Programming industrial robots using advanced input-output devices: test-case example using a CAD package and a digital pen based on the Anoto technology

J. Norberto Pires[1], Tiago Godinho[1], Klas Nilsson[2], Mathias Haage[2] and Christhian Meyer[3]
[1] University of Coimbra/Mechanical Engineering Department, Coimbra, Portugal
[2] Lund University/Computer Science Department, Lund, Sweden
[3] Fraunhofer Institute/ Production Engineering and Automation, Stuttgart, Germany

Abstract— **Interaction with robot systems for specification of manufacturing tasks and motions needs to be simple, to enable wide-spread use of robots in SMEs. In the best case, existing practices from manual work could be used, to smoothly let current employees start using robot technology as a natural part of their work. Our aim is to simplify the robot programming task by allowing the user to simply make technical drawings on a sheet of paper. Craftsman use paper and raw sketches for several situations; to share ideas, to get a better imagination or to remember the customer situation. Currently these sketches have either to be interpreted by the worker when producing the final product by hand, or transferred into CAD file using an according tool. The former means that no automation is included, the latter means extra work and much experience in using the CAD tool.**

Our approach is to use the digital pen and paper from Anoto as input devices for SME robotic tasks, thereby creating simpler and more user friendly alternatives for programming, parameterization and commanding actions.

To this end, the basic technology has been investigated and fully working prototypes have been developed to explore the possibilities and limitation in the context of typical SME applications. Based on the encouraging experimental results, we believe that drawings on digital paper will, among other means of human-robot interaction, play an important role in manufacturing SMEs in the future.

***Index Terms* — CAD, Human machine interfaces, Industrial Robots, Robot programming.**

I. INTRODUCTION

The success of using robots with flexible manufacturing systems especially designed for small and medium enterprises (SME) depends on the human-machine interfaces (HMI) and on the operator skills. In fact, although many of these manufacturing systems are semi-autonomous, requiring only minor parameterization to work, many other systems working in SMEs require heavy parameterization and reconfiguration to adapt to the type of production that changes drastically with time and product models. Another difficulty is the average skill of the available operators, who usually have difficulty adapting to robotic and/or computer-controlled, flexible manufacturing systems. This reality configures a scenario in which flexible automation, and robotics in particular, play a special and unique role requiring manufacturing cells to be easily used by regular non-skilled operators, and easier to program, control and monitor. One way to this end is the exploitation of the computer input-output devices to operate with industrial robotic equipment. With this approach, developers can benefit from the availability, and functionality of these devices, and from the powerful programming packages available for the most common desktop and embedded platforms. On the other hand, users could benefit from the operational gains obtained by having the normal tasks performed using common devices, and also from the reduction in prices that can arise from using consumer products [1].

Industrial manufacturing systems would benefit greatly from improved interaction devices for human-machine interface even if the technology is not so advanced. Gains in autonomy, efficiency, and agility would be evident. The modern world requires better products at lower prices, requiring even more efficient manufacturing plants because the focus is on achieving better quality products, using faster and cheaper procedures. This means having systems that require less operator intervention to work normally, better human-machine interfaces, and cooperation between humans and machines sharing the same workspace as real coworkers [1].

Also, the robot and robotic cell programming task would benefit very much from improved and easy-to-use interaction devices. This means that availability of SDKs and programming libraries supported under common programming environments is necessary.

This paper explores the utilization of programmable digital pens [2, 3] for the task of programming industrial robot manipulators. The objective is to demonstrate the usefulness and potential of these devices, defining a

suitable platform that could be used to implement user services to support new and advanced features.

II. THE ANOTO DIGITAL PEN

A digital pen is a device that stores the user hand writing in a digital format, providing a mechanism for the user to access the stored data from a computer. The concept developed by the Anoto [2] is very innovative and suitable for robotic applications. The technology developed by Anoto, entailing interpretation and transmission of handwritten text and images, is based on a special digital pen and a paper printed with a pattern that is almost invisible to the eye (Figure 1). The digital pen uses ink and handles just like a normal ballpoint pen, but it also contains a digital camera, an advanced image processing system and a communication unit, for example for wireless Bluetooth connection to a mobile phone.

