Lesson-3&4

Atlas 200dk

NPU Postprocessing and hardware periphery

01 Working Flow of Sample hand-pose RC Car

Sample hand-pose RC Car

1. Initialize all resources;
2. Capture Image via Camera;
3. Pre-processing (e.g. color space conversion, data-type conversion);
4. Neural network inference;
5. Post-processing;
6. Send command through peripheral (UART,I2C, and SPI)
7. Jump to step 2.

Same with convert black-and - white images

Project Description

- data
  - Configure the IP address of present servicer
- Inc
  - Including all the head files
- Model
  - The .om neural network model file
- Out
  - Build file after compiling
- src
  - **Camera.cpp** – Manage the camera
  - **handpose.cpp** – Operations of the entire neural network
  - **handpose_decode.cpp** - postprocess
  - **Main.cpp**
  - **I2c.cpp and uart.cpp** – Manage the hardware peripheral;
  - **Utils.cpp** - handle file operations
- CMakeLists.txt

https://github.com/Atlas200dk/sample-handposeRCcar/tree/c73/sample-handposeRC/Atlas200DK/sample-handposeRC

02

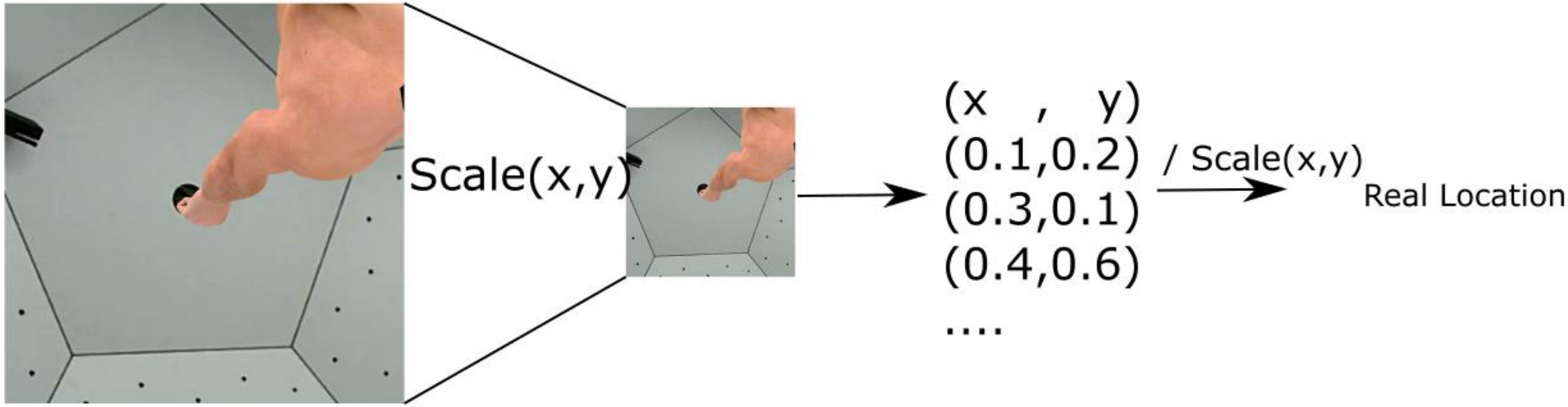
Hand pose Estimation
Post-processing

Post-processing

hand_pose.cpp#L190: Result HandPose::Postprocess(cv::Mat& origImage, aclmdlDataset* modelOutput)

Detection-based vs. Regression-based

- Output: 21 x (x,y) – location of key-points of hand (32-bits float point); (hand_pose.cpp#L195)
- Re-construct the points location (Scale)



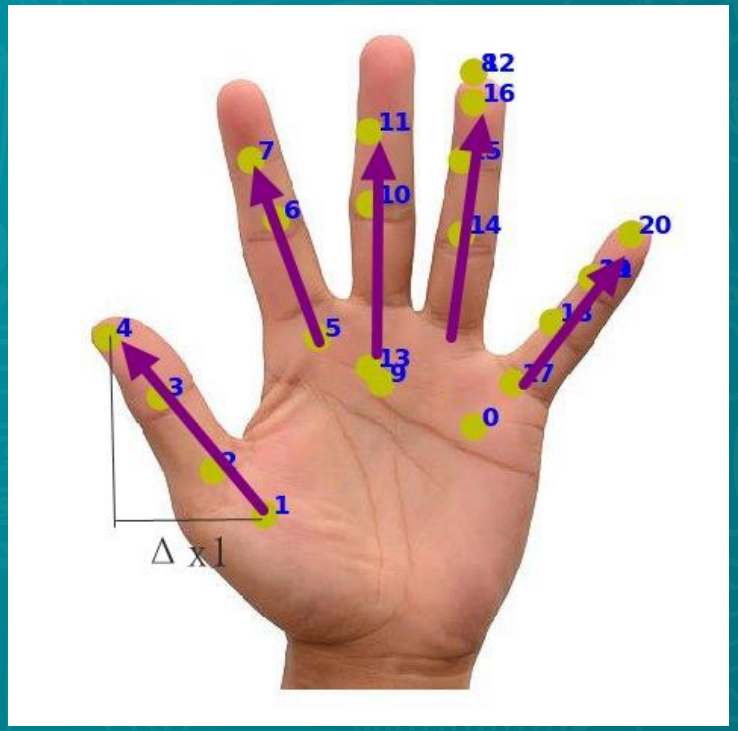
Post-processing

```
unsigned char HandposeDecode::get_rc_command(){
    int angle = get_fingers_angle_scale();
    int thumb_bool = thumb_status();
    if(!validate()){
        return cmd_stop;
    }
    else if((left_thresh < angle < right_thresh) && thumb_bool){
        return cmd_backward;
    }
    else if(angle > right_thresh){
        return cmd_right;
    }
    else if(angle < left_thresh){
        return cmd_left;
    }
    else {
        return cmd_forward;
    }
```

```
int HandposeDecode::get_fingers_angle_scale(){
    int finger_top_total = x_arr[8] + x_arr[12] + x_arr[16] + x_arr[20];
    int finger_bottom_total = x_arr[5] + x_arr[9] + x_arr[13] + x_arr[17];
    int angle_scale = finger_top_total - finger_bottom_total;
    return angle_scale;
}
```

