Leo Schultz
11/4/14
Parallel Programming Project
Real-time Video Filtering

Abstract:
Photo and video filters are common in all modern graphics applications, some popular applications include Instagram and Photo Booth. Performance is most important when streaming real-time video. I found when processing images it was such fractions of a second in performance differences, it almost didn't make a difference if the program was optimized; therefore, I decided to try filtering a real-time video feed, where performance would be measured in the amount of frames per second the computer could output the video feed to the screen. I started with a simple RGBA bit flipping algorithm for the filters and used OpenGL to parallelize the algorithm, then tested the different results on both a 15 inch Macbook Pro with on board graphics and a 27 inch iMac with an NVIDIA graphics card.

Introduction:
The problem can be solved in many ways most of these all use OpenGL. You can use sequential bit flipping of the RGBA values; however, in real-time video the feed comes close to breaking. This may be acceptable for some very low power computing devices where you don't have access to a GPU or multi-core processor. However, it does not work very well for real-time video slowing the computers feed down to 3 to 7 frames per second the eyes refresh rate can easily catch the effects low performance and the CPU is highly taxed slowing down the computers resources.

I choose the middle performance implementation to limit scope and increase portability of the application. I used a sequential for loop to populate an array; however, I passed the array back to OpenGL to parallelize the output vector which filters the video feed. A slightly higher performance way of using OpenGL would have been to use GSLS shaders with the OpenGL library; however, GSLS has an intense learning curve and I was not able to get around to implementing GSLS because I had a large scope with using the video live video feed already.

The highest performance way which would have been extremely parallel, but only portable to NVIDIA machines is using the CUDA kernel to parallelize the bit flipping algorithm one more step by assigning separate blocks and then doing vector math on the addition. This could still use OpenGL; however, it would be a lower level implementation requiring a lot of time to implement and debug for an unportable result.

Implementation:

In the header file I declared the dependant frameworks, all of the global functions I was using and the global variables used. I am calling many dependent frameworks from open framework which has a scaffolding for OpenGL libraries and framework which I call from the “ofMain.h”.

main.cpp | ofApp.cpp | ofApp.h

```
#pragma once

#include "ofMain.h"

enum videoMode{
    STANDARD_MODE = 0,
    INVERTED_MODE,
    OUTLINE_MODE,
    TOON_MODE
};

class ofApp : public ofBaseApp{

    public:
        void setup();
        void update();
        void draw();

        void keyPressed(int key);
        void keyReleased(int key);
        void mouseMoved(int x, int y );
        void mouseDragged(int x, int y, int button);
        void mousePressed(int x, int y, int button);
        void mouseReleased(int x, int y, int button);
        void windowResized(int w, int h);
        void dragEvent(ofDragInfo dragInfo);
        void gotMessage(ofMessage msg);

        void setMode(videoMode newMode);

        ofVideoGrabber      vidGrabber;
        unsigned char *     videoInverted;
        ofTexture           videoTexture;
        int                 camWidth;
        int                 camHeight;
        int                 frameNum;
        string              currentModeStr;
        videoMode           currentMode;
};

```

In the “main.cpp” file I call the main function where I set the window size to the 2880 x 1800 pixel ratio of the Apple Retina display. I do this intentionally because the windows pixel width requires for increased processing of the bits. This gives me a display window controlled by OpenGL which is critical to the application. I also call an “ofRunApp();” where I declare the app to be in the “ofApp.cpp” file.

```
4   //========================================================================
5   int main( ){
6       ofSetupOpenGL(2880,1800,OF_WINDOW);         // <--------- setup the GL context
7
8       // this kicks off the running of my app
9       // can be OF_WINDOW or OF_FULLSCREEN
10      // pass in width and height too:
11      ofRunApp(new ofApp());
12
13  }
14
```

I then built a setup function for the ofApp file. In the setup I populate a vector with the video feed with the vidGrabber library and then allocate the pixels in the vector to the GL_RGB function.

```
3   //--------------------------------------------------------------
4   void ofApp::setup(){
5       camWidth        = 2880; // try to grab at this size.
6       camHeight       = 1800;
7
8       //we can now get back a list of devices.
9       vector<ofVideoDevice> devices = vidGrabber.listDevices();
10
11      for(int i = 0; i < devices.size(); i++){
12          cout << devices[i].id << ": " << devices[i].deviceName;
13          if( devices[i].bAvailable ){
14              cout << endl;
15          }else{
16              cout << " - unavailable " << endl;
17          }
18      }
19
20      vidGrabber.setDeviceID(0);
21      vidGrabber.setDesiredFrameRate(60);
22      vidGrabber.initGrabber(camWidth,camHeight);
23
24      videoInverted   = new unsigned char[camWidth*camHeight*3];
25      videoTexture.allocate(camWidth,camHeight, GL_RGB);
26      ofSetVerticalSync(true);
27  }
28
```

