Final Project

Motion Tracking Camera System

Haoran Wu, Haoran Zhou, Anderson Cone
Motivation - Core Problem

- Professors teaching remotely struggle to show students what they’re writing
- Must adjust camera regularly to show appropriate blackboard
Motivation - Additional Uses

Entertainment

- Tracking camera, selfies
- Personal filming
- Game (Dodgeball)

Security

- Motion sensitive, enhanced security monitoring ability (Home, Childminding, etc.)
- Switching to manually control, enable user to check by themselves
Introduction - Motion Tracking Camera

2 modes:

(1) **Motion tracking mode**: Camera + OpenCV track the subject; Raspberry Pi controls stepper motors move the camera autonomously to focus on the subject.

(2) **Interactive mode**: Arduino and hand-held buttons send signals to Raspberry Pi to control the stepper motors to move camera manually.
# Component List

<table>
<thead>
<tr>
<th>Item #</th>
<th>Name</th>
<th>Quantity</th>
<th>Price (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cana Raspberry Pi 4 4GB Starter Kit</td>
<td>1</td>
<td>99.99</td>
</tr>
<tr>
<td>2</td>
<td>Arduino UNO board</td>
<td>1</td>
<td>23.00</td>
</tr>
<tr>
<td>3</td>
<td>Adafruit Stepper Motor</td>
<td>2</td>
<td>28.00</td>
</tr>
<tr>
<td>4</td>
<td>Adafruit Motor Hat for Raspberry Pi</td>
<td>1</td>
<td>22.50</td>
</tr>
<tr>
<td>5</td>
<td>USB Webcam 1080p</td>
<td>2</td>
<td>50.00</td>
</tr>
<tr>
<td>6</td>
<td>Voltage converter</td>
<td>1</td>
<td>6.50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item #</th>
<th>Name</th>
<th>Quantity</th>
<th>Price (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Buttons</td>
<td>4</td>
<td>1.00</td>
</tr>
<tr>
<td>8</td>
<td>Wood board</td>
<td>2</td>
<td>4.00</td>
</tr>
<tr>
<td>9</td>
<td>Jump wires</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>10</td>
<td>Portable charger</td>
<td>1</td>
<td>18.77</td>
</tr>
<tr>
<td>11</td>
<td>Standoffs for Pi HATS</td>
<td>1</td>
<td>0.75</td>
</tr>
</tbody>
</table>

**Total** 242.53
Main Component - Stepper Motor HAT

- Since the Raspberry Pi does not have a lot of PWM pins, we use a fully-dedicated PWM driver chip on board to both control motor direction and speed.
- This chip handles all the motor and speed controls over I2C. Only two pins (SDA & SCL) are required to drive the multiple motors, and since it's I2C you can also connect any other I2C devices or HAT to the same pins.
- External 12V to power the HAT
Main Component - Stepper Motor and Camera

Stepper motor - NEMA-17 size - 200 steps/rev, 12V 350mA

ELP USB with Camera 2.1mm Lens
Arduino and Pi are using serial communication via USB cable
Key Features: Hardware Anatomy

Raspberry Pi with Adafruit motor HAT

Base structure, web cameras, Adafruit stepper motors, Raspberry Pi and USB dock
Communicate between Arduino UNO and Raspberry Pi 4B

Left, right, up and down buttons for interactive mode control
Test - Motion Detection

- **Find contours** by applying greyscale, blur, threshold, and dilate.

- **Compare** each frame of the captured video flow with the *initial frame*.

- All frames are captured from the base camera.
class VideoUtils(object):
    
    Helper functions for video utilities.
    