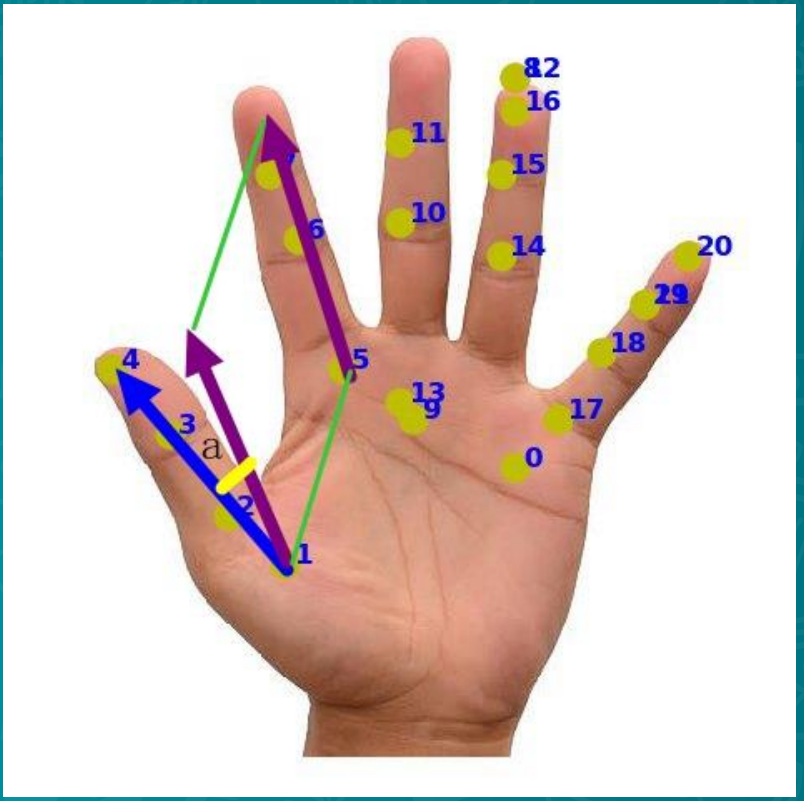
$\Delta x = \Delta x1 + \Delta x2 + \Delta x3 + \Delta x4$

Average filter



Post-processing

```
int HandposeDecode::thumb_status(){
    // get vector of thumb
    int vec_thumb_x = x_arr[4] - x_arr[3];
    int vec_thumb_y = y_arr[4] - y_arr[3];
    // get vector of index finger
    int vec_index_x = x_arr[8] - x_arr[7];
    int vec_index_y = y_arr[8] - y_arr[7];
    // length of thumb and index finger vector
    float len_thumb = sqrt(pow(vec_thumb_x,2)+pow(vec_thumb_y,2));
    float len_index = sqrt(pow(vec_index_x,2)+pow(vec_index_y,2));
    // thumb inward (direction = -1) or outward (direction = 1), using cross product of vectors
    int direction = (vec_thumb_x*vec_index_y-vec_index_x*vec_thumb_y)>0 ? -1:1;
    // get angle between thumb and index figure vector
    float thumb_index_angle
=direction*acos((vec_thumb_x*vec_index_x+vec_thumb_y*vec_index_y)/len_thumb/len_index)*18
0.0/3.1415926;
    if(thumb_index_angle < thumb_threshold){
        return 1;
    }
```



03 UART 1 on Atlas 200dk

Universal Asynchronous Receiver/Transmitter

```
class uart {
public:
    uart(void) ;
    ~uart(void) ;

    int uart_open(void);
    int uart_close(void);
    int uart_send(char *buffer, int size);
    int uart_read(char *buffer, int size);
    int uart_set_option(int nSpeed, int nBits, char
nEvent, int nStop);

private:
    int fd;
};
```

The program could only use Atlas 200dk *UART1* because of UART0 has been occupied by the Linux system.

1. Open UART1;
2. Set UART1 Baud rate 9600, 8-bits and 1 stop bit;
3. Send commends;
4. Receive commends;

Atlas 200dk

THANK YOU
FOR WATCHING