Adaptive Robot Based Visual Inspection of Complex Parts

M. Gauss[1], A. Buerkle[1], T. Laengle[1], H. Woern[1],
J. Stelter[2], S. Ruhmkorf [3], R. Middelmann[4]

[1]Institute for Process Control and Robotics (IPR), Universitaet Karlsruhe (TH), Germany
[2]AMATEC ROBOTICS GmbH, Germering, Germany
[3]ISRA VISION SYSTEMS AG, Karlsruhe, Germany
[4]VITRONIC Dr.-Ing. Stein Bildverarbeitungssysteme GmbH, Wiesbaden, Germany

Abstract

In this paper an architecture for robot based visual inspection of complex parts is presented. Based on this architecture two systems have been implemented, one for visual engine compartment inspection and another for visual weld seam inspection for car manufacturing. The two main goals of the work are: (1) Defining open interfaces between a host computer, an industrial robot and a vision system. (2) Automating the adaptation of a robot based visual inspection system to new tasks.

1. Introduction

The use of image processing systems in industrial manufacturing and assembling has significantly increased in recent years. Used as inspection systems they can enhance product quality and minimize loss through waste. The employment of such systems is profitable only under certain conditions. Fixed costs of acquisition, adaptation and initial operation are to be compared to saving of costs of the ongoing operation.

In order to not only use the inspection systems at huge lot sizes reasonably, it is necessary that the systems can be adapted to a certain problem with little effort.

The goal of the ARIKT[1] project is to develop an architecture for a robot based visual inspection system which is easy to adapt to new tasks. The problem of inspecting also large objects and inspecting objects from changing points of view is solved by the employment of a robot as carrier of a camera. In order to make possible that robots and image processing systems from different manufacturers work together in ‘plug and work’ manner, a standard communication profile is suggested for the control of these components. To achieve easy adaptation to new tasks the user is supported in the determination of machine vision procedures and parameters by the system. Therefore, a knowledge base associated with the vision system is used. The position of the camera in an inspection task is determined by the necessary distance and the location of the part to be inspected. Thus the automatic programming of a robot to reach the desired camera position is possible, when workcell geometry and robot are modeled.

In section 2 a short overview of the system is given. The main components host, robot and vision system and their collaboration are introduced.

In section 3 two inspection applications are presented, which are implemented based on the ARIKT architecture: a engine compartment inspection and a weld seam inspection.

In section 4 a protocol profile for communication between a host, a vision system and an industrial robot is proposed. On application level XML[2] [1][2] is used to express robot and vision commands. As communication base Ethernet with TCP/IP is used.

Section 5 shows how the user can be supported in

[1] ARIKT = Adaptive Roboter-Inspektion komplexer Teile (German: adaptive robot inspection of complex parts)

[2] XML = eXtensible Markup Language

adapting the inspection system to new tasks. The knowledge base is described which helps with the determination of vision procedures/parameters for a given inspection task. The automatic computation of robot trajectories and the integrated model for workcell and product are illustrated.

2. System Overview

In the framework of the ARIKT project the following main components are provided: Host, robot and vision system.

During an inspection procedure (*online phase*) the collaboration between the main components works simplified as follows: First, the host sends robot commands to the robot control (see Figure 1). The robot moves the camera resp. sensor head to the desired position. It announces reaching this position to the host. The host then transmits and starts the procedure of the vision system. When finished, the vision system returns the inspection result. The determination, whether a part is faulty or not, is only task of the vision system.

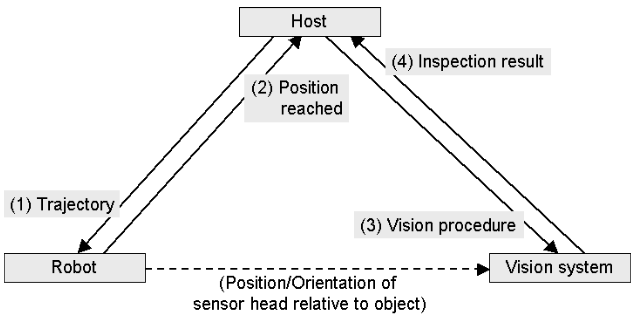

Figure 1: Collaboration during the online phase of the inspection

In some applications one image is acquired from one single position for analysis, e.g. the engine compartment inspection. In other applications the camera has to be moved on a continuous path and a series of images is acquired, e.g. the weld seam inspection. In the latter case cyclic transmission of position and orientation of the sensor head from robot to vision system can be provided.

