Reducing overexposure and underexposure during image-guided surgery

October 29, 2023

Kai Kitagawa-Jones
student ID: i6275822

1 Abstract

When capturing images during surgery, it is often the case that many areas are either underexposed or overexposed. This happens because of the combination of strong lighting and deep wounds. In this report, I look into techniques to process video in real-time to reduce these issues. In order to be run in real-time, the techniques will be optimized so that each pixel can be processed in parallel on the GPU.

2 Introduction

When operating in deep wounds that do not receive sufficient light, the exposure of the camera has to be increased. However, when the exposure is increased, parts of the wound that are closer to the surface are often overexposed. This makes it challenging to capture images that are well-exposed. Therefore, I aim to answer the following:

- What approaches can be used to reduce underexposure and overexposure?
- How do they compare to each other?
- How can they be implemented in practice?

One approach to solve this problem is to adjust the brightness of each pixel by applying a transformation. In order to reduce underexposed areas, the transformation will simply increase the brightness of the dark pixels. However, for overexposed pixels, this technique is less effective. This is because as the camera being used outputs a 24 bit RGB (red, green, blue) colour, and when overexposed, very little information about the original colour remains, and reducing the brightness will simply turn the pixels gray.

Another approach is exposure fusion [1, 2]. This is a technique that can be used to avoid loosing information in areas that are not well exposed. This technique merges a number of frames taken at different exposure levels into one image. If done correctly, the resulting image will contain the best parts of each image, greatly reducing both underexposure and overexposure.

In [1], the image is divided into blocks. Then, for each block of the output image, one of the corresponding blocks from the input images is selected. The selection is based on entropy, with the block with the highest value being selected. Once all blocks have been selected, these are blended together using bilinear interpolation. On the other hand, in [2], the images are combined through pyramid blending, and the selection happens at a pixel level rather than a block level. Here, the selection is based on a weight computed for each pixel. In order to compute this value, three parameters are used: contrast, saturation, and well-exposedness.

All experiments performed in this report were carried out using the DHM system made by I-Med Technology, with custom software that uses OpenGL [3] for image processing. This is different to the original software, that uses CUDA [4], making the custom software compatible with more hardware configurations. It is also well optimized, allowing it to be run on low performance systems.

The software also implements the image processing pipeline using a modular system, allowing image processing steps to be written and tested individually. These steps can then be combined in any order, making it easy to experiment with new techniques. Using this, a filter that increases the contrast of colours in the red spectrum was implemented. This is done by shifting the hue of the pixels away from red. This results in an image where it is easier to see small differences between various shades of red. See Appendix A for details.

3 Methods

3.1 Transformation function

The most common technique used to adjust the brightness of each pixel is histogram equalization [5]. In order to use this technique with images captured during surgery, it is necessary to specify an area of interest. This is because over half the image may consist of extremely bright or extremely dark areas that are of no interest to the user. However, for the use case discussed in the report, the region of interest is hard to define. In addition, histogram equalization results in the loss of colour depth. Because of these reasons, histogram equalization is not a good approach to solve this problem.

An approach that will not need a region of interest specified is gamma correction [6]. However, this offers little control over the overall shape of the transformation function. Therefore, I propose a transformation function inspired by gamma correction that combines a linear function ($y = x$) and a monomial ($y = x^b$):

$$y = 1 - (1-a)(1-x) - a(1-x)^b \tag{1}$$

- y ($0 \le y \le 1$): The brightness of the pixel after transformation is applied.
- x ($0 \le x \le 1$): The brightness of the pixel before transformation is applied.
- a ($0 \le a \le 1$): $1 - a$ has a similar effect to the γ parameter in the gamma function.
- b ($0 \le b$): This controls how much the brightness of the dark pixels is increased.

The transformation function is designed so that when $a = 0$, the monomial $-(1-x)^b$ does not contribute to the final output. Because of this, only the linear function $-(1-x)$ affects the output, leaving the input unmodified. On the other hand, when $a = 1$, only the monomial affects the output of the transformation. Any value between 0 and 1 represents a combination of both the linear function and the monomial.

