All about Anigma-View

Chang Wook Seo, Jungjin Park, Seonghyeon Kim, Gyeonghun Im, Myunggyun Seo, Yerim Shin, Kyungmin Cho, Seung Han Song

All about Anigma-View

Chang Wook Seo
Anigma Technologies Inc.
lgtwins@anigma-ai.com

JungJin Park
Anigma Technologies Inc.
jinpark@anigma-ai.com

Seonghyeon Kim
Anigma Technologies Inc.
okdalto@anigma-ai.com

Gyeonghun Im
Anigma Technologies Inc.
nsoar@anigma-ai.com

Myunggyun Seo
Anigma Technologies Inc.
jonathan@anigma-ai.com

Yerim Shin
Anigma Technologies Inc.
shema117@anigma-ai.com

Kyungmin Cho
Anigma Technologies Inc.
ckm@anigma-ai.com

Seung Han Song
Anigma Technologies Inc.
song@anigma-ai.com

ABSTRACT

This technical report offers an overview of the functions implemented in Anigma-View. Each function is thoroughly described. Please cite this report when referencing any Anigma-View functions in your academic papers or manuscripts. Note that this report solely explains the technical logic behind the software's value measurement. It does not provide detailed information on optimizing or fine-tuning the functions (e.g., alignment, segmentation).

CCS CONCEPTS

• **Applied computing**; • **Computing methodologies** → **Image processing**;

KEYWORDS

Computer Vision, Medical Image

1 BASE PIPELINES

1.1 Alignment

All input face images to Anigma-View undergo a face alignment process, which aligns the images to the proper position and angle. More details about face alignment systems can be found in Jin et al. [JT17].

1.2 Segmentation

Measurement functions in Anigma-View are processed after segmenting the input face image. Anigma-View utilizes the segmented image to measure facial features. See Figure 1 for an example.

1.3 Measurement&Settings

This technical report is written with the assumption that the coordinate values are based on the OpenCV [Its14] library. In OpenCV, the x and y coordinates start from the top-left corner rather than the bottom-left. See Figure 2 for the coordinate system used in OpenCV.

Authors' addresses: Chang Wook Seo, Anigma Technologies Inc., lgtwins@anigma-ai.com; JungJin Park, Anigma Technologies Inc., jinpark@anigma-ai.com; Seonghyeon Kim, Anigma Technologies Inc., okdalto@anigma-ai.com; Gyeonghun Im, Anigma Technologies Inc., nsoar@anigma-ai.com; Myunggyun Seo, Anigma Technologies Inc., jonathan@anigma-ai.com; Yerim Shin, Anigma Technologies Inc., shema117@anigma-ai.com; Kyungmin Cho, Anigma Technologies Inc., ckm@anigma-ai.com; Seung Han Song, Anigma Technologies Inc., song@anigma-ai.com.

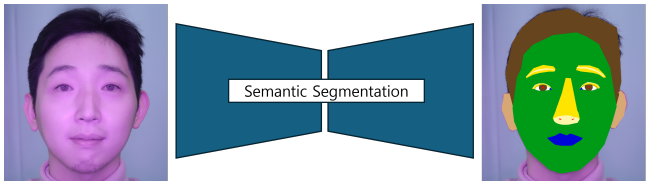

Figure 1: Anigma-View segments the input face image before measuring the features of the face.

Values in Anigma-View are calculated relative to an assumed iris radius of 11.5 mm. Compared to ImageJ [SRE12], a well-known image measurement tool that relies on user-defined scaling settings, Anigma-View shows similar measurement values. We customized the ImageJ scale by setting the iris radius to 11.5 mm. The results are shown in Figure 3. The two software tools show only a slight difference in measurement values. Note that ImageJ's scaling setting is based on the user's manual input for each image, which can introduce slight errors in the values.

