DEFINITION OF A PROGRAMMING LANGUAGE FOR TELEROBOTS APPLICATION TO ROBOTICS IN BUILDING CONSTRUCTION

B. TONDU, R. FOURNIER, G. CLEMENT

Comissariat à l'Energie Atomique, CEN/FAR
Unité de Génie Robotique Avancé
92265 Fontenay-aux-Roses, France

Abstract

This paper deals with the programming of construction robots for finishing work on façades of buildings. The difficulty of automating such tasks, and the high potential occurence of incidents on the building site, lead to consider the use of hybrid telerobots, programmable like autonomous robots but also controlable in teleoperation mode . Programming telerobots is a new field which must allow the combination of the two modes of control: the automatic mode and the teleoperated mode. We propose in this paper the definition of a telerobot programming language, and a first simulation is presented to illustrate its application in construction robotics.

1. Introduction

Within the EUREKA/GEO project, a façade working robot /1/, we have to consider the problem of programming the machine: GEO is intended to perform finishing works on façades in an outdoors environment. Tasks and environment may present high degrees of unstructurations due, for example, to unexpected obstacles, misfunctionment of the process or particular local tasks. This lack of structuration makes it very difficult to consider a purely autonomous mode of operation of the robot. To overcome the local difficulties or incidents encountered during a session we propose to get help from an operator through a teleoperated mode of control. Then the initial programming will not have to be reconsidered, the machine will come back to its nominal procedure once the local problem is solved.

Subject of this paper is to present a first definition of a so called "telerobot programming language" that is able to integrate combinations of automatic and computer aided teleoperation (CAT) modes /2/,/3/. State of the art is not presenting general telerobot programming language adapted to our problem. The teleoperation mode of control is not fully integrated as part of the instructions set of the language; it is only considered as back up solution to complete a given task when the autonomous operation is no more achievable. LARTS /4/,/5/ is moving further and does integrate CAT modes but only to ease the learning and programming procedures. The objective of the concept we develop here is to directly integrate within the program sequence CAT modes that will be considered at the same level as autonomous instructions. The two approaches are then unified, allowing to switch continuously between autonomous and teleoperated modes of control.

2.Basic principles of the language

We call "robot mode" the autonomous mode of control of the telerobot, and "CAT mode" the advanced computer aided teleoperated mode of control of the telerobot.

Basic principles of the language are twofold:
1. A task is a sequence of "automatic subtasks" performed in robot mode and/or "CAT subtasks" performed in CAT mode.
2. For each subtask (automatic or CAT) a set of takeovers procedures can be associated. These procedures are performed in CAT mode. The aim of takeovers is to offer to the operator the possibility to complete an interrupted subtask. The interruptions may come from the occurence of an incident (detected by sensors) or may be forced by the operator himself. When the problem is overcome then a return procedure allows to come back to the current subtask.

Figure 1 illustrates those principles. It shall be noted that application range of takeovers is limited to the current subtask; the operator cannot jump to any other subtask in sequence. This very streng limitation has been chosen in order to simplify the control process of the program and to increase the safety of the system.

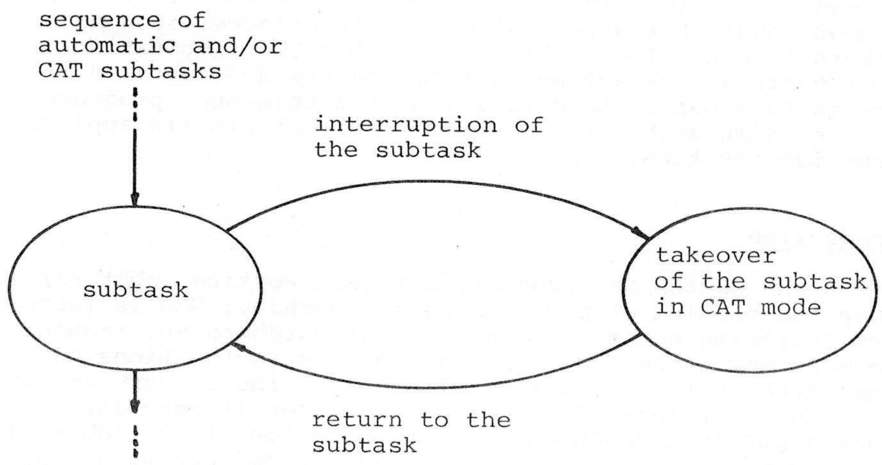


Figure 1: Association of a takeover to a subtask

We will further distinguish the programmer who performs the task programming, generally off-site, from the operator who supervises the task execution on-site. Depending on the problem, programmer and operator are the same persons or not.

