

# コンピュータビジョン説明書

## 0. 提出作品

/images/forWall/wall+mr+wd.jpg



- 実際にアイドルがそこにいるかのような画像を目指した

/images/forGif/PKB.gif



- Red Velvet の「Peek-A-Boo」という MV がコンセプト
- GIF 画像

/images/forTile/Tiled-alpha.jpg



- モザイクアートを目指した

# 1. 使用ファイル

## 1.1. ファイル一覧

### 既存のものを改良したファイル

- ScaleEX.java
- AlphaBlendingHole.java
- RotatinoX.java
- TilingX.java

### 自作ファイル

- AddWhite.java
- Size.java
- Noise.java
- And.java
- Or.java
- StainedGlass.java
- GifSequenceWriter.java
- Gray.java
- Tuple.java
- Hole.java
- Celare.java
- VoidChromakey.java
- CelareX.java
- WhiteEdge.java
- CircleCut.java
- Mosaic.java
- Cut.java
- CutX.java
- MosaicX.java
- ShadeCircle.java
- ShadeSquare.java

### 既存ファイル

- SpaceFiltering.java
- GammaCorrection.java
- Binalization.java
- Chromakey.java

- KMeans.java

## 1.2. ファイル概要

既存のものを改良したファイル

- ScaleEX.java
  - 拡大率を引数に取れるように
- AlphaBlendingHole.java
  - 2枚の画像の大小、(z軸方向の)上下に関わらず、重ね合わせ可能に
- RotatinoX.java
  - 任意の角度に対応
- TilingX.java
  - 任意の枚数、大きさに対応

自作ファイル

- AddWhite.java
  - 任意の場所に白を追加する(クロマキー用)
- Size.java
  - 入力画像の大きさを
- Noise.java
  - 任意の割合でノイズを追加
- And.java
  - 入力画像配列の白い部分を AND 結合
- Or.java
  - 入力画像配列の白い部分を OR 結合
- StainedGlass.java
  - 指定したコア数でステンドグラス画像を作成
- GifSequenceWriter.java
  - 入力画像配列で GIF 画像を作成
- Gray.java
  - 入力画像をグレースケール
- Tuple.java
  - 組という型を導入
- Hole.java
  - 穴(一定の大きさの黒い部分)の初期座標を組で返す
- Celare.java
  - ケラレ加工(円形に外側に行くにつれて暗くなる)
- VoidChromakey.java

- 任意の大きさ・場所に白を加える(クロマキー用)
- CelareX.java
  - ケラレ加工改善版
- WhiteEdge.java
  - 入力画像を縁取り(任意の形に対応)
- CircleCut.java
  - 指定された半径の円で入力画像を切り取り
- Mosaic.java
  - モザイク後の画素の一片の長さを指定し、モザイク処理
- Cut.java
  - 入力画像を任意の大きさ、初期座標で切り取り
- CutX.java
  - 入力画像の黒い部分が終わる上端下端右端左端の座標を組で返す
- MosaicX.java
  - 入力画像についてy軸方向の境界線より右側をモザイク処理
- ShadeCircle.java
  - 円形の影を入力画像内の任意の位置に任意の大きさで追加
- ShadeSquare.java
  - 四角形の影を入力画像内の任意の位置に任意の大きさ・傾きで追加

### 1.3. 使用例

#### 改良したクラス・使用例

- ScaleEX.java
  - x, y 軸方向への拡大率を受け取るように変更
- RotationX.java
  - 入力画像と任意の角度[°]を受け取り、入力画像を反時計回りに角度分回転したものを返す
  - 任意の角度(メソッド内で%360→to radian)に対応するように
  - 角度を受け取るようにしただけでなく、回転後の画像の大きさ等も自動で算出し、回転し、長方形でなくなった部分は黒くなるように改良した
  - タプルを利用し、各 4 頂点の回転後の頂点のうち、x(y)座標が最も大きいものと最も小さいものの差の絶対値を output の画像サイズとした
    - ✧ 画像座標系は y 軸が逆なので注意が必要！



- TilingX.java
  - 画像の縦、横、file の置いてあるパス、縦に配置する file 数、横に配置する file 数を受け取り、各画像が正方形になるよう ScaleEX→Cut したものを対応する大きさと file 数に並べた画像を返す
  - ✧  $[j/dwidth + (i/dwidth) * (width/dwidth)]$  番目の画像について  
 $(j \% dwidth, i \% dwidth)$  の部分の色を取ってくる(大変だった)  
300 x 400, 縦 3 つ, 横 4 つ



