Decreasing time-of-flight 3D-camera range ambiguity

Gerben van den Broeke

July 16, 2011

Abstract

This paper is about a SPAD-based time-of-flight 3D-camera. The purpose of this paper is to document a bit of how the camera works, and to give some ideas for improvements. The measurement method of this camera introduces an ambiguity in the result, because the distance modulo 5 meters is measured. A method is implemented that reduces this ambiguity by alternating between coarse and fine measurements. Also, two other ideas for improvement of the camera are given, which are both not yet implemented.

Overview

This paper follows from a small student research project, concerning an improvement to a 3D-camera. This paper is intended to provide a bit of documentation for anyone continuing with the project or planning to do anything else with the camera, and to amuse anyone interested. Because the design of the camera itself is not well documented, a short explanation of how it works is included. The project consisted mostly of figuring out how the camera works, writing a software driver to be able to control it and to display the measurements, and implementing a method to overcome the range ambiguity that is inherent to the used ranging method. Unfortunately, I had not enough time to get the implementation fully functional, so no measurements results are included.

The structure of the paper is as follows. First the used depth measurement method is described, then how this method is implemented in the camera. Then follows a short description of the software that I wrote. Finally, the disambiguation improvement is described, and two ideas for other possible improvements are given.

Time-of-flight phase shift measurement

The device in use is a camera capable of measuring depth images, using a time-of-flight principle: Infrared light is emitted by the illuminator situated on the camera, reflected by some object to be measured, and received by the camera. The time between emitting and receiving the light is measured to obtain the distance to the reflecting object. This particular camera (figure 1) uses a matrix of 60×48 sensors to create a 3D-image, where each sensor is a single-photon avalanche diode (SPAD). For more information on the principles of this camera, I refer to the thesis by Cristiano Niclass [1].

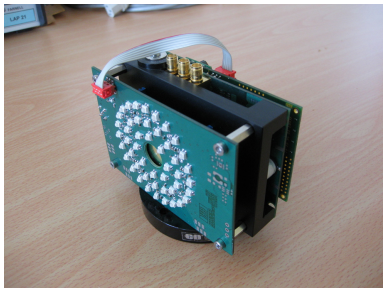

Figure 1: The camera

To measure the time of flight the SPSD (single photon synchronous detection) method is used, as described in Cristiano's thesis. In short, the principle is to emit the light modulated with a 30MHz

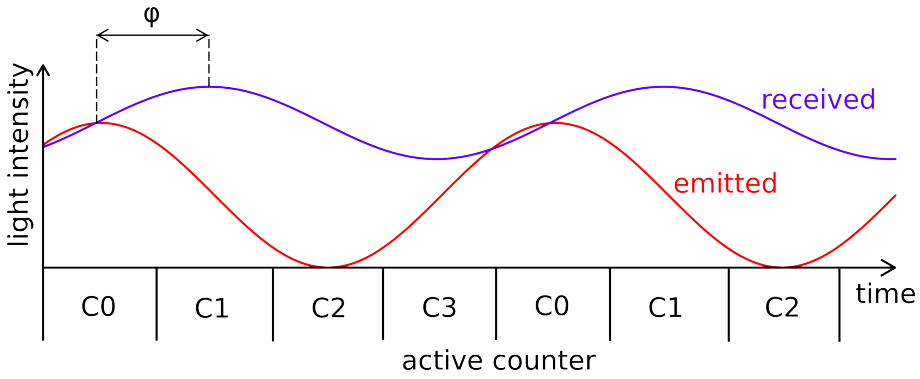

Figure 2: The sinewave modulation of the light and activation of the counters

sine wave. The received light is then again this sine wave, but shifted in time, plus a noise factor from background light. To measure the distance, the phase shift $-\phi$ of this received wave is measured. The distance then simply follows as

$$d = \frac{\phi c}{2 \cdot 2\pi f_{mod}} \tag{1}$$

where f_{mod} is the modulation frequency of the sine wave, normally 30MHz, and c the speed of light. To measure the phase shift, four counters are used, with each counter counting the amount of incoming photons during a part of the emitted sine wave, as shown in figure 2. Using these counts, the amplitude and phase shift of the received wave can be obtained, as well as the amplitude of the background light. The phase shift, modulo 2π, can be found from the counter values:

$$\phi = \arg\left(C_0 - C_2 + (C_1 - C_3)\, i\right) \tag{2}$$

In this method, the assumption is made that the background light is not correlated to the sine wave. Measurement accuracy is dependent on the integration time of the counters. Practical values are between 100 and 1000 modulation periods, depending on the requested accuracy and amount of background light.

