBW, where *long_max.tdrb, long_max.tdcb* are the maximum travelling distances for the two type of parts if they move on the belt with their minimum inertia axes parallel to the direction of motion of the conveyor belt. In this case, the image acquisition instruction in the application program waits for an interrupt from the fast digital-input line.

Two techniques have been experimented for triggering the strobe light:

- *Asynchronous triggering* with respect to the read cycle of the video camera, i.e. as soon as an image acquisition request appears. Using the fast digital input interrupt mechanism, a delay of 0.2 milliseconds has been considered. For a 51.2 cm width of the image field, and a line resolution of 512 pixels, the pixel width is of 1 mm. For a 2.5 m/sec high-speed displacement of objects on the conveyor belt, the most unfavourable delay of 0.2 milliseconds corresponds to a displacement of only one pixel (and hence one object-pixel might disappear during the *dist* travel above defined), because

$$(0.0002 \text{ sec}) * (2500 \text{ mm/sec}) / (1 \text{ mm/pixel}) = 0.5 \text{ pixels}$$

- *Synchronous triggering* with respect to the read cycle of the video camera, inducing a variable time delay between the image acquisition request and the strobe light triggering. The most unfavourable delay was in this case 16.7 milliseconds, which, for the same conveyor belt speed causes a potential disappearance of 41.75 pixels from the camera's field of view (this field of view covers a window on the conveyor belt, in the reach area of the robot).

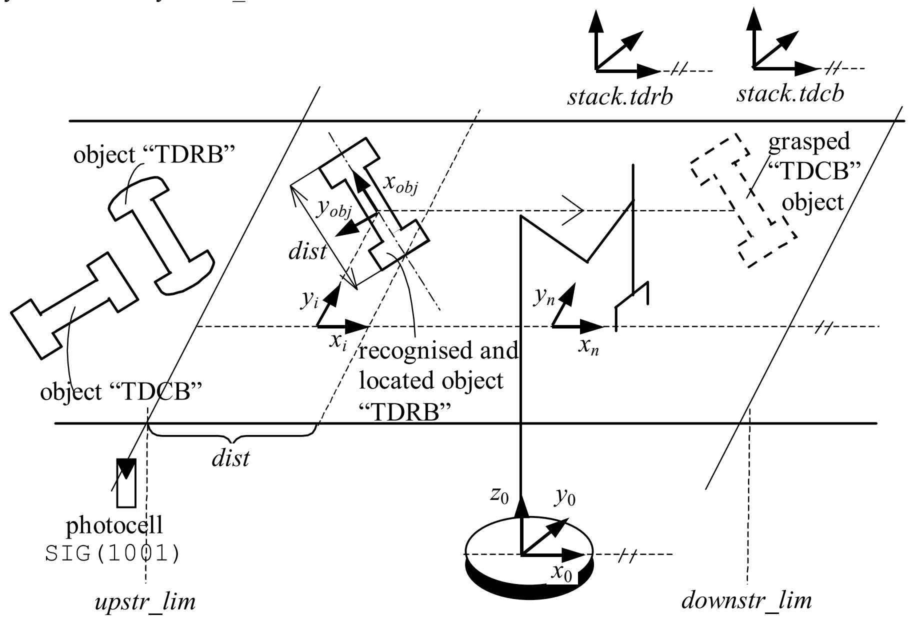

Figure 3. Representation of the experimental robot task with visual tracking of a conveyor belt.

Fig. 4 depicts the statistics about the sum of the visual locating errors (computation of the object's location relative to the image frame (x_{vis}, y_{vis}) and of the motion planning errors (destination downstream *downstr_lim*), function of the object's dimension (length *long_max.obj* along the minimal inertia axis) and of the belt speed (four high travelling speed values have been considered: 0.5 m/sec, 1 m/sec, 2 m/sec and 3 m/sec). As can be observed, at the very high motion speed of 3 m/sec, for parts longer than 35 cm there was registered a percentage of near 10% cases of unsuccessful object locating or planning of robot destinations outside the CBW, from a total number of 1500 experiments.

In the same conditions of visual field size and conveyor belt speed, instead of using the fast digital-input interrupt line, a WAIT operation has been experimentally used to synchronise the image acquisition operation with the state change of an external signal (coming from the photocell).