The paper consists of an ordinary paper printed with a dot pattern that is either pre-printed or printed on a laser printer. The displacement of the dots, 0.1 millimeters in size, from the relative position enables them to be programmed to tell the pen the exact location on the page – or the whole pad of papers – one is writing on.

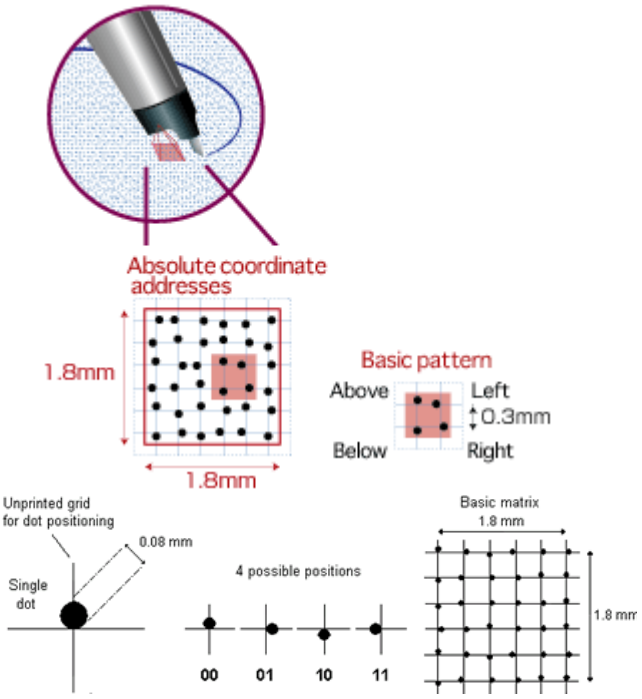

Figure 1. Dot pattern used by the digital pen [2]

Briefly, a device like this allows the user to extract the following information (Figure 2):

By registering the pen's movement across the paper, and also the pressure, the writing is interpreted and digitalized. Hence the technology is not based on characters having to be written in a special way, in contrast to various other applications such as hand-held computers. Even drawn sketches can be interpreted and transferred. Since the pen's movements are stored as a series of map coordinates and the paper defines where on the paper one is writing, it is possible to go back and complete previous notes in a pad. The technology is capable of interpreting this correctly and putting it in the right context when transmitting it to the digital media. The pattern enables different parts of the paper to be assigned different functions such as predefining various parts of a form.

1. Absolute position of the pen on the paper page.
2. The sequence of pen movements as a map of coordinates.
3. Each coordinate has information about the position, the time stamp (both relative to the start of the stroke and current time when it is written), and the line color and width.

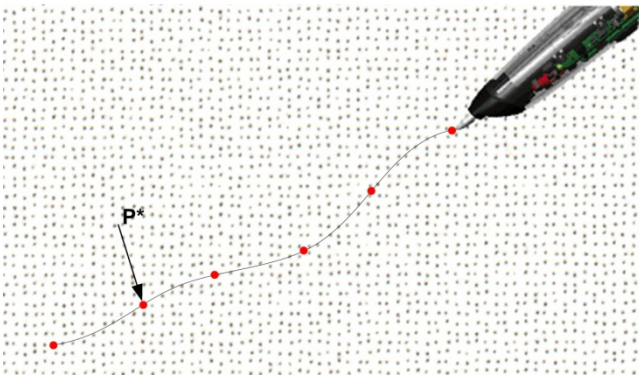

Figure 2. Sample line stroke: P* contains position, time, status and stroke properties information

Consequently, if the user technical drawing is a robot path over a CAD drawing, for example, then the extracted sequence of positions and time stamps, etc., is a complete definition of the motion specified by the user. Selecting a proper set of reference points/orientations this information can be used to allow users to specify a sequence of robot actions just by drawing them on a paper. If a few rules are observed the information extracted from the pen can be used to generate robot programs, i.e., the user can program a robot just by drawing the robot actions on a paper, like it's common for any engineer. Anoto developed a software platform (Figure 3) to interface their pens with personal computers, pocket PC's, mobile phones, etc., and released the necessary APIs to allow users to fully explore the device features [4, 5].