I then built an update function for the ofApp file. I first set a global background window then called in the video grabber function to update in real-time. I then set some local variable which grab pixels just to have a structure to flag off of to send back flag back to the vector structure that was called in the setup function. I create three filters by bit flipping pixels in an array and then send the array back to the video texture using GL_RGB.

```
29  //--------------------------------------------------------------
30  void ofApp::update(){
31
32      ofBackground(100,100,100);
33
34      vidGrabber.update();
35
36      int totalPixels = camWidth*camHeight*3;
37      unsigned char * pixels = vidGrabber.getPixels();
38
39      if( currentMode == INVERTED_MODE ){
40          for (int i = 0; i < totalPixels; i++){
41              frameNum ++;
42              videoInverted[i] = (int)ofRandom(150,210) - pixels[i];
43          }
44          videoTexture.loadData(videoInverted, camWidth,camHeight, GL_RGB);
45      }
46      else if( currentMode == OUTLINE_MODE ){
47          for (int i = 0; i < totalPixels; i++){
48              frameNum ++;
49              videoInverted[i] = 155 + pixels[i];
50          }
51          videoTexture.loadData(videoInverted, camWidth,camHeight, GL_RGB);
52      }
53      else if( currentMode == TOON_MODE ){
54          for (int i = 0; i < totalPixels; i++){
55              frameNum ++;
56              videoInverted[i] = 20 * pixels[i];
57          }
58          videoTexture.loadData(videoInverted, camWidth,camHeight, GL_RGB);
59      }
60  }
```

For a controller I built a keyPressed function I set on ‘spacebar’ to a reset function, I set the keys ‘1, 2 and 3’ to toggle variables which pass back to the update function to control the computations also for the draw functions to control the OpenGL rendering engine.

```
79  //--------------------------------------------------------------
80  void ofApp::keyPressed(int key){
81      if (key == ' '){
82          currentMode = STANDARD_MODE;
83          currentModeStr = "Reset Mode";
84      }
85      if( key == '1'){
86          currentMode = INVERTED_MODE;
87          currentModeStr = "1 - Inverted Mode";
88      }
89      if( key == '2'){
90          currentMode = OUTLINE_MODE;
91          currentModeStr = "2 - Outline Mode";
92      }
93      if( key == '3'){
94          currentMode = TOON_MODE;
95          currentModeStr = "3 - Toon Mode";
96      }
97      if (key == 'f'){
98          ofToggleFullscreen();
99      }
100 }
```