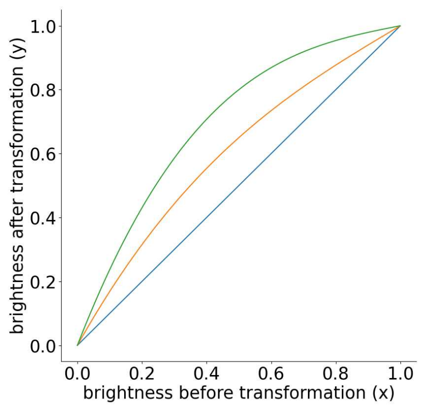

Figure 1: Transformation function with $b = 3$ and $a = 0.8$ (top), $a = 0.4$ (middle), $a = 0$ (bottom)

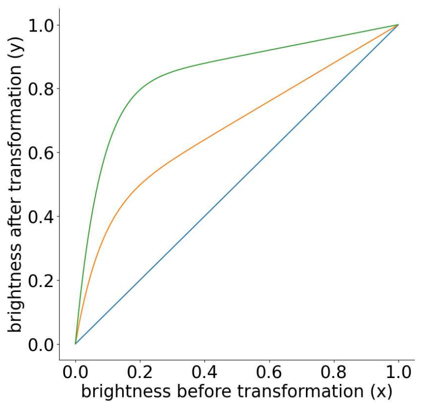

Figure 2: Transformation function with $b = 13$ and $a = 0.8$ (top), $a = 0.4$ (middle), $a = 0$ (bottom)

Visit `desmos.com/calculator/ewzhqjhkpc` for an interactive demo of the transformation function.

In order to reduce the underexposed parts of an image, a relatively high b value of around 13 is ideal, as it does not affect the lighter pixels as much. The parameter a can then be used to adjust how much the brightness is increased, where the larger the value of a, the more the brightness of each pixel is increased. In addition, this transformation function leaves the brightness unmodified when $a = 0$.

3.2 Exposure fusion

Images with different exposures are needed to perform exposure fusion. Because real-time video processing is the focus of this report, each frame of the video will alternate between two exposures: one with high exposure, and one with low exposure. For every frame, exposure fusion will be performed using the current frame and the previous frame.

For both the current and previous frame, a weight for every pixel is calculated. This is done by estimating two parameters: saturation and well-exposedness. These measures are based on [2], with the contrast parameter removed. This parameter was removed because a Laplacian filter is used, requiring that each operation access multiple pixel values, reducing performance.

- Saturation: By calculating the standard deviation between the R, G, and B channels.

  $$p_s = \sqrt{(\bar{c} - R)^2 + (\bar{c} - G)^2 + (\bar{c} - B)^2} \tag{2}$$

  Where R, G, and B are the channels of the RGB colour, and $\bar{c} = \frac{R+G+B}{3}$.

- Well-exposedness: When comparing two pixels, the pixel with a brightness closer to 0.5 is desired. Therefore, this measure is calculated by how close the average of the R, G, and B channels is to 0.5, based on a Gauss curve.

$$p_e = \exp\left(\frac{(\bar{c} - 0.5)^2}{2\sigma^2}\right) \tag{3}$$

  $\sigma = 0.2$ was used in this implementation.

After estimating the two parameters p_s and p_e, the weight of each pixel is calculated:

$$W = p_s^{w_s} \cdot p_e^{w_e} \tag{4}$$

The values of w_s and w_e control how much each parameter contributes to the final weight. In this implementation, $w_s = 0.01$ and $w_e = 0.1$, however, further experiments should be performed with different weights.

Finally, the pixels from each frame are combined based on the weights of each pixel W_1 and W_2:

$$\begin{aligned} R_{\text{fusion}} &= R_1 \cdot \frac{W_1}{W_1 + W_2} + R_2 \cdot \frac{W_2}{W_1 + W_2} \\ G_{\text{fusion}} &= G_1 \cdot \frac{W_1}{W_1 + W_2} + G_2 \cdot \frac{W_2}{W_1 + W_2} \\ B_{\text{fusion}} &= B_1 \cdot \frac{W_1}{W_1 + W_2} + B_2 \cdot \frac{W_2}{W_1 + W_2} \end{aligned} \tag{5}$$

4 Implementation

Both approaches are designed to be run on a GPU using OpenGL compute shaders. In addition, all operations can be run in parallel per-pixel, without information from the surrounding pixels. In order to capture video and control the camera settings, libuvc [7] was used.

