Making PiNAOqio Read: Digitizing Text From Physical Books

Mihnea Rusu (mr3313)
Imperial College London

Abstract: Today's families spend more time connected and less time together [1]. Parents, therefore, have less time available to read to their children. We aim to solve this problem with PiNAOqio, the robot that reads paper books to young ones.

Paper has been and will continue to be an essential part of modern society. Having our robot read from physical books is as challenging from an engineering perspective as it is advantageous and rewarding to the users. The problems of reading from an image will be tackled with a special illumination rig and optical character recognition software running in an embedded Linux system.

I. Introduction

In [2] we hypothesized that by having a storytelling robot, PiNAOqio, read books to children, information retention and user experience would be improved over traditional audiobooks. One may now argue that e-books should be the obvious candidate for our project. However, with traditional printed book sales on the rise [3], it offers convenience to the user of our robot to be able to pre-load their existing titles as well as know that future hardcopy volumes they purchase will be reverse compatible. Additionally, a wider selection is available to the reader in print, as opposed to digitally in the form of e-books or audiobooks due to the ubiquity of paper.

It is mentioned in [4] that physical books are unlikely to go out of print. The people interviewed mention factors like smell, weight, and the feel of turning pages as key reasons for their continued pursuit of printed material. Furthermore, paper is currently the go-to solution when it comes to learning e.g. in a classroom environment, due to the spatial markers books provide [5]. Seeking to read physical books as part of our project may enable children born in today's increasingly digital environment to read more paper copies, become better learners and (re)discover the joys of physical books.

Arguably the most essential sub-system of PiNAOqio is that which reads books. This report will lay out the requirements and constraints we impose and are faced with in constructing this system. A brief explanation behind the reasoning behind each one will be provided and candidate hardware and software solutions will be explored. Finally, the results of preliminary testing will be briefly discussed and future work will be laid out.

II. Requirements

The book reading sub-system of PiNAOqio needs to perform two broad tasks: turn the pages of a book (which will not be discussed in this report) and extract the text of the currently opened pages. We shall restrict our scope to only include printed books in English, as opposed to manuscripts or printed material in other languages.

In order to digitize a page into text, a photograph will be taken of that page and OCR will be applied. The reading system will be built separately from the NAO robot. We have decided this for two reasons: it's required to perform computationally intensive tasks and it gives us more freedom with the entire concept of our robot.

In an ideal situation, our requirements would be as follows (in no particular order):

A. Text to be available to the speech synthesis system with unnoticeable latency from when the page-turning system reaches a rest state after flipping a page.

B. The throughput of words should be constant and in real-time, as PiNAOqio is speaking.

C. There should be no difference between the book and the spoken word i.e. text accuracy should be 100%.

With more development time and resources available (both hardware and financial) all of the above requirements are achievable, as will be explained below. As it stands though, a more realistic set of requirements would be to prioritize real-time throughput, allow a minimum level of noticeable latency and assume not all text will be fully accurate. Throughput would hide the high latency during reading. The increased latency would only pose a problem the first time the robot begins reading. Turning the page would be timed to begin from a specific time prior to the estimated current page's ending time in order to eliminate the otherwise-apparent delay.

Furthermore, to prevent obvious text inaccuracies (e.g. non-alphanumeric symbols in the middle of a word) from being spoken, pre-determined placeholder actions can be performed by PiNAOqio such as coughing, sneezing, "uhm"-ing, etc. These will also enhance the human-like qualities of the robot.

There are a number of challenges associated with

achieving accurate OCR. Firstly, is the need for a high pixel-per-inch (PPI) count. A major use of OCR technology is in creating searchable text documents from scanned material. Images from a scanner are advantaged by a high PPI count due to the proximity of the scanner's sensor array to the page. In our application, the page flipping mechanism needs a minimum clearance from the camera to operate. Additionally, we are limited by the focal length of the available lens and the requirement to capture an open book.

To obtain comparable OCR results to those obtained from a scanned page, our solution is to use a high resolution camera. Processing large image files however is a more intensive process and will increase the system's latency further. To increase effective PPI, we could opt for a mobile camera axis that would capture the open pages section by section – the camera could be positioned at smaller height away from the page. This solution, while attractive, increases mechanical complexity and is simply unfeasible given the development time available.

