Near Infrared Imaging

Dajana Vlajic Florian Zimmermann Josselin Vallee Rebecka Rőnnbäck
Sahand Kashani Yann Perret

June 3, 2016

1 Introduction

Silicon sensors commonly used for photographic purposes are naturally sensitive to a wide band of the light spectrum. This band contains both *visible* light (RGB, 390-700 nm), and *Near-infrared* light (NIR, 700-1100 nm).

As the human visual system is only sensitive to visible light, camera manufacturers have traditionally covered their sensors with a filter that blocks out all NIR light, therefore preventing radiation outside the RGB band from affecting the sensor's output.

However, recent work has shown that combining the information contained in both the RGB and NIR bands of the light spectrum is beneficial for various image enhancement algorithms, as it allows one to extract more accurate information about the scene. For example, the combined use of RGB- and NIR-based imagery was used in [1] for skin smoothing, in [2] for detecting shadows in scenes, and in [3] for image dehazing.

In this project, we sought to verify some of these results experimentally. To this end, we put together a custom camera setup and implemented a few of the aforementioned image enhancement algorithms. We present our setup and results below.

2 Image acquisition

The main problem with NIR imaging is actually the acquisition process: The image processing algorithms presented earlier take as inputs both an RGB and an NIR image. These images must have been taken *simultaneously* to avoid any temporal shift in the scene, and need to be *aligned* in such a way that a pixel in the RGB image and the corresponding pixel (located at the same coordinates) in the NIR image both measure the same position within the captured scene.

Capturing 2 images simultaneously is not easily possible with a single camera, so the acquisition of distinct RGB and NIR images is generally performed with a dual-camera setup. Although the cameras are facing the exact same scene, we will always observe a slight shift between the 2 captured images. This is normal as mechanical constraints make it impossible for the 2 cameras to capture the exact same light beams.

Unfortunately, this creates an issue for dual-picture image processing algorithms. Indeed, the algorithms presented earlier are adaptive and take advantage of locality within a scene for better results. The fact that the RGB and NIR images are not aligned causes incorrect "local" information to be used, therefore resulting in unpleasant outputs.

So the first step of any NIR imagery is to solve the acquisition problem.

2.1 Hardware Setup

Obtaining 2 identical cameras that are available with and without an NIR filter is more difficult than it seems. Thankfully such cameras exist for the Raspberry Pi (RPi) embedded system, so we chose this platform for our setup. These camera are the standard Raspberry Pi camera, and the NoIR camera.

The RPi can only be connected to one camera, so we will use 2 such devices in order to connect our RGB and NIR cameras.

2.1.1 Dual-camera mount

We 3D-printed a custom support for mounting the 2 cameras side-by-side. Special care was taken to ensure that the 2 cameras are aligned, and are mechanically mounted the closest possible to each other. This is done to reduce the shift between the 2 cameras the most possible.

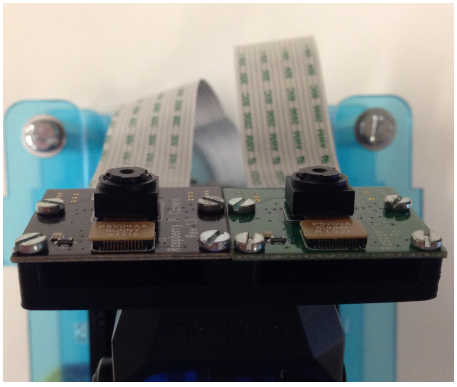

Figure 1: Top view.

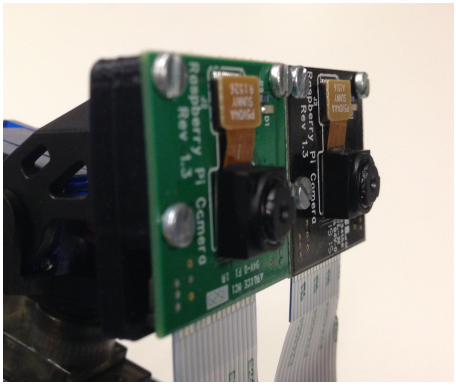

Figure 2: Side view.