4.1 Transformation function

Based on Equation 1, the transformation function was implemented with the following GLSL code:

```
float transform(float x) {
    return 1 - (1 - a) * (x - 1)
              - a * pow(1 - x, b);
}
```

For each pixel, the following operations are carried out:

1. The RGB values for each pixel are converted into HSL (hue, saturation, lightness) values.

   ```
   vec3 hsl_in = rgb2hsl(rgb_in);
   ```

2. The transformation function is applied to the lightness value L.

   ```
   float hsl_out = vec3(
       hsl_in.x, hsl_in.y,
       transform_function(hsl_in.z)
   );
   ```

   Note: `hsl_in.x`, `hsl_in.y`, and `hsl_in.z` correspond to the H, S, and L values.

3. The new RGB values are calculated based on the HSL values obtained in the previous step.

   ```
   vec3 rgb_out = hsl2rgb(hsl_out);
   ```

Step 1. and 3. are the most computationally expensive parts of the operations. In addition, these steps are also somewhat unnecessary since only the lightness value is used. Therefore, a way to optimize these steps should be looked into.

4.2 Exposure fusion

Based on Equation 2, 3 and 4, the weight for each pixel is calculated by the following GLSL code:

```
float weight(vec3 c, float ws, float we) {
    float avg = (c.r + c.g + c.b) / 3.0;
    float ps = sqrt(
        pow(avg - c.r, 2.0)
      + pow(avg - c.g, 2.0)
      + pow(avg - c.b, 2.0)
    );
    float pe = exp(-pow(avg - 0.5, 2) / 0.08);
    return pow(ps, ws) * pow(pe, we);
}
```

For each pixel c_1 and c_2 from the two input frames, the following operations are carried out:

1. The weight for each pixel is calculated.

   ```
   float w1 = weight(c1, ws, we);
   float w2 = weight(c2, ws, we);
   ```

2. The weights are normalized.

```
float tot = w1 + w2;
float n1 = tot == 0 ? 0.5 : w1 / tot;
float n2 = tot == 0 ? 0.5 : w2 / tot;
```

`tot == 0` is checked because both `w1` and `w2` can be 0, causing division by 0. Because this leads to undefined behaviour, the weights are manually normalized to 0.5.

3. The output colour is calculated based on the normalized weights.

```
vec3 co = c1 * n1 + c2 * n2;
```

5 Results

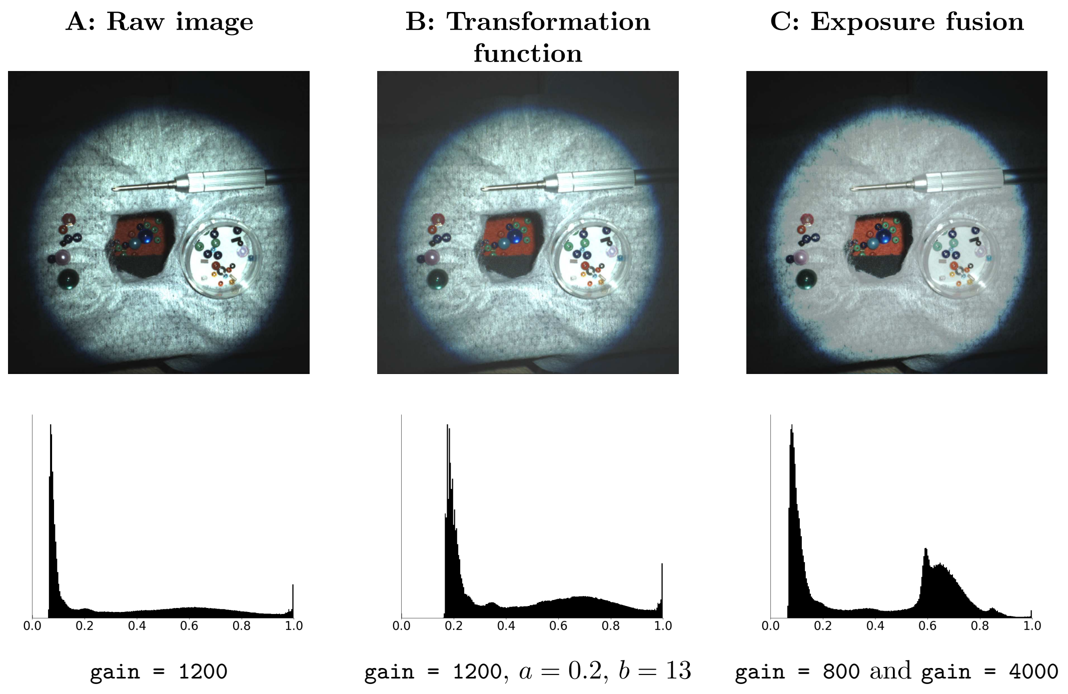

Figure 3: The results obtained from a test setup.

