～Dlibを触ってみた～

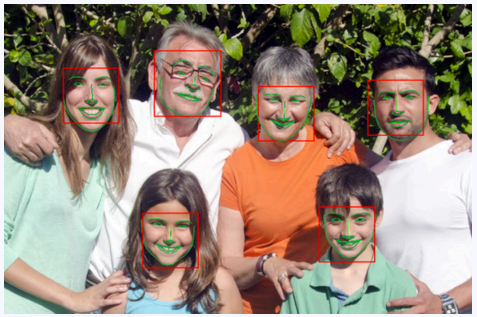
Dlibって何？

画像処理、機械学習などの機能をもったC++ライブラリ。
オープンソースで提供されていて、
ライセンスはBoost Software License 1.0という、
かなり緩いもの。
※商用利用でも著作権表示が要らないほど

Dlib
http://dlib.net/

んで何ができるん？

リアルタイム顔認識ができちゃいます。



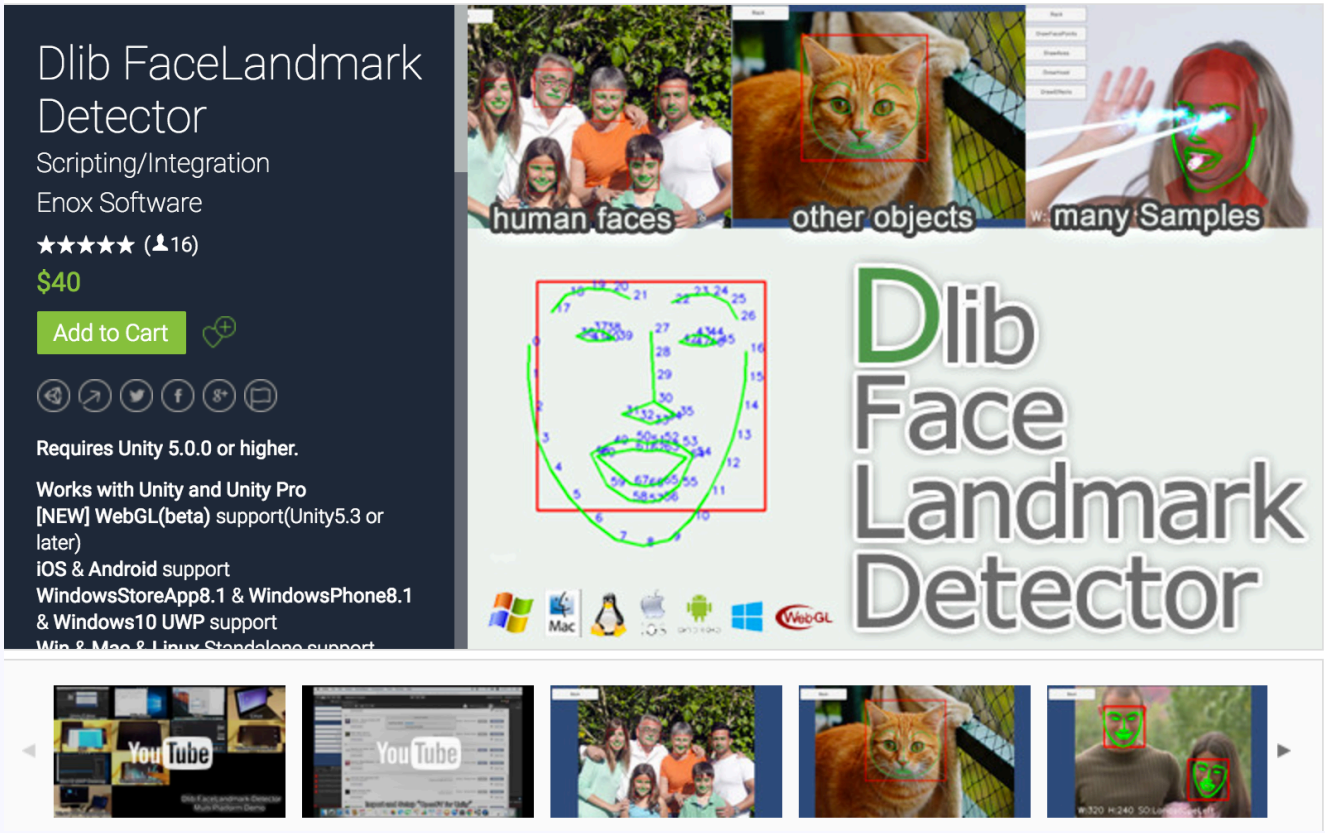
「snow」とか「B612」とか「MSQRD」とか「FaceRig」とか
のアプリで行われてるアレです。
「目でか」「顔痩せ」「アバター」「フェイススワップ」とか。
OpenCVでも顔認識できるけど、精度はDlibの方が優秀。
他にも色々できるみたいだけど、今回は顔認識のみに焦点を当てます。