2.1.2 Camera movement

To support pointing the camera in any direction, we needed a way to mechanically control the horizontal and vertical angle of the mount. This could be done by manually holding and orienting the 2 units, but results in unstable support. To get around this issue, we added a dedicated pan-tilt module.

The pan-tilt module is composed of 2 servomotors (1 horizontal, and 1 vertical). A servomotor is a small engine that can rotate by 180 degrees subject to an input pulse of varying width. The figure below shows how a single servomotor responds to such a control pulse.

Figure 3: Pan-tilt module.

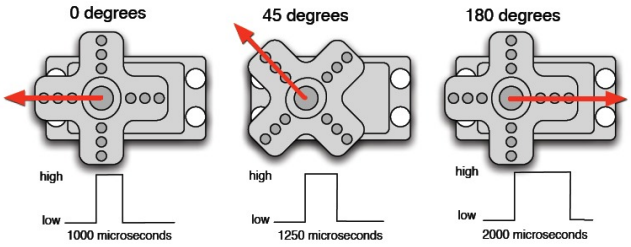

Figure 4: Servomotor rotation under control pulse. [6]

To control the width of the input pulses for the two servomotors, we added a joystick to the system. One of the most commonly used joysticks on the market is the one that was available on the PlayStation 2 game console, so we went for the same model. Since the joystick has an analog interface, we needed to use an Analog-Digital Converter (ADC) to interface it with the RPi's GPIO pins. The RPi unfortunately doesn't have an internal ADC, so we added a very popular external one, the MCP3204.

The ADC outputs 14-bit converted digital samples with a standard serial communication protocol (Serial Peripheral Interface, SPI). We use the CPU on the RPi to decode and save these converted samples. Figure 5 shows a short extract of the communication performed by the CPU with the ADC over the serial bus.

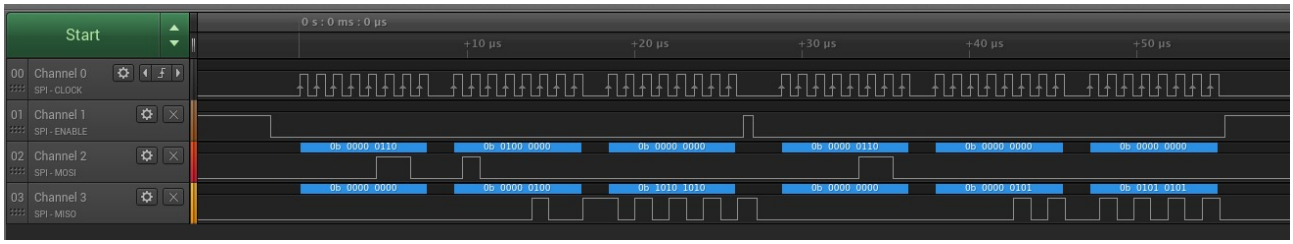

Figure 5: SPI communication with the ADC. The first 3 communication pulses read channel 0 from the ADC (joystick x-coordinate), and the next 3 communication pulses read channel 1 from the ADC (joystick y-coordinate).

Once we have a digital reading of the values obtained from the joystick, we need to generate 2 control pulses for the pan-tilt module's servomotors. There are many ways to do this.

1. Use the CPU to explicitly create this control pulse: this unfortunately results in very high CPU utilization, as the pulse needs to be updated at a very high frequency.
2. Use a dedicated hardware unit designed for generating PWM pulses. This provides the best performance as the CPU has almost nothing to do, but unfortunately the RPi only has 1 such unit, whereas we need 2 of them as we have 2 servomotors to control.
3. Use a hybrid design where we employ a Direct Memory Access (DMA) unit to generate the control pulses. A DMA unit is a dedicated controller designed to efficiently copy data from one memory location to another. Since modern systems use *memory-mapped* peripheral addressing, each peripheral is accessible at a certain address. This means that we can also use the DMA unit to write to a peripheral as if it were "memory". We programmed the DMA controller of the system to periodically write a modulated bit pulse to a GPIO port. This results in moderate CPU utilization, while being able to target any number of GPIO ports. We finally went for this solution.

Figure 6 shows the global hardware control loop used in our system.