The commands send from the host to robot and vision system are stored in an *inspection plan*. It consists of one or more *tasks* which are build up of one or more *vision commands* and one or more *robot commands* and the *analysis result*.

The inspection plan is build up *offline* and stored on the host (see section 5).

3. Inspection Applications

3.1. Engine Compartment Inspection

The task is comprised of several inspections of different components within the engine compartment, the assembly of which can be faulty. The assembly faults are unknown ahead of inspection. Additionally the occurrence of assembly faults can vary strongly with regard to the timing aspect.

The inspection tasks chosen as examples within the ARIKT project can be subdivided into three different classes. The first class is comprised of the inspection of correct positioning of tube connectors and clamping pieces (see Figure 2). The second class takes care of the verification of distances between cables, tubes and pipes. The third class embraces inspections of presence of sealing plugs, cables or tubes in gaps of the car body.

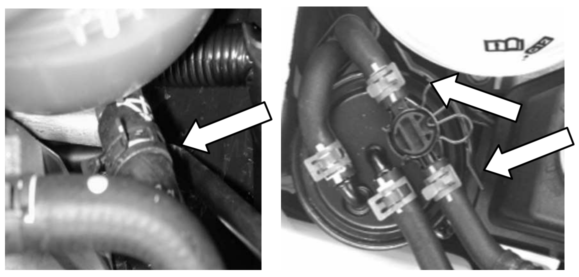

Figure 2: Inspection of clamping pieces

The control algorithms are started by a control job from the host during which they are fed with all relevant data regarding the objects to be inspected. This data consists of size, position, shape of the object as well as of information regarding the surface properties of object and background. On the basis of this information the chosen inspection algorithm is parameterised and executed automatically. The inspection results are sent back to the host computer. The host can select different inspection algorithms. The algorithm itself can be either a simple task as a light meter or a blob analysis or a complex multi-task inspection as for example a complete control of the clamping piece.

The sensor head, as integral part of the vision system, features a shuttered camera and an externally controlled LED ring light. The ring light is manufactured with multi-colour LEDs, thereby enabling an automatic adaptation of the illumination to the inspection task at hand.

The basis for the modular and adaptive vision system is a standard vision system for general assembly control and quality inspection. The adaptation to several different tasks as presence/absence control, gauging, dimension control, quality inspection of objects and object surfaces is possible through the multitude of integrated inspection tools and algorithms.

3.2. Weld Seam Inspection

The requirements for weld inspection are primarily drawn from standards such as EN30042 [3] / ISO5817 [4], combined with experience gained in developing inspection systems for automotive requirements. Using the principal of laser triangulation, a high speed, high resolution 3D representation of the surface of the weld is generated. A series of feature extraction algorithms each determine different properties of the weld at every location where a cross sectional profile was obtained from the sensor. The measured properties of the weld are compared with the tolerances specified in the standards, to arrive at an overall assessment of the acceptability of the weld.

A specifically designed sensor that integrates a laser with line generating lens and a high resolution camera produces a stream of 2D cross sectional profiles of the surface of the weld and the adjacent unwelded material at the location where the line is projected. Relative movement between the sensor and the part being inspected is required to build up the third dimension in the measurement data. For complex parts, this relative movement is ideally provided by a robot which is able to maintain a constant distance and viewing angle between the sensor and the weld.

As the measurement system is optical, only the surface of the weld is mapped. The applied inspection criteria therefore form subset of the criteria specified in the standards. Typical examples of these criteria include measurement of the height and cross sectional area of the weld, together with detection of porosity density and undercuts.

Whilst calibration of the sensor as such results in accurate 2D data within each profile, the accuracy of the third dimension depends entirely on the relative movement speed between the sensor and the robot. In some applications a constant velocity is achievable, however the frequently encountered complex geometry of automotive components and the kinematics of the robot result in a non-constant velocity. To prevent this causing measurement errors, the robot is required to constantly communicate its Tool Center Point (TCP) to the inspection system. The actual data transmitted is the distance traveled, relative to a defined weld start or trigger point. This data is cyclically transmitted as an XML packet, containing at a minimum a single distance value. Accuracy of the transmitted data packets is further enhanced through inclusion of a time stamp.