This paper explores the digital pen as a robot input device, showing 3 different approaches/implementations designed to program robot manipulators. The first implementation presented explores the possibility to specify robot motions on CAD drawings, and using specially design computer applications to include the robot paths with the CAD package, generate the robot program code, download it to the robot controller and run it remotely. The second approach moves toward creating an industrial test case instructing cutting operations in a foundry application using the digital pen based on the Anoto technology. The third implementation deals with

the generation of CAD data and robot programs from technical hand drawings. Geometric shapes and their measures are taken from a simple sketch. The user is able to define robot programs without dealing with complex interfaces.

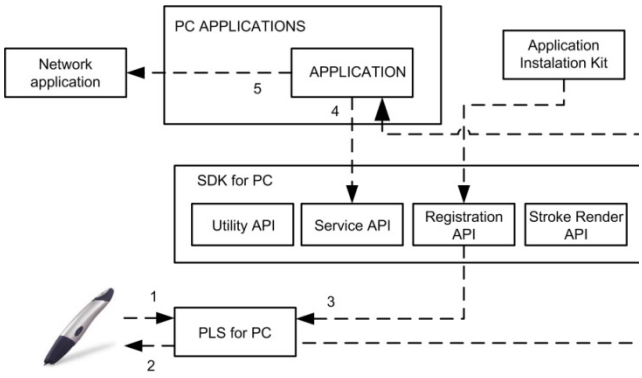

Figure 3. The PC architecture used by the digital pen [2]

III. ROBOT PROGRAMMING: CAD INTERFACE

This implementation uses the Anoto platform and uses the generated PGC (Pattern Generated Coordinates) files as input, which contains the data from the pen and is obtained when the user synchronizes the pen with the PC infrastructure (just by putting the pen in the docking station or requesting synchronization by Bluetooth).

Using the Anoto APIs an application was designed [6] to perform the following services (Figure 4):

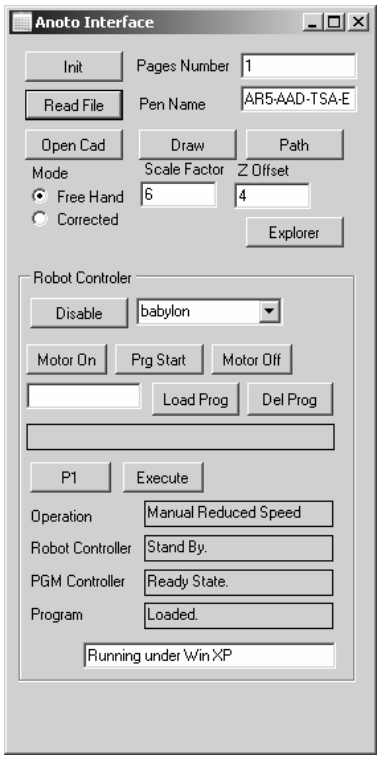

Figure 4. Application developed for the CAD interface

1. Extract the data from the PGC file.
2. Import the user strokes to AUTOCAD [7] using the specified file and layer. The strokes are printed in the screen using also the user selected color.
3. Generate the robot code based on the received information, using two different operating modes: *Free mode*, where the strokes are sent to the robot as they come from the pen and the robot is commanded to move in line between points, and *Corrected Mode*, where the type of stroke is identified and mapped to known robot types of motions (linear, arc, circular, etc.).

To access data from the PGC file the following routine is used:

```
Dim page As Anoto.Common.Page
Dim data As Object
file = New Anoto.Util.File
penRequest = New Anoto.Service.PenRequest
data = file.Read(currentFileName)
penRequest.Initialize(data)
PagesNumber = penRequest.Pages.Count()
txt_PagesNumber.Text =
Convert.ToString(PagesNumber)
PenName = penRequest.PenId
txt_PenName.Text = PenName
If Not penRequest Is Nothing Then
  For Each page In penRequest.Pages
    Number_Strokes = page.PenStrokes.Count
    strokes = page.PenStrokes
      For Each stroke In strokes
        Read_Strokes(k_stroke).x = stroke.X
        Read_Strokes(k_stroke).y = stroke.Y
      Next
  Next
End If
```

The position information obtained from PGC files is expressed in Anoto coordinates units (Figure 1) and referenced to the paper left upper position. Each unit corresponds to 0.3 mm or 84.67 dpi. If the user selects the *Free Mode* of operation (Figure 4), the start and end positions of each stroke are sent directly to AUTOCAD using the following method [8]:

```
line =
Acad_Doc.ModelSpace.AddLine(StartPoint, EndPoint)
```

and constitute also the start and endpoint of a robot *move_line* instruction (a simple computation of the orientation is obtained from AUTOCAD just by aligning with the predefined *workobject*).