Programming a task corresponds to the following steps:
1. Definition of the complete set of subtasks and of takeovers that are required for the given task.
2. Attachment of appropriate takeovers to each subtask and definition of the sequencing.

On-site the execution phasis is performed under operator's supervision. When required the operator is sollicited by the machine to adjust a parameter, to complete a CAT subtask or a particular takeover.

3. Definition of subtasks and takeovers

The definition of subtasks and takeovers is based on a double library of elementary instructions:
- robot mode instructions: elementary automatic instructions similar to those of an end-effector level robot programming language such as VAL /6/.
- CAT mode instructions: elementary CAT constraints of an advanced teleoperation system /2/ (freezing degrees of freedom, distance servoed to a plane,...etc).

3.1. Definition of automatic subtasks

An automatic subtask is defined as a sequence of robot mode instructions. A set of messages will explain to the operator the content of the subtask. The following syntax is proposed for the definition of an automatic subtask:

```
define automatic subtask(name)

(set of messages indicating
what realizes the subtask,
and the adjustable parameters)

(sequence of automatic
instructions)

end define
```

3.2. Definition of CAT subtasks

A CAT subtask represents an action which must be performed by the operator. Its programming consists in indicating to the operator:
- the aim of the subtask,
- the CAT assistance that is proposed to achieve the subtask.

This CAT assistance is composed of a set of CAT mode instructions that combine their effects. For example, the aided painting of a surface can consist in the combination of following constraints: tool orientation frozen perpendicularly to the surface and distance of the tool to the surface servoed at a preseted distance. The following syntax is proposed to define such a CAT subtask:

```
define CAT subtask(name)

(set of messages indicating
the aim of the subtask)

(set of CAT instructions
combining their effects)

end define
```

3.3. Definition of takeovers

A takeover is defined as a CAT assistance composed by a set of CAT mode instructions combining their effects. The definition of a takeover is then similar to the CAT subtask definition, the difference being that there is no predifened aim as in the case of a CAT subtask. A set of messages is also here explaining to the operator what the programmed takeover consists in. The following syntax is proposed for the definition of a takeover:

```
define takeover(name)

(set of messages indicating
in what consists the takeover)

(set of CAT mode instructions
combining their effects)

end define
```

4.Association of takeovers to a subtask

4.1.Basic rules

The association of a takeover to a given subtask can be simply represented by a programming line such as:

(execute subtask(name))(takeover(name)

However, to be really efficient, we complete this programming principle with the two following rules:
- At each subtask can be associated several possible takeovers, which will be proposed to the operator. Depending on the reason why the subtask has been interrupted, the operator will choose the most adapted proposed takeover. This possibility of choice can be expressed by an 'or' in the previous programming line.
- If the proposed takeovers are not sufficiently pertinent, the operator will have the possibility to define on-line a more adapted takeover. This on-line definition will be possible thanks to a menu that will propose a set of CAT mode instructions. However, for safety reasons and depending on the programmed subtask, it can be important to foresee during the programming phase the limitations of this on-line definition process by imposing or forbidding given CAT assistances. This will be expressed by adding to the programming line an instrcution called "limit-takeover". The "limit-takeover" instruction is detailled in the next subparagraph.

After having introduced these two rules, the proposed programming line becomes:

(execute subtask(name))(takeover(name1) or takeover(name 2)
or takeover(name3)...)(limit-takeover(name))

One can remark that the "limit-takeover" instruction can be directly associated to a subtask without takeovers have been specified:

(execute subtask(name))(limit-takeover(name))

This association means that the programmer has no idea about the pertinent takeovers in case of an incident, but he however prepares the definition of them.