### CvMain 内の新設メソッド

- PutOnChekis
  - 壁画像、チェキ画像、チェキ位置、回転の有無を受け取り、それらを考慮してチェキ画像+適切な位置に ShedeSquare.java による影を壁画像に貼り付けた画像を返す



- CheckKmeans
  - クラスタ数、ファイルパス、入力画像を受け取り、Kmeans を実行して i 番目の id に対応する画像をファイルパスに i.jpg として保存し、かつその画像配列を返す





### 増設クラス・使用例

#### - Celare.java

- 入力画像、半径を受け取り、対応するケラレ加工をした画像を返す
  - ✧  $3 - (\sqrt{(j - centerX)^2} + (i - centerY)^2 - \sqrt{r^2})/100$
  - ✧ 汎用性がない

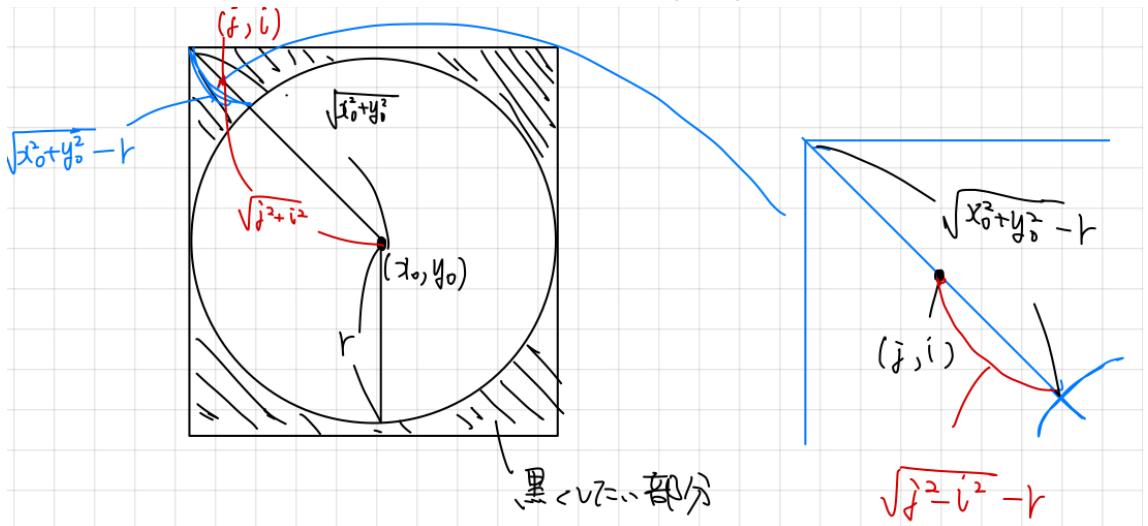


#### - CelareX.java

- 入力画像、半径を受け取り、対応するケラレ加工をした画像を返す
  - ✧ トーンカーブ+幾何変換

✧ 以下のような関数を設定

$$rgb' = rgb - rgb * \frac{\sqrt{(j^2 + i^2)} - r}{\sqrt{(x_0^2 + y_0^2)} - r}$$



- Cut.java
  - 入力画像、切り取り後の幅、高さ、初期 x 座標、y 座標を受け取り、それに応じて切り取った画像を返す
  
- StainedGlass.java
  - ゆるゆるプログラミング参照



- CutX.java
  - 入力画像(通常、クロマキー画像)を受け取り、黒い部分が始まる左端、右端、上端、下端の整数配列を返す
  - CutX と Cut を併用することで、不要な部分の除去が可能になる



- CircleCut.java
  - 入力画像、初期 x, y 座標、半径を受け取り、入力画像を半径分円形に切り取った画像を返す



- AlphaBlendingHole.java
  - 入力画像 1(クロマキーでない)、入力画像 2(クロマキー処理した元画像)、クロマキー後の画像、重ね合わせる始点 x 座標、y 座標、上に重ねる画像番号(1 or 2)、大きい方の画像番号(1 or 2)、クロマキー画像の方が大きいか否かの boolean 値を受け取り、全て考慮した重ね合わせをした画像を返

