NON-INTRUSIVE TEST EXECUTION OF GUI APPLICATIONS USING OFF-THE-SHELF HARDWARE

Presented by Raphael Zeiger
PROBLEMS DURING TEST EXECUTION?

If not, be happy and don’t change your toolchain
Problems might be (but not restricted to):

- **Stability**
  - Interaction between test execution tool and application under tests has side effects, i.e. erroneous behavior that does not appear in production
  - GUI elements are not detected reliably

- **Maintainability**
  - How easily breaks your test automation after operating system updates, e.g. from Windows 7 to Windows 10?
Problems might be (but not restricted to):

• Security Restrictions
  • Application process is running using unknown credentials
  • Protected screen context

• Company Policies
  • Fixed set of allowed test execution tools
TRIGGER USER INPUT

Using a microcontroller to pretend a USB keyboard
Arduino Leonardo

- Can act as a mouse or keyboard
- Hardware USB
- Platform independent
  - Windows, Linux, OS
- Also available during BIOS
- Drawbacks:
  - No absolute position
  - Variable translation speed
Arduino Leonardo

```c
switch (platform) {
  case OSX:
    Keyboard.press(KEY_LEFT_GUI);
    // Shift-Q logs out:
    Keyboard.press(KEY_LEFT_SHIFT);
    Keyboard.press('Q');
    delay(100);
    Keyboard.releaseAll();
    // enter:
    Keyboard.write(KEY_RETURN);
    break;
  case WINDOWS:
    // CTRL-ALT-DEL:
    Keyboard.press(KEY_LEFT_CTRL);
    Keyboard.press(KEY_LEFT_ALT);
    Keyboard.press(KEY_DELETE);
    delay(100);
    Keyboard.releaseAll();
    // ALT-I:
    delay(2000);
    Keyboard.press(KEY_LEFT_ALT);
    Keyboard.press('I');
    Keyboard.releaseAll();
    break;
```
Arduino Leonardo

- Combine with Ethernet shield
- Firmware (<=300 loc)
- Listens for UDP packets and generates keyboard events
- Can be controlled by an arbitrary scripting language like perl or python
RETRIEVING VISUAL FEEDBACK

Using an FPGA board to extract pixel data from HDMI stream
PYNQ

- Can extract raw pixel data from HDMI stream
- Image can be analyzed using python
PYNQ

**Step 1: Load the overlay**

```python
from pynq.overlays.base import BaseOverlay
from pynq.lib.video import *

base = BaseOverlay("base.bit")
hdmi_in = base.video.hdmi_in
hdmi_out = base.video.hdmi_out
```

**Step 2: Initialize HDMI I/O**

```python
hdmi_in.configure(PIXEL_RGB)
hdmi_out.configure(hdmi_in.mode, PIXEL_RGB)

hdmi_in.start()
hdmi_out.start()
```

**Step 3: Show input frame using IPython Image**


```python
import PIL.Image
frame = hdmi_in.readframe()
img = PIL.Image.fromarray(frame)
img.save("/home/xilinx/jupyter_notebooks/base/video/data/face_detect.jpg")
```

User Conference on Advanced Automated Testing
USING OFF-THE-SHELF HARDWARE

There is a UVC USB specification
UVC Frame Grabber

- Can extract almost (YUY2) raw pixel data from HDMI stream
- No proprietary drivers/api
UVC Frame Grabber

• Template Matching

• OCR

![Image of bios setup utility with boot and Kingston Data Traveler]

User Conference on
Advanced Automated Testing
```python
import time
import numpy as np
import cv2

cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)

if cap.isOpened() == 0:
    exit(-1)

cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1920)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 1080)

template_boots = cv2.imread("template_boots_selected.png", cv2.IMREAD_GRAYSCALE)
w_boot, h_boot = template_boots.shape[:2]

while True:
    result = []
    retval, frame = cap.read()
    gray_img = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray_img = gray_img[0:1200, 0:1920]
    res = cv2.matchTemplate(gray_img, template_boots, cv2.TM_CCORR_NORMED)
    loc = np.where(res >= 0.3)
    for pt in zip(*loc[::-1]):
        cv2.rectangle(frame, pt[0] + w_boot, pt[1] + h_boot), (0, 255, 0), 3)
    if len(loc[0]) != 0:
        result.append("Boots")
    frame2 = cv2.resize(frame, (int(1920/2), int(1080/2)), interpolation=cv2.INTER_AREA)
    cv2.imshow("frame", frame2)
    print(result)
    time.sleep(0.1)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```
WINDOWS 10 INSTALLATION

As a proof of concept
User Interactions

- Plug in USB install medium
- Supply power
- Press power button
- Check power LED
- Use Keyboard and Mouse
- Analyze video signal
  - Template matching, color matching, ocr
User Interactions

• Plug in USB install medium
• Supply power
• Press power button
• Check power LED
• Use Keyboard and Mouse
• Analyze video signal
  • Template matching, color matching, ocr

© All rights reserved
Enclosure
Overview
NEXT STEPS

More Hardware ...
Next Steps

• Understand USB HID Communication
  • PhyWhisperer-USB
  • GreatFET
• Simulate USB Stick Removal
  • Programmable USB Hub
• Analyze GUI structure
  • Pix2code
• OCR Training
  • Train used fonts