4. Communication between Main Components

4.1. Communication Protocol Profile

Communication between devices can be divided into layers according to the ISO/OSI reference model [5]. To allow collaboration between components in ‘plug and work’ manner communication on each layer has to be defined. The protocol profile of the ARIKT system is listed in Table 1. On layers 1-6 existing standard protocols are used. The application level will be considered in section 4.2 and 4.3.

Layer	Task	Protocol
7	Application	ARIKT
6	Presentation	XML
5	Session	TCP
4	Transport	TCP
3	Network	IP
2	Data link	Ethernet 10Mbit/s
1	Physical	10BaseT

Table 1: ARIKT communication profile

Ethernet with TCP/IP in general is not real time compliant because of the used medium access protocol CSMA [6]. But, the cyclic transmission of position and orientation of the sensor head from robot to vision system (see Figure 1) shall run in assured short periods of time. To realize real time transmission with Ethernet/TCP/IP two techniques can be used:

- Communication of certain partners is paused according to an application level protocol. While transmission of position of the sensor head from robot to vision system the host does not contact robot or vision system.
- Through the use of an (Ethernet-) *switch* one *collision domain* can be split up into several smaller ones.

XML as syntactical format for application telegrams compared to binary format brings some advantages: The content is readable and understandable by humans with a simple text editor. The structure of content can be defined in *XML schemas* [7]. (XML documents look like HTML documents with self-defined tags.) For syntactical parsing free software modules are available. Disadvantages are: The size in bytes for XML messages is significantly larger than that of binary data. Software complexity is increased by the need for syntactical parsing.

4.2. XML Robot Commands

The robot commands, which have been defined, are designed for remote control of industrial robots. The goal is to control robots from different vendors with the same commands. The commands are intended to allow the definition of trajectories and to support synchronization between a robot and external devices.

A condensed example for an XML robot program is given in Figure 3. It is used for weld seam inspection.

With the *TCP* element the Tool Center Point can be defined. A *Vector* element is used to describe displacements in mm and a rotation around the axis x,y and z. The *Velocity* element determines the velocity of the TCP in mm per second. The *Accuracy* element declares the distance which the TCP may diverge from defined waypoints. It results in rounding or fly-by behavior. The *Linear* element defines a linear movement. Beside linear movements there are existing commands for circle and point-to-point (PTP) movements. To define waypoints and orientation of the TCP the *Point* element is used. Also Vector elements could be used here instead.

All commands listed until now allow to define movements and trajectories. The *OutputSignal* element accomplishes synchronization between the robot and the host instance. After the completion of the last movement before an OutputSignal the robot sends a message to the host. The host knows then the status of the robot and can send for example a 'start-your-work' command to the vision system.

```
<?xml version="1.0" encoding="UTF-8"?>
<AR:RobotCommand ...>
  <TCP>
    <Vector XPos="0" YPos="-77" ZPos="43" RX="90" RY="0" ... />
  </TCP>
   <Velocity Nominal="150" Min="0" Max="115"/>
   <Accuracy ToleranceXYZ="10"/>
   <Linear>        <!-- accelaration phase -->
     <Point XPos="953.68" YPos="-144.42" ZPos="588.02"
       RX="81.8305" RY="-5.4639" RZ="174.4402" />
       ...
     <Point XPos="1474.54" YPos="-106.98" ZPos="482.48" ...  />
   </Linear>
   <OutputSignal Value="Position reached"/> <!-- begin of weld  -->
   <Linear>
     <Point XPos="1474.91" YPos="-106.94" ZPos="469.19" ... />
     <Point XPos="1475.44" YPos="-106.87" ZPos="448.87" ... />
     ...
     <Point XPos="1331.11" YPos="-270.06" ZPos="-32.53"  ... />
     <Point XPos="1326.17" YPos="-302.55" ZPos="-48.2" ... />
   </Linear>
   <OutputSignal Value="Position reached"/> <!-- end of weld seam -->
</AR:RobotCommand>
```

Figure 3: XML robot commands sample

The leaving out of robot joint-angles in commands and the exclusive use of Cartesian coordinates allow to use the same robot program with different robot types.