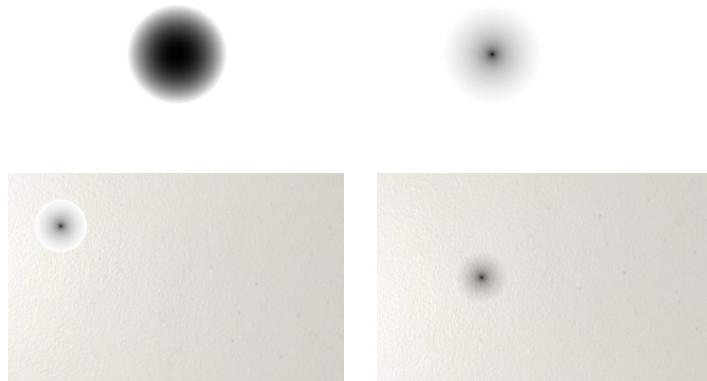
す

- あらゆる画像の重ね合わせ(入力画像 1, 2 の大きさやクロマキー処理した画像の大きさ、始点位置に依存しない)に対応するよう汎用的にプログラム



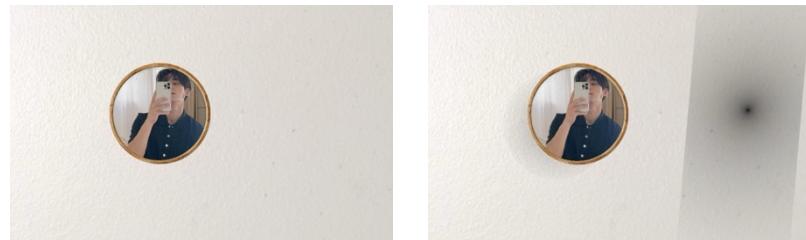
- ShadeCircle.java

- 入力画像、初期 x, y 座標、半径、傾き[°]を受け取り、入力画像に対応した円形の影を追加した画像を返す
- 最初は x 軸 $(j - x_0)^2 + (i - y_0)^2$  y 軸 r( $=g=b$ )値, 定義域  $0 \leq x \leq r^2$ , 値域 $0 \leq y \leq 255$ の線形変換
- つまり  $y = 255 * \frac{dx^2 + dy^2}{r^2}$ 
  - ✧ 影にしては暗い部分が大きすぎるため対数関数による変換に変更
- $\log_{10}(a * (dx^2 + dy^2))$ 
  - ✧ 白を除去できない
- 対数関数により壁紙画像の任意の場所に直接影を描写する方法に変更



- ShadeSquare.java

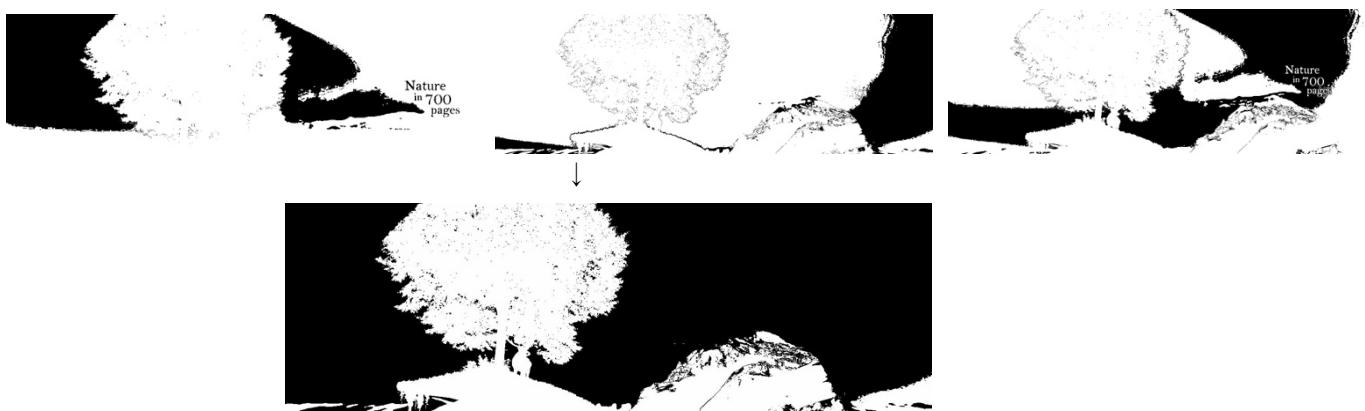
- 入力画像、初期 x, y 座標、影の幅、縦、傾き[°]を受け取り、入力画像に対応した四角形の影を追加した画像を返す
- 傾きはアフィン変換(スキー)に利用
  - ◆  $x' = x + y * \tan \theta$



