Spatio-Temporal Video Amplification

Akashnil Dutta, Rishi Patel, Pranav Kaundinya

I. Introduction

Visualizing small changes that are indiscernible to the human eye is an interesting challenge in video processing. Given that modern cameras have the ability to detect minute motions, processing video to amplify these motions can give significant insight. This has a wide range of applications. For instance, in biological imaging of cells, one may be interested in visualizing tiny movements as seen through a microscope. This project implements a real-time video-processing system that will enhance small changes in videos.

Our system will be implemented using an FPGA and will generate output video that amplifies two aspects of visible changes– a change in intensity or color over time, and translational motion of objects. However, as described below, these two aspects are closely related. Our system will implement discrete-time equations to obtain and amplify the effect of translational and temporal changes on intensity and combine the effects to produce an amplified image. The system will consist of a preprocessing stage, a translation amplifier, and a magnitude change amplifier. Initially we will work on processing grayscale video, and later on color video. If time allows, we will extend this framework to allow frequency selective amplification. In such a scheme, the user could adjust a parameter that lets them see phenomena that occur around frequencies of interest.

We begin with an explanation of the theory behind our project. We then describe the modules in the hardware implementation, followed by a timeline that outlines when various components will be expected to work. Results from some software simulations in MATLAB are also discussed.

II. Theory

In this section we'll discuss the theoritical basis of amplitude magnification. Assume that the video input is represented as a continuously differentiable ($\mathbf{C}^1$) function

$$I : [0, X] \times [0, Y] \times [0, T] \to V$$

such that $I(x, y, t)$ represents the intensity of pixel (x, y) at time t. V can be either a scalar field for a grayscale video or in general a vector space with RGB components for a colored video. We want to obtain a function $J(x, y, t)$ from I which amplifies the small changes in I with respect to time.

First consider the problem of magnitude amplification. In this case, we can amplify the temporal variation of each point independently. We just add a gain of μ to rate of change of I with respect to time.

$$J(x, y, t) = I(x, y, t) + \mu \frac{\partial I}{\partial t}(x, y, t)$$

To extend this to motion amplification, we will first consider I in an implicit form, $f(x, y, z, t) = I(x, y, t) - z$. The video stream is precisely the space of solutions of the equation $f = 0$. In general we have to map each point (x, y) and magnitude z to another point (x', y') and corresponding magnitude z' so that

$$x' = x + \lambda \frac{dx}{dt}, y' = y + \lambda \frac{dy}{dt}, z' = z + \mu \frac{dz}{dt}$$

Here, λ is the translational gain factor. The derivatives are obtained by implicit differentiation of the constraint relation $f(x, y, z, t) = 0$, i.e.

$$\left.\frac{dx}{dt}\right|_{f=0} = -\frac{\partial f/\partial t}{\partial f/\partial x}$$

etc. Let the partial derivatives $\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}, \frac{\partial I}{\partial t}$ by I_x, I_y, I_z respectively. We can simplify the above as

$$\frac{dx}{dt} = -\frac{I_t}{I_x}, \frac{dy}{dt} = -\frac{I_t}{I_y}, \frac{dz}{dt} = I_t$$

Hence we get

$$\begin{aligned} x' &= x - \lambda I_t / I_x \\ y' &= y - \lambda I_t / I_y \\ z' &= z + \mu I_t \end{aligned}$$

To produce an output video from this mapping, we need to find the intensity z' of an arbitrary point (x', y') in the proposed image. To do that, we first find pre-image point (x, y) in the original image from (x', y') of the proposed image, then take $z = I(x, y, t)$ and use the forward mapping to get proposed magnitude z' from z.

The inverse (x, y) of (x', y') is approximately $(x' + \lambda I_t/I_x, y' + \lambda I_t/I_y)$.[1] The corresponding z for (x, y) is then given by $I(x' + \lambda I_t/I_x, y' + \lambda I_t/I_y, t)$. z', or J is the image of this under the map is $z + \mu I_t$. Hence the final function is given by

$$J(x, y, t) =$$

$$\boxed{I\left(x + \lambda \frac{\partial I/\partial t}{\partial I/\partial x}, y + \lambda \frac{\partial I/\partial t}{\partial I/\partial y}, t\right) + \mu \frac{\partial I}{\partial t}} \qquad (1)$$

