Core and Completion Reflection

I layered out the detection for the Ruby in my code, using RGB values stated in my for-loop statements.

```
12      for ( int row =0 ; row < image.height ; row++)  {
13        for ( int column = 0; column < image.width ; column++) {
14          totred = totred + (int)get_pixel(row,column,0); // get redd
15          totblue = totblue + (int)get_pixel(row,column,2); // get blue
16          totgreen = totgreen + (int)get_pixel(row,column,1); // get green
17          totint = totint + (int)get_pixel(row,column,3); // get luminocity
18          redness = (float)totred/(float)totgreen;
```

This is stated on lines 14 – 16 and on line 17 the totint returns the number of white pixels in each individual image. When the code is executed, it returns a value through the redness line (line 17) which uses both line 14 and 16 to return a value for how many pixels in the image that are red (ruby). When the ruby is present it gives a value over 1. When the ruby is not present it gives a value less than 1. These values also reflect within the false alarms and theft images within the assignment through the RGB values being able to detect the amount of red within the pixels even if there are other colors and darker shades to the image.

```
22      if(redness>1){
23           cout<<"Ruby is present"<<endl;
24         }
25         else{
26            cout<<"Ruby is stolen"<<endl;
27         }
```

On lines 22-27 there is an if statement which prints in the execution to state whether the Ruby is present or stolen. This is done though whether the redness is greater or lesser than 1.

The changes I made to the initial code

```
 7      int totred = 0;
 8      int totgreen = 0;
 9      int totblue = 0;
10      int totint = 0;
11      double redness = 0;
```

Added totgreen, totblue as integers since they are being stated as numbers when executed and double redness since it is analyzing the number of red pixels in each image.

```
14          totred = totred + (int)get_pixel(row,column,0); // get redd
15          totblue = totblue + (int)get_pixel(row,column,2); // get blue
16          totgreen = totgreen + (int)get_pixel(row,column,1); // get green
17          totint = totint + (int)get_pixel(row,column,3); // get luminocity
18          redness = (float)totred/(float)totgreen;
```

Tot blue and tot green were added into the for loop to determine the RGB values within each image. They were written out the same as the totint and totred that were already provided within the code and were given their color provided within the video_proc.hpp file. Redness was stated on line 18 which got the totred pixels over the number of green pixels.

```
29          cout<<"Total red ="<<totred;
30          cout<<"Total blue ="<<totblue;
31          cout<<"Total green ="<<totgreen;
32          cout<<"Total int ="<< totint;
33          cout<<" Redness ="<<redness<<endl;
```

Cout statements were added on line 30, 31 and 33 to execute the code with the added blue, green and redness statements.

What my code is lacking, what can be improved

My code can successfully determine the number of red pixels in a non-moving series of images, but it cannot determine moving or multiple amounts of red pixels on different areas of an image. There is no alert to state whether there are multiple red pixels on images. Another thing that can be improved on this code is determining whether there are different pixels with different colors in each image. This can be done by adding different double statements (like redness) to determine different colors. These are things that can be improved on this code to determine multiple aspects of images that can be run through the code and to provide a more robust detection system within it.