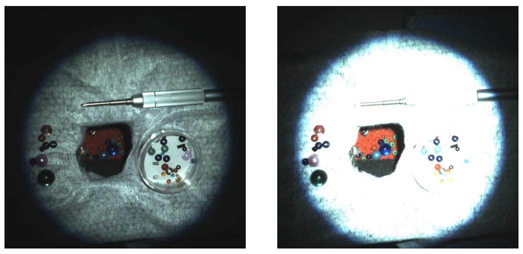

Figure 4: The frames used in Figure 3, column C

6 Discussion

6.1 Transformation function

As can be seen in Figure 3, column B, passing the frame through the transformation function results in an increase in the brightness of the red area at the centre of the image compared to the raw image. The histogram also shows how the dark values have been shifted towards lighter values.

However, because this is a simple transformation function, the overexposed areas, such as the plastic dish, remain the same. In addition, the transformation results in desaturation over the entire image.

6.2 Exposure fusion

Looking at Figure 3 column C, it is clear that there is an improvement compared to both the raw image and the transformation function. The brightness of the red area at the centre of the image has been increased even more compared to the transformation function. In addition, the overexposure of the plastic dish has been greatly decreased.

The two frames used for exposure fusion are shown in Figure 4. The information from the darker frame was used in brighter areas, such as the plastic dish and surface cloth, while information from the brighter frame was used for dark areas, such as the centre of the image. In addition, by combining the two frames, details such as the small beads can become clearer.

Another important benefit of this method is that there is no loss of colour information in the form of desaturation. When using the transformation function, colours are shifted towards brighter values, leading more pixels to have the same colour [8]. This is especially important when using 24 bit colours, where each channel is assigned only 8 bits and the number of colours that can be expressed is relatively low.

7 Conclusion

As shown in the previous section, both the transformation function and exposure fusion can reduce the underexposure of an image or video. It was also shown that in addition to underexposure, overexposure can also be reduced using exposure fusion. Because of this, both techniques can be used for situations where there is often a large contrast between light and dark areas, such as during surgery, although, exposure fusion produces a better result.

A benefit of the transformation function is that the single parameter a in Equation 1 can be used to control how much the brightness of the dark colours is increased. The second parameter b, can be set in advance according to the use case. This allows the transformation function to be intuitively controlled by people who may not necessarily understand the implementation details of this function. In addition, compared to exposure fusion, the transformation function has a much simpler and higher performance implementation.

Exposure fusion can also be controlled by changing the difference in exposures between the two frames. A small difference will produce an output that is close to simply taking the average between the two frames, while a large difference can produce an output where some areas of the output are only taken from one input frame.

8 Next steps

8.1 Exposure fusion anomalies

In some cases, when looking at a surface with a smooth brightness gradient, exposure fusion outputs an image where the brightness changes rapidly at a certain point. By adjusting the two weights w_s and w_e, it is possible to reduce this effect, however, the issue remains. Although this is not a critical issue, it is necessary to look into this further.

8.2 Auto exposure

In order for exposure fusion to be used practically, it is also necessary to adjust the exposure of the camera automatically (AE). There are many implementations of AE for both single-shot image capture [9, 10] and video capture [11]. However, there are no implementations of AE designed for exposure fusion.

In order to effectively perform exposure fusion, the AE has to select two exposure values that provide the most information about the view when combined. In addition, the target hardware does not have a light meter, which is used in many cameras to perform AE [9]. Because of this, the two exposure values will have to be chosen based on the histogram, or other information obtained from the frames captured.

References

[1] A. K. Bachoo. “Real-time exposure fusion on a mobile computer”. In: Dec. 2009. URL: `http://hdl.handle.net/10204/3857`.

[2] T. Mertens, J. Kautz, and F. Van Reeth. “Exposure Fusion: A Simple and Practical Alternative to High Dynamic Range Photography”. In: *Computer Graphics Forum* 28.1 (Mar. 2009), pp. 161–171. DOI: `10.1111/j.1467-8659.2008.01171.x`. URL: `https://doi.org/10.1111/j.1467-8659.2008.01171.x`.

