

Rysowanie w aplikacji, śledząc ruch markera przy pomocy systemu wizyjnego

Grupa projektowa: Wojciech Niedbała, Albert Millert, Michał Mirończuk, Patryk Romaniak

Część I

Przyrost

- Widok, umożliwiający przemieszczanie się po aplikacji,
 - Odczytywanie klatek z kamery,
 - Rozpoznawanie markera.
-
- Śledzenie ruchu markera,
 - Możliwość podglądu w okienku na żywo ścieżki, po której prowadzony był znacznik.

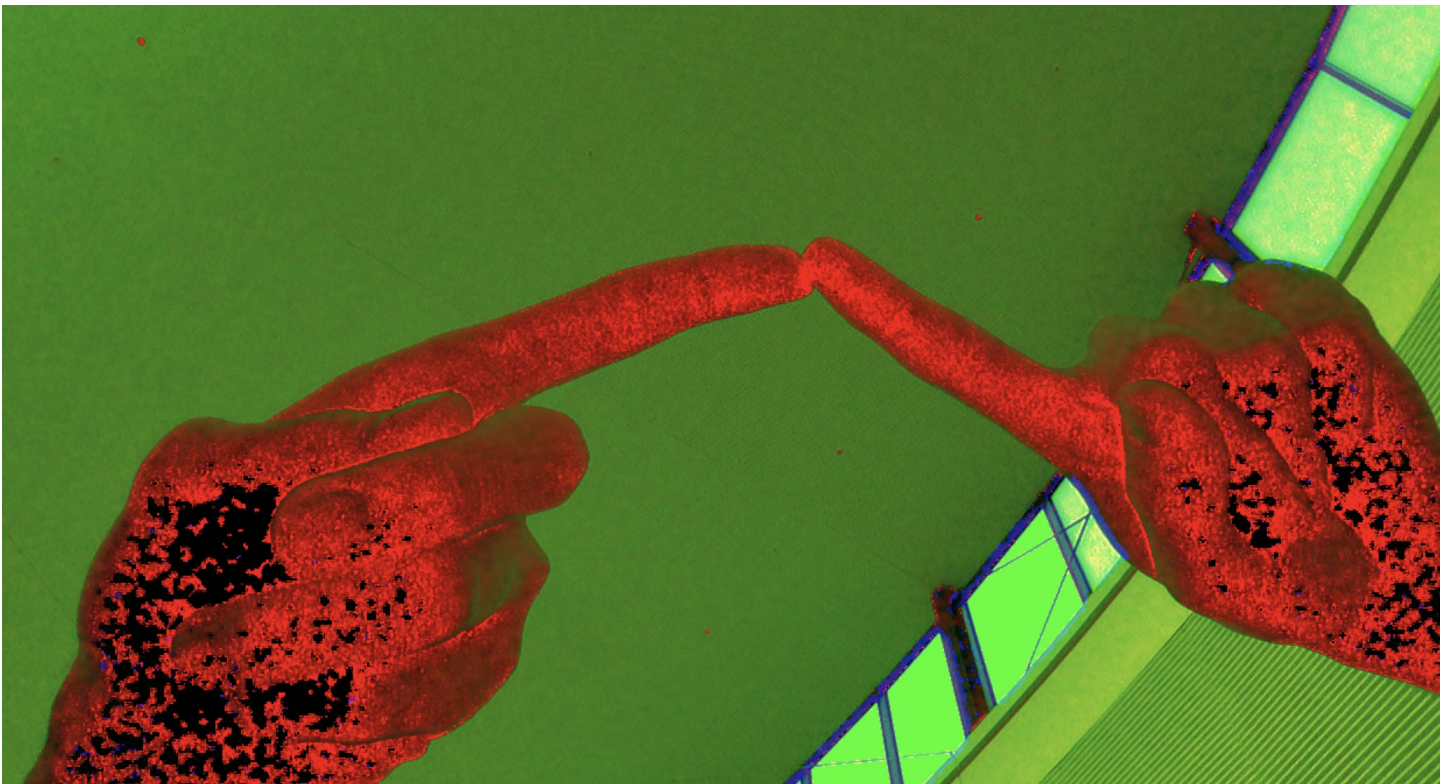


Wykrywanie ręki

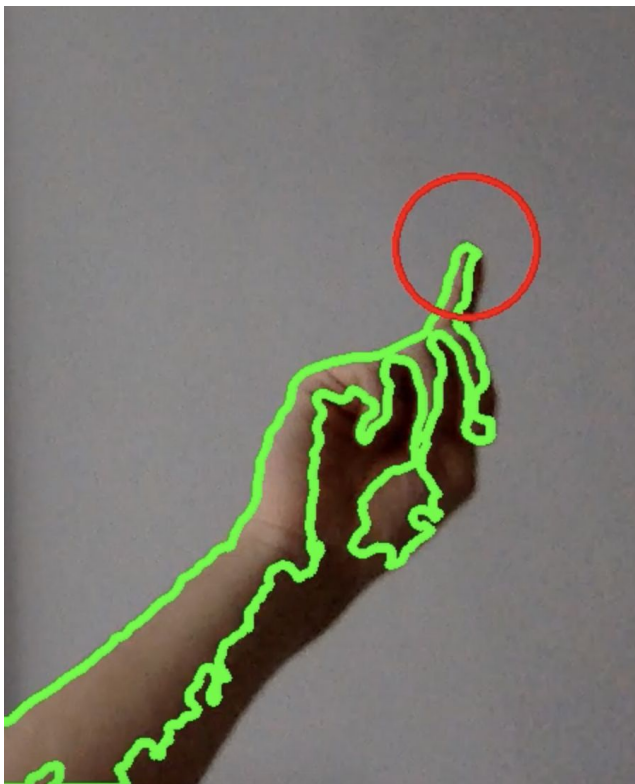


```
func skinColorUpper(hue float64) gocv.Scalar {  
    return gocv.NewScalar(hue, 0.8*255, 0.6*255, 200)  
}  
  
func skinColorLower(hue float64) gocv.Scalar {  
    return gocv.NewScalar(hue, 0.1*255, 0.05*255, 200)  
}  
  
func MakeHandMask(img gocv.Mat) gocv.Mat {  
    imgHLS := gocv.NewMat()  
    rangeMask := gocv.NewMat()  
    gocv.CvtColor(img, &imgHLS, gocv.ColorBGRToHLS)  
  