There are already existing attempts to form vendor independent robot programming languages. In DIN 66312 [8] the *Industrial Robot Language* (IRL) is defined. In contrast to the ARIKT robot command set the IRL provides a 'full' robot programming language. IRL allows to program robots in a type specific way and is not intended for remote control. IRL is not commonly supported by robot vendors (maybe due to its complexity).

4.3. XML Vision Commands

The enormous variety of optical sensors and their broad range of image processing algorithms has predestinated the class of vision sensors for many different applications such as object position detection, object recognition, barcode reading, seam-inspection and so on. Against this background, the usability of a vision system in conjunction with an industrial robot depends highly on the integration of robot and vision system in terms of:

- communication
- operation
- service and diagnosis

To ensure a transparent and intuitive communication of robot applications based on vision sensors a set of abstract vision commands has been designed. The main goal of those vision commands is to provide:

- application independency
- robot independency
- consistent command syntax
- flexibility through command parameterization

The command syntax is based on an abstract view onto a common vision based robot application. The center of all considerations is a physical object (Obj), that is dealt with by the inspection system, i.e. a workpiece. Figure 4 shows exemplarily the structure of the "LoadObj"-command whereas Table 2 provides an overview of all predefined commands.

Command	Description
LoadObj(<objName>)	Loads an object pattern (image processing program) into the vision system. Onto this pattern all further image processing tasks are performed
Trigger()	Triggers the image acquisition

StartTrigger()
StopTrigger()
	Starts and stops the cyclical triggering of the vision system by the robot

SetSysParameter()	Sets global parameters on the vision system, e. g. operation mode, timeout, etc.
SetParameter()	Sets task specific parameters, e. g. setting for integrated illumination devices
GetStatus()	Gets the status (e. g. idle, busy, waiting) of the vision system
Reset()	Resets the vision system to an initial state

Table 2: Vision commands

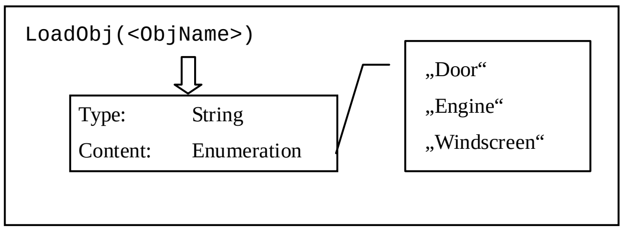

Figure 4: Command structure “LoadObj”

The return value of a dedicate command is defined similarly within a schema definition file. The types of the return value are basically defined accordingly to fundamental types of higher programming languages. Only the representation of coordinate systems or transformation matrices respectively, a fundamental data type in robot applications, and a variable type to indicate the status of the vision system are defined separately.

Type	Description
BOOLEAN	Designates boolean variable
INTEGER	Designates integer variable
REAL	Designates floating point variable
STRING	Designates character- based variables
FRAME	Contains a homogenous transformation matrix
VISIONSTATUS	Contains detailed information about the vision system

Table 3: Data types

Advanced mechanisms of the schema definition can be used to specify lower boundaries for either range limitation (Figure 4) by setting upper and lower boundaries or restrictions in size of a type by predefining minimal and maximal data length (Figure 5). Those mechanisms can be used to customize the type definitions accordingly to the conditions of the robot controller.

```
<xs:simpleType name="REAL"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:restriction base="xs:double">
        <xs:minInclusive value="-3.4E+38" />
        <xs:maxInclusive value="3.4E+38" />
    </xs:restriction>
</xs:simpleType>
```

Figure 5: Schema definition for type REAL

```
<xs:simpleType name="STRING"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:restriction base="xs:string">
        <xs:maxLength value="80" />
        <xs:minLength value="0" />
    </xs:restriction>
</xs:simpleType>
```

Figure 6: Schema definition for type STRING

For each type shown in Table 3 an accessing function is provided which is structured consistently:
```
get<TYPE>(<ObjName>.<TaskName>.
                              <VariableName>)
```