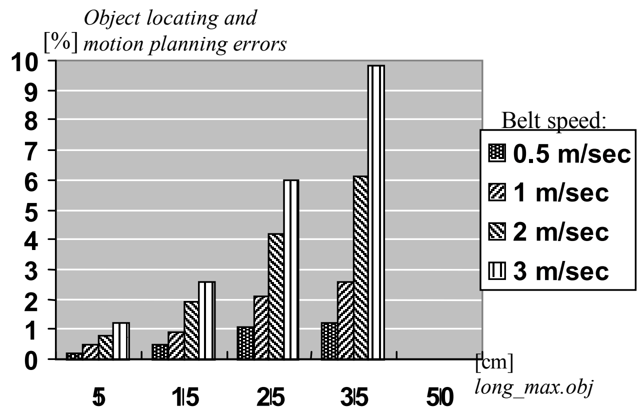

Figure 4. Error statistics for object locating and motion planning at various high belt speeds.

The most unfavourable delay between the occurrence of the external event "object inside the CBW" and the image snap was of 16 milliseconds, which leads to a displacement of the object with 40 pixels outside the predefined conveyor belt window.

This work is funded by the National Council for Scientific University Research, in the framework of the National Plan for Research, Development and Innovation, grant 69/2007.

REFERENCES

[1] Th. Borangiu and M. Dupas, "*Robot – Vision. Mise en œuvre en V+*", Romanian Academy Press, Bucharest, 2001

[2] Th. Borangiu, "Visual Conveyor Tracking for "Pick-On-The-Fly Robot Motion Control", *Proc. of the IEEE Conf. Advanced Motion Control AMC'02*, Maribor, pp. 317-322, 2002

[3] Th. Borangiu and P. Kopacek (Eds), Proceedings Volume from the IFAC *Workshop "Intelligent Assembly and Disassembly - IAD'03*", Bucharest, Elsevier Science, Pergamon Press, Oxford, UK, 2004

[4] Th. Borangiu, "*Intelligent Image Processing in Robotics and Manufacturing*", Romanian Academy Press, Bucharest, ISBN 973-27-1103-5, 2004

[5] Th. Borangiu, "Guidance Vision for Robots and Part Inspection", Proceedings volume of the 14th Int. Conf. *Robotics in Alpe-Adria-Danube Region RAAD'05*, pp. 27-54, Bucharest, ISBN 973-718-241-3, 2005

[6] S.K. Nayar, "Subspace Methods for Robot Vision", *IEEE Trans. on Robotics & Automation*, **12**, *5*, pp. 750-758, 1996

[7] Adept Technology Inc., "*AdeptVision User's Guide Ver. 16.0, Technical Publications*", Part Number 00964-03300, Rev. B, San Jose, CA, 2008

[8] M. Lindenbaum, "An Integrated Model for Evaluating the Amount of Data Required for Reliable Recognition", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2006

[9] West, P., "High Speed, Real-Time Machine Vision", *CyberOptics – Imagenation*, pp. 1-38, Portland, Oregon, 2008

[10] Th. Borangiu, M. Manu, F.D. Anton, Silvia Tunaru and Anamaria Dogar, "High-speed Robot Motion Control under Visual Guidance", Proceedings of the *12th International Power Electronics and Motion Control Conference - EPE-PEMC'06,* Portoroz, Slovenia, 2006

[11] S.A. Hutchinson, G.D. Hager and P. Corke, "A Tutorial on Visual Servo Control", *IEEE Trans. on Robotics and Automation*, **12**, pp. 1245-1266, 1996

[12] J. Lastra, J. and I. Delamerm, "Semantic web services in factory automation: Fundamental insights and research roadmap", *IEEE Transaction on Industrial Informatics*, **2**, 1-11, 2006

[13] Th. Borangiu, G. Pascal, N.-A. Ivanescu, and A. Rosu, "An Implementing Framework for Holonic Manufacturing Control with Multiple Robot-Vision Stations", Journal IFAC, *The International Journal of Intelligent Real-Time Automation - Engineering Applications of Artificial Intelligence*, Vol. 22, Issue 4-5, Elsevier, Pergamon Press, ISSN 0952-1976, 2009

[14] M. Rahayem, "Geometric Reverse Engineering Using a Laser Profile Scanner Mounted on an Industrial Robot"*, Proceedings of DAAAM Conference*, Tallinn, Estonia, 2008

[15] H. Nylund, K. Salminen and P.H. Andersson, "A multidimensional approach to digital manufacturing systems", Proceedings of the 5th International Conference on Digital Enterprise Technology Nantes, France 2008