If the *Corrected Mode* is selected, then the software tries to identify the type of stroke. In this example, 2 types of strokes are used: lines and arcs. The identification procedure is very simple and based on the method

presented in [9] which computes the linearity of a stroke (k_l) given by:

$$k_l = \frac{\Delta L}{\sum_{i=1}^{n} \Delta l_i} \quad (1)$$

where n is the obtained number of segments of the stroke, ΔL is the computed distance between the stroke start and end points and Δl_i is the computed distance between the i-segment start and end points.
The following rule is used:

If $k_i \geq \varsigma \Rightarrow$ the stroke is a line
If $k_i < \varsigma \Rightarrow$ the stroke is a arc

The parameter ς is adjusted by the user (we used $\varsigma = 0.98$ in this example) to optimize the identification procedure. If the stroke is identified as an arc, it's necessary to compute the midpoint of the arc. The procedure, depicted in figure 5, is the following:

1. Define the vector **v** from the stroke start point to the stroke end point.
2. Define the vector $\mathbf{v_1}$ collinear with **v**, but with origin the in the midpoint of **v** and pointing to the stroke end point.
3. Define the vector $\mathbf{v_n}$ with the same origin as $\mathbf{v_1}$ but pointing to the stroke point n.
4. Compute the angle (α) between $\mathbf{v_1}$ and $\mathbf{v_n}$.
5. The arc mid point is the stroke point defined by $\alpha = 90º$

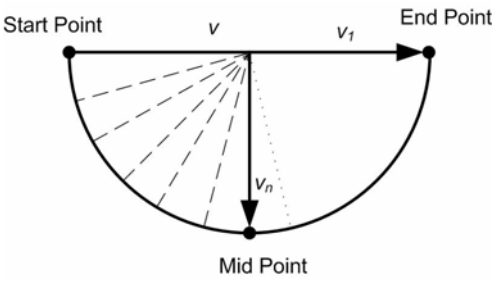

Figure 5. Computing the arc mid point

The start, mid and end points of the arc are the necessary parameters to parameterize the *move_arc* instruction. On AUTOCAD, when using the Autocad API, the parameters necessary to define an arc are the center of the arc, the radius of the arc and the angles correspondent to the start and end point (angles are measured from X axis in the anti-clock direction). These parameters are easily computed from the previously defined 3 points. To draw an arc in AUTOCAD using the obtained parameters the following method is used:

arc = Acad_Doc.ModelSpace.AddArc(CPoint, radius, alfa, beta)

where *CPoint* is the arc central point, *radius* is the arc radius, and *alfa* and *beta* are the arc initial and end angles.

From our experiments it was observed that if the user is careful and makes drawings with firm hand, the identification method works without faults.

To map the obtained points to robot positions a user defined *workobject* is used. The *workobject* is defined in AUTOCAD and is composed by a point and a XYZ reference frame. The origin of the *workobject* is printed in the working paper, and the first thing that a user must do is to mark that point with the pen. Our software computes all positions and orientations relative to the defined *workobject*. The obtained positions and computed points, depending if the user is using the *Free Mode* or the *Corrected Mode*, are stored in a RAPID matrix with the following structure:

Const num pp{284, 4}:=[[442.35, -43.87501, zz, 0],[442.125, -43.65001, zz, 1],[441.225, -43.42501, zz, 1], [441.45, -43.42501, zz, 1],[441.45, -43.87501, zz, 1],…

where the first value is the X coordinate, the second value is the Y coordinate, the third value (zz) is the Z coordinate (the user defined distance from the robot tool to the working piece) and the fourth value defines the type of motion to execute (0 – joint motion, 1 – linear motion, 2 – circular motion). This last value results from the identification procedure, i.e., if the user selects the *Free Mode*, then this value is equal to one (linear motion) to all the points except the first; on the other hand, if the user selects *Corrected Mode* the value is equal to 2 (circular motion) except the first one. All flying trajectories between consecutive robot paths (correspondent to different pen strokes) are set as joint motion trajectories (parameter 4 of the above mentioned matrix equal to zero).