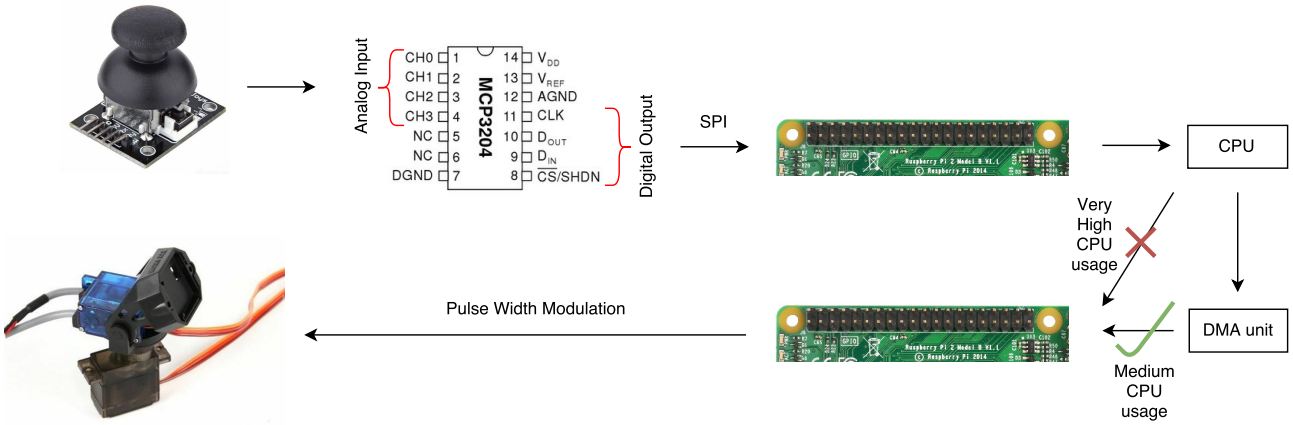

Figure 6: Hardware control loop (simplified).

2.1.3 Final hardware system

Finally, we put all components together in a stackable case and fixed all peripherals to it for easy transportation. Figure 7 shows the full system once mounted.

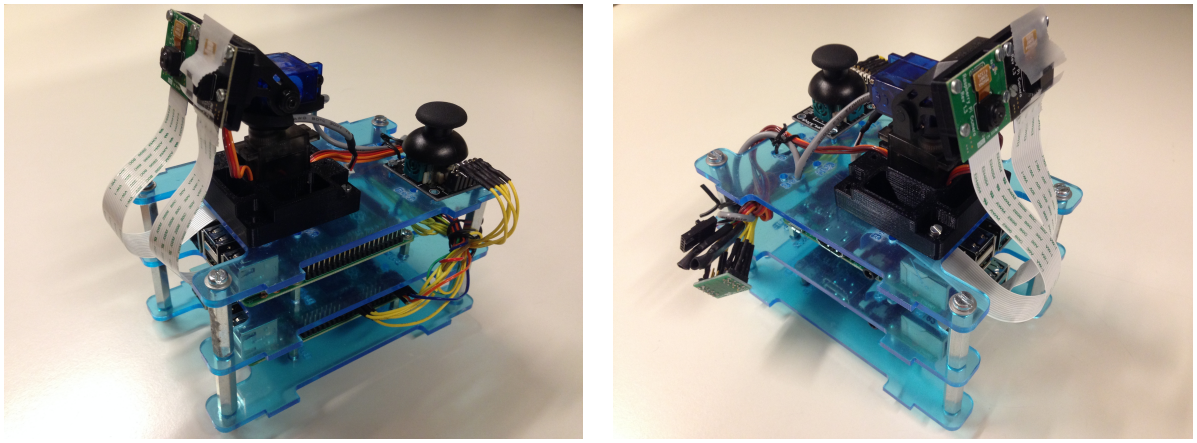

Figure 7: Full acquisition system.

We need to obtain an RGB-only image, and an NIR-only image as the output of the acquisition system. The standard RGB camera comes with an NIR filter, so all is good on that front. However, the NoIR camera captures both visible and NIR light, instead of exclusively capturing NIR light

To filter out the visible light, we use a small piece of exposed camera film. This camera film is fixed onto the NoIR camera with small pieces of cello-tape.

