A Hardware-based Image Perspective Correction System

Matt Hollands, Patrick Yang

1 Overview

The prevalence of smart phones, equipped with a camera and powerful processor, has led to the development of applications which enable digitisation of a physical document simply by taking a photo. One can simply point the phone's camera at an object to perform the functions of a scanner, enabling easier sharing and collaboration. While convenient and powerful, this method's drawback is that obtaining a direct perspective (as opposed to a skew perspective) with a phone camera can present a challenge. A system utilizing mathematical transformations and edge detection can correct this issue for the user. While this system has previously been implemented in software, we are aiming to implement this solution in hardware, using an FPGA. We will build a system that can take a camera input, automatically detect the corners of a visible rectangular object, and output a perspective-corrected version of that object.

This process requires specifying the location of a document in the image, and performing transformations on the original image to produce a new image of the original document from a direct perspective.

Stretch goals include exporting the data to an SD card, which would allow the system to actually be used for digitization and sharing of documents, as well as performing the operation on live video in real time, which is a greater technical challenge that closer mimics a real-life scenario where a user is holding a phone unsteadily.

Aside from a camera and VGA-equipped monitor, no external hardware is expected to be required for this project.

2 Design

2.1 Complexity and Challenges

The process of rectifying an image's perspective requires a number of computer vision techniques as well as complex mathematical transformations on the pixels of the original image.

Both the mathematics and computer vision required in order to transform the image are complex by themselves; however, implementation on hardware provides further limitations on the mathematical operations available as well as the available memory and time per frame, making the task more difficult.

2.2 Design Decisions

The overarching philosophy that guides our design is to mimic the behavior of mobile software systems that perform the same function, and to design with a human user in mind. To that end, our aim is to automatically detect the corners whenever possible, but to enable the user to have an override if those are unsatisfactory. This will involve displaying the original captured image on screen with an overlay interface showing the locations of the corners. We rely on the user to advance through the flow of data, similar to how the user would need to take a picture and confirm the perspective-corrected image on a mobile phone. At least in our baseline system, we will sacrifice efficiency for ease of use.

Due to the time constraints, we will prioritize development of certain features over others. Because the image transformation module is such a large part of the system and relies on knowing the location of the documents corners, we have decided to develop the manual corner detection system utilizing a direct human interface before attempting automatic detection. This allows one team member to work on the image transformation section, while the other team member works on the computer vision modules, and has the further benefit of being more easily testable and iterable.

Furthermore, due to the complexity of the image transformation algorithm, it may prove infeasible to perform this operation in real time as data comes from the camera. Therefore we have chosen to have the user press a button to capture an image to be rectified. Real time processing will be a stretch goal.

2.3 Block Diagram

Below is a high-level block diagram for the system. Data flow within it is approximately clockwise starting at the upper left (with the camera interface input) and ending at the lower left (with the VGA to monitor output).

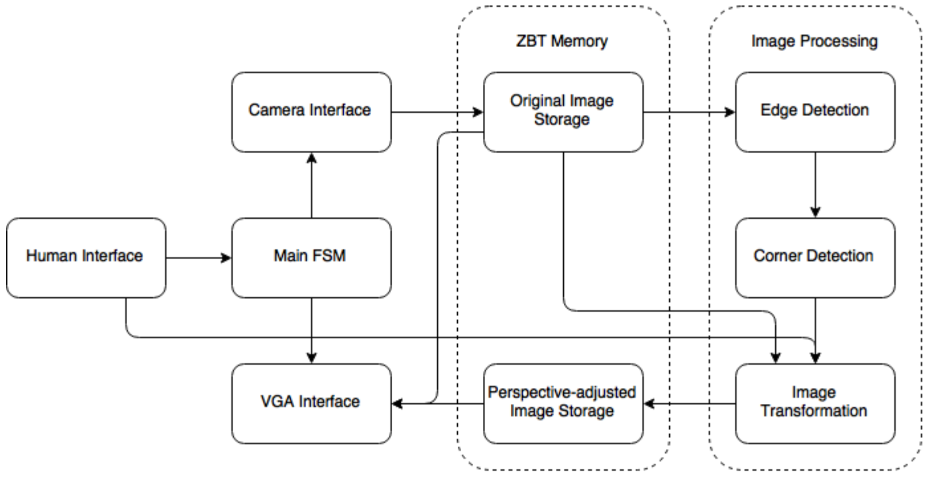

Figure 1: Block diagram for the system.

3 Implementation

3.1 Camera Interface Module

This module will be responsible for interfacing with the camera. It will load data for a single frame from the camera into one memory bank. It is very important to keep this module synchronised with any module reading from memory. If it is not adequately synchronised then the camera interface may overwrite half of a frame with the next frame before the processing of that frame is complete. The synchronisation will be handled by the main FSM.

This module will be tested by outputting the image to the display by using the VGA output module.

3.2 Memory

Both banks of ZBT memory on the labkit will be used. One bank will hold the original frame, as captured by the camera. The other bank will hold the processed frame. This will allow us to keep the original data from the camera while producing the rectified image.

