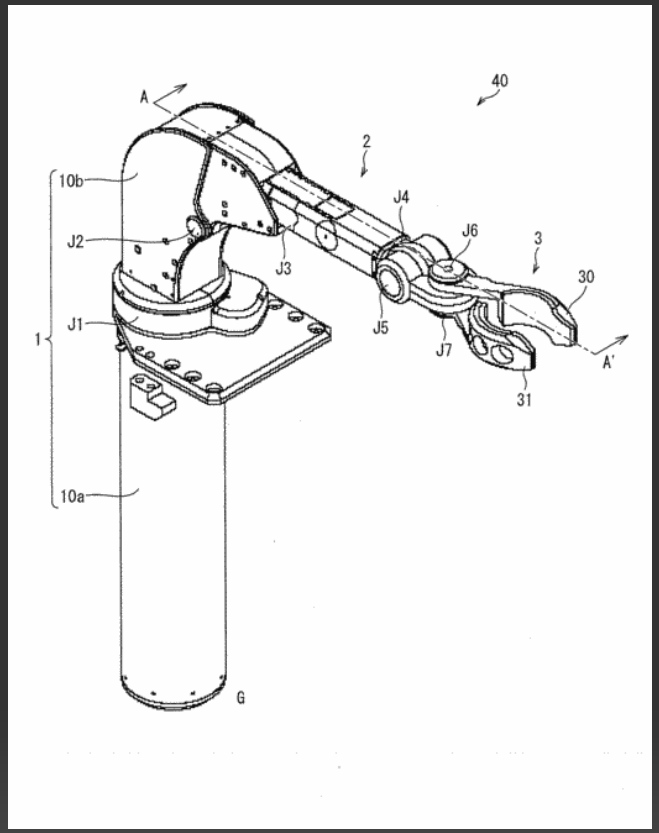
Automated Testing Platform for Quality Control WP7

Jeremy Bridon
Summer 2010
aXelerate Solutions & Microsoft

Quality Control

- New WP7 devices need QC checks
- Hundreds of test procedures
  - Battery life
  - Accelerometer accuracy
  - Touch screen resolution
  - Etc...
- Requires time, money, and people
  - Solution: robots!
- Project: *proof-of-concept automated QC testing platform*
- Note: Though robots are still inferior to humans, automation **saves time, money, and HR**



Platform & Project Description

- Goals: Automatically test as many QC cases / procedures as possible
  - Hardware Platform: existing **Adept Cobra** s350 SCARA 4-Axis Robot
  - Software Platform: **Windows** & **V+**
  - Languages: **C# (.net 4) & V+**
  - Interface: **USB / TTY Serial**
- Subgoal: prove it can be fast and can test complex multi-device cases (i.e. calling)
- Image processing needed, closed-loop interface required, error recovery, ...
- Test a "**true**" vanilla device; no specialty software needed

Robot Controller

- Controlled by V+ scripting / programming language
  - Physical robot controller does movement interpolation
  - Language is similar to a state-machine
- Wrote a C# front-end that could execute movements (poses)
  - Test procedures evolved into a special test description language that also described robot poses
- Error checking and recovery
  - Physical protection
  - Lowest level software boundary
  - High level software boundary

General Control Loop

- **Close-loop system**; i.e. internal feedback so the robot isn't moving blindly

1. For each procedure
    1. Reset the robot position and device state
    2. Execute the procedure
        1. For each step
            1. Save device state (screenshot)
            2. Execute movement: press, slide, or complex path (with arguments like speed
            3. Verify resulting state (screenshot comparison)
    3. Save the result of the procedure

Screen Comparison

- How do we check screen states?
  - *Can't do pixel-by-pixel tests, or average tests: inaccurate*
  - *Solution: break-down the scene based on known UI elements (i.e. buttons, boxes, icons, etc.)*

AForge.NET
FRAMEWORK

tesseract-ocr

An OCR Engine that was developed at HP Labs between 1985 and 1995... and now at Google.

Computer Visions & Image Processing

- Libraries:
  - OpenCV, AForge.NET, Tesseract-OCR
- Take the image and apply the projection matrix so that it's more "flat"
  - Done based on three color dots on the device screen
- Blur and average the image to remove "complex" details
  - Not a good approach, but necessary given camera resolution
- Do blob detection & edge detection
  - Merge the two together to build screen geometry (i.e. buttons, text, icons, etc..)
- Apply the detection rules (icon-specific detection)
  - Match geometry based on thresholds

Results & Extensions

- Tests work with high success rate
  - False-positive of 5%-10%!
  - Multi device works!
- Known Issues:
  - **Slow**; robot controller shouldn't be interfaced using the V+ scripting / command language
    - Solution: use direct USB control
  - **Error Recovery**; some test cases make error recovery hard
    - Solution: add special firmware, which defeats the
  - **Setup is Costly**; setting up a device is costly and annoying to do so
    - Solution: minimize overhead of the system initialization