2.2 Synchronization

2.2.1 Clock synchronization

In order to enable both the RGB-only and the NIR-only images to be captured simultaneously, the clocks of both RPi units need to be synchronized. Since both units are on the same LAN, the PTPd project was perfectly suited for our needs. The PTP daemon (PTPd) is an implementation the Precision Time Protocol (PTP) version 2 designed primarily for instrumentation and control systems [5]. It can achieve clock coordination of a group of LAN connected computers on the microsecond level. One RPi unit runs PTPd configured as a master and periodically broadcasts the current time, while the other unit is configured as a slave and coordinates its clock with the master.

2.2.2 Simultaneous image capture

For the simultaneous capture of both images, the RPi unit connected to the analog input runs as a client while the other functions as a server (figure 8). In idle mode, the server unit listens on a specified port for messages from the client. When the user initiates the image capture at time $t = t_s$, the client unit decides upon a capture time $t_c = t_s + \Delta t$ and transmits it to the server. Then both units wait until $t = t_c$, at which point they simultaneously initiate the image capture and write the obtained images to a file. Once the server has finished capturing the image, it transmits it to the client and returns to its initial listening state. The client is then in possession of both the RGB and the NIR image and applies the operations requested by the user.

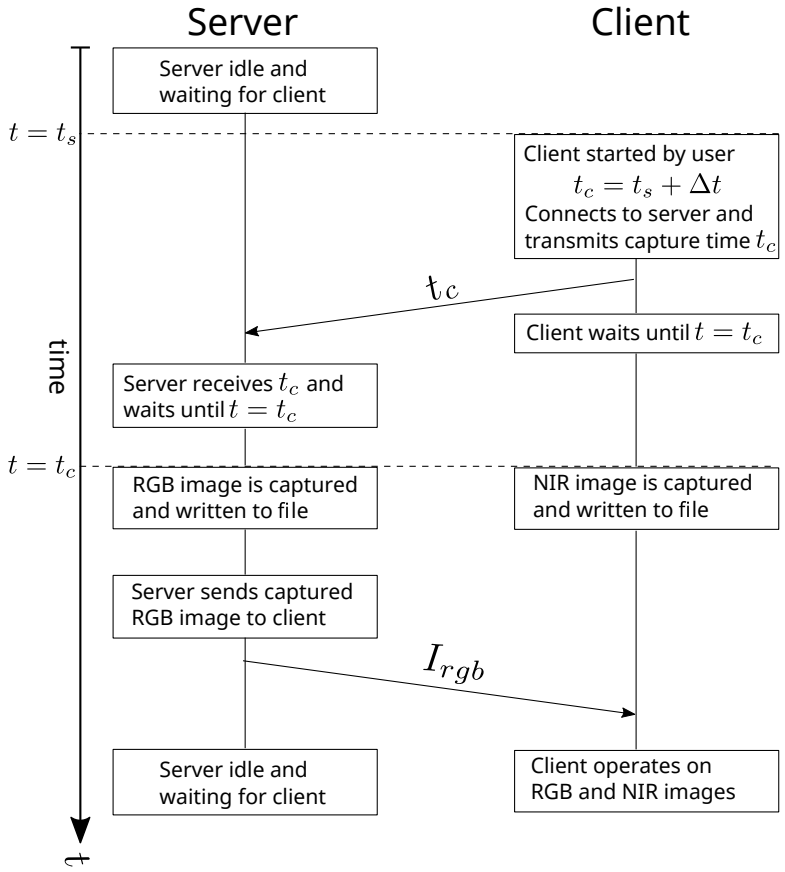

Figure 8: Protocol for simultaneous image capture.

2.3 Registration algorithm for image alignment

In the registration part of the project, the main purpose is to align the two images so they can later be used for various image processing algorithms. To do this, a motion parameter needs to be found that can move one image to suit the other. For this purpose, the OpenCV library was used.

Firstly, the imported pictures are converted to byte images, regardless of previous format. This is saved into an array, where each element is a list with all values for the pixels of the first row of the image. Then, the feature points and descriptors are located using the SIFT-algorithm. SIFT has the size of 16 x 16 windows around the feature points. The algorithm computes the gradient for each pixel, which is important in the preprocess for matching. Feature points are pixels who stand out in the image, often due to a change in color, e.g edges. The descriptors are the surrounding pixels, creating an unique patch.