The RAPID code below was generated for the example depicted in Figure 6:

```
WHILE i < moves +1 DO
  tempx:= pp{i, 1} - p3.trans.x;
  tempy:= pp{i, 2} - p3.trans.y;
  tempz:= pp{i, 3} - p3.trans.z;
  p3:=Offs(p3, tempx,tempy, tempz);
  IF pp{i, 4} = 0 THEN
    ConfJ \Off;
    MoveJ p3,v100,fine,trj_tool\WObj:=trj_wobj;
    ConfJ \Off;
  ENDIF
  IF pp{i, 4} = 1 THEN
    MoveL p3,v100,fine,trj_tool\WObj:=trj_wobj;
  ENDIF
  IF pp{i, 4} = 2 THEN
    i :=i+1;
    tempx:= pp{i, 1} - p3.trans.x;
    tempy:= pp{i, 2} - p3.trans.y;
    tempz:= pp{i, 3} - p3.trans.z;
    p4:=Offs(p3, tempx,tempy, tempz);
    MoveC p3,p4,v100,fine,trj_tool\WObj:=trj_wobj;
```

```
    ENDIF
    p3:=CRobT(\Tool:=trj_tool\WObj:=trj_wobj);
    i := i +1;
  ENDWHILE
```

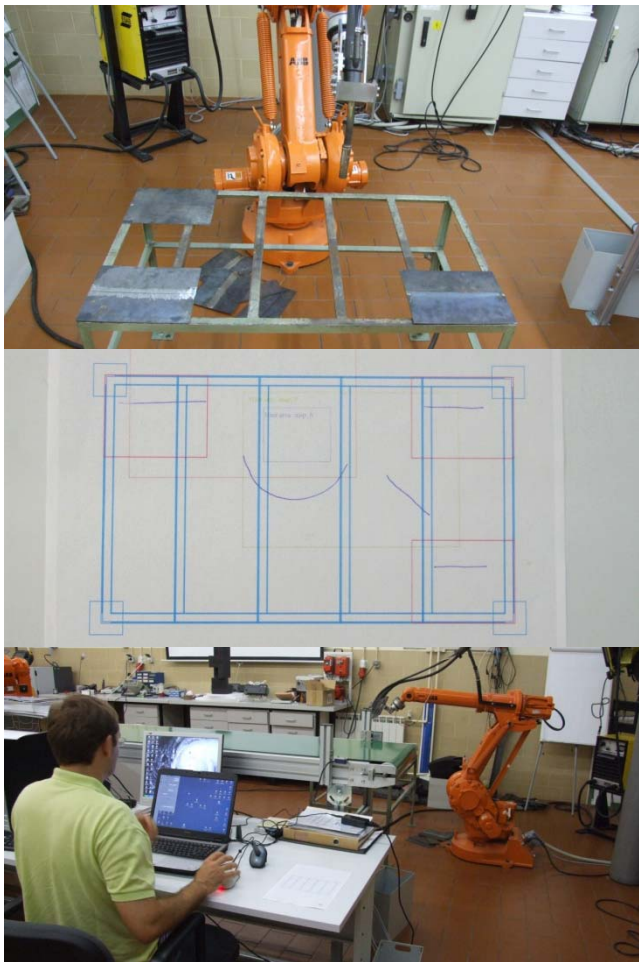

Figure 6. Setup used for the demonstration

The demonstration presented here uses a welding robot (ABB IRB1400 robot equipped the S4 robot controller [10]), connected to a local area network and controlled from a laptop using software based on Remote Procedure Calls (RPC). The working table is modeled in AUTOCAD and the user is able to:

1. Draw robot paths in 2D using a CAD printout of the working table and the Anoto digital pen.
2. The obtained robot paths are transferred to AUTOCAD and identified using the procedure explained above.
3. An application is used to generate the robot code necessary to run the programmed paths, using a procedure as explained above.
4. Download the generated robot code to the robot controller and the execution commanded from the PC.