Unityで使えるのかい？

アバターとか目でかとか、Unity使えたら表現の幅拡がるし
実装楽だよね。
Dlib自体C++でできてるので、動かそうと思えば動かせる。
ただ、C#とC++を繋ぐインターフェース（ラッパー）作らなきゃ。

まあ世の中にはそんなこと考える人は当然いるわけで、
AssetStoreに既にありました。
※Enox Softwareさんが販売してます。($40+tax)

自分でインタフェース作ることを考えたら$40なんて安い安い！



ってことで買いました。

「DlibFaceLandmarkDetector」の中には
ライブラリとラッパーとサンプルスクリプトが色々入っていますした。

ちなみに「OpenCV for Unity」というAssetがあれば、
さらに色々なことができちゃいます。
※これもEnox Softwareさんで販売してます。

「OpenCV for Unity」はすでに持っていたので、
「目からビーム」「口からハート」などのARサンプルも
動作させることができました。

体験してみよう！

DlibFaceLandmarkDetector/Samples/WebCamTextureSample/WebCamTextureSample.unity
DlibFaceLandmarkDetectorWithOpenCVSample/WebCamTextureARSample/WebCamTextureARSample.unity

今後Dlibを使ってやりたいこと

スマホで

- **顔認識でリアルタイムで顔にメイクさせる**
- **アバター的な変身**
- **「目でか」「顔痩せ」機能の実装**

をやりたい。

なんてことを書いているうちに、
「DlibLandMarkFaceDetector」を使ったデモプロジェクトが
AssetStoreに増えていました。

FaceMask（顔にマスクをかぶせる）
FaceSwap（顔交換）

の二つが増えてた。

前ページでやりたいって言った
- **顔認識でリアルタイムで顔にメイクさせる**
- **アバター的な変身**

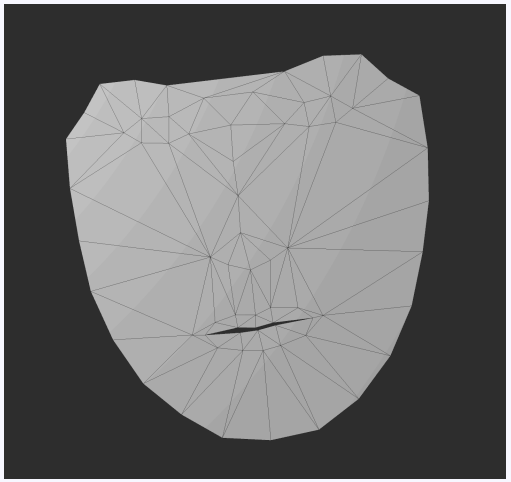
が叶いそうだ。

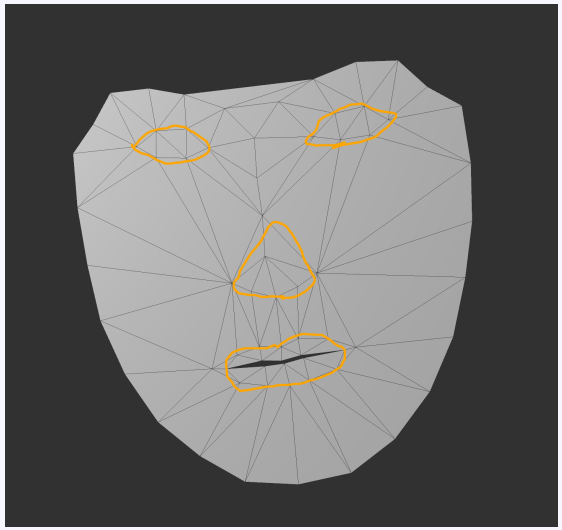
やってみた。

FaceMaskもFaceSwapも、
顔の形状のポリゴン用意して、顔認識した位置に合わせて
テクスチャ貼っているんだろうな

と予想はしていたが、
入っていたデータを見て、予想が確信に変わった。

こんなポリゴンデータが入っていました。
顔の形状のポリゴンですね。





これも体験してみよう！

FaceMaskSample/Scenes/WebCamTextureFaceMaskSample.unity
FaceSwapperSample/Scenes/WebCamTextureFaceChangerSample.unity
FaceSwapperSample/Scenes/WebCamTextureFaceSwapperSample.unity