The next step involves extraction of the feature points and descriptors, through detector, extractor and computing functions. Furthermore, the patches of feature points and descriptors are patched over the images, whereafter a Flann Index Tree and a ratio test is performed. The Flann Index is a fast way of approximating the nearest neighbor to a point in a large data set and for high dimensional features.

The constructed index is a randomized kd-tree, a data structure for organizing points in a k-dimensional space, which is searched through in parallel in our case in five trees. It computes the distances between pairs of features. The ratio test put a threshold on the matches, only choosing the closest 70% in distance.

When appropriate matches are found, the motion parameter with the best accuracy in regard to translation and rotation is obtained. Homography and RANSAC are used, since the relationships between the motions are nonlinear. The homography evaluates the parameters in regard to how big errors they provide, choosing the best motion parameter. RANSAC is a method to remove outliers in the matching process. Then a transformation takes place, i.e. the source image is then warped to the destination image. The resulting image has been created, consisting only of the shared content of the two input images, making it appear to be taken from the same perspective as one of the input images and with the possibility to merge them.

3 Applications

3.1 Skin Smoothing

Once the sequence of operations previously described has been successfully carried out, we obtain a classical RGB image and an aligned NIR image. The next step in the pipeline is to combine these two images in order to perform a skin smoothing operation.

3.1.1 Theory

The theory behind skin smoothing using NIR imaging is presented in the paper Combining visible and near-infrared images for realistic skin smoothing by Clement Fredembach, Nathalie Barbuscia and Sabine Susstrunk [1]. In fact, portraiture is often capricious and improving its quality plays a big part in the digital photography field.

The idea is to realistically smooth skin tones and filter out unwanted features by combining an RGB image and an NIR one. The main argument for using NIR is that melanin and hemoglobin only absorb a little of its wavelength and that NIR penetrates deeper inside the dermis therefore suppressing surface imperfections. More generally, unwanted skin features often stand out in visible light but remain invisible in the near infrared image.

The global concept is to divide our image into two layers: a base layer of low frequency features extracted from the RGB image and a detail layer of high frequency features obtained from the NIR image. For that purpose, a bilateral (low-pass) filter is used on both the RGB and NIR images. This is a more fine-grained and appropriate tool for portraiture than a simple Gaussian filter, as it is edge-aware therefore yielding a sharper result. In the next part, the complete process of merging the two images is described in a more detailed fashion.

3.1.2 Methodology

Our implementation for the skin smoothing algorithm follows the flowchart visible on Figure 9

3.1.3 Pre-processing

As part of the project pipeline, the merging routine takes as input two aligned and therefore superposable images. As a side note, because we use a physical filter that considerably darkens the NIR image, normalization is performed as a pre-processing step to the registration and skin-smoothing routines of our project.

Following this brightness adjustment, the NIR image has its brightness corrected by computing its mean and shifting it to match the brightness of the RGB image. Moreover, there is no need for chrominance information during the merging phase of the project, thus, the RGB image is converted to YC_bC_r, a three channel

