コンピュータビジョン 自由制作課題説明書

目次

1　作品 1 について

1.1　制作作品コンセプト

この画像のテーマはハワイです。日の入りの海面で踊るフラダンサー達が水面に反射しているのを表した、ファンタジーな作品です。落ちていく太陽の光によってシルエットになっているダンサーをイメージしました。

今回のコンセプトが壁紙画像だったので、PC で作業後の疲れ目に、まるで遠くを見ているかのような画像を壁紙に設定することで、視力の回復を見込みたいと考えました。また、水面の模様と影に透過処理を用いて、水面に映る画像処理に挑戦してみたかったことと、私自身がハワイに行きたいという願望から、このテーマに決めました。

1.2　使用したクラスについて

作品 1 に用いたクラスと、その目的を記します。

・Binalization.java

シルエットが欲しかったので、画像を白と黒の 2 値のみにするために用いた。元の画像はシルエットの画像だが、背景や境界線に白黒以外の色も混じっているので、この作業は必要である。

・Scale.java

シルエット画像の大きさの調整を行なった。

・Rotation.java

水面に映る影を作るためにまずは 180 度回転の幾何変換を施すために使った。

・Kmeans.java

Chromakey 処理に使うために、画像を領域分割させて減色させた。今回は引数に 6 を指定したので、6 色に減色されている。

・Chromakey.java

先に述べた Kmeans クラスを用いて、背景からシルエットの部分のみを抜き出す処理を行うために用いた。

・AlphaBlending(画像に依存した投資番号).java

合成するフラダンサーのシルエット画像によって固有の AlphaBlending のクラスを作成した。
倒立していないフラダンサーの画像に関しては、合成位置指定と透過処理のために用いた。
背景の地平線の画像の上に 4 枚の画像がバランスよく並ぶように位置を指定して合成した。
透過処理では、水面の波の模様とダンサーの画像が混じるように処理を施した。
それに対して、倒立している反射面の方のフラダンサーの画像のために、上記のような合成位置指定と透過処理の目的に加えて、180 度幾何変換させた画像をさらに、左右反転させる処理を施した。
左右反転の処理では、x 軸方向の出力を操作して実現することができた。

1.3 用いた画像

1.4 使用手順

以下に適用したクラスとその効果をピックアップしたものです。

図 1

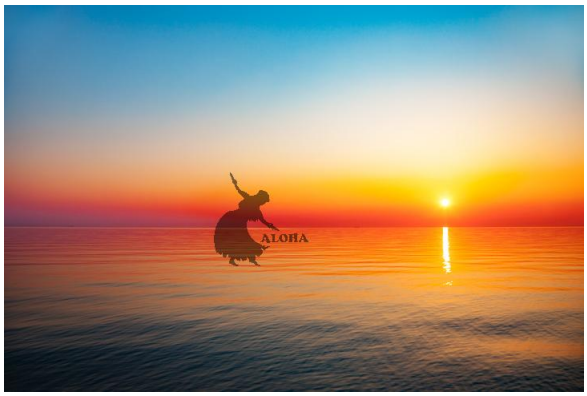

図 2

図 1 は、hawaii2.jpg に Scale.java を用いて大きさの調整、AlphaBlending.java を用いて合成位置の調整をして sunset.jpg に合成をしたものである。
図 2 は、図 1 の hawaii2.jpg の背景が邪魔なので Binalization.java で背景を切り取ったものである。

図 3

図 4

図 3 は、図 2 のように AlphaBlending クラスを用いて、合成位置と透過処理を施して、複数のシルエット画像をバランスよく並べたものである。図 4 は、180 度回転・反転させて複数をちょうど影になるような位置に配置したものである。

2　作品 2 について

2.1　制作作品コンセプト

この作品は、元の画像をステンドグラス風にする加工を施してから、時刻によって画像の色味を変えたものです。窓のステンドグラスが時間によって陽の光の加減で写り方が変わる様子を壁紙にしたいと考え、この作品を作成しました。

2.2　使用したクラスについて

・StainedGlass.java
入力画像をステンドグラス風にするためのクラス。
第一引数に、ステンドグラス風にしたい JPEG 画像のファイル名、第二引数に作成する JPEG 画像ファイル名、第三引数にどのくらい細かくステンドグラス風にするかを指定するための 2 以上の値を指定する。
この第三引数の値によってどのように変化するかは、後述の使用手順において記述する。

・GammaCorrection.java
時刻によって変化させる、画像全体の色味を調整するために用いた。

・StainedMain.java
入力画像の色味を現在時刻に応じて変化させるために使う。

2.3　用いた画像

2.4　使用手順

　まずは、ステンドガラス風にする処理に関して説明する。
プログラム実行時に第三引数として、ステンドグラスの細かさを指定するが、その値をコア数と呼ぶ。
以下、図 5 から図 10 は、コア数を徐々に大きくしていった結果である。粗いステンドグラス風の画像から細かくなるにつれて、入力画像に徐々に近づいていくのがわかる。