4.2.Definition of on-line takeovers

The definition of on-line takeovers will take place when the operator on the site will have refused all the programmed takeovers. Thanks to a menu, the operator selects CAT mode instructions and their combination will form a new programmed takeover. This on-line definition will be limited by choosable compulsory or forbidden CAT mode instructions.

We so complete the definition of the takeover presented in paragraph 3 by introducing the "limit-takeover" instruction. The proposed syntax for this instruction is:

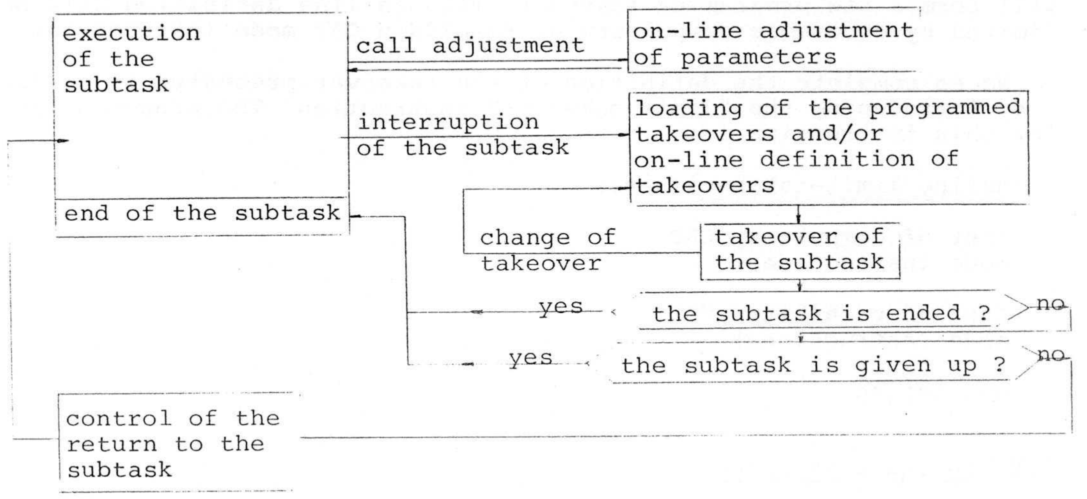
```
define limit-takeover(name)

(set of compulsory CAT
mode instructions)

(set of prohibited CAT
mode instructions)

end define
```

5. Execution principles

5.1. General organization

Three cases may occur during the execution of a subtask:
1. The subtask is entirely achieved in its nominal mode (automatic or CAT),
2. The subtask is interrupted and the call for a takeover allows to overcome the difficulty and the subtask can be achieved,
3. The subtask is interrupted but cannot be completely achieved: it must be abandoned and special response has to be envisaged in the case (off-site reprogramming, direct "hands-on" work, give up,...).

The call for takeovers is processed in the following way: when the interruption of the subtask happens, the systems loads the set of programmed takeovers that are contained in the "takeover" instructions, and the set of foreseeable CAT modes that are contained in the "limit-takeover" instruction. When a takeover has been loaded, the operator has the possibility to change for an other programmed takeover or to create on-line a new takeover thanks the specific menu. When the takeover procedure is ended, return to the subtask is controlled by a specific procedure that has to propose a choice of return criteria. Figure 2 presents the general organization of the subtask execution.



execution of the subtask
call adjustment
on-line adjustment of parameters
interruption of the subtask
loading of the programmed takeovers and/or on-line definition of takeovers
end of the subtask
change of takeover
takeover of the subtask
yes
the subtask is ended ?
no
yes
the subtask is given up ?
no
control of the return to the subtask

Figure 2: General organization of the subtask execution

5.2. Man-machine dialogue

The operator follows the task execution thanks to a control screen. He recieves the sucessive messages prepared during the programming that indicate to him the current state of the subtasks execution and guide him in his interventions. The operator can intervene at following levels:
- for adjusting a parameter of the current subtask, without interrupting the subtask (for example, concerning the spraying process),
- for realizing a programmed CAT subtask,
- for choosing or creating a takeover and continuing the interrupted subtask in CAT mode.

The set of these operations will be realized through an interactive dialogue.

6. Simulation example

A simulation program is being developped to evaluate the proposed concepts. We present in this paragraph a first example of simulation intented to illustrate the basic idea of associated takeover to an automatic subtask.