- VoidChromaky.java
  - 入力画像、左上座標(タプル)、右下座標(タプル)を受け取り、その部分を白くした画像を返す

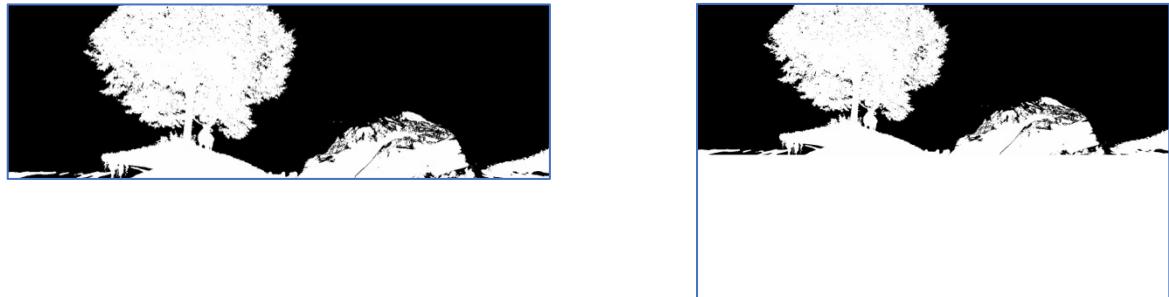


- And.java
  - 入力画像配列を受け取り(配列内の画像はクロマキー処理後の画像かつ全て大きさが同じと仮定)、白い部分を AND した画像を返す
    - ◆  $rgb$  それぞれについて  $r[i] = r[i-1]*input.getRed()$  した後、 $rgb$  それぞれの n 乗根を setColor する



- AddWhite.java
  - 入力画像、追加後の幅、縦、入力画像の初期 x 座標、y 座標を受け取り、正

しい大きさになるよう入力画像を初期座標に配置し、他の部分に白い部分を追加した画像を返す



- Or.java
  - 入力画像配列を受け取り(配列内の画像はクロマキー処理後の画像かつ全て大きさが同じと仮定)、白い部分を OR した画像を返す
    - ✧  $r1[j, i] += r0[j, i]/n$
- WhiteEdge.java
  - 入力画像、入力画像のクロマキー画像、ずらし幅[px]を受け取り、ずらし幅分周りを白く(任意の柄で)縁取った画像を返す
    - ✧ 自作クラス Cut.java と AddWhite.java を用いて、ずらし幅分左、右、上、下にクロマキー画像をずらした画像配列を作る。その配列を Or.java に渡して白い部分を OR 結合した画像を受け取り、元画像と縁取り分ずらしたクロマキー画像を参照し、ずらしたクロマキー内か否か、ずらす前のクロマキー内か否かで場合分け

15px, 新聞画像で縁



- Noise.java

- 入力画像とノイズ割合を受け取り、入力画像にノイズを加えた画像を返す
  - ✧ (j, i)ごとに乱数を発生させ、ノイズ割合より大きければ黒くする



- Gray.java

- 入力画像と境界線( $x=k$ )を受け取り、境界線より右側をグレースケールにした画像を返す



- Mosaic.java

- 入力画像、カーネルサイズを受け取り、カーネルサイズに対応したモザイク処理した画像を返す
- アルゴリズム：分割後平滑化



- MosaicX.java

- 入力画像、境界線( $x=k$ )、カーネルサイズを受け取り、境界線より右側をカーネルサイズに対応したモザイク処理した画像を返す
  - ❖ Cut.java→Mosaic.java→Tiling.java