To be more specific, the use of the vision commands will be demonstrated with the following application scenario: a vision equipped robot has to check, if the oil-filter has been mounted to an engine block. The first step is to prepare the vision system by implementing an appropriate image processing mechanism. This mechanism is named intuitively, e. g. “CheckOilFilter” and might consist of several tasks that operate upon the oil-filter. E. g. one task is for checking the existence of the oil-filter (`FindFilter`), another task might be reading the serial number of the oil-filter (`ReadSerial`). Each task possesses a number of variables that are used to represent the result of the owning task, e. g. `BOOL OilFilterExistance` Figure 7 displays exemplarily the sequence and structure of commands to execute the described job.

```
/* Declaration */
BOOLEAN OilFilterExistance
-------------------------------------------
/* Checking Oilfilter */
LoadObject(CheckOilFilter)
Trigger()
result=getBoolean(CheckOilFilter.FindFilter.
Existance, OilFilterExistance)
if (result == OK)
         if (OilFilterExistance == TRUE)
                  /* complete!! */
         else
                  /* NOT complete
if (result == NOT_OK)
         /* Errorhandling */
```

Figure 7: Inspection operation sequence

.

```
<RobData>
 <PosAct X="1620.00" Y="120.00" Z="620.00"
      A="0.00" B="70.00" C="-10.00"/>
 <PosDesired X="1620.10" Y="120.10" Z="619.90"
      A="0.00" B="70.00" C="-10.00"/>
 <AxisAngleAct A1="132.00" A2="20.00" A3="65.00"
      A4="2.00" A5="-13.00" A6="-21.00"/>
 <AxisPosDesired A1="131.84" A2="20.20" A3="64.70"
      A4="2.00" A5="-13.20" A6="-21.02"/>"
 <ExternalAxisAngleAct E1="0.00" E2="15.30"
          E3="57.00" E4="0.20"
          E5="5.90" E6="0.00"/>
 <ExternalAxisAngleDesired E1="0.00" E2="15.00"
          E3="57.20" E4="0.23"
          E5="6.00" E6="0.00"/>
</RobData>
```

Figure 8: Position and orientation of TCP in XML format

For certain applications it is absolutely necessary that the vision system receives robot specific information, e. g. Cartesian position of the TCP, joint angles or position of external axis. This can be handled by establishing an cyclic data export from robot to vision system. The corresponding XML-based protocol description is shown in Figure 8. .

The advantages provided by our approach are extremely important in industrial applications which are dominated by heterogeneous environmental conditions such as employment of different robot and vision systems as well as installation of diverse communication media (Ethernet, fieldbus etc.):

- Consistent set of commands to employ vision systems within robot applications
- Adaptability and configurability of the data exchange due to the use of advanced protocol definition mechanisms (XML & schema)
- Extensible structure; protocol modifications can easily be introduced by rearranging underlying schema-files

XML schemas for robot control and vision control will be published in full at the end of the ARIKT project. To demonstrate the universality of the proposed protocol profile two demonstrators are build up whereby robot and vision system are exchanged crosswise. The weld seam inspection system from Vitronic is run with both a Kuka robot (KR6/1) and a Reis robot (RV6L). As well the engine compartment inspection from Isra is run with both robot systems.

5. Adaptation to New Inspection Tasks

To adapt the inspection system to a new task means to determine appropriate machine vision procedures/parameters and to find suitable sensor positions/trajectories. These activities shall be partially automated [9].

5.1. Determination of the Vision Procedure by a Knowledge Base

The determination of the vision procedure for a specified inspection task is supported by a ‘vision knowledge base’. Because it is difficult to implement too generalized knowledge, to each problem domain an own knowledge base is associated. For example a vision knowledge base dealing with weld seam inspection shall not be used in engine compartment inspection and vice versa.

In Figure 9 we see the data flow in general in determination of a vision procedure.

Exemplarily now the determination of vision procedures/parameters in engine compartment inspection is described:

The knowledge base consists of numerous inspection routines to be executed by the vision system.