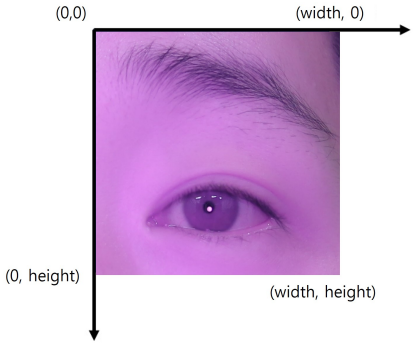

Figure 2: Coordinate system in OpenCV

2 FUNCTIONS

2.1 Base functions

2.1.1 Circle detection. To detect the coordinates of circle shape object center (x,y) and radius, we used circle detection function

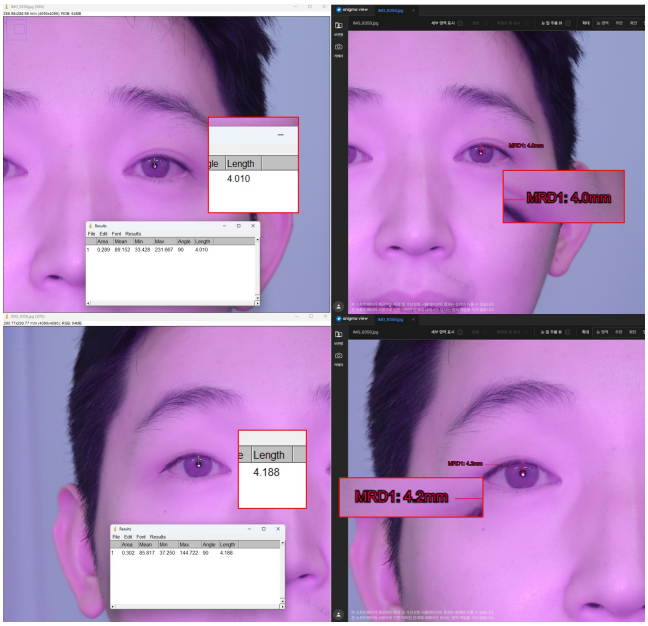

Figure 3: Comparing measure scale with imageJ, Anigma-View SW shows only slight differences

which is well known from previous studies [DH72]. The detection process is written below.

(1) Edge Detection: Apply an edge detection algorithm [Can86] to the input image to highlight the edges.
(2) Voting in Hough Space: For each edge pixel (x, y), vote in the Hough parameter space for all possible circles that could pass through the point. This involves varying the radius r and voting for the center points (a, b) such that:

$$a = x - r\cos(\theta)$$

$$b = y - r\sin(\theta)$$

where θ ranges from 0 to 360 degrees.
(3) Finding Circles: Identify peaks in the Hough parameter space, which correspond to the most likely circle centers and radii.
(4) **Return:** x,y and radius of detected circle.

2.2 Distance

2.2.1 MRD1. To measure MRD1, we detected the **UpperEyelidMargin** coordinate using an algorithm with a segmentation mask as input. Based on the segmentation mask, the y-coordinates of the pupil, iris, white area, and upper eyelid crease, whose x-coordinates correspond to the center of the pupil (estimated via the circle detection function mentioned above), are extracted and stored as lists. The initial value of the upper eyelid margin coordinate is set to the minimum y-coordinate of the iris and pupil (min(IrisYCoords[0], PupilYCoords[0])). If there are y-coordinates for the white area (WhiteYCoords), the **UpperEyelidMargin** is updated to the minimum y-coordinate among the current **UpperEyelidMargin** and the white area (min(**UpperEyelidMargin**, WhiteYCoords[0])). See Algorithm 1 for more details.

By calculating the distance between center of pupil and **UpperEyelidMargin**, MRD1 value is measured.

Algorithm 1 Detect Upper Eyelid Margin

Input: PupilMask, IrisMask, WhiteAreaMask, UpperEyelidMask
Output: UpperEyelidMargin
Estimate PupilCenterX via circle detection function
Initialize lists PupilYCoords, IrisYCoords, WhiteAreaYCoords, UpperEyelidYCoords
for each row y in PupilMask **do**
  if PupilMask[y][PupilCenterX] is not 0 **then**
    Append y to PupilYCoords
  end if
end for
for each row y in IrisMask **do**
  if IrisMask[y][PupilCenterX] is not 0 **then**
    Append y to IrisYCoords
  end if
end for
for each row y in WhiteAreaMask **do**
  if WhiteAreaMask[y][PupilCenterX] is not 0 **then**
    Append y to WhiteAreaYCoords
  end if
end for
for each row y in UpperEyelidMask **do**
  if UpperEyelidMask[y][PupilCenterX] is not 0 **then**
    Append y to UpperEyelidYCoords
  end if
end for
Set UpperEyelidMargin ← min(IrisYCoords[0], PupilYCoords[0])
if len(WhiteAreaYCoords) > 0 **then**
  UpperEyelidMargin ← min(UpperEyelidMargin, WhiteAreaYCoords[0])
end if
return UpperEyelidMargin

