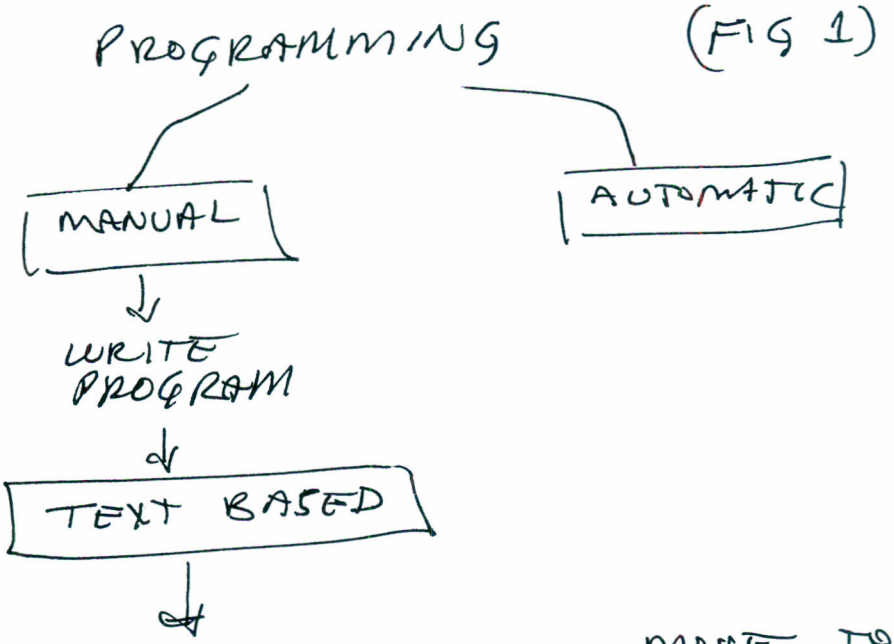
Biggs + Mac Donald
Robot Programming



(1) CONTROLLER SPECIFIC LANGUAGES
P93

a) MOVE TO P1
OPEN GRIPPER
⋮
(LIKE ACL OF ESHED)

b) WOULD SELECT COMMANDS — KUKA OR ABB INTERFACES

(2) GENERIC LANGUAGES + ROBOTIC EXTENTIONS (JAVA, C++, etc)
Page 3-4

(3) BEHAVIOUR-BASED
Page 4
— HOW SHOULD THE ROBOT REACT
(i.e. FOLLOW A LINE OR WALL)

Review of Robot Software Klafter pgs 523 -557

Last week we reviewed the ACL software for the Eshed Robot. This "Advanced Control Language" is similar to Val as described by Klafter page 539-546. It is a programming language with extensions for robotics.

The basic robotic programs allow:

Execution of Robot commands

Input and Output of data

Creation of programs

Simple file management.

ACL is also a multitasking system and can respond to inputs (Trigger and Wait).

We expect the software to do at least the following:

Overall control of the robotic system and Operator Interface

Robot control – Position control (coordinate transformations) with various motion types, speed control, end effector control

Normal computer operations – programming, User I/O, program control (RUN, etc.), debugging

Sensor I/O (Analog) and Digital I/O and recognition of interrupts

Safety features – protect motors from overheating and mechanical parts from being driven beyond their limits

More advanced:

Acceleration, force and torque control

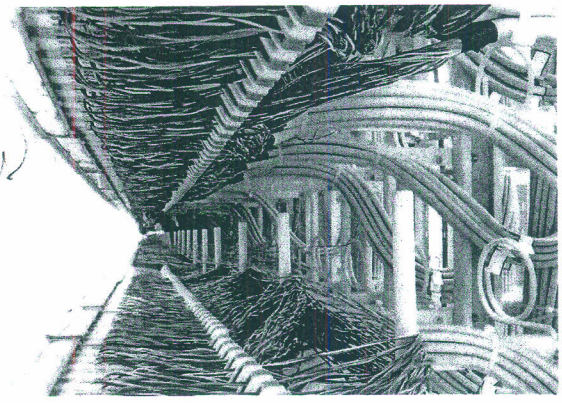
Vision system with coordination with robot coordinates

Digital signal processing to filter inputs and reduce electrical noise from sensors

Obstacle avoidance (mobile robots or to protect robot)

Safety inputs from guards such as light or acoustic fences to protect personnel.

Robot OS: A New Day for Robot Design



Lee Garber

The Robot Operating System promises to make designing robotic software easier and less expensive.

Robotics has undergone many advances in recent years, with robots gaining new skills and the ability to undertake new tasks. For example, they are helping with surgery, and are even flying and executing complex instructions as drones.

However, robot use has not become as widespread as many people have predicted. Even the development of industrial robots—formerly a bright spot—has stagnated.

One reason is that robots have not had a standard platform on which developers could build. Instead, they typically have had to start from scratch with each project. This has frequently made the development process time consuming and expensive.

In addition, the lack of a standard platform has frequently kept different types of robotic software from being able to communicate with one another, which has limited developers' ability to use multiple applications within a single robot. This has also kept engineers from being able to reuse robotics software and otherwise collaborate.

This may all be about to change, though, as an increasing number of companies and researchers are working with the open source Robot Operating System (ROS) platform—first released in 2010—to provide software for their robots. And as this has occurred, the pace of robotics innovation has increased.

Despite its name, ROS isn't just an operating system, although it includes some OS functionality. It is also a framework that provides software modules for performing typical robot activities such as object recognition. The technology also helps control and coordinate the modules.

"ROS is a software framework that simplifies the task of writing complex robot software. With ROS, it is possible to quickly grab state-of-the-art software for many aspects of the system, which frees up R&D time for the novel aspects of a particular project," said Morgan Quigley, who developed the first version of ROS and is now chief architect of the Open Source Robotics Foundation (OSRF).

Proponents hope ROS will enable developers to flexibly and inexpensively create more capable robots.

However, the technology faces challenges before it can become the dominant robotics platform.

A BRIEF HISTORY

When Quigley arrived at Stanford University to study machine learning for his doctoral program, he joined associate professor Andrew Ng's Stanford Artificial Intelligence Laboratory (SAIL). Students there were working on the Stanford Artificial Intelligence Robot.

STAIR reflected Ng's desire to design a general-purpose robot that could perform different types of tasks, rather than the one narrow job that office and industrial robots typically do.

To accomplish this, Quigley realized that SAIL required a software framework that could integrate programs from various students in a robot that would keep working even if one of the applications failed. In 2006, he designed a distributed, peer-to-peer system called Switchyard to meet this need.

The next year, in a collaboration that led to ROS's creation, Quigley began working with Willow Garage, a research lab that developed personal robotics hardware and open source software.

Willow Garage, which has since closed, released ROS version 1.0, called Box Turtle, in 2010. There

Published by the IEEE Computer Society