Because phase shifts are ambiguous, any multiple of $\frac{c}{2f_{mod}}$ could have to be added to the measurement. In the case of $f_{mod} = 30$MHz this limits the usable range of the camera to 5 meters.

Structure of the camera

The device hardware consists of three circuit boards: one board with the camera chip, one with the infrared LEDs for illumination, and FPGA4U board with an Altera Cyclone FPGA.

In the camera chip, every pixel has two 8-bit counters to count the amount of received photons. By switching the SMOD signal, one can select which counter each pixel should use, as shown in figure 4.

To get the four counter values, two measurements are done after another. The first measurement counts in phase with the illumination signal, the second 90° out of phase, as shown in figure 5. This way, values for C_0 and C_2 are measured first, then C_1 and C_3. Even though the counters now each count during half the wave period instead of a quarter, equation 2 can still be

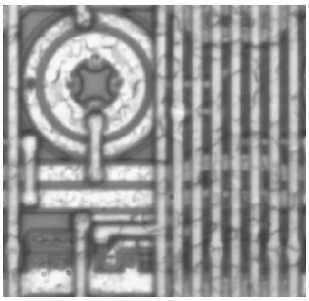

Figure 3: One pixel of the camera *(source: [1])*

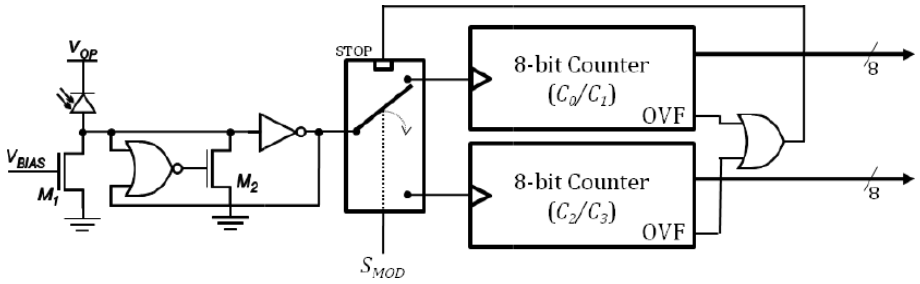

Figure 4: Schematic of one pixel *(source: [1])*

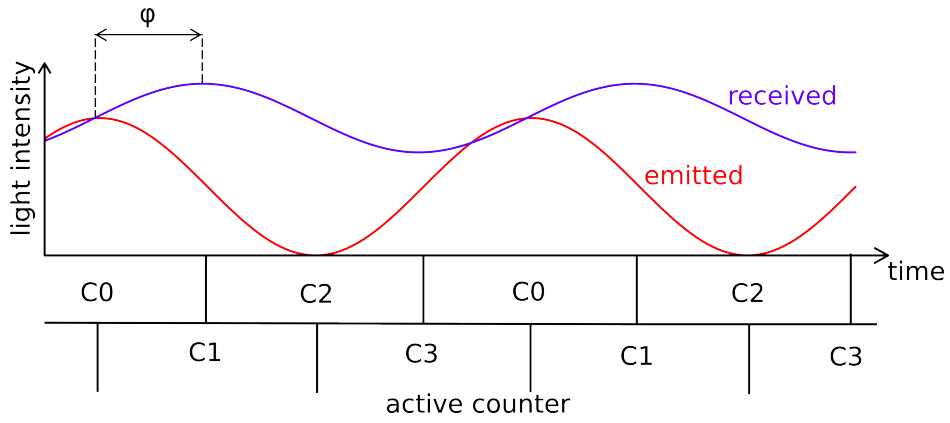

Figure 5: Counter activation with only two counters

used. Please note that the real implementation
differs a little bit from this explanation: the counters C_1 and C_3 are actually switched, so the difference between them is negated.

The counter values are read into the FPGA via a set of multiplexers on the camera board. The counters are read out sequentially, and accumulated in 16-bit registers. Two values are accumulated: the sum of the two counters, and the difference of the two counters. Only the difference is needed for the distance calculation, the sum is used for measuring background light intensity and signal-to-noise ratio.

The system in the FPGA consists of a Nios2 processor connected to a custom piece of hardware. This custom hardware generates the illumination signal and SMOD signal, reads and accumulates the counter values, and interfaces with the processor.