2.2.2 MRD2. We detected the **LowerEyelidMargin** coordinate using a segmentation mask as input to measure MRD2. Similar to the **UpperEyelidMargin**, the y-coordinate lists from the pupil, iris, and white area, where the x-coordinates correspond to the center of the pupil, are stored as lists. Therefore, the **LowerEyelidMargin** is detected using these y-coordinate lists with a similar algorithm to the **UpperEyelidMargin**, but by finding the maximum value in this case. See Algorithm 2 for more details.

By calculating the distance between center of pupil and **LowerEyelidMargin**, MRD2 value is measured.

Algorithm 2 Detect Lower Eyelid Margin

Input: PupilMask, IrisMask, WhiteAreaMask
Output: LowerEyelidMargin
Estimate PupilCenterX via circle detection function
Initialize lists PupilYCoords, IrisYCoords, WhiteAreaYCoords
for each row y in PupilMask **do**
  if PupilMask[y][PupilCenterX] is not 0 **then**
    Append y to PupilYCoords
  end if
end for
for each row y in IrisMask **do**

```
    if IrisMask[y][PupilCenterX] is not 0 then
        Append y to IrisYCoords
    end if
end for
for each row y in WhiteAreaMask do
    if WhiteAreaMask[y][PupilCenterX] is not 0 then
        Append y to WhiteAreaYCoords
    end if
end for
LowerEyelidMargin ← max(IrisYCoords[−1], PupilYCoords[−1])

if len(WhiteAreaYCoords) > 0 then
    LowerEyelidMargin ←
    max(LowerEyelidMargin, WhiteAreaYCoords[−1])
end if
Set UpperEyelidMargin ← min(IrisYCoords[0], PupilYCoords[0])
return LowerEyelidMargin
```

2.2.3 PFH. PFH is measured by calculating the distance between value of **UpperEyelidMargin** and **LowerEyelidMargin**.

2.2.4 DH. To measure DH, we detected the **DoubleEyelidMargin** coordinates by using the detected **UpperEyelidMargin** value and the double eyelid area from the segmentation mask. The y-coordinate list from the double eyelid area, where the x-coordinates correspond to the **UpperEyelidMargin**, is stored as lists. Therefore, the **DoubleEyelidMargin** is determined by using the smallest value from this list. See Algorithm 3 for more details.

By calculating the distance between the **DoubleEyelidMargin** and the **UpperEyelidMargin**, the DH value is measured.

Algorithm 3 Detect DoubleEyelidMargin

```
Input: DoubleEyelidMask, UpperEyelidMargin
Output: DoubleEyelidMargin
Initialize list DoubleEyelidYCoords
if DoubleEyelidMask exists then
    for each row y in PupilMask do
        if DoubleEyelidMask[y][UpperEyelidMarginX] is not 0
        then
            Append y to DoubleEyelidYCoords
        end if
    end for
    Set DoubleEyelidMargin ← min(DoubleEyelidYCoords)
    return DoubleEyelidMargin
end if
```

2.2.5 IPD. Based on the pupil area in the segmentation mask, we detected the center of the pupil coordinates using a circle detection function. Therefore, the IPD is measured by calculating the distance between the centers of each pupil.

2.2.6 ICD. Based on the detected medial canthus in the segmentation mask, ICD is measured by calculating the distance between the medial canthus of each eye.

2.2.7 PFW. Based on detected medial canthus and lateral canthus in the segmentation mask, PFW is measured by calculating the distance between medial canthus and lateral canthus.

2.2.8 HL. Based on detected medial canthus and lateral canthus in the segmentation mask, HL is measured by calculating the x-axis distance between medial canthus and lateral canthus.

2.2.9 VL. Based on detected lateral canthus in the segmentation mask, VL is measured by calculating the y-axis distance between **LowerEyelidMargin** and lateral canthus.

2.2.10 Pupil to canthus. Based on detected medial canthus in the segmentation mask, Pupil to canthus is measured by calculating the distance between medial canthus and center of pupil.