A second challenge we are faced with in obtaining accurate OCR is that of adequate page illumination. Scanners achieve consistent illumination across the entire page by evenly illuminating only the area being scanned at the time – which is small; the page also lies perfectly flat inside. Our system requires even levels of illumination across a larger area – an open book - ideally without gradients, highlights or shadows.

III. Hardware

For the system's hardware we are using a Raspberry Pi 3, together with a Raspberry Pi v2 camera which features an 8MP sensor and a maximum photo resolution of 3280x2464 pixels. We could have opted to use NAO's built-in camera, but this was disadvantageous for three main reasons. First of all, its camera is of inferior quality to the one we are currently using. Secondly, because NAO would be forced to look at the book at an acute angle, a skewed image would be obtained. Deskewing algorithms are applied by OCR software as a standard feature, however this would lead to uneven text PPI across the page, and thus could potentially harm our text recognition performance.

Finally, the robot would be forced to look down at the book every time it needs to photograph an open page spread – which always occurs after a page flip. To the reader, this would look scheduled and unnatural after a number of pages. With a separate system, we have the freedom to do the action of looking down at the book more randomly and naturally in order to build the 'believability' of PiNAOqio to the human user.

Initial experimentation in creating adequate light for OCR involved using an LED ring light around the Raspberry Pi Camera. Due to the lack of optics on the diodes, the light was not directional and would cause camera glare. Moreover, the light output from a dozen LEDs in the ring did not provide sufficient illumination from our camera's focus height.

Further thought revealed that the engineering challenge of evenly illuminating a large area with diffuse light has already been tackled. Liquid Crystal Display (LCD) backlights are ubiquitous and feature such technology. A backlight is typically constructed of several layers, as shown in Figure 1.

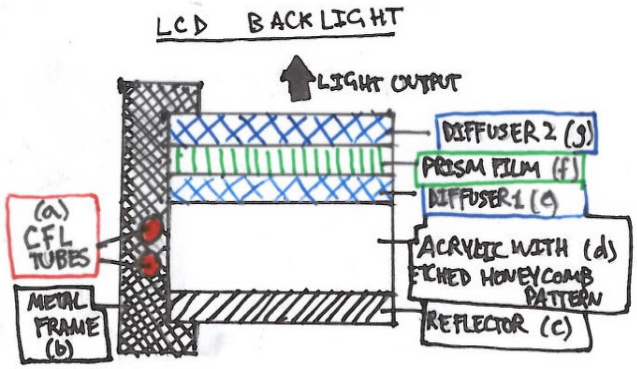

Figure 1: Side view of the LCD backlight layer stack-up. (a) are two Compact Fluorescent Lightbulb (CFL) tubes; (b) is a thin metallic frame that houses (a) and holds the entire stack-up; (c) is a sheet of white opaque plastic serving as a reflector; (d) is an acrylic plate with a honeycomb pattern etched on the side facing towards (c) to give it the properties of a diffuse light pipe; (e) and (g) are two diffuser sheets; (f) is a prism lens (also known as a brightness enhancing film or BEF for short) that focuses light akin to a Fresnel lens.

We obtained a backlight from a recycled LCD display. The backlight originally used CFL tubes as a source of light (denoted as (a) in Figure 1); these require a source of high voltage and as such would not be adequate for a prototype that is to be evaluated with people, especially children. To fix this issue, we replaced the source of light with white LED strips around the entire perimeter of the acrylic sheet (d). Downsizing the backlight was done by laser cutting the acrylic sheet to the dimensions required to fit inside the frame that was built to accommodate one book; a central hole was cut to allow the camera to peer through. Sealing the edge of the acrylic inside the hole was necessary to avoid camera glare and was achieved with opaque white tape (also to serve as a side reflector).

IV. Software

The software component of the system can be broken down in three components: a) image acquisition; b) pre-processing; c) OCR. A software pipeline links all the components together; all scripts are designed to run on the

Raspberry Pi. In the eventuality that computational load should diminish user experience or there is sufficient time remaining for developing further enhancements, the intensive aspects of processing may be offloaded to more powerful machines in the cloud using technologies like Amazon Web Services (AWS). It is helpful in that regard that no additional hardware will be needed – the Raspberry Pi 3 has an on-board Wi-Fi radio. Lastly, it should be mentioned that our code is version controlled using GitHub, for good software engineering practice.