    gocv.InRangeWithScalar(imgHLS, skinColorLower(0), skinColorUpper(15), &rangeMask)  
    gocv.Blur(rangeMask, &rangeMask, image.Point{10, 10})  
    gocv.Threshold(rangeMask, &rangeMask, 220, 255, gocv.ThresholdBinary)  
  
    return rangeMask  
}
```

Wykrywanie ręki



Wykrywanie palca



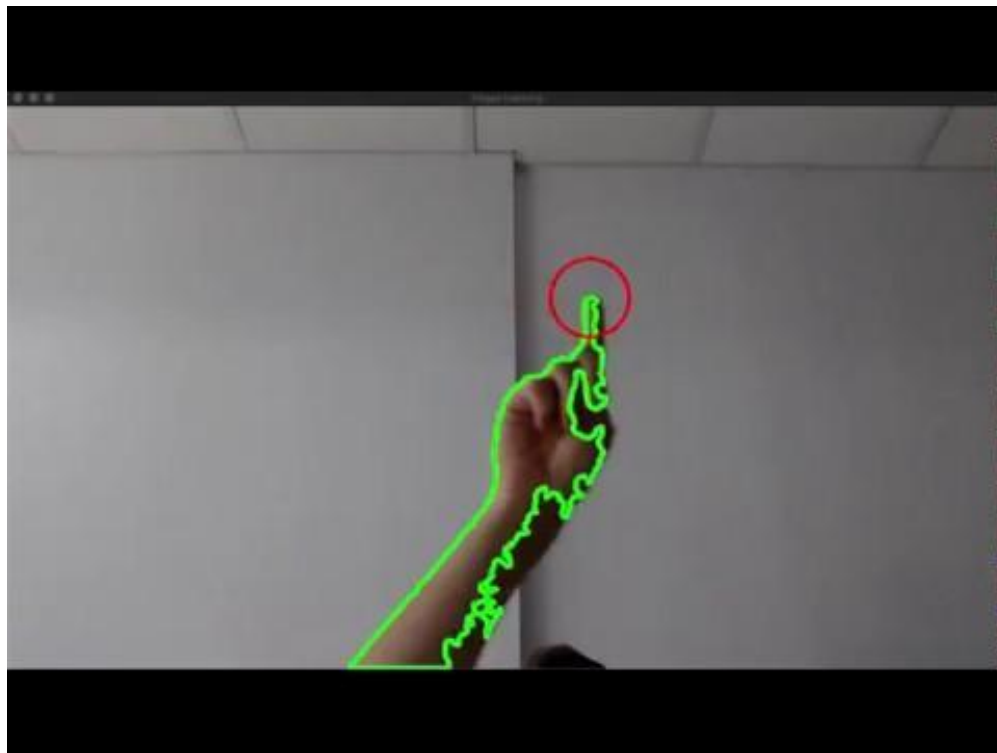
```
func GetHandContour(img *gocv.Mat, handMask gocv.Mat) []image.Point {  
    points := gocv.FindContours(handMask, gocv.RetrievalExternal, gocv.ChainApproxSimple)  
    if len(points) == 0 {  
        return []image.Point{}  
    }  
    ind, area := 0, gocv.ContourArea(points[0])  
    for i, a := range points {  
        tempArea := gocv.ContourArea(a)  
        if tempArea > area {  
            area = tempArea  
            ind = i  
        }  
    }  
    gocv.DrawContours(img, points, ind, color.RGBA{0, 255, 0, 1}, 5)  
    return points[ind]  
}
```