Related to these inspection routines is information stored in the knowledge base regarding the input data modelled for each routine. This input data stream does not require specialized vision information, but rather a general description of the objects, the inspected scene and the camera and illumination hardware:

- Object data:
  - Dimensions and shape
  - Material
  - Surface properties and colour
  - Texture / Structure
- Scene data:
  - Position of the object in the area
  - Position of the camera with regard to the object
  - Number of objects
  - Inspection area(s)
  - Position of illumination with regard to the object
- Hardware data
  - Camera resolution
  - Camera distance
  - Image size
  - Focus length
  - Position of illumination with regard to the camera
  - Bright field / Dark field
  - Illumination intensity

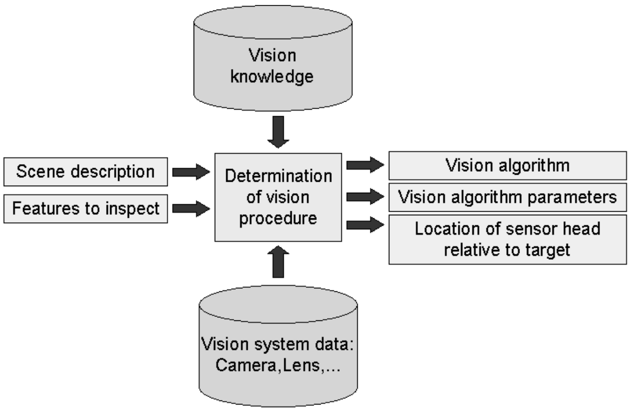
Vision knowledge
Scene description
Features to inspect
Determination of vision procedure
Vision algorithm
Vision algorithm parameters
Location of sensor head relative to target
Vision system data: Camera,Lens,...

Figure 9: Determination of a vision procedure in general

Each generation of a new inspection task starts with the assignment of this task to an inspection type. This is shown in the following with the example of the „ clamping piece “. According to the knowledge base the following data is needed for the inspection of the clamping piece:

- Dimensions and shape
- Surface properties and colour
- Material
- Position of the camera with regard to the object
- Image size

Based on this data input the inspection task is configured automatically with the position where the object is expected and the search for the actual object will start. The information about colour and material generates the parameter for the inspection algorithm (such as contrast between object and background). The information about correct dimensions and correct position finally allows the verification of correct shape and correct assembly. Feedback data of the inspection routine is either dimensional data or ok/nok information. For the latter the input parameter have to be coupled with tolerances.

During the whole process of generating an inspection plan, the user is supported in several instances by the knowledge base.

In the first step a sub-sample of objects to be inspected is selected out of the information that is archived regarding the whole engine compartment. Through this sub-sample of objects to be inspected the sub-sample of relevant data is automatically generated. Based on the specific information about the inspection object the knowledge base now offers all fitting inspection algorithms for this object. When choosing a rectangular metallic object, for example, the inspection „tube distance“ can be neglected, because for this inspection task two lengthy objects are mandatory.

After the inspection task is selected every other further step is executed automatically, since all relevant data can be achieved out of the knowledge base.

In order to allow manual interaction there is the additional option to modify the inspection plan and the automatically generated parameter via user interface.

5.2. Computation of Trajectories of the Sensor Head and Integrated X3D Model

Result of the determination of a vision procedure is amongst others the location of the sensor head relative to the part to be inspected. For example the weld seam inspection sensor head shall be away 60 mm from the seam. The line of sight to the seam shall be perpendicular.

By the location of the sensor head relative to the target and the given geometry of the part to be inspected the positions of the sensor head in Cartesian scene coordinate can be computed (see Figure 10). Result is either a set of single positions (engine compartment inspection) or a continuous path trajectory (weld seam inspection).

In the case of the single positions it is not yet determined how the robot moves the sensor head from position to position. A collision free path between the single positions has to be computed. Therefore the whole scene geometry (workcell, robot, part) and the robot kinematics is needed. In the case where a continuous path is given no path planning is needed anymore but the path has to be verified as collision free. Result is in both cases a collision free robot trajectory (or the conclusion that no collision free trajectory exists).