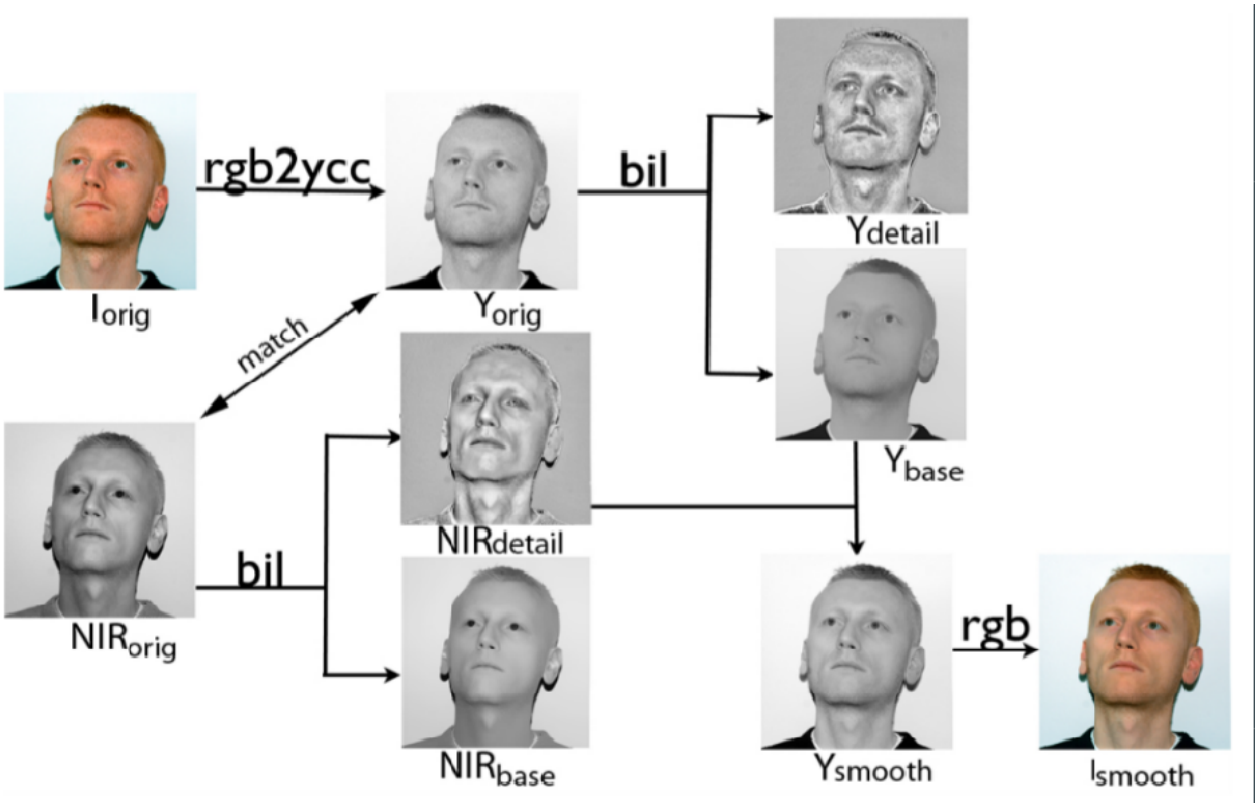

Figure 9: Flowchart of the procedure for combining NIR and RGB images as described in [1].

format for images where Y is the luminance and C_b/C_r the chrominance (blue and red levels). All subsequent transformations/computations are done exclusively on the Y channel of the RGB image.

3.1.4 Bilateral filter

Out of the pre-processing step, we get two brightness-equalized grayscale images. The base and detail layers must now be extracted. To that extent we use the implementation of bilateral filter of the OpenCV library. This filter is non-linear, edge preserving and noise-reducing, where the intensity of each pixel is replaced by a weighted average of its neighborhood.

One of the main difficulties is to adjust the parameters correctly in order to obtain natural looking results. We have to choose an appropriate neighborhood size d (in pixels) and tune parameters σ_r, the range parameter for smoothing differences in intensity, and σ_d ,the parameter for smoothing spatial features. This step is crucial and closely linked to the quality of our output as badly selected parameters cause unwanted effects. Although the bilateral filter is edge-preserving, being overzealous with the smoothing parameters yields unnatural effects such as blurring, unrealistic facial hair/eyebrows, gradient reversal, etc... This step is computationally heavy and costly even if the library used provides runtime optimizations for the filter.

The RGB base layer containing only low frequency features (e.g face shape such as cheeks, nose forehead etc...) is obtained after running the filter. The same instance is used on the NIR image, the result base layer residual is subtracted from its original to obtain the NIR-detail image containing high frequency features (e.g. eyes, facial hair, eyebrows etc...). Finally, the two resulting images have simply their pixel values added together in order to obtain the skin smoothed grayscale version of the portrait, the chrominance information is ultimately added by using the C_b/C_r channels of the original result of the original RGB image.

3.1.5 Practical results

On Figure 10 and Figure 11 we can see practical results of the skin smoothing algorithm, features like freckles, shadows under the eyes and other irregularities are completely removed yielding a natural looking result. Pictures were taken in strong incandescent lighting conditions in order to grasp as much detail as possible from both the RGB and NIR images.