Software

In the Nios2 processor a program runs which communicates with the connected computer. First, it waits for commands to set the integration time and some other settings. After a start command it configures the custom hardware and signals it to start measuring. After the measurement, the program in the Nios processor reads the accumulated data and sends it via a USB connection to the attached computer.

On the computer the data is received and for each pixel the distance is calculated. I wrote a C-program to communicate with the device using *libusb*, process the data and display a grayscale image, with darkness of each pixel representing its distance. It can also display the measured total light intensity, and the signal-to-noise ratio.

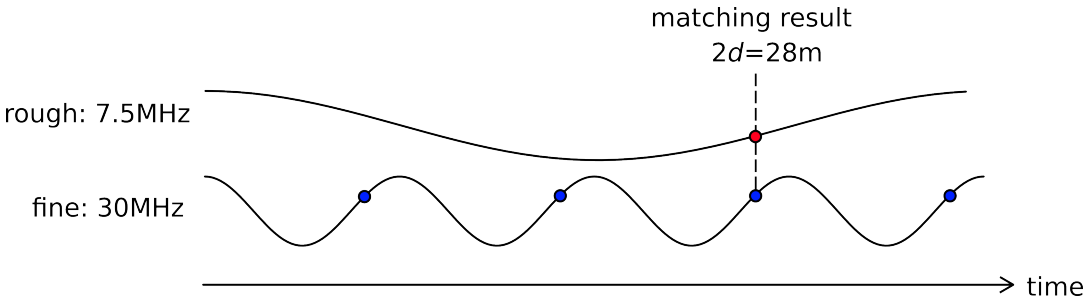

Figure 6: Using a rough and a fine measurement to decrease ambiguity

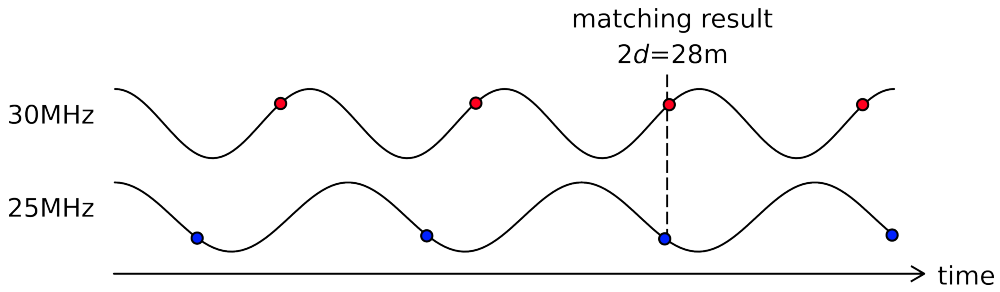

Figure 7: Using two fine measurements to decrease ambiguity

Because some pixels in the camera did not work correctly, I wrote a function to find these dead pixels. This function finds the oversensitive pixels and prints a string with their coordinates. These can then be filtered (set to zero) from next measurements using *clear_ dead_ pixels*.

Range disambiguation

A problem intrinsic to the phase measurement method is that several distances map to the same phase shift. With a modulation frequency of 30MHz, the modulation wavelength is $\frac{c}{f_{mod}} = 10$m, which results in a 5m ambiguity. A calculated distance of 2 meters might thus as well result from an object that is actually 7 or 12 meters away. The current illuminator is not so strong, so in indoor light condition, distances bigger than about 10 meters are usually lost because of noise. Especially when the illuminator would be improved, the ambiguity of the result becomes a problem.

A simple method to decrease this ambiguity is to use a lower modulation frequency. This however reduces the measurement accuracy. A solution is to switch between a low and a high frequency, to get a precise measurement with less ambiguity. For example one can switch between 7.5MHz and 30MHz to get a range of 20 meters, as depicted in figure 6.

Instead of switching between a rough and a fine measurement, one could also switch between two high frequencies to get two fine measurements. For example, by choosing 30MHz and 25MHz , the range can be increased to the least common multiple of their ranges, which is 30 meters. To find the object's distance, a distance is sought which is both a solution to the 30MHz measurement and to the 25MHz measurement, as depicted in figure 7. This method results in a higher accuracy than the rough/fine measurements, given the same integration time. The downside is that if, because of noise, the phase shift measurements are a bit off, then the resulting distance could be guessed totally wrong. By using frequencies closer to each other, say 29MHz and 30MHz, the theoretical range will increase a lot, but the noise will become a bigger problem.