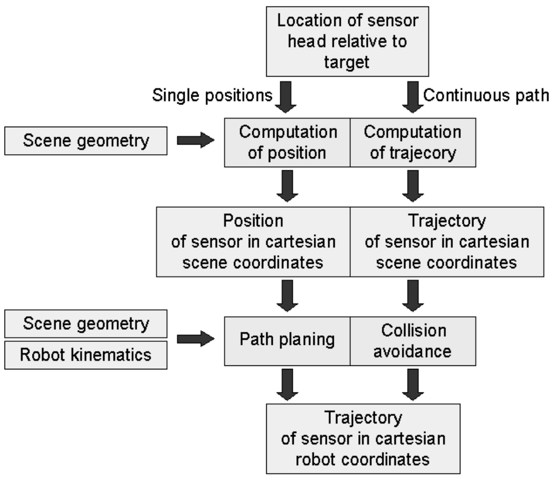
Location of sensor head relative to target
Single positions
Continuous path
Scene geometry
Computation of position
Computation of trajecory
Position of sensor in cartesian scene coordinates
Trajectory of sensor in cartesian scene coordinates
Scene geometry
Robot kinematics
Path planing
Collision avoidance
Trajectory of sensor in cartesian robot coordinates

Figure 10: Data flow with computing robot trajectories

As we have seen in the sections above, for automating the adaption to new inspection tasks the following data have to be provided: (computation of trajectories) geometry of workcell, product and robot; the kinematics of the robot; (determination of vision procedure) geometry of parts to inspect; surface, texture and special attributes of parts to inspect; attributes of vision system sensors.

Computation of trajectories and determination of vision procedures partially need the same data: geometry of parts. Surface and texture attributes of parts are naturally associated to part objects. In a optimal way all this data should be stored together in one model to avoid redundancies and to achieve consistence.

The use of X3D[3] [10] allows to integrate the needed data in one model. X3D is the 'XML variant' of VRML[4] [11]. VRML allows among others the description of:

- geometry of primitive volume bodies (sphere, cone, box etc.)
- geometry of surfaces (neighbouring polygons: *indexed face sets*)
- surface attributes like texture, color, brightness, transparency, reflexivity etc.

X3D has all modelling facilities which VRML offers, but because it is XML based, it is extensible. That means you can make additions to the X3D schema [12]. For example you can introduce attributes like 'weld seam porosity' to the geometric element 'line set'. Also complex data structures, for example to define the (direct) kinematics of a robot, can be described in XML and can be added to a X3D document. But, common viewers for X3D or VRML documents don't know how to handle these self-defined X3D/XML additions. They can be only handled by own applications.

In Figure 11 we see a clipping of the surface of an part with a weld seam. The surface is modelled as an X3D *indexed face set*. Each face is defined by three points. The weld seam is modelled as X3D *indexed line set*. A set of points is used to define a set of connected lines.

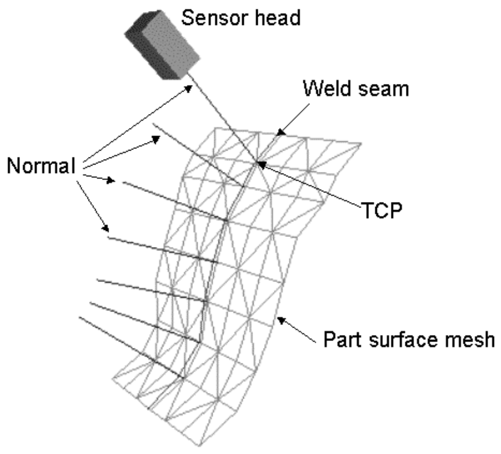

Figure 11: X3D surface model: indexed face set

To determine a sensor head trajectory the normal for each point on the indexed line set is computed. The TCP of the robot is set to a position so that he is on the weld seam, when the sensor head is in the correct distance (60 mm) and direction. The positions of the sensor head are defined by the position of the point on the weld seam and the normal of the point. The robot system will compute the position of the sensor head by itself with given normal and 'target point', when the TCP is set properly.

6. Summary

In this paper an architecture for robot based visual inspection is introduced. This architecture shall allow to create easy to adapt inspection systems.

A communication profile (Ethernet, TCP/IP, XML and application level commands) for interaction of a host instance, a robot and a vision system has been proposed.

[3] X3D = Extensible 3D

[4] VRML = Virtual Reality Modeling Language

Robots and vision systems, which follow these communication profile, can work together in ‘plug and work’ manner. Robot commands allow to define movements and trajectories. Also synchronisation between robot and other devices is handled. Vision commands allow to tell vision systems what tasks to perform, to start and stop analysis and to communicate status and results of an inspection analysis.