残りの、
・「目でか」「顔痩せ」機能の実装
について。

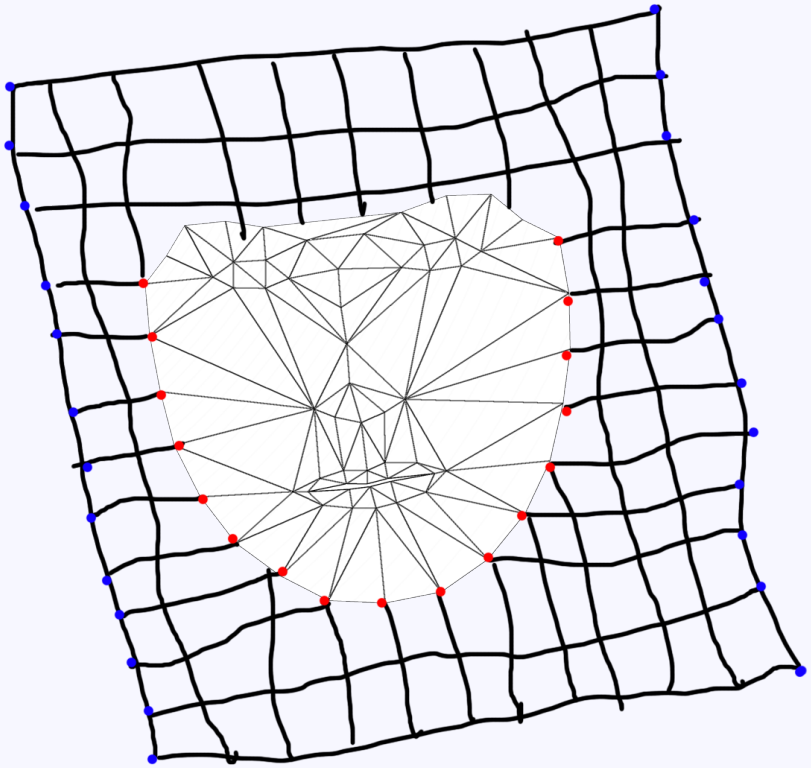
目でかはメッシュデータの目の部分をプログラムで動かせば
実現できるでしょう。

顔痩せについては、このメッシュデータでは実現できません。
顔のメッシュはカメラ画像の上に乗せているだけなので、
顔の輪郭を細めたら、下のカメラ画像が出てきてしまい、
ダブって表示されてしまうかと。

どうやって実現するのか悩みに悩んだ末、
予想がつきました。

顔のメッシュだけでなくその周りも含めたメッシュを用意し、
顔の輪郭の頂点だけを内側に細めてあげればいいんじゃないか。
※下の図の赤い頂点だけ動かし、青い頂点はそのままの位置をキープ
　その他の頂点は青と赤と影響を半々受けるようにしておけば

そうすれば、顔痩せさせても
下のカメラ画像は出てきませんね。



もう一つの課題。

「スマホで」

という部分。

Dlibの動作は重いです。
iPhone6でそのままサンプルを動かした場合、
10fpsも出ませんでした。

なので高速化の方法を考えて見た。

４案考えて見ました。

A.カメラの解像度を下げる（ただし見た目が悪くなる）

B.Aの代わりにDlibに渡す画像の解像度を下げる

C.２フレームに１回のように、顔認識処理の回数を減らす

D.カメラ画像のデータの扱いを工夫する

簡単に試せそうな、AとCをを試して見た所効果あり！

次に、
「A.カメラの解像度を下げる」の代わりに
「B. Dlibに渡す画像の解像度を下げる」を試そうと、
参考になりそうなサンプルソースがないかと見てると、
なんと、Bを実装しているサンプルがありました。
ってか「C. ２フレームに１回のように、顔認識処理の回数を減らす」
も実装されてるし。

試してみよう！

DlibFaceLandmarkDetectorWithOpenCVSample/OptimizationSample/OptimizationSample.unity

残すは、
「D.カメラ画像のデータの扱いを工夫する」
だけど、~~こっちはまだ未調査です。~~

調査しました。
WebCamTextureはOSからの
生データを持っていました。
なのでこれ以上の高速化は
難しそうです。

~~UnityのWebCamTextureの仕様とDlibのサンプル調べないとわからないけど、~~

~~もし、OSのカメラAPIから受け取ったデータを
WebCamTextureでGPU転送して、そのデータを使う場合は、
WebCamTexture経由でGPUからデータを受け取るとなっているなら
改善の余地あり。~~

~~他にもOpenCVでなんやかんやしてるとしたら、
これも改善の余地あり。~~

~~OSのカメラAPIから受け取ったデータを
直接Dlibに投げ込めるようにしたらきっと速くなるはず。~~

まとめ

・Dlib使えば顔認識もちょろいぜ。

・高速化にはそれなりの工夫が必要。

とはいえ、こういった後術が手軽に扱えるようになったのは嬉しい。
おバカアプリのアイディアが出まくります。

■参考サイト

Dlib
http://dlib.net/

NegativeMindException
https://blog.negativemind.com/2016/06/28/dlib-cpp-machine-learning-library/

UnityでDlibFaceLandmarkDetectorを利用した顔器官検出アプリ事始め
http://qiita.com/utibenkei/items/2f392d6fa00960f39e45

ご清聴ありがとうございました