Dynamic Compositing Report

This report investigates and present the problem of colour matching between the natural content and synthetic content and dynamic composting of natural content. The report also investigates on different type of colour correcting algorithm to see which produces a better result.

In the Dynamically Composite Content (DCCS) :-

Colour grading is the process of matching of colour between different shots of the image for aesthetic purposes. In post-production purposes is digitally manipulated to have consistent colour reproduction.

The main code used in the sketch 2 for colour transfer and couler correcting is as follows :-

```
void applyScalingsFromTo(PImage r, PImage t) {
 float [] s = new float[3];
 float userScaling = 0.4;

 float [] mRef = getMeansFrom(r);
 float [] sdRef = getVariancesFrom(r);

 float [] mTarget = getMeansFrom(t);
 float [] sdTarget = getVariancesFrom(t);

 for (int j = 0; j < 3; j += 1) {
  s[j] = 1 - userScaling + userScaling * sdRef[j] / sdTarget[j];
 }

 //println(mRef); println(sdRef);
 //println(mTarget); println(sdTarget);

 t.loadPixels();

 for (int i = 0; i < t.pixels.length; i += 1) {
  float [] cp = rgb2Lab(red(t.pixels[i]), green(t.pixels[i]), blue(t.pixels[i]));

  for (int j = 0; j < 3; j += 1) {
   cp[j] = s[j] * (cp[j] - mTarget[j]) + mTarget[j];
  }

  float [] rgb = lab2RGB(cp[0], cp[1], cp[2]);
  t.pixels[i] = color(rgb[0], rgb[1], rgb[2]);
 }

 t.updatePixels();
}
```

This code has a function applyScalingsFromTo() which takes in two images as parameters. PImage R as the background image (that exact frame of the video in that time) and PImage T the synthetic content image(that exact frame of the video in that time).

$$
\begin{aligned}
L_{\text{target}} &= s_L \left(L_{\text{target}} - \bar{L}_{\text{target}}\right) + \bar{L}_{\text{target}} \\
\alpha_{\text{target}} &= s_\alpha \left(\alpha_{\text{target}} - \bar{\alpha}_{\text{target}}\right) + \bar{\alpha}_{\text{target}} \\
\beta_{\text{target}} &= s_\beta \left(\beta_{\text{target}} - \bar{\beta}_{\text{target}}\right) + \bar{\beta}_{\text{target}}
\end{aligned}
$$

To implement this we have code

```
for (int i = 0; i < t.pixels.length; i += 1) {
   float [] cp = rgb2Lab(red(t.pixels[i]), green(t.pixels[i]), blue(t.pixels[i]));

   for (int j = 0; j < 3; j += 1) {
     cp[j] = s[j] * (cp[j] - mTarget[j]) + mTarget[j];
   }

   float [] rgb = lab2RGB(cp[0], cp[1], cp[2]);
   t.pixels[i] = color(rgb[0], rgb[1], rgb[2]);
 }
```

We store the rgb values of each pixel from the PImage T in cp as an array. After that we we change the alpha and colour of the target PImage T by correcting it against the average mean of the target image with the background image. This process is done three times for each values of R, G and B. Then we store each of the cp array containing colour corrected RGB values into a RGB colour space. Then we update the RGB colour space of that pixel in the target image.

<u>The resultant image by using this default method gives different results for different lighting condition or sources. Some of them are :-</u>