Śledzenie

```
func reader(conn *websocket.Conn) {
    for {
        messageType, p, err := conn.ReadMessage()
        if err != nil {
            log.Println(err)
            return
        }
        p = bytes.Replace(p, imgPrefix, []byte{}, 1)
        unbased := make([]byte, len(p))
        _, err = base64.StdEncoding.Decode(unbased, p)
        if err != nil {
            log.Println("Cannot decode b64")
            return
        }
        img, err := png.Decode(bytes.NewReader(unbased))
        if err != nil {
            log.Println(err)
            continue
        }
    }
}
```

```
        imgMat, err := toRGB8(img)
        if err != nil {
            log.Println(err)
            continue
        }
        point := finger.Detect(imgMat)
        body, _ := json.Marshal(point)
        err = conn.WriteMessage(messageType, body)
        if err != nil {
            log.Println(err)
            continue
        }
    }
}
```

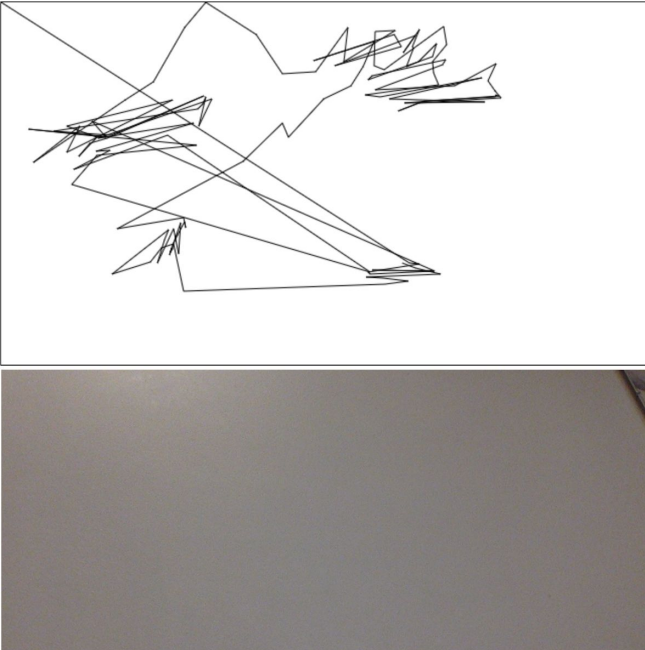
Śledzenie



Strona internetowa

← → ↻ localhost:3000

START VIDEO STOP VIDEO



Elements Console Sources Network >>

top Filter Default le 2 h

```

sending msg ws.
got ▶ {prevX: 155, prevY: 259, x: 162, y: 236} Canvas.
sending msg ws.
got ▶ {prevX: 162, prevY: 236, x: 166, y: 226} Canvas.
sending msg ws.
got ▶ {prevX: 166, prevY: 226, x: 110, y: 270} Canvas.
sending msg ws.
got ▶ {prevX: 110, prevY: 270, x: 148, y: 258} Canvas.
sending msg ws.
got ▶ {prevX: 148, prevY: 258, x: 159, y: 244} Canvas.
sending msg ws.
got ▶ {prevX: 159, prevY: 244, x: 171, y: 240} Canvas.
sending msg ws.
got ▶ {prevX: 171, prevY: 240, x: 181, y: 287} Canvas.
sending msg ws.
got ▶ {prevX: 181, prevY: 287, x: 381, y: 280} Canvas.
sending msg ws.
got ▶ {prevX: 381, prevY: 280, x: 404, y: 277} Canvas.
sending msg ws.
got ▶ {prevX: 404, prevY: 277, x: 379, y: 274} Canvas.
sending msg ws.
got ▶ {prevX: 379, prevY: 274, x: 362, y: 273} Canvas.
sending msg ws.
got ▶ {prevX: 362, prevY: 273, x: 436, y: 270} Canvas.
sending msg ws.
got ▶ {prevX: 436, prevY: 270, x: 411, y: 264} Canvas.
sending msg ws.
got ▶ {prevX: 411, prevY: 264, x: 398, y: 259} Canvas.
sending msg ws.
stopping App.
Socket Closed Connection: ws
▶ CloseEvent {isTrusted: true, wasClean: false, code: :
  reason: "", type: "close", ...}

```


Harmonogram



- Widok, umożliwiający przemieszczanie się po aplikacji,
- Czytywanie i zapisywanie danych z kamery.
- Rozpoznanie i śledzenie ruchu markera,
- Możliwość podglądu w okienku na żywo ścieżki, po której prowadzony był znacznik,
- Możliwość zmazywania rysunku
- Zapisywanie utworzonego obrazu.
- Dopracowanie aplikacji,
- Możliwość zmiany koloru rysunku,
- Możliwość wyboru trybu rysowania.

Podział prac

Wojciech Niedbała

- Przesyłanie obrazu z kamery internetowej
- Widoki

Albert Millert

- Przetwarzanie klatek z otrzymanego obrazu
- Rozpoznawanie znacznika
- Śledzenie ruchu

Patryk Romaniak

- Możliwość podglądu ścieżki (rysowanie)
- Zapisywanie utworzonego obrazu

Michał Mirończuk

- Możliwość zmazywania rysunku
- Możliwość wyboru trybu rysowania
- Zmiany stylu, koloru, grubości markera