In this example, we consider the subtask consisting in painting a rectangular surface in robot mode. For performing this subtask, a straight-line trajectory is programmed at constant speed. The execution program is simulated thanks to a cartesian trajectory generator written on the model of the VAL language (the two basic instructions "MOVES" and "SPEED" have been simulated).

Besides, we introduce a specific model of the painting spraying cone illustrated on Figure 3.a, inspired from studies about robot painters /7/.

This model considers that the trace of the spraying cone on the surface is the following: within an inner circle of radius "r", the coating thickness is uniform and within a margin between the radius "r" and the radius "r'" it decreases outwards irregularly. This model being considered, the painting strategy consists in sweeping the surface with a step of "r+r'" that insures the overlaping of non uniform circle regions, as illustrated on Figure 3.b.

Furthemore, we assume that an obstacle, for example a pipe extremity, obstructs the programmed trajectory. We consider that a sensor system allows for the detection of the obstacle. When the obstacle has been detected, the robot arm stops, the spraying process is interrupted and the takeover procedures are loaded. The operator can so overcome the obstacle. At the end of this phase, he considers that the subtask can be pursued in the robot mode. A return criterium is then selected to perform the repositionning of the tool-effector on the nominal trajectory. The painting process starts again. Figure 4.a shows the simulated interruption of the automatic trajectory and its pursuit after the takeover. The points Q, Q', Q" respectively represent the interruption point of the programmed trajectory, the final point after moving the arm in takeover and the repositionning point on the trajectory.

However, during the takeover, the painting process was interrupted in the neighbourhood of the obstacle, as it is illustrated on Figure 4.b. It can be necessary to achieve this part after the subtask has been ended. It is then important to remark that this finishing phase can be more or less difficult:
- In the case where the process is not dirty or the obstacle is protected, this finishing phase can be easily made by defining an aided painting takeover that, for example, combines a freezing of the tool orientation and a distance to the surface servoing.
- In the case where the process is dirty and the obstacle is not protected, the finishing phase becomes more difficult, and can necessitate direct human intervention.

Lastly, Figure 5 presents the corresponding simulated robot program.

7. Conclusion

We have presented in this paper the definition of a programming language for telerobots. Our approach considers at the same level a double set of instructions corresponding to a robot mode and a CAT mode. It seems particularly adapted to the robotization of delicate or hazardous tasks as encountered in construction robotics. Further developments will lead to elaborate an experimental site in order to practically validate the proposed concepts.

References

/1/ B. Tondu, G. Clement, C. Rouveau, R. Colas, D. Ahsworth
"Presentation of EUREKA/GEO Project: A Façade Working Robot",
Proc. 4th International Symposium on Robotics and I.A. in Building
Construction, Haifa, Israel, June 1987, pp. 94-102.

/2/ J. Vertut, P. Coiffet
"Computer Aided Teleoperation", Volume 3b in "Robot Technology",
HERMES, Paris, 1985.

/3/ Jean Vertut Memorial Session, RoManSy '86, Cracow, Poland,
September 1986, pp. 3-41.

/4/ T. Sato, S. Hirai
"Language-Aided Robotic Teleoperation System (LARTS) for Advanced
Teleoperation", Proc. '85 ICAR, Tokyo, Japan, June 1985, pp. 329-336.

/5/ S. Hirai, T. Sato
"Advanced Master-Slave Manipulator Augmented with World Model",
Proc. ISIR 15, Tokyo, Japan, September 1985, pp. 137-144.

/6/ VAL Manual, Unimation Inc., 1980.

/7/ A. Klein
"Off-line Programming for Robot Painters", in Artificial Intelligence
and Information-Control Systems for Robots, Elsevier Science, 1984,
pp. 211-214.