図 5　入力画像

図 6　コア数=200

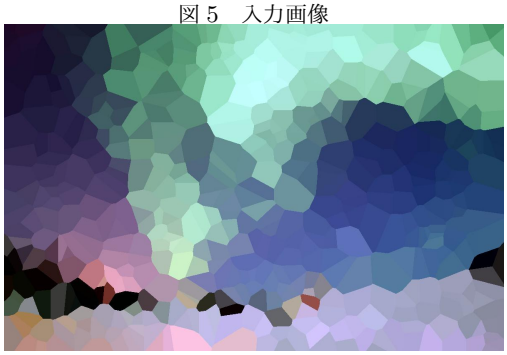

図 7　コア数=500

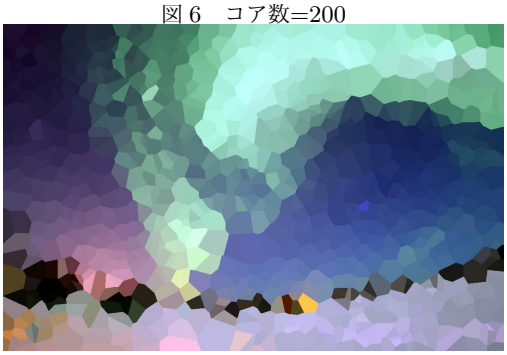

図 8　コア数=1000

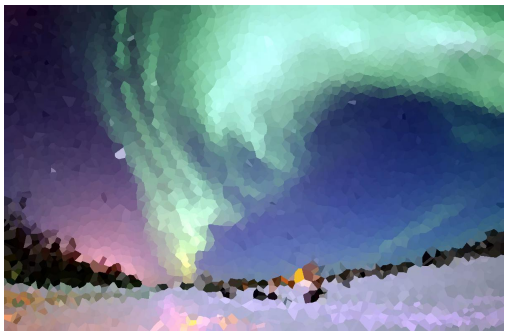

図 9　コア数=5000

図 10　コア数=10000

ステンドグラス風に画像を処理したあと、StainedMain.java によって、取得した現在時刻を使い、時間帯によって画像の色味が変化するようにした。
ここでは、コア数 50000 でステンドグラス風画像処理を施したものを使用した。

図 11　時刻が 19:00-24:00 または 0:00-6:00 の間

図 12　時刻が 6:00-16:00 の間

図 13　時刻が 16:00-19:00 の間

3　最後に

今回最も凝ったのは、作品 1 の水面に反射している画像の作成でした。180 度幾何回転させる段階までは順調に進んだのですが、左右反転させる処理が想像以上に難しく、座標を書いて試行錯誤を繰り返しました。普

段、画像処理ソフトのボタン一押しで簡単に行なっているせいか、こんな簡単な処理がなぜ容易に実現できないのか、大変苦しく感じられました。クロマキー処理や背景との合成処理を行っているためか、複雑に考えさせられました。この経験のおかげで、普段使っているものの仕組みに興味が湧いたり、身の回りのソフトウェアの便利さに感謝の気持ちを持つきっかけとなりました。

作品 1 では、ハワイと日没の地平線というお気に入りの画像を複数集めて、合成した、課題の趣旨に沿ったものを作成しました。しかし、どうしても今までの課題で行ってきた画像処理の枠を超えた画像処理を行いたく、一枚の画像のみを使用した作品 2 を作成しました。複数の画像を使っても良かったのですが、ステンドグラス風にしたかったため、画像処理の特性上、複数の画像を扱うメリットがなかったので一枚としました。
私は、ゲームの設定や、端末のデフォルトの壁紙などが時間帯によって変化していくのが、コンピュータ世界と現実の融合に感じられ、昔から好きでした。よって、作品 2 で時刻取得を組み込みました。自分でこのような挙動がプログラムできたのはとてもワクワクしました。

画像処理に加えて、本説明書作成のために、LaTeX に挑戦できたことも大きな成長となりました。日本語出力を可能にするためのセットアップが大変だったり、コーディングのようなことをしながら WYSIWYG ではない LaTeX を使って、画像の挿入位置を操作したりすることに慣れず、諦めかけたこともありました。しかし、今後論文の作成をすることなどを見越して、この経験は大変価値のあるものだったと考えています。

4　おまけ

図 14　理想形

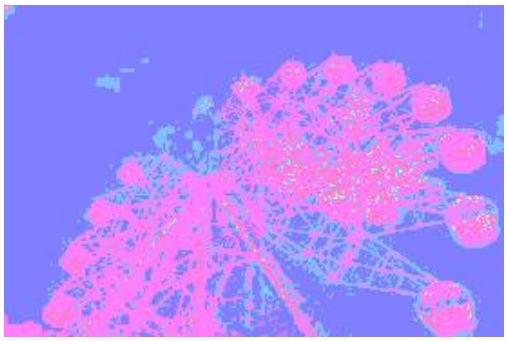

図 15　自作 BGColor.java 使用結果

図 16　図 15 の元画像

この課題のコンセプトを固めるにあたって、様々な画像処理を見てきました。その中で、見慣れた配色や画像処理ではありましたが、いざ自力でやってみようとするとどのように実装していいのか、大変考えさせられるものが多々ありました。その一つが図 14 です。

完成させられたらとても綺麗だろうと考え、挑戦してみたのですが、観覧車の中心が円形のようにピンクになっていることや、色の変化が縦横で共通しているものではなく、座標を使ってどのように色を調整していいのか分からず、自作ではなかなか実現できませんでした。
しかし、パステルカラーで行う、と決めて色を研究して作ったフィルターが BGColor.java というクラスなので、ここに記させていただきます。座標を使わずに実現する方法をこれから模索し、今後もチャレンジしてみたいと思います。