Figure 10: Original RGB image

Figure 11: Skin smoothed result

3.2 Shadow Detection

Another feature that we implemented in our project is a shadow detection algorithm. From one pair of registered RGB and NIR images, this part aims to automatically detect shadows in a fast way by using the properties of the near-infrared information. The idea is that dark objects that are often detected as shadows in traditional RGB shadow detection algorithms have a much higher reflectance in the NIR spectrum. This method is built around three main observations: The first one is that shadows are generally darker than their surroundings. Second, a considerable number of dark objects in the visible spectrum are brighter in the NIR one. The last one is that most illuminants have a specific behavior in the NIR.

Our algorithm follows the one described in [2]. It takes as input the pair of registered RGB and NIR images computed in 2.3 and outputs a binary mask revealing the pixels estimated as under shadow. Its operation is briefly explained in the following paragraphs. We can easily divide it in three parts: Shadow candidates, Color to NIR Ratios and Binary Shadow Mask.

Figure 12 represents the different parts of the algorithm to achieve shadow detection.

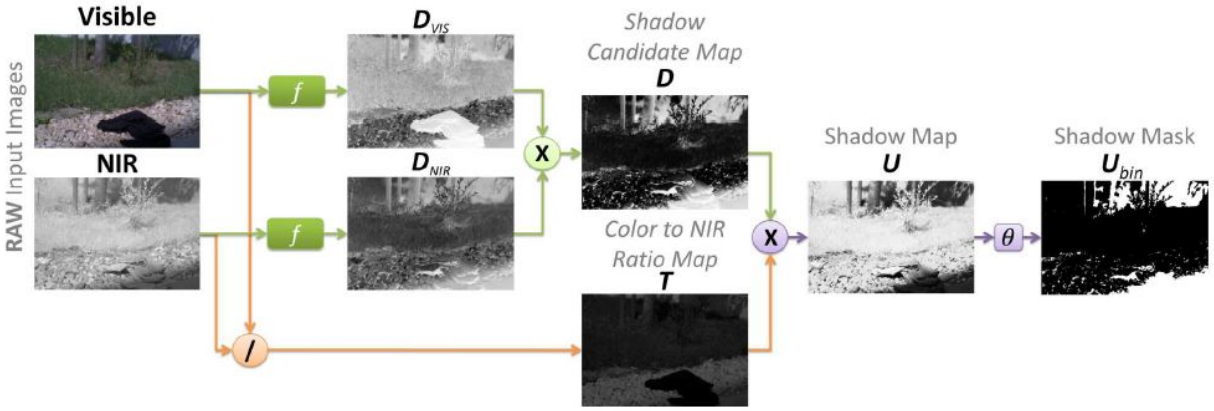

Figure 12: Block diagram showing the shadow detection algorithm as stated in [2]

3.2.1 Shadow Candidates

The first task is to find the areas where the probability of finding a shadow at this place is high. We call this the shadow candidate map. Because darkness in both visible and NIR images is a condition of shadow presence,

we need to compute a shadow candidate map for both images. For the visible one we compute the brightness of the image, to whom we then apply a non-linear mapping to obtain a better distribution. The NIR one is pretty straightforward because we only need to apply the non-linear mapping on the near-infrared image. The final shadow candidate map is simply the multiplication of the two to take the two conditions into account.

3.2.2 Color to NIR Ratios

The shadow candidate map is already giving a good intuition about shadow presence but it is possible to improve it with the help of color to NIR ratios. The key insight here is that the difference between the visible and NIR bands for many shadow creating illuminants is distinctive. For example, the illuminant spectral power of the skylight is greater for the visible light than for the NIR while they are close to equal under sunlight. We can exploit this difference and compute the ratio image between the RGB and the NIR image. This ratio will be greater than 1 for skylit regions and around 1 for sunlit regions. Let's not forget to take the maximum ratio over the three color channels to obtain the better results. We also have to handle the case where the ratio could tends to infinity when the pixel intensity of the NIR image tends to zero. The shadow map found in 3.2.1 contains all possible shadows but can also include some dark objects. By combining this with the ratio image, we obtain a pretty nice shadow detector .