2.2.11 Sup border, Inf border and Mid-portion. Based on the detected medial canthus of both eyes, draw a straight line to establish the base height of the eye border. By calculating the distance from points in the eyebrow (upper and lower), the Sup and Inf border values are measured. The Mid-portion value is calculated from the median point between the upper and lower eyebrow area points. The eyebrow points are detected based on the segmentation map in the order of A, C, F, and G, named according to Asaad et al. [AKJ+19]. The X-axis of these points are aligned based on the medial canthus (A), pupil (C), and lateral canthus (F). Point G is not aligned to a specific point from the eye part. See Figure 4 for a detected example

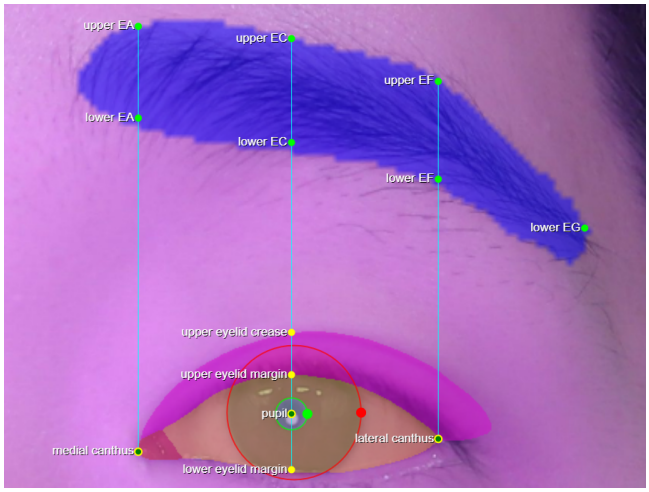

Figure 4: Detected and aligned points of eyebrow area (upper and lower). Points are named in order of A,C,F and G.

2.3 Angle

2.3.1 ES. Based on the detected medial canthus and lateral canthus in the segmentation mask, we created another point using the x-value from the medial canthus and the y-value from the lateral canthus. Using this newly created point and the existing two canthus, we measured the ES angle value.

2.4 Ratio

2.4.1 CER. To measure the Cornea Exposure Ratio (CER), we detected the pupil and iris coordinates using circle detection on the segmented image. We combined the pupil and iris masks to create a cornea mask. For the exposed eye mask, we combined the cornea

mask with the white area mask. We then initialized a boolean mask of the same shape as the cornea mask to create the full iris mask, setting each pixel to True if it lies within the iris radius from the center coordinates of the iris. Subsequently, we updated the cornea mask by performing a logical AND operation with the full iris mask and the exposed eye mask. We computed the areas of the full iris and the cornea by counting the non-zero elements in their respective masks. Finally, we calculated the Cornea Exposure Ratio (CER) as the ratio of the cornea area to the full iris area, multiplied by 100. See Algorithm 4 for details.

Algorithm 4 Measuring Cornea Exposure Ratio (CER)

Input: pupil coordinates $(pupil_x, pupil_y, pupil_r)$, iris coordinates $(iris_x, iris_y, iris_r)$, PupilMask, IrisMask, WhiteAreaMask
$mask\_cornea \leftarrow mask\_pupil \vee mask\_iris$
$mask\_exposed\_eye \leftarrow mask\_cornea \vee mask\_white\_area$
Initialize $mask\_full\_iris$ as a boolean array of the same shape as $mask\_cornea$
for each pixel (x, y) in $mask\_full\_iris$ **do**
  if $(x - iris_x)^2 + (y - iris_y)^2 < iris_r^2$ **then**
    $mask\_full\_iris[y, x] \leftarrow$ True
  end if
end for
$mask\_cornea \leftarrow mask\_full\_iris \wedge mask\_exposed\_eye$
$full\_iris\_area \leftarrow$ count_nonzero$(mask_full_iris)$
$cornea\_area \leftarrow$ count_nonzero$(mask_cornea)$
$CER \leftarrow \frac{cornea\_area}{full\_iris\_area} \times 100$
return CER

2.4.2 MWR. To measure the Medial White Area Ratio (MWR), we calculated the ratio of the Medial White Area (MWA) to the total area, multiplied by 100. To find the MWA, we counted the non-zero elements in the white area mask to the left of the pupil center (:pupil_x), if the eye is the left eye (eye_is_left is True), W . If the eye is the right eye (eye_is_left is False), count the non-zero elements in the white area mask to the right of the pupil center (pupil_x:). The total area is defined by the sum of the non-zero elements in the white area mask and the cornea mask. Check Algorithm 5 for more details.

