1. Implementatieplan titel: Week 2 lokaliseren van de ogen

1.1. Namen en datum

Martijn van der Struijk & Tim Hasselaar

16-05-2015

1.2. Doel

Het doel van deze opdracht is om een gezicht van een persoon te herkennen. Dit kan door middel van meerdere manieren.

1.3. Methoden

1. Het lokaliseren van de bovenkant van het hoofd en de linker en rechter zijkant;
2. Het lokaliseren van de neus, mond en kin;
3. Het lokaliseren van de kincontoeren;
4. Het lokaliseren van de linker en rechter neusvleugel, deel van de wang en deel van de ogen;
5. Het lokaliseren van de ogen.

1.4. Keuze

Wij hebben gekozen om de ogen binnen een gezicht te gaan herkennen. Dit lijkt ons de leukste en de meeste uitdagende methode die er is. Je hebt als je het oog wilt gaan herkennen al snel last van het feit de wenkbrauwen er dichtbij zitten. Dit lijkt ons leuk om te gaan onderzoeken en te kijken hoe we dit kunnen implementeren.

1.5. Implementatie

Wij trekken eerst een vierkant om het hoofd van de persoon in. Daarin weten war de mondhoeken zitten. vanaf dat punt starten we met naar boven gaan tot we bij een pixel komen die onder een bepaalde pixel waarde komt. Dit is het begin van een oog. de breedte pakt die van de mondhoeken tot aan de zijkant van het hoofd. Aan de hand van deze informatie zal er een vierkant om het oog worden getekend.

```
RGBImage * debugImage = ImageFactory::newRGBImage();
       ImageIO::intensityToRGB(image, *debugImage);

       //Create the eye features to return
       Feature featureLeftEye = Feature(Feature::FEATURE_EYE_LEFT_RECT);
   Feature featureRightEye = Feature(Feature::FEATURE_EYE_RIGHT_RECT);

   Feature top_head = features.getFeature(Feature::FEATURE_HEAD_TOP);
   Feature bottom_nose_left = features.getFeature(Feature::FEATURE_NOSE_END_LEFT);
   Feature bottom_nose_right = features.getFeature(Feature::FEATURE_NOSE_END_RIGHT);
   Feature bottom_head_left =
features.getFeature(Feature::FEATURE_HEAD_LEFT_NOSE_BOTTOM);
   Feature bottom_head_right =
features.getFeature(Feature::FEATURE_HEAD_RIGHT_NOSE_BOTTOM);

       //Add the left eye rect
   int x_top_left_left_eye = bottom_head_left.getPoints()[0].getX();
```

```
   int y_top_left_left_eye = top_head.getPoints()[0].getY();
   int x_bottom_right_left_eye = bottom_nose_left.getPoints()[0].getX();
   int y_bottom_right_left_eye = bottom_nose_left.getPoints()[0].getY();

   double IntensityEyeLocalizationStart = 0.3;
   double IntensityEyeLocalizationEnd = 0.07;

   bool found = false;
   for (int i = y_bottom_right_left_eye; i > y_top_left_left_eye; i--){
      int intensity = 0;
      for (int j = x_top_left_left_eye; j < x_bottom_right_left_eye; j++){
         if (debugImage->getPixel(j, i).b == 0){
            intensity++;
         }
      }
      double result = (double)intensity / (double)(x_bottom_right_left_eye -
x_top_left_left_eye);
      if (!found){
         if (result >= IntensityEyeLocalizationStart){
            y_bottom_right_left_eye = i;
            found = true;
         }
      }
      else {
         if (result <= IntensityEyeLocalizationEnd){
            y_top_left_left_eye = i;
            break;
         }
      }
   }

   for (int i = x_top_left_left_eye; i < x_bottom_right_left_eye; i++){
      int intensity = 0;
      for (int j = y_top_left_left_eye; j < y_bottom_right_left_eye; j++){
         if (debugImage->getPixel(i, j).b == 0){
            intensity++;
         }
      }
      double result = (double)intensity / (double)(y_bottom_right_left_eye -
y_top_left_left_eye);
      if (result > 0){
         x_top_left_left_eye = i;
         break;
      }
   }

   featureLeftEye.addPoint(Point2D<double>(x_top_left_left_eye, y_top_left_left_eye));
   featureLeftEye.addPoint(Point2D<double>(x_bottom_right_left_eye,
y_bottom_right_left_eye));


   //Draw rectangles on RGB_Map
   for (int i = x_top_left_left_eye-1; i <= x_bottom_right_left_eye+1; i++){
      debugImage->setPixel(i, y_top_left_left_eye-1, RGB(255, 255, 0));
      debugImage->setPixel(i, y_bottom_right_left_eye+1, RGB(255, 255, 0));
   }
   for (int i = y_top_left_left_eye-1; i <= y_bottom_right_left_eye+1; i++){
      debugImage->setPixel(x_top_left_left_eye-1, i, RGB(255, 255, 0));
```

```
      debugImage->setPixel(x_bottom_right_left_eye+1, i, RGB(255, 255, 0));
   }

   for (int i = x_top_left_right_eye-1; i <= x_bottom_right_right_eye+1; i++){
      debugImage->setPixel(i, y_top_left_right_eye-1, RGB(255, 255, 0));
      debugImage->setPixel(i, y_bottom_right_right_eye+1, RGB(255, 255, 0));
   }
   for (int i = y_top_left_right_eye-1; i <= y_bottom_right_right_eye+1; i++){
      debugImage->setPixel(x_top_left_right_eye-1, i, RGB(255, 255, 0));
      debugImage->setPixel(x_bottom_right_right_eye+1, i, RGB(255, 255, 0));
   }
```

1.6. Evaluatie

We zullen een afbeelding maken van waar het programma denkt dat de ogen zitten en dit met de werkelijkheid vergelijken. Ook zullen we onze manier met de bestaande manier vergelijken.