A. Image acquisition

To obtain images from the camera, the built-in Raspberry Pi Bash command `raspistill` is used. This command captures an image from the camera, with a provided set of exposure parameters i.e. shutter speed, ISO, exposure compensation, etc.; image quality and image size. A list of available output formats may be found in [6].

The raw format is useful for preserving all information from the sensor at the cost of increased memory usage and computation time further in the pipeline. Lossy compressive formats may be specified for boosting throughput and significantly reducing memory cost, but they are not without fault. According to [7] lossy formats that use the Discrete Cosine Transform (DCT) such as JPEG are to be avoided for photographing subjects featuring high spatial frequencies such as text. Ringing around sharp edges becomes visible with increased compression ratios. A proposed alternative to JPEG is JPEG2000, which uses the discrete wavelet transform (DWT) to ameliorate the problem of ringing, while keeping or even improving the memory cost and throughput benefits of JPEG. Unfortunately, `raspistill` does not provide the option of JPEG2000 compression and as a result we have decided to use lossless PNG compression as a trade-off between the two extremes.

B. Pre-processing

Following image acquisition, the raw image of the open book is processed before the OCR stage to make text recognition faster and more robust. Firstly, the position of the camera will remain constant and approximately so will the book's (some movement may occur as the spine of the book shifts). We can use a divide and conquer approach to crop the single input image into a series of smaller sub-images. OCR can then be applied on each sub-image in sync with the requirements of NAO's speech synthesis engine, reducing the page-flip-to-speaking latency and as a result the system's start-up time from rest.

The challenge with this idea arises from sectioning text such that it doesn't get cut off in any sub-image. Since the book location is only approximately constant, we cannot assume that all lines of text will be in the same location across subsequent input images.

Secondly, the pre-processing stage should apply image enhancement filters for bettering OCR performance. Such enhancements have been shown in [8] to boost the performance of open-source OCR software to commercial standards. We have determined empirically that increasing contrast, brightness and black levels, as well as converting the image to greyscale has a positive effect on the resulting text accuracy. Under the assumption that the pages' surface is evenly lit, the previous statement is true; otherwise in some cases it may provide worse results than without image enhancements. For our prototype, the actual filter parameters would need to be tweaked before the demonstration, to accommodate the lighting conditions of the surrounding environment. Despite having the backlight panel as a fixed artificial source of diffuse light, its power output is not enough to overpower that of the sun's, for example, on a clear day.

All our pre-processing algorithms are performed using a combination of OpenCV in Python and command-line programs including Textcleaner (as in [8]).

C. Optical Character Recognition

The final stage of the pipeline is converting the image (or sub-image) with all its applied enhancements into text. There are numerous OCR packages available for Linux. Despite that, only a small number of open-source OCR engines are employed by the majority of these packages. Other proprietary engines for which one must pay a licensing fee for usage permission do exist, though they do not fit within our target budget.

An analysis shown in [9] compares the accuracy of 5 common optical character recognition engines: ABBYY, Cuneiform, GOCR, Ocrad and Tesseract; the first being the only closed-source one on the list. As one would expect, the proprietary software from ABBYY scored the highest – 100% accuracy, but also took on average approximately 5 times longer than the second slowest (Tesseract) to return an answer. A further evaluation of engines ABBYY, Tesseract and the open-source OCRopus in [10] draws a similar conclusion – proprietary OCR engine ABBYY is the superior software available on the market, but Tesseract is the best open-source alternative. Finally, a comparison between ABBYY and Tesseract in [11] concludes that both engines have their merits and demonstrates that there are areas where Tesseract outperforms ABBYY in accuracy, such as when fonts contain glyphs. It is apparent that Tesseract is likely

the best performing open-source engine.

We have decided to choose Tesseract as the backbone of our reading system. The software benefits from being actively maintained and updated by the community as well as from the ability to easily train its classification stage using arbitrary fonts and languages. Furthermore, over 100 languages are supported by Tesseract without any further training, enabling us to expand the scope of our requirements with ease.