Algorithm 5 Measuring Medial White Area Ratio (MWR)

Input: pupil coordinates $(pupil_x, pupil_y, pupil_r)$, WhiteAreaMask, $mask\_cornea$ (from Algorithm 4)
if eye_is_left **then**
  $MWA \leftarrow$ count_nonzero$(WhiteAreaMask[:, : pupil_x])$
else
  $MWA \leftarrow$ count_nonzero$(WhiteAreaMask[:, pupil_x :])$
end if
$total\_area \leftarrow$ count_nonzero$(WhiteAreaMask)$ + count_nonzero$(mask_cornea)$
$MWR \leftarrow \frac{MWA}{total\_area} \times 100$
return MWR

2.4.3 LWR. To measure the Lateral White Area Ratio (LWR), we calculated the ratio of the Lateral White Area (LWA) to the total area, multiplied by 100. To find LWA, we counted the non-zero elements in the white area mask from the pupil center to the right edge (:pupil_x), if the eye is the left eye (eye_is_left is True), W . If the eye is the right eye (eye_is_left is False), count the non-zero elements in the white area mask from the left edge to the pupil center (pupil_x:). The total area is defined by the sum of the non-zero elements in the white area mask and the cornea mask. Check Algorithm 6 for more details.

Algorithm 6 Measuring Lateral White Area Ratio (LWR)

Input: pupil coordinates $(pupil_x, pupil_y, pupil_r)$, WhiteAreaMask, $mask\_cornea$ (from Algorithm 4)
if eye_is_left **then**
  $LWA \leftarrow$ count_nonzero$(WhiteAreaMask[:, pupil_x :])$
else
  $LWA \leftarrow$ count_nonzero$(WhiteAreaMask[:, : pupil_x])$
end if
$total\_area \leftarrow$ count_nonzero$(WhiteAreaMask)$ + count_nonzero$(mask_cornea)$
$LWR \leftarrow \frac{LWA}{total\_area} \times 100$
return LWR

2.4.4 EER (written as EEA in Anigma-View). Based on Eyeball Exposure Area (EEA) and whole area in the segmentation mask, the Eyeball Exposure Area Ratio (EER) is calculated by the ratio of EEA to the whole area, multiplied by 100. See Figure 5 for segmentation of EEA and whole area.

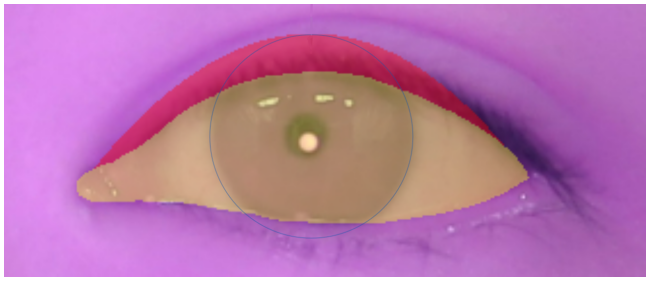

Figure 5: The segmented image is used for measuring EER. The red segmented area represents the EEA, while the combined red and yellow areas represent the whole area.

REFERENCES

[AKJ+19] Malke Asaad, Ahmad Beshr Kelarji, Cham Shaban Jawhar, Joseph Banuelos, Editt Taslakian, Waseem Wahood, Krishna S Vyas, and Basel Sharaf. Eyebrow height changes with aging: a systematic review and meta-analysis. *Plastic and Reconstructive Surgery–Global Open*, 7(9):e2433, 2019.

[Can86] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[DH72] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[Its14] Itseez. *The OpenCV Reference Manual*, 2.4.9.0 edition, April 2014.

[JT17] Xin Jin and Xiaoyang Tan. Face alignment in-the-wild: A survey. *Computer Vision and Image Understanding*, 162:1–22, 2017.

[SRE12] Caroline A Schneider, Wayne S Rasband, and Kevin W Eliceiri. Nih image to imagej: 25 years of image analysis. *Nature methods*, 9(7):671–675, 2012.