This example, developed at the Mechanical Engineering Department of the University of Coimbra and briefly demonstrated in the video included with this paper, shows fully the usefulness of this type of devices for robotic programming applications [11].

IV. DEVELOPMENT TOWARDS FOUNDRY APPLICATIONS

Due to the bad working conditions, cutting and deburring in foundry industries is an important robot application, and this is one of the key demonstrators for new easy-to-use robots. To validate the usefulness of the digital pen and paper approach for robot programming and interaction in that area, initial experiments have been carried out using the work pieces from one of our end users (Norton Casts in the UK). The application includes material removal by cutting the input metal reservoirs, which are needed for the casting process but not part of the product (compare left and right picture in Figure 7). A second stage of the process is deburring, which we also think can be simplified by using the Anoto technology, but in this paper we focus on specification and/or programming of the cutting.

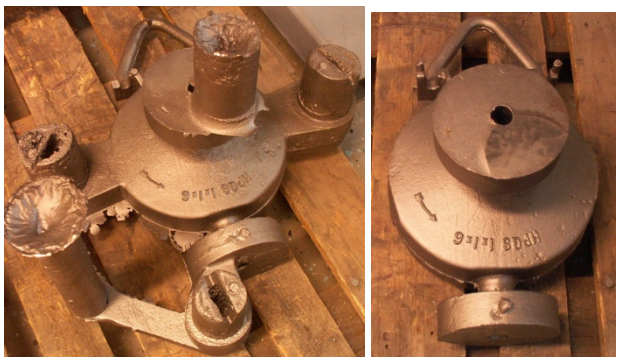

Figure 7. Example foundry workpieces. Left: before cutting. Right: after cutting.

Since the foundry workpieces are produced in very small batches (1-10), it is desirable to find a quick and easy way of specifying the task such that it can be carried out by the automation equipment, including the robot, the fixture, and the cutting tool (*oxyfuel* burner in our case). An initial task specification experiment is shown in Figure 8. The experienced simplicity of system development and data import, in combination with the natural and simple way the system can be used, clearly indicate the potential and applicability of the suggested techniques.

The data to the lower left in Figure 8, as obtained from the Anoto interfaces, is then mapped to the coordinates used in the robot program. This is similar to the other applications is this paper.

Taking a closer look at the system and further developments, Figure 9 shows an example of how the data flow could look like in the prototype system. The layout and contents of the form enabling the Anoto technology need to be defined to allow efficient definition of the task. In order to do that one or several images or poses of the workpiece is necessary to print on the paper enabling the Anoto technology, however, in the

foundry industry CAD models featuring metal reservoirs are very rarely produced, indicating that a method is necessary for either reconstructing geometry or using camera images of the workpiece directly printed on paper. The filled in form needs post processing for interpreting the obtained data (pen strokes), paths or symbols.

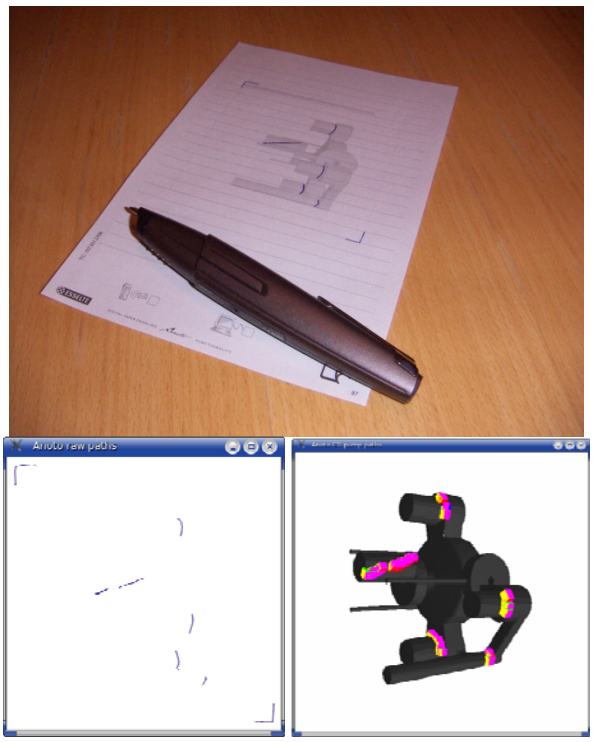

Figure 8. Foundry workpiece experiment. In upper picture the pen with the CAD model of the foundry workpiece printed on the paper enabling the Anoto technology and the pen strokes specify the cuts to be made. In the lower part, the imported pen strokes (left) and the surface mapping on workpiece 3D image.