V. Evaluation

Initial testing of our system using Tesseract, prior to installing the updated backlight panel illumination solution provided hopeful results. High contrast, clear text on the pre-processed image resulted in 100% accurate OCR, whereas dimly or unevenly lit areas were not at all correct. While the final stage of the software pipeline is slow indeed, it is not slower than the time for a human to orate the open pages at a normal rate. This is favorable as it will manage to keep up with a throughput as per our requirements, while also providing us ample time for the page turning action.

The latency from when a page is flipped to when the pages have been read is high, however, as mentioned in the requirements section, this is only a problem when the reading pipeline begins from rest or from an invalid state (i.e. user has forcefully flipped the page). After the pipeline has been initialized, the latency becomes masked by the throughput of text – the only visible observation will be that what the robot is reading may not necessarily be what the book's currently open pages read.

Our conclusion from the preliminary tests was that lighting plays a more important role in optical character recognition than we initially anticipated. It is the reason we sought to find an adequate source of illumination i.e. the backlight panel.

VI. Future Work

A bright source of diffuse light shining upon a book may lead to good OCR, but could be considered a distraction by the user. An alternative pursuit of adequate illumination could involve building our bespoke backlight panel for infrared (IR) wavelengths and using the IR version of the Raspberry Pi camera instead of the current one. This would make the panel less obtrusive to the user.

Work still needs to be done in linking all components of our system together and getting their output to NAO. This will be done over the Ethernet connection that links the Raspberry Pi to the robot by means of standard networking packets. Finally, the book reading software subsystem will require the introduction of additional logic to enable high-level communication with the speech synthesis engine in the NAO robot.

VII. Conclusion

In conclusion, making NAO read printed books with an embedded system is no mean feat. There are several challenges in achieving this while keeping the robot human friendly, which include computational expense and the need to create special lighting conditions. We mask the latency induced by the former with a sufficient level of throughput and we tackle the latter with a rig made from a recycled LCD backlight. While further work is necessary before PiNAOqio is literate, initial experimentation showed that our aim is achievable.

VIII. Bibliography

[1] J. Elgot, "One In Five See Family For 'Less Than An Hour A Day'," *The Huffington Post,* 1 April 2015.

[2] X. L. Soler, P. Korkontzelos, N. Warner and M. Rusu, "PiNAOqio: Story-telling for children using physical books and HRI," 2016.

[3] M. Sweney, "Printed book sales rise for first time in four years as ebooks decline," *The Guardian,* 13 May 2016.

[4] J. Catone, "Why Printed Books Will Never Die," *Mashable UK,* 16 January 2013.

[5] C. Myrberg and N. Wiberg, "Screen vs. paper: what is the difference for reading and learning?," *Insights,* vol. 28, no. 2, pp. 49-54, 2015.

[6] Raspberry Pi Foundation, "Camera Module v2 Documentation," [Online]. Available: https://www.raspberrypi.org/documentation/hardware/camera/README.md. [Accessed 16 November 2016].

[7] T. Ebrahimi and M. Kunt, "Visual Data Compression for Multimedia Applications," in *Proceedings of the IEEE*, 1998.

[8] vBridge, "How we tuned Tesseract to perform as well as a commercial OCR package," 5 November 2012. [Online]. Available: http://vbridge.co.uk/2012/11/05/how-we-tuned-tesseract-to-perform-as-well-as-a-commercial-ocr-package/. [Accessed 16 November 2016].

[9] A. Gohr, "Linux OCR Software Comparison," 15 June 2010. [Online]. Available: https://www.splitbrain.org/blog/2010-06/15-linux_ocr_software_comparison. [Accessed 16 November 2016].

[10] U. Springmann, "Other OCR engines: ABBYY, Tesseract," Ludwig-Maximilians-Universität München (LMU), 14 September 2015. [Online]. Available: http://www.cis.lmu.de/ocrworkshop/data/slides/m6-abbyy-tesseract.pdf. [Accessed 16 November 2016].

[11] M. Heliński, M. Kmieciak and T. Parkoła, "Report on the comparison of Tesseract and ABBYY FineReader OCR engines," PCSS, 2012.

[12] H. Furness, "Books are back: Printed book sales rise for first time in four years as ebooks suffer decline," *The Telegraph,* 13 May 2016.