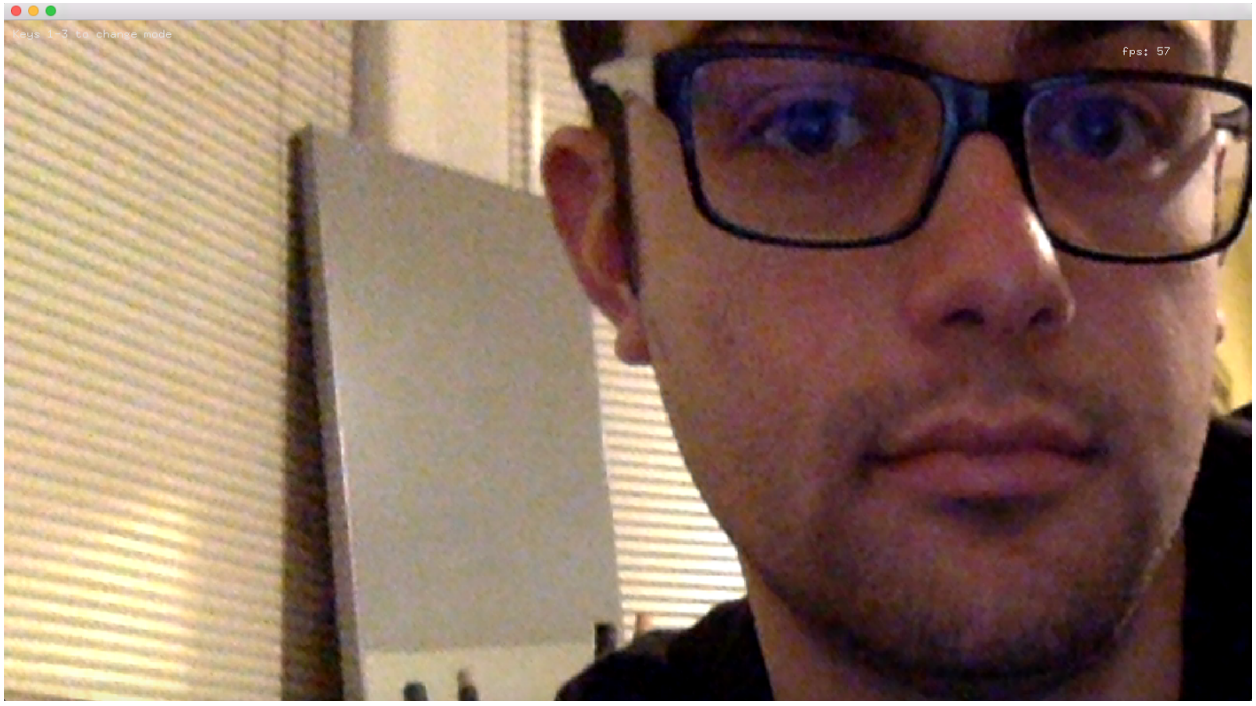
Last I build a “draw” function to render out the results created by the computations in the update function. I render out the standard video feed, which can be replaced with the textured feed, then the framerates and control values.

```
62  //--------------------------------------------------------------
63  void ofApp::draw(){
64      ofSetHexColor(0xffffff);
65
66      if( currentMode == STANDARD_MODE ){
67          vidGrabber.draw(0,0);
68      }
69      else if( currentMode == INVERTED_MODE || currentMode == OUTLINE_MODE || curren
70          videoTexture.draw(0,0,camWidth,camHeight);
71          ofDrawBitmapString("Pixel Counter: " + ofToString(frameNum),ofGetWidth()-1
72      }
73      ofSetColor(230);
74      ofDrawBitmapString("Keys 1-3 to change mode\n" +   currentModeStr, 10, 20);
75
76      ofDrawBitmapString("fps: " + ofToString((int)ofGetFrameRate()),ofGetWidth()-15
77  }
```

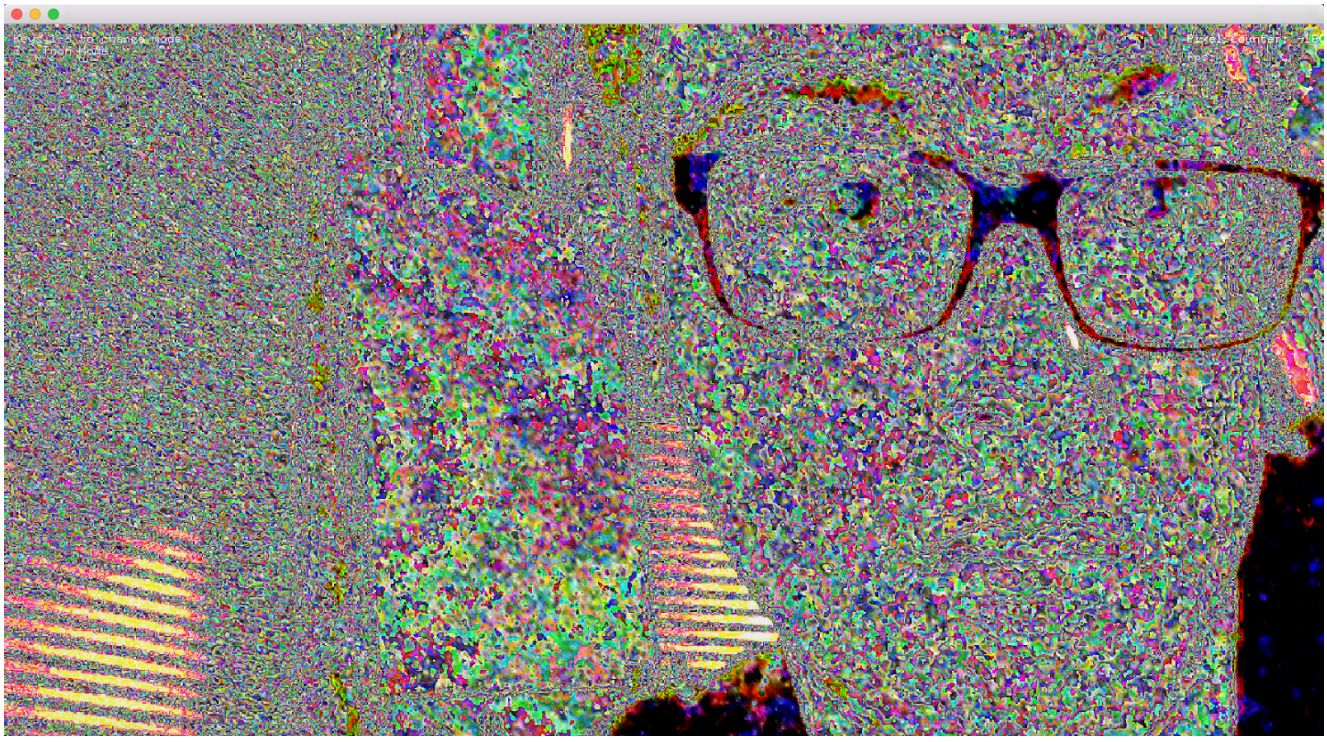
Results:
Standard Video Output at 60 fps