input,

kernel size = 10,

kernel size = 40

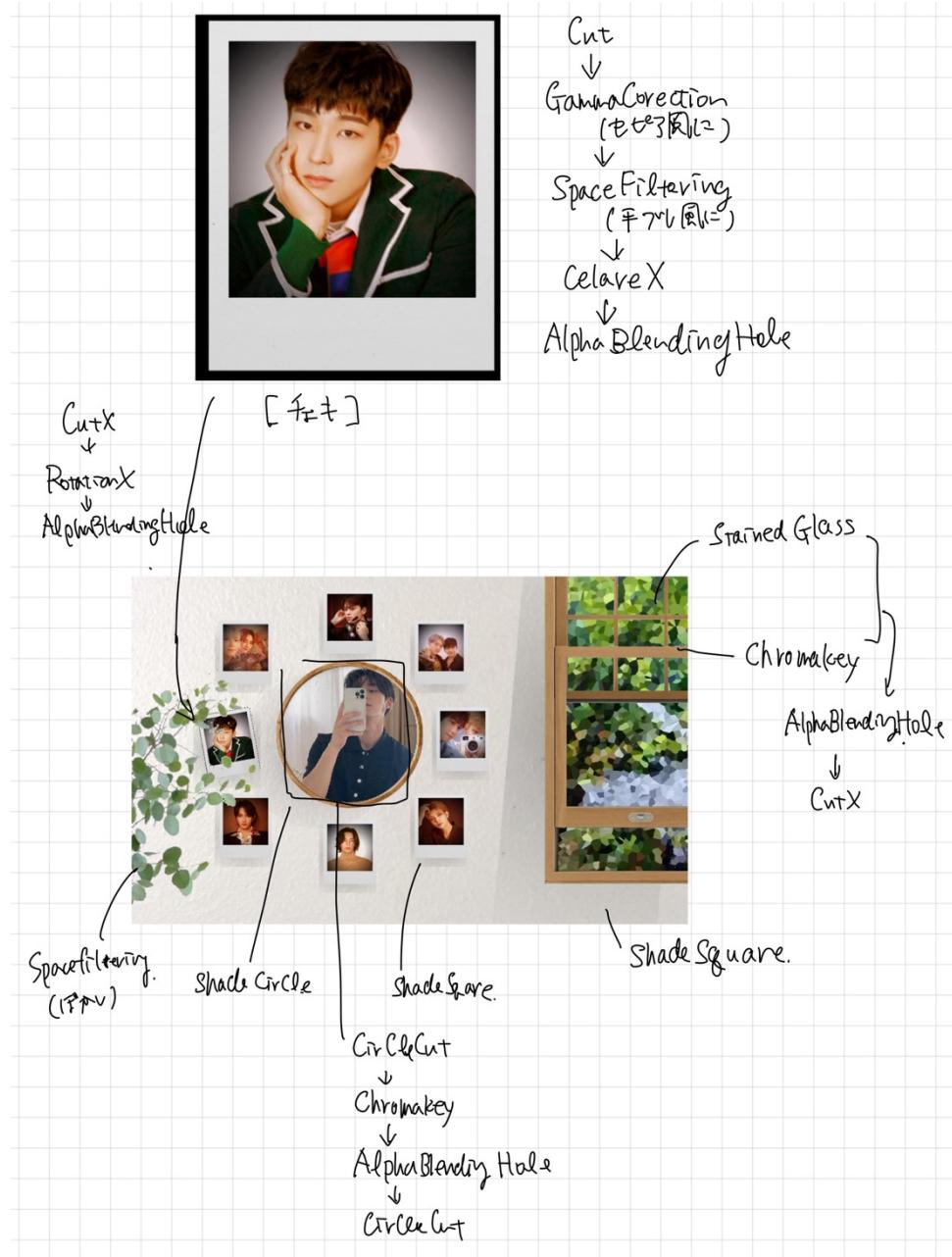


- GifSequenceWriter.java

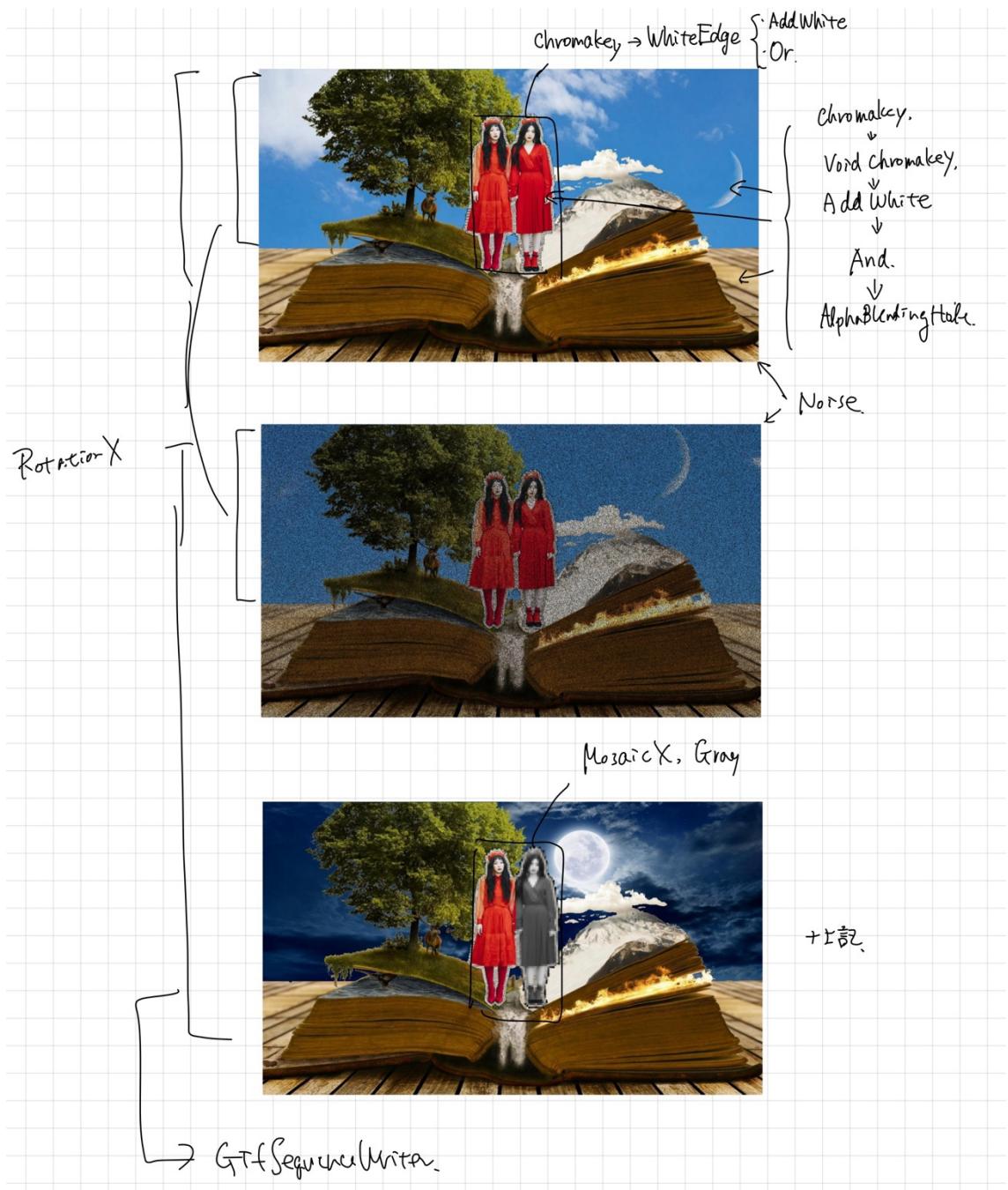
- 画像配列と output 名を受け取り、それらを結合して GIF 画像にしたものを作成する
- [サイト](#) を参考に、execute メソッドを追加

## 2. 作品概要

### 2.1. Polaroid-Selfie



## 2.2. Peek-A-Boo



### 2.3. Mosaic-Art



### 3. 感想

#### 3.1. 大変だったもの

- RotationX.java
  - 任意の角度に対応するようプログラムするのが難しかった。y 軸が逆であるから、逆向きに回転したり、座標が負になったりした時の getColor の仕方やその後の画像の幅と高さをどうすれば良いか悩んだ。
- TilingX.java
  - すべての画像を自動で読み込み、自動で並ぶようにしたかったので、 $j$  行  $i$  番目の画像をどうに getColor し、何番目の画像と対応つけたら良いのか悩んだ。
- CelareX.java
  - 中心からの距離に RGB 値が線型に減るように、かつ正規化されるように数式を定義づけるのが難しかった。
- ShadeCircle.java, ShedeSquare.java
  - 影らしくなるような数式の定義づけに悩んだ。
  - Square に関しては、傾けた際に、傾ける前の部分の色がどうしても真っ黒になってしまい、悩んだ。
- AlphaBlendingHole.java
  - あらゆるパターンに対応するように、かつ初期座標も考慮されるように何度も図を書いた。ずっと悩んでいた。

#### 3.2. 汎用性の高いもの

- 上記すべて
  - なるべく汎用性が高くなるよう試行錯誤した。画像ごとのちょっとした調整等は面倒だと感じたから。
- ScaleEX.java
- Tuple.java
  - java にも組という型があれば楽(あるのかも)

#### 3.3. 全体

推しながら頑張ることができた。汎用性の高いプログラムを書けるようになったと思う。画像処理には想像以上に数学が使われていることを実感した。

#### 3.4. 参考サイト

デザインの参考になったもの

[peek-a-boo 考察](#)

プログラムの参考になったもの

[ゆるゆるプログラミング](#)

[空間フィルタリングによる平滑化とエッジ検出](#)

[線形変換](#)

[Is there a way to create one Gif image from multiple images in Java? \[closed\]](#)