It is shown how a knowledge/data base can support the user in adapting an inspection system to new tasks. The knowledgebase helps to determine vision system procedures and parameters. However, the application field of a vision knowledgebase is bounded to one domain. Trajectories, to move a sensor head in a desired position, will be computed by a geometric model of the scene. To store geometric data and data for the vision system X3D is proposed as format. It allows to create *one* integrated model. Data used for computing robot trajectories as well as determining vision procedures and parameters have not to be copied in different data bases.

Acknowledgment

The ARIKT system is a result of the cooperation of the IPR and the companies AMATEC, ISRA/RMV, KUKA, REIS and VITRONIC. We want to thank the following people involved in the ARIKT project: H. Mueller (Amatec); G. Jansen, B. Doerffel (Isra/Rmv); H. Tradt, M. Eberl (Kuka), Dr. M. Dresselhaus, F. Som (Reis), Dr. B. Minge, J. Schlesinger (Vitronic), Th. Rosenbusch (FZK/PFT ‘project executing organization’), M. Salonia, B. Hein (IPR).

The ARIKT project is partly funded by the ‘Bundesministerium für Bildung und Forschung’ (project no. 02PD2465).

Contact

Dipl.-Inform. Michael Gauss,
Dipl.-Inform. Axel Buerkle,
Dr.-Ing. Thomas Laengle,
Prof. Dr.-Ing. Heinz Woern
Universitaet Karlsruhe (TH)
Institute for Process Control and Robotics (IPR)
Kaiserstrasse 12, 76128 Karlsruhe, Germany
{gauss | abuerkle | laengle | woern}@ira.uka.de
http://wwwipr.ira.uka.de/

Dr. Johannes Stelter
AMATEC ROBOTICS GmbH
Landsberger Straße 63a, 82110 Germering, Germany
Johannes.Stelter@amatec.de
http://www.amatec.de/

Dipl.-Ing. Stefan Ruhmkorf
ISRA VISION SYSTEMS AG
Schoemperlenstr. 12a, 76185 Karslruhe, Germany
sruhmkorf@isravision.com
http://www.isravision.com

Richard Middelmann B.Eng. (Hons)
VITRONIC Dr.-Ing. Stein Bildverarbeitungssysteme GmbH
Hasengartenstraße 14, 65189 Wiesbaden, Germany
rpm@vitronic.com
www.vitronic.com

7. References

[1] World Wide Web Consortium (W3C): URL: http://www.w3.org/

[2] H. E. Rusty: XML in a Nutshell. O'Reilly, Beijing, Köln, 2001.

[3] EN30042:1994 Arcwelded joints in aluminium and ist weldable alloys – Guidance on qualitz levels for imperfections (ISO 10042 : 1992)

[4] prEN ISO 5817 : 2000 Welding – Fusion welded joints in steel, nickel, titanium and their alloys (beam welding excluded) – Quality levels for imperfections (ISO/DIS 5817 : 2000)

[5] ISO/IEC 7498-1: Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model, 1994.

[6] F. J. Furrer: Ethernet-TCP/IP für die Industrieautomation. Hüthig Verlag, Heidelberg, 2000.

[7] XML schema definition and development: URL: http://www.w3.org/XML/Schema#dev

[8] DIN 66312, Industrieroboter - Programmiersprache - Industrial Robot Language (IRL), Beuth, 1996-09.

[9] H.Wörn, T. Längle, M. Gauß: Adaptive Robot Based Visual Inspection – ARIKT. KI - Künstliche Intelligenz, ISSN 0933-1875, 2/2003, in press.

[10] Extensible 3D (X3D™) Graphics : URL: http://www.web3d.org/x3d.html

[11] The Virtual Reality Modeling Language. International Standard, ISO/IEC 14772-1:1997, URL: http://www.web3d.org/technicalinfo/specifications/vrml97/index.htm

[12] Extensible 3D (X3D) Schema (normativ), URL: http://www.web3d.org/technicalinfo/specifications/ISO_IEC_19776/Part01/x3d-3.0.xsd