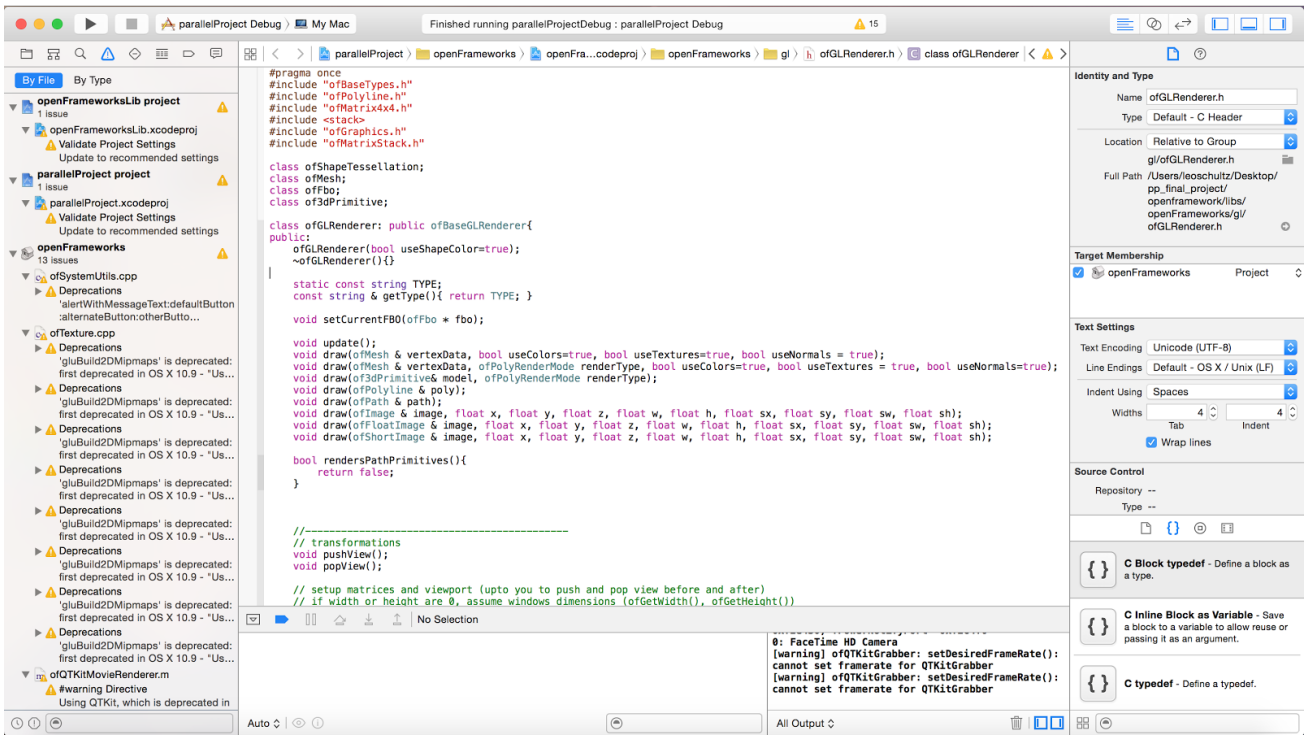
iMac w/NVIDA Graphics Card -
With filter after Parallelization = 22 fps
With filter before Parallelization = 7 fps