For this project I have implemented the first method. For this I introduced a configurable clock divider, so the range can be chosen. The divided clock signal is used for generating both the illumination

signal and the SMOD signal. The division factor can be set during the configuration of the camera, but is not yet switched during runtime. In the driver software one can choose the distance range. It then calculates the required division factor, which is sent to the camera during the configuration. The division factor can be 1 or an even number up to 30.

I tested the system with the normal range of 5 meters, and with a range of 10 meters (15 MHz modulation). The 5 meter worked as it should, but at 10 meter the results showed errors of more than a meter. The problem is probably that the illuminator is actually fed a square wave instead of a sine wave. The LEDs are slow enough to emit almost a sine wave when they are fed a 30MHz square wave signal, but when lowering the frequency the wave becomes more square-like, so the calculation of the phase using equation 2 is no longer correct. Part of this problem could be corrected for, but I did not investigate any further in this problem because of a lack of time. A better illuminator would be a good idea anyway, preferably one that is stronger and can emit sine waves of several frequencies.

Other possible improvements

I have thought about two other possible improvements to the camera, which I have not implemented. I write them here so someone else might try them out. The first is related to the range disambiguation as it also uses multiple frequencies. The second is about the camera chip hardware, so it can not be implemented without building a new camera chip.

Multi-object detection

A problem with the phase detection method is that looking at edges of objects, or at partly transparent materials, results in wrong calculated distances. Because in these situations the sine wave is reflected at two (or more) different distances, the reflected waves superpose and again give a sine wave with a different phase. The distance that will be measured is then somewhere between the objects if they are close, or even anywhere if the objects are further than a quarter of the wavelength apart. If the reflection of one object is much stronger than that of the others, the resulting distance will be close to that object's distance.

By treating the reflected waves as phasors, one finds that the measured phase shift and amplitude are:

$$\phi_r = \tan^{-1}\left(\frac{\sum_n \sin\phi_n \cdot a_n}{\sum_n \cos\phi_n \cdot a_n}\right) \tag{3}$$

$$a_r = \sqrt{\left(\sum_n \sin\phi_n \cdot a_n\right)^2 + \left(\sum_n \cos\phi_n \cdot a_n\right)^2} \tag{4}$$

where ϕ_n and a_n are the phase shift and amplitude of the sine wave as reflected by the n^{th} object. These phase shifts are a function of the distances of the objects and of the used modulation frequency, as follows from equation 1. By choosing a different modulation frequency, the values of ϕ_n will change linearly, and will cause a nonlinear change in the resulting ϕ_r and a_r. The method of switching between frequencies can thus be used to detect multiple objects and determine their distances. The number of frequencies that is used determines the amount of objects that can be distinguished. When disambiguation is also wanted, even more frequencies have to be used.

Preventing pixel saturation

In each pixel of the camera, there are two counters, and ideally there should be four. The main constraint is the area usage of these counters, they should be kept small to give space to the SPAD. The

counters should however have enough bits be able to count the incoming photons without saturating. The current system uses two 8-bit counters, of which one is active at a time. The pixels are read out sequentially, and at each readout the counters are reset. When there are too many photons, the counters saturate at 255 and the measurement becomes unusable. This happens when looking at a mirroring object, or, more often, when operating in bright daylight. Because usually most of the photons, say 90% of them, come from background light, it seems silly to count them. For the distance calculation we are actually only interested in the difference between the two counters, so I thought about other ways to implement them.

One obvious way is to use a bidirectional counter, counting up when SMOD is high, and down when it is low. The background light will have a net effect of about zero, so the counter will not saturate as quickly, and can thus have less bits. A problem with this idea is that a bidirectional counter takes a lot more space than a simple up-counter. It might be worth to compare the hardware size of two 8-bit up-counters with one up/down-counter having less bits.

Another approach is to stick with the two up-counters, but to let them roll over instead of saturating at their maximum. The assumption that is made here is that the reflected light will not cause the counter to overflow, but the background light may do so. The background light will overflow both counters the same number of times, or perhaps one of them once more than the other. To figure out which of these options is the case, the assumption is made that the difference between the photon counts is small. If the two counter values are more than half of the counter range apart, the one with the lowest value is assumed to have rolled over one more time than the other. For this method, the size of the counter thus needs to be just one bit more than what would be needed to count the maximum expected amount of reflected photons, instead of the maximum total amount of photons.

Reference

[1] Cristiano L. Niclass, *Single-Photon Image Sensors in CMOS: Picosecond Resolution for Three-Dimensional Imaging* (2008)