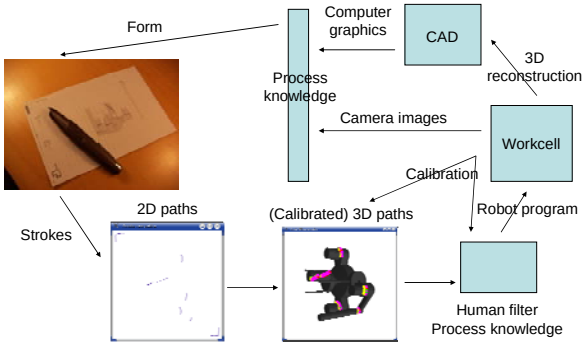

Figure 9. Approach for further development. Colored boxes are part of ongoing work, whereas the non-colored boxes are part of the initial experiments.

Based on these findings the proposed approach for further development includes:

1. Obtain metal reservoir workpiece geometry and add to CAD model.
2. Assemble and print form for defining cut task
3. Interpret pen strokes and identify cut paths on the workpiece
4. Generate the robot program

Note that the end user simply works with pen and paper in combination with images and the real equipment; no programming skill required.

V. CONCLUSION

In this paper the authors explore the utilization of digital pens for the task of programming industrial robot manipulators. Three different implementations were presented to assess the digital pen features using different robotic setups. The results clearly show that the digital pen based on Anoto technology is very useful and powerful, being an interesting solution for certain advanced applications involving the task of programming robotic installation just by making technical drawings on a sheet of paper. The next steps will be to adopt a software infrastructure and develop the necessary services to allow system integrators to consider this type of device an advanced user-friendly user programming method.

ACKNOWLEDGMENT

This work has been partially funded by the European Commission's Sixth Framework Program under grant no. 011838 as part of the Integrated Project SMErobot.

The authors want also to thank Anoto for the support and for the access to the SDKs developed for the digital pen based on the Anoto technology.

REFERENCES

[1] JN Pires, "Industrial Robot Programming, Building Applications for the Factories of the Future", Springer, New-York, 2006.
[2] Anoto, http://www.anoto.com
[3] Logitech, http://www.logitech.com
[4] API reference, Anoto SDK for PC applications (3.2), Anoto, 2006
[5] Anoto SDK for PC applications, Anoto, 2006
[6] Visual Basic Reference Guide, Visual Studio .NET 2005 Documentation, MSDN, Microsoft, 2005
[7] Autodesk, http://www.autodesk.com
[8] AutoCad 2006 Developers Help, Autodesk, 2006
[9] S. F. Qin, "Intelligent Classification of Sketch Strokes", Eurocon 2005: The International Conference On Computer As A Tool, Vol 1 And 2 , Proceedings: 1374-1377, 2005
[10] ABB IRB1400 Users and Programming Manual, ABB Robotics, 1995.
[11] Demonstration videos at http://robotics.dem.uc.pt/docs/video8/

AUTHORS

J. Norberto Pires is with the Mechanical Engineering Department, Faculty of Sciences and Technology, University of Coimbra, Polo II Campus, Coimbra, Portugal (e-mail: norberto@robotics.dem.uc.pt).

Tiago Godinho is with the Mechanical Engineering Department, Faculty of Sciences and Technology,

University of Coimbra, Polo II Campus, Coimbra, Portugal (e-mail: norberto@robotics.dem.uc.pt).

Klas Nilsson is with the Computer Science Department, University of Lund, Sweden (e-mail: klas@cs.lth.se).

Mathias Haage is with the Computer Science Department, University of Lund, Sweden (e-mail: mathias@cs.lth.se).

Cristhian Meyer is with the Fraunhofer Institute, IPA, Stuttgart, Germany (e-mail: crm@ipa.fhg.de).

Manuscript received on June 26, 2007.
Published as submitted by the author(s).
Paper presented at REV2007 conference, Porto, Portugal, June 2007.