Macbook w/On-Board Graphics-
With filter after Parallelization = 15fps
With filter before Parallelization = 3 fps




Keys 1-3 to change mode
2 - Outline Mode




parallelProject Debug
My Mac
Finished running parallelProjectDebug : parallelProject Debug
parallelProject
openFrameworks
openFra...codeproj
openFrameworks
gl
ofGLRenderer.h
class ofGLRenderer
By File
By Type
openFrameworksLib project
1 issue
openFrameworksLib.xcodeproj
Validate Project Settings
Update to recommended settings
parallelProject project
1 issue
parallelProject.xcodeproj
Validate Project Settings
Update to recommended settings
openFrameworks
13 issues
ofSystemUtils.cpp
Deprecations
'alertWithMessageText:defaultButton
:alternateButton:otherButto...
ofTexture.cpp
Deprecations
'gluBuild2DMipmaps' is deprecated:
first deprecated in OS X 10.9 - "Us...
ofQTKitMovieRenderer.m
#warning Directive
Using QTKit, which is deprecated in
#pragma once
#include "ofBaseTypes.h"
#include "ofPolyline.h"
#include "ofMatrix4x4.h"
#include <stack>
#include "ofGraphics.h"
#include "ofMatrixStack.h"
class ofShapeTessellation;
class ofMesh;
class ofFbo;
class of3dPrimitive;
class ofGLRenderer: public ofBaseGLRenderer{
public:
ofGLRenderer(bool useShapeColor=true);
~ofGLRenderer(){}
static const string TYPE;
const string & getType(){ return TYPE; }
void setCurrentFBO(ofFbo * fbo);
void update();
void draw(ofMesh & vertexData, bool useColors=true, bool useTextures=true, bool useNormals = true);
void draw(ofMesh & vertexData, ofPolyRenderMode renderType, bool useColors=true, bool useTextures = true, bool useNormals=true);
void draw(of3dPrimitive& model, ofPolyRenderMode renderType);
void draw(ofPolyline & poly);
void draw(ofPath & path);
void draw(ofImage & image, float x, float y, float z, float w, float h, float sx, float sy, float sw, float sh);
void draw(ofFloatImage & image, float x, float y, float z, float w, float h, float sx, float sy, float sw, float sh);
void draw(ofShortImage & image, float x, float y, float z, float w, float h, float sx, float sy, float sw, float sh);
bool rendersPathPrimitives(){
return false;
}
//--
// transformations
void pushView();
void popView();
// setup matrices and viewport (upto you to push and pop view before and after)
// if width or height are 0, assume windows dimensions (ofGetWidth(), ofGetHeight())
No Selection
0: FaceTime HD Camera
[warning] ofQTKitGrabber: setDesiredFrameRate():
cannot set framerate for QTKitGrabber
[warning] ofQTKitGrabber: setDesiredFrameRate():
cannot set framerate for QTKitGrabber
Auto
All Output
Identity and Type
Name ofGLRenderer.h
Type Default - C Header
Location Relative to Group
gl/ofGLRenderer.h
Full Path /Users/leoschultz/Desktop/pp_final_project/openframework/libs/openFrameworks/gl/ofGLRenderer.h
Target Membership
openFrameworks Project
Text Settings
Text Encoding Unicode (UTF-8)
Line Endings Default - OS X / Unix (LF)
Indent Using Spaces
Widths 4 4
Tab Indent
Wrap lines
Source Control
Repository --
Type --
C Block typedef - Define a block as a type.
C Inline Block as Variable - Save a block to a variable to allow reuse or passing it as an argument.
C typedef - Define a typedef.

References:

http://lnx.cx/docs/opengl-in-xcode/
http://www.lighthouse3d.com/tutorials/glut-tutorial/
http://www.openframeworks.cc/
https://www.opengl.org/
https://www.opengl.org/resources/libraries/glut/
http://glew.sourceforge.net/

Appendix:

Project Source Code located at http://www.github.com/leorue/videofilter/