ERVA – Hackathon Approach Pseudocode

Chris Cadonic and Cassandra Aldaba

November 30, 2018

Algorithm 1 Frame pre-processing

```
procedure PRE-PROCESS($frame$)
    Resize
    Filter noise
    $mask$ = background_subtract()
    return $frame$, $mask$
```

Algorithm 2 Face detect

```
procedure FACE_DETECT($frame$)
    Convert to grayscale
    Build Cascade Classifier
    Detect faces with classifier
    for face in $detected\_faces$ do
        Draw ROI rect on raw frame
        Mask processing frame
```

Algorithm 3 Simple Motion Detection

```
procedure M(o)tionDetect($video$)
    while frame $cur$ left in $video$ do
        $diff = cur - bg$
        $threshold_img$ = Threshold $diff$ image
        Dilate($threshold_img$)
        $contours$ = findContours($threshold_img$)
        for $contour$ in $contours$ do
            Determine if good contour to track (parameter specification)
            Add bounding rectangle
```

Algorithm 4 Hierarchical LK

```
procedure LK($video$)
    $prev$ = video.initialFrame()
    Convert $prev$ to grayscale
    Preprocess to mask initial frame
    $features$ = goodFeaturesToTrack()                ▷ SIFT or ShiTomasi
    while frame $cur$ left in $video$ do
        Convert $cur$ to grayscale
        run calcOptimalFlowPyrLK($prev$, $cur$, $features$)
        Select feature points $old\_pnts$ and $new\_pnts$ from flow where $st == 1$
        Draw flow tracks onto images
        $prev = cur$
        $features = new\_pnts$.reshape()
```

Algorithm 5 Polynomial Expansion Flow

```
procedure FARNEBACK($video$)
    $prev$ = video.initialFrame()
    Convert $prev$ to grayscale
    $hsv = [0..., 0..., 0..., [0, 255], ...]]$
    while frame $cur$ left in $video$ do
        Convert $cur$ to grayscale
        $flow$ = calcOpticalFlowFarneback($prev$, $cur$)
        $magnitude, theta$ = cv2.cartToPolar() using output flows
        update $hsv$ using angle and cv2.normalize()
        Convert $hsv$ to HSV color space
        $prev = cur$
```