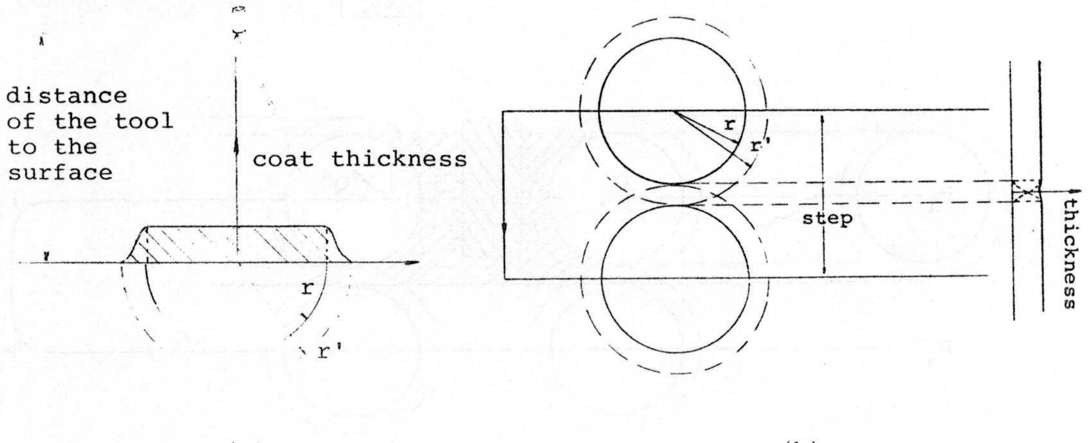


Figure 3: Modelisation of the process, (a) Model of the spray, (b) Method of surface coating