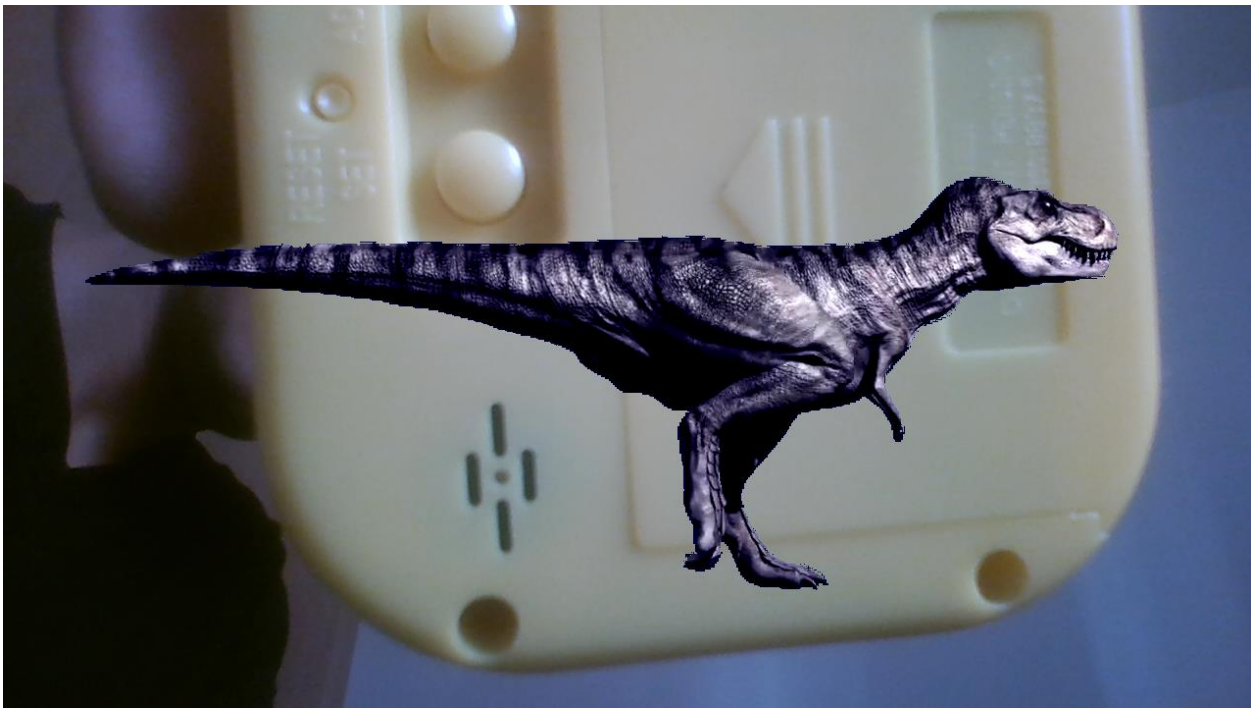
1.

There is a blue/purple tint when using in a bright but morning room. The colour correcting is aggressive towards the blue RGB colour space with a mix of slight red that is been detected from the scene.

2.

An almost total blue background results into slit change in color of the synthetic content. The difference is very little, but red levels of the synthetic material has been reduced. This experiment can assert a hypothesis that the default colour correcting algorithm when using in live camera feed, processes the blue RGB colour space very aggressively.

3.



When having a majority White/yellow with some blue in the background, we see the synthetic content changing colour and appearing more white but still having the blue tint on it. The red and blue RGB colour space is properly matched but the blue RGB colour space is dominant. This confirms the hypothesis that the default algorithm is very aggressive on the blue RGB colour space. And the morning day light tend to have more blue elements in it.

Now to recheck and test the hypothesis to see if the camera content is not the factor for generating more blue in the synthetic content, a prerecorded natural video will be used. And the results are :-

4.

This image displays a good amount of color correction on the synthetic image while having high dark areas in the natural image. The bluish tint is not observed.

5.

Another natural image when used asserts that the video feed from camera has a higher number of blue pixels.

6.

When using a static image with predominantly green pixels. Here the emergence of blue in the synthetic colour corrected image is again observed.

Hence to make the algorithm more accurate, some modifications were made to the code. These were :-

```
for (int j = 0; j < 3; j += 1) {

    if(j==0)
     cp[j] = (s[j] * (cp[j] - mTarget[j]) + mTarget[j])-5;
    else if(j==1)
      cp[j] = (s[j] * (cp[j] - mTarget[j]) + mTarget[j])-5;
    else if(j==2)
      cp[j] = (s[j] * (cp[j] - mTarget[j]) + mTarget[j]);

  }
```

Adjusting the amount of change in the relative RGB values for corresponding RGB values in target results in the following :-

7.

The same room the same amount of light but with a white background results in close to the same colour as of the synthetic material originally was. There is colour correction applied and the image has a slight darker and more greenish tint to it.