3.2.3 Binary Shadow Mask

In the last step, we combine the results of 3.2.1 and 3.2.2 by multiplying the complementary of both. This give us a nice continuous shadow map. Finally, we want to threshold this shadow map to distinguish shadow from non-shadow area as a binary shadow mask. The key to find a good threshold value is to compute the histogram of the image and then set the threshold value to the first valley of the histogram. The first valley is defined as the smallest valued bin of the histogram where the two neighboring bins on both sides have larger, increasing values. The resulting binary image indicates which pixels are in shadow and which aren't.

3.2.4 Results

Finally we obtain a nice and convenient way to automatically detect shadows in a captured image. The method runs rather fast, in the order of few seconds depending on the original picture's dimensions. The quality of results obtained by this algorithm are quite variable due to a dependency on a lot of parameters. The first one is that the algorithm uses the assumptions that the two images are perfectly well registered which may not be the case in some situations. Another issue come from the type of illuminant of the scene. The best results are obtain with an outdoor sunny lighting thanks to NIR reflectance properties.

Figure 13: RGB image

Figure 14: Registered NIR image

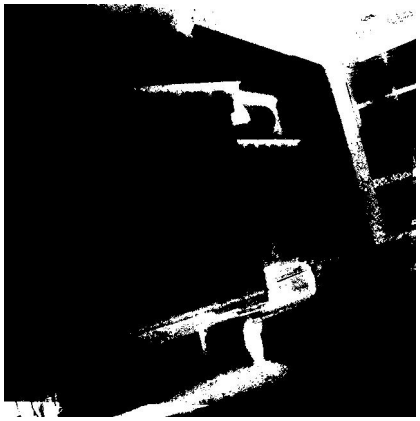

Figure 15: Shadow detection

4 Issues

Various issues were encountered during the project.

1. Due to memory limitations in the ARM version of OpenCV, we were unable to apply the algorithms to full-resolution (2592×1944) pictures. We had to limit ourselves to lower resolution frames.

2. The quality of the final images depends greatly on the registration stage. Unfortunately, registration does not always work, as too few keypoints are available at times.

3. Transporting the camera setup is not practical, especially for outdoor shots (would have been useful for shadow detection)

4. Compared to a standard laptop, the RPi takes a huge amount of time to process the images, sometimes disproportionally.

5 Conclusion

Near-infrared information is intrinsically available to the silicon sensors commonly used in digital cameras, but is currently not acquired or used in any way. However, research has shown that information contained within the NIR band of the light spectrum is useful to obtain more information about a scene, and to therefore be able to devise more sophisticated image processing algorithms.

In this project, we implemented such a system on both a hardware and software front, and demonstrated 2 applications of near-infrared imaging, namely skin smoothing and shadow detection. Upon analysis, we were able to reproduce the results from the papers that initially presented the various methods. Results were satisfactory, proving yet again that NIR imagery has good potential for implementation within digital cameras, even more so as not having to filter out this extra information does not have any additional cost in a camera's fabrication process.

We look forward to seeing how camera manufacturers respond to these new research results in order to provide more information for use within imaging applications.

6 Credits

Dajana Vlajic & Rebecka Rőnnbäck Image registration

Florian Zimmermann Raspberry Pi operating system setup & Synchronization

Josselin Vallee Skin smoothing

Sahand Kashani Dual-camera mount, Joystick & Dual-servomotor control

Yann Perret Shadow detection

References

[1] Clement Fredembach, Nathalie Barbuscia, Sabine Susstrunk, *Combining visible and near-infrared images for realistic skin smoothing*,

[2] Dominic Rufenacht, Clement Fredembach, Sabine Susstrunk, *Automatic and accurate shadow detection using near-infrared information*, IEEE Transactions on Pattern Analysis and Machine Intelligence.

[3] Chen Feng, Shaojie Zhuo, Xiaopeng Zhang, Liang Shen, Sabine Ssstrunk, *Near-Infrared Guided Color Image Dehazing*.

[4] R. Szeliski, *Computer vision: algorithms and applications*, Springer Science & Business Media, 2010, p. 311-320.

[5] IEEE 1588 Precision Time Protocol (PTP) Version 2 Specification, IEEE, March 2008.

[6] http://sweb.cityu.edu.hk/sm2240/4/