3.3 Main Finite State Machine

The Main FSM module will hold the state of the overall system. Some high level states include:

- Camera Viewfinding - Replicate the camera view on the output display
- Image Capture - Store the camera view when transitioning to this state; present the stored view on the output display
- Image Processing/Transformation - Using either human-provided corner locations (human interface) or automatically detected corners, signal the image transformation module to begin computing transformation.
- Processed Image - Present the processed image view on the output display when image transformation is done.

The main input to this module will be from the Human Interface Module because the transition between high level states will generally be directed by the user.

This module will be tested by using test benches to stimulate the inputs and check that the FSM performs the correct state transitions.

3.4 Human Interface Controller

The human interface controller will handle all input from the user, including human-directed FSM transitions as well as selection of corner locations (before automatic corner detection is implemented). The main method of input from the user will be a PS/2 mouse. Initially, however, the system may only use the buttons built into the lab kit as the mouse input is not core functionality. This module will likely be broken into a number of smaller modules, such as an FSM to handle the PS/2 protocol, and debouncing modules for button inputs.

As a human interface component, this module may be tested manually by providing feedback for the outputs on the LED and seven-segment display areas of the labkit.

3.5 VGA Output Module

As well as the camera image, the display needs to present a GUI to the user. Therefore this module will have to read either the original or processed image data from the ZBT memory (depending on Main FSM output), and also handle sprites for the mouse cursor and onscreen buttons. This module will be broken down into further modules, such as sprites and an FSM to handle the protocol.

This module will be tested by placing an image in memory and attempting to draw it directly to the screen.

3.6 Edge Finder/Corner Finder Modules

These modules are the image processing stage. They will take data from the main memory and attempt to locate the corners of the document in the image. The chosen algorithms for this stage will be Canny Edge Detection and Hough Transform. These will detect the edges of document, which should be the longest lines in the image. Then by finding the intersections of these lines, the corners can be identified. The coordinates of the four corners will be outputted to the Transformer Module.

This module will be tested by inputting known images from memory and checking that the correct values are being outputted.

3.7 Transformer Module

This module takes as input the original image from memory and the coordinates of the four corners. This module will transform the original image into an undistorted image of the original document. This image will then be outputted to the second bank of ZBT memory, which holds a processed image for display. This module may be divided into two submodules, one to compute the transformation matrix between the original and undistorted rectangles, and one to actually perform the pixel mapping.

If possible, this module will be pipelined so that (after some per-frame latency to compute the transformation parameters) it can output the mapped pixels without needing to write them to memory.

This module will be tested by inputting a known image from memory with hard-coded corner locations and checking that the output has correct perspective.

4 Timeline and Responsibilities

4.1 Timeline

Our goal is to complete the desired functionality in the following timeframes.

- Week of Nov 2

  Camera, VGA interfaces with static image; Memory modules

  In this stage, we set up the camera to write to a memory bank and the VGA interface to read from a memory bank, with static images and a very basic human interface or state-controller (e.g. press a button to store camera image to memory, and press a different button to switch between replicating camera and presenting stored image). Since knowledge of memory and camera operation is core to the project, both members are responsible for this.

- Weeks of Nov 9, Nov 16

  Human-interface controller, Image Transformation

  Here we set up the human-interface controller to allow for manual corner selection, updating the VGA interface to display any necessary spriting. The image transformation module (both computing the transformation matrix from corner locations as well as the pixel mapper) is also done. These two modules constitute absolute bare-minimum functionality: with them together, the system can be used for rectilifying an image's perspective. The Image Transformation module is Patrick's responsibility, while the human-interface controller is Matt's responsibility.

- Weeks of Nov 16, Nov 23

  Edge/Corner detection

  In this stage, which may be replaced by a buffer if the previous stage requires it, we set up automatic edge and corner detection based on the input image. These modules together would form the core of real-time image perspective correction, and as such they are not considered top-priority functionality. The edge detection module is Matt's responsibility, while the corner detection module is Patrick's responsibility.

- Week of Nov 30

  Integration, Main FSM, End-to-end testing

  In this stage, we bring together the components using the main FSM, so that the system can function standalone. End-to-end testing is performed at this time, and must be done via collaboration of both members. Any remaining time before checkoff will be dedicated to testing and optimization.

4.2 Responsibilities

As the camera, VGA, memory, and main FSM modules are core to the project, correct functionality and understanding is the responsibility of both team members.

The image transformation and corner detection modules will be Patrick's responsibility.

The human interface and edge detection modules will be Matt's responsibility.

5 Resources

Resources needed for this project include the NTSC camera and a VGA monitor. If attempting the stretch goal of using a mouse for the human interface controller, then a PS/2 mouse will also be required. No other electronic components will be necessary; as for non-electronic resources, black construction paper will be used during testing of the edge detection.

6 Conclusion

Rectifying an object's perspective is a process with real-life application which hides substantial complexity behind an easy to use interface in most software implementations. Our main challenge with this project will be to optimize the calculations to run smoothly and accurately on the FPGA, which is limited in computation resources and mathematical precision. We believe that this project will teach us much about computational complexity in digital systems.