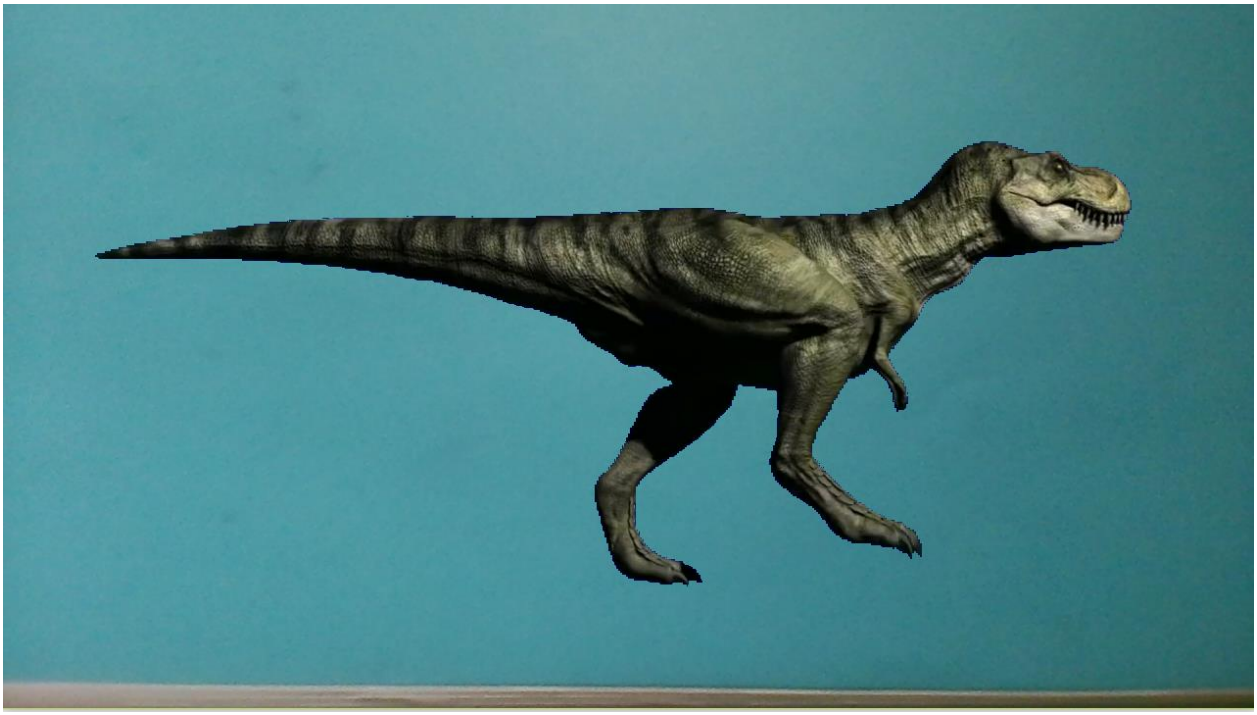
8.

When using a full black background no amount of colour correction is applied and the synthetic material is presented almost the same. There is a bit of green tint but it can be possible because of the code :-

```
float [] mTarget = getMeansFrom(t);
 float [] sdTarget = getVariancesFrom(t);
```

which gives the whole mean for the colour space a green value as the synthetic content is green and after colour correcting is applied to it again.

9.



Having a blue colour as background the colour correction is done resulting into a near perfect synthetic material for the background. The amount of blue is less on the synthetic content though.

10.

With having a bright lit room, and significant shade of red,, it is been observed the colour correction is too the point indicating the updated algorithm is more suitable in this scenario . But the synthetic material and the background are relatively different and can be distinguished as the synthetic material being superimposed on the background. This might happen because of the type of synthetic material been used.

11.

When having a large amount of red colour space in the background the synthetic material is colour matched successfully to the background. This totally indicates that the updated algorithm is quite good in these scenarios.

12.

When using a natural video as a source the synthetic material generated by colour correction using the updated algorithm has very little difference from the default one.

13.

When using another natural video as a source and comparing it to the previous experiment with the same source it was noted that the synthetic image generated was clearer, and less bluish than the image generated using the default algorithm. Giving a more definitive proof that the algorithm has more features to be added and can be made more efficient and accurate.

The major drawback of the DCCS is the slow process and subsequently slow frame rate and glitches like frame drop of the synthetic material. Which makes use of the algorithm for real time usages very less likely.

When using JavaScript for colour correcting in a Web based Application:-

The major advantage this has is the quick response time and simplicity to work with. The following image is from the Web Application –

Its quite clear that the JavaScript based HTML application is lot faster and efficient in colour correcting the same materials. The colour correction is also quite accurate.

Conclusion:-

The advantages JavaScript has over DCCS is faster compute time and better colour reproduction when comparing default algorithms. The advantages of DCCS is it is a lot more programmable environment and algorithms can be adjustable according to the users needs.