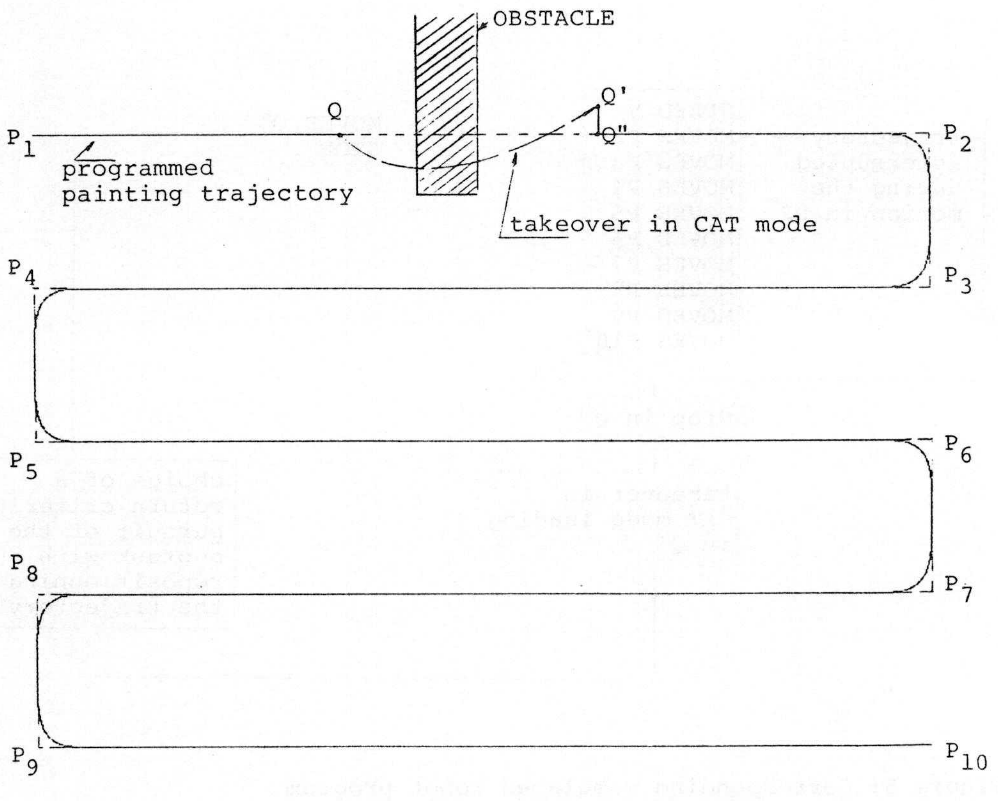


Figure 4.a: Example of automatic subtask and its takeover in CAT mode

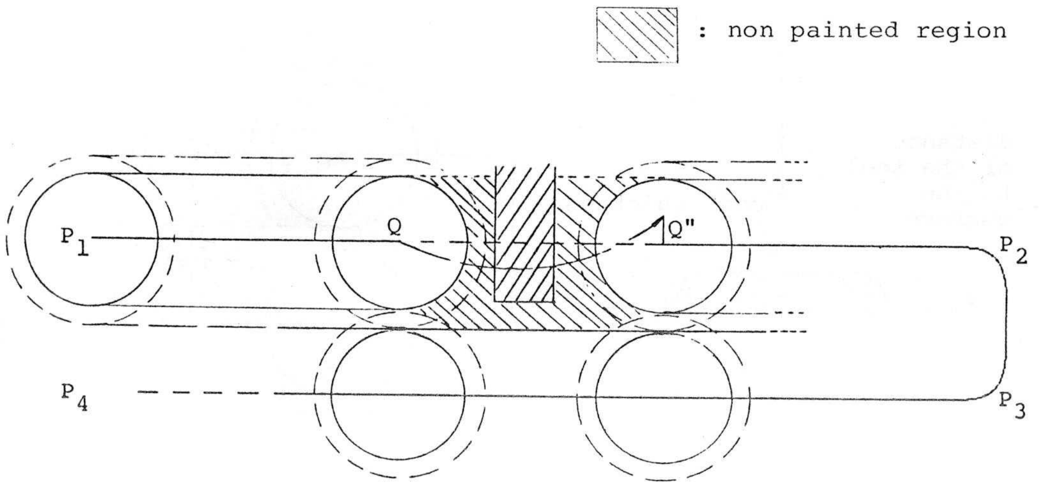


Figure 4.b: Trajectory part showing the spray trace and the region to be finished in the neighbourhood of the obstacle

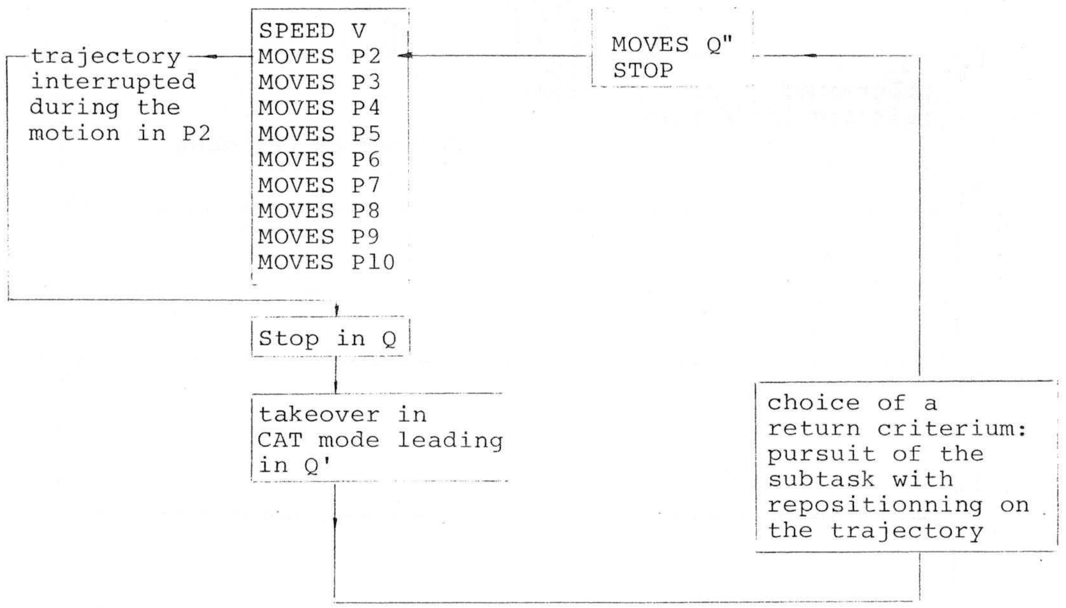

SPEED V
MOVES P2
MOVES P3
MOVES P4
MOVES P5
MOVES P6
MOVES P7
MOVES P8
MOVES P9
MOVES P10
trajectory interrupted during the motion in P2
Stop in Q
takeover in CAT mode leading in Q'
choice of a return criterium: pursuit of the subtask with repositionning on the trajectory
MOVES Q"
STOP

Figure 5: Corresponding simulated robot program