[3] John Kessenich, Graham Sellers, and Dave Shreiner. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.5 with SPIR-V.* 9th ed. Addison-Wesley Professional, 2016. ISBN: 9780134495491.

[4] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. *CUDA, release: 10.2.89.* 2020. URL: `https://developer.nvidia.com/cuda-toolkit`.

[5] Rafael C Gonzalez. *Digital image processing.* Pearson education india, 2009.

[6] Poynton Charles. *Digital Video and HD : Algorithms and Interfaces.* Vol. 2nd ed. Morgan Kaufmann, 2012. ISBN: 9780123919267. URL: `https://mu.idm.oclc.org/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=453857&site=ehost-live&scope=site`.

[7] libusb developers. *libuvc.* Version 0.0.6. Sept. 21, 2017. URL: `https://libuvc.github.io`.

[8] James W. Walters and Ronald S. Harwerth. "The mechanism of brightness enhancement". In: *Vision Research* 18.7 (1978), pp. 777–779. ISSN: 0042-6989. DOI: `https://doi.org/10.1016/0042-6989(78)90116-5`. URL: `https://www.sciencedirect.com/science/article/pii/0042698978901165`.

[9] Jarosław Bernacki. "Automatic exposure algorithms for digital photography". In: *Multimedia Tools and Applications* 79.19-20 (Jan. 2020), pp. 12751–12776. DOI: `10.1007/s11042-019-08318-1`. URL: `https://doi.org/10.1007/s11042-019-08318-1`.

[10] JiaYi Liang, YaJie Qin, and ZhiLiang Hong. "An Auto-exposure algorithm for detecting high contrast lighting conditions". In: *2007 7th International Conference on ASIC.* 2007, pp. 725–728. DOI: `10.1109/ICASIC.2007.4415733`.

[11] Marc-André Bégin and Ian Hunter. "Auto-Exposure Algorithm for Enhanced Mobile Robot Localization in Challenging Light Conditions". In: *Sensors* 22.3 (Jan. 2022), p. 835. ISSN: 1424-8220. DOI: `10.3390/s22030835`. URL: `http://dx.doi.org/10.3390/s22030835`.

A Red contrast enhancement

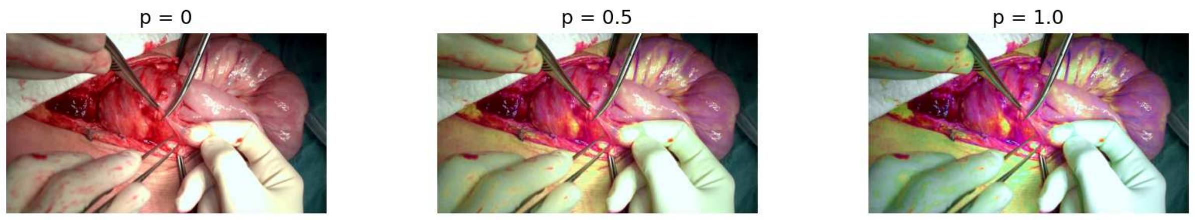

Figure 5: The effects of red contrast enhancement

The p value ($0 \leq p \leq 1$) controls how strong this effect is a way that when $p = 0$, the colours are unmodified. The following histograms show how the colours have been shifted away from red (`hue = 0`):

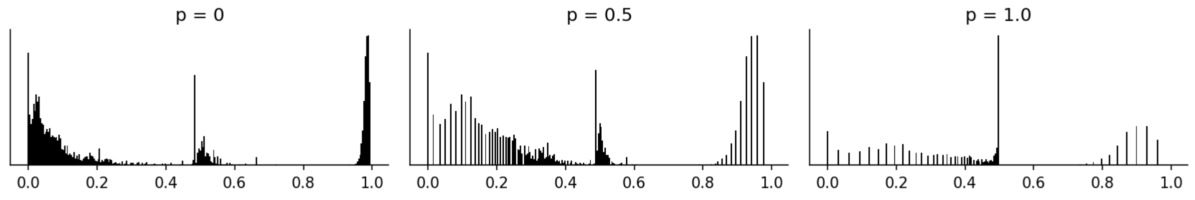

Figure 6: Hue histogram of the images in Figure 5

As can be seen in Figure 5, with $p = 0.5$, it is easy to see the difference between different shades of red. This filter will be looked into further, and improved so that colours that are not in the red spectrum are not distorted as much.