    @staticmethod
    def live_video(camera_port=0):
        
        Opens a window with live video.
        :param camera:
        :return:
        
        video_capture = cv2.VideoCapture(camera_port)
        video_capture.set(cv2.CAP_PROP_FRAME_WIDTH,800)
        video_capture.set(cv2.CAP_PROP_FRAME_HEIGHT,800)
        video_capture.set(cv2.CAP_PROP_FPS,60)

        while True:
            # Capture frame-by-frame
            ret, frame = video_capture.read()

            # Display the resulting frame
            cv2.imshow('Video', frame)

            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

        # When everything is done, release the capture
        video_capture.release()
        cv2.destroyAllWindows()
Demo 1: Motion Detection
Prototype - Interactive Mode

Interactive mode: Arduino and buttons send signals to Raspberry Pi to control the stepper motors to move the upper camera manually.
Demo 2: Interactive Mode Controlled with Arduino
**Prototype - Motion Tracking Mode**

**Motion tracking mode**: using **Base camera** to track the motion, and the stepper motors are controlled to move the **Upper camera** autonomously to focus on the subject.
Demo 3 - Motion Tracking
Difficulties

- Control of Adafruit Stepper motor
- Lack of material to build structure
- Motion tracking algorithm affected by light conditions
Future Work

- Replace wood/cardboard structure with 3D printed housing
- Replace Upper camera with smartphone
- Use RF modules for wireless connection between Arduino and Raspberry Pi
- Improve motion tracking algorithm to enhance reliability in low light conditions
- Add mode switch button to handheld user controls
ME-GY 6933: Advanced Mechatronics (Spring 2020)

Final Project

Motion Tracking Camera System

Haoran Wu, Haoran Zhou, Anderson Cone
Appendix
Key Features: Code Anatomy

- Basic video display

```python
class VideoUtils(object):
    
    Helper functions for video utilities.

    @staticmethod
    def live_video(camera_port=0):
        
        Opens a window with live video.

        :param camera:
        :return:

        video_capture = cv2.VideoCapture(camera_port)
    video_capture.set(cv2.CAP_PROP_FRAME_WIDTH, 800)
    video_capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 600)
    video_capture.set(cv2.CAP_PROP_FPS, 60)
```

- Motion find

```python
def find_motion(callback, camera_port=3, show_video=True):  
    
    camera = cv2.VideoCapture(camera_port)
    
    time.sleep(0.1)

    # initialize the first frame in the video stream
    firstFrame = None
    tempFrame = None
    count = 0

def get_best_contour(imgmask, threshold):
    
    contours, hierarchy = cv2.findContours(imgmask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    
    best_area = threshold
    best_cnt = None
    
    for cnt in contours:
        area = cv2.contourArea(cnt)
        if area > best_area:
            best_area = area
            best_cnt = cnt
    
    return best_cnt
```
Key Features: Code Anatomy

- **Base control (Motion tracking mode and Interactive mode)**

```python
class Base(object):
    """
    Class used for Base control.
    """
    def __init__(self, friendly_mode=True):
        self.friendly_mode = friendly_mode

    # create a default object, no changes to:
    self.mh = Adafruit_MotorHAT()
    self.mh.setPower(1)
    self.mh.setPulseWidth(50)
    self.mh.setPWMFreq(50)

    # Stepper motor 1
    self.sm_x = self.mh.getStepper(400, 1)
    self.sm_x.setSpeed(10)
    self.current_x_steps = 0

    # Stepper motor 2
    self.sm_y = self.mh.getStepper(400, 1)
    self.sm_y.setSpeed(5)
    self.current_y_steps = 0

    # motion detection
    def motion_detection(self, show_video=False):
        """
        Uses the camera to move the Base. OpenCV.
        """
        VideoUtils.find_motion(self.__move_axis,
```

- **Adafruit stepper motor control**

```python
def move_forward(motor, steps):
    """
    Moves the stepper motor forward the specified
    :param motor:
    :param steps:
    :return:
    """
    motor.step(steps, Adafruit_MotorHAT.FORWARD, Adafruit_MotorHAT.HALFSTEP)

def move_backward(motor, steps):
    """
    Moves the stepper motor backward the specified
    :param motor:
    :param steps:
    :return:
    """
    motor.step(steps, Adafruit_MotorHAT.BACKWARD, Adafruit_MotorHAT.HALFSTEP)

def turn_off_motors(self):
    """
    Recommended for auto-disabling motors on shut
    :return:
    """
    self.mh.getMotor(1).run(Adafruit_MotorHAT.RELEASE)
    self.mh.getMotor(2).run(Adafruit_MotorHAT.RELEASE)
    self.mh.getMotor(3).run(Adafruit_MotorHAT.RELEASE)
    self.mh.getMotor(4).run(Adafruit_MotorHAT.RELEASE)
```