In the discrete domain this formula is equivalent to

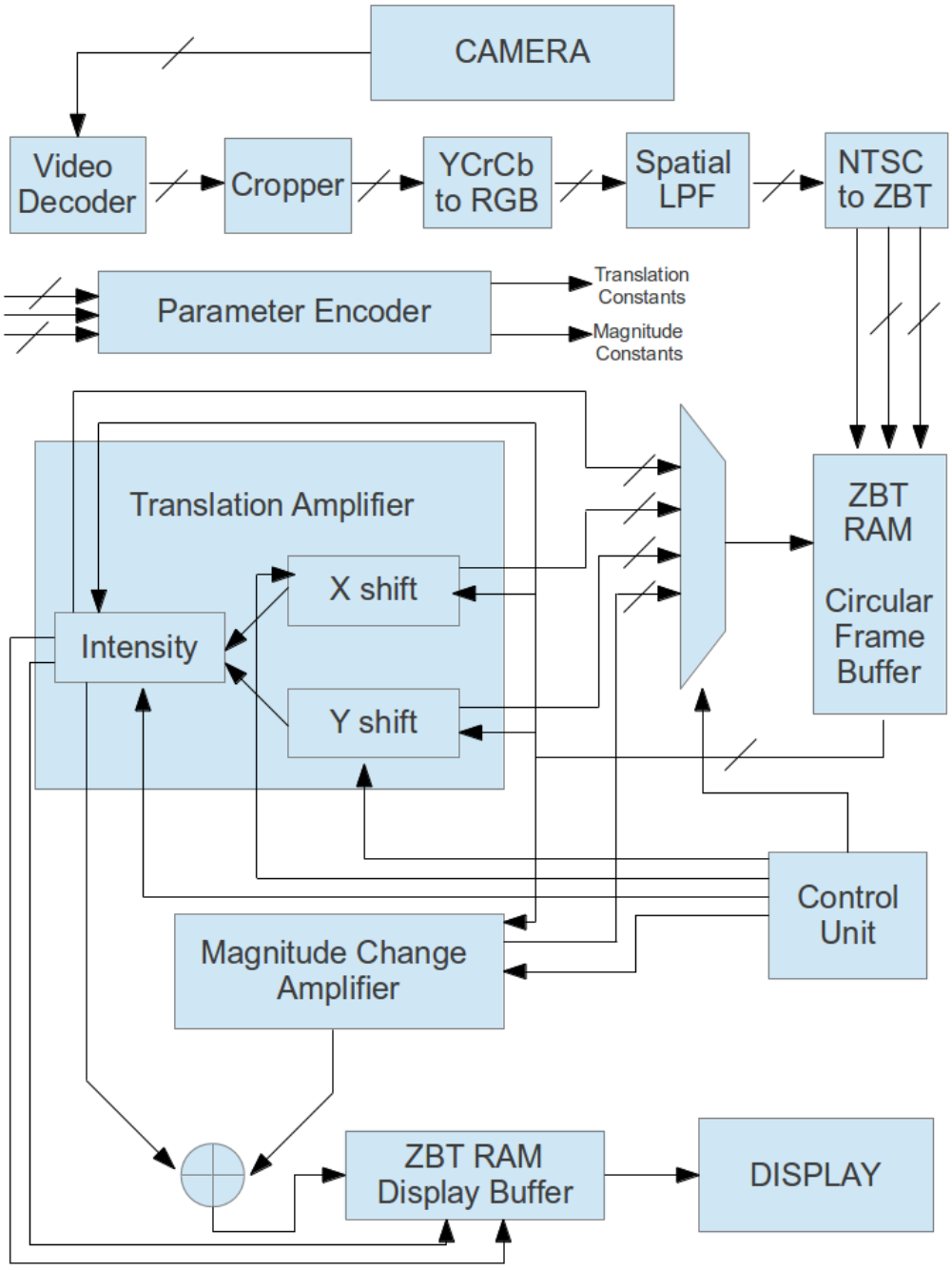
$$J(x, y) = I(x + \lambda \frac{I(x, y, t) - I(x, y, t-k)}{I(x, y, t) - I(x-k, y, t)},$$

$$y + \lambda \frac{I(x, y, t) - I(x, y, t-k)}{I(x, y, t) - I(x, y-k, t)}, t)$$

$$+\mu \frac{I(x, y, t) - I(x, y, t-k)}{k} \qquad (2)$$

Here k is a parameter which specifies the window length for calculating the derivative. Its value depends on the noise characteristics. It is also possible to correct for noise with multiple data points using Richardson Extrapolation for instance.

[1] Here we are making an assumption that I_x, I_y, I_t values are the same at (x', y', t') as (x, y, t). This assumption does not work well for large λ, but makes the processing simpler in our implementation. However, it is not necessary to assume this for the theoritical treatment. We can invert the mapping exactly to do this as well, at the cost of a more complex hardware.



Block Diagram

III. Implementation

We will use above discrete-time equations to implement a system as illustrated in the block diagram above. The various modules in our implementation are proposed below.

3.1 Video Decoder

- Inputs - NTSC Video
- Outputs - NTSC Video-decode
- This module has been pre-written, and will be implemented in our design.

3.2 Cropper

- Input - NTSC Video-decode
- Output - VIDEO-cropped
- The cropper module will crop input images to the desired size. This will allow faster processing, and ease requirements on memory storage. It will take NTSC input signal VIDEO and output a gray-scale VIDEO-cropped.

3.3 YCrCb to RGB

- Input - VIDEO-cropped
- Output - VIDEO-RGB
- This module performs color-code conversion, and has been written already.

3.4 Spatial LPF

- Input - VIDEO-RGB
- Output - VIDEO-filtered
- The spatial low-pass-filter module will smooth out the input image to reduce noise. It will implement an averaging difference equation that replaces the intensities of each pixel with the average of surrounding pixels. The module will implement weak-averaging so that significant blur is not introduced.

3.5 NTSC to ZBT

- Input - VIDEO-filtered
- Outputs - WE, WAddr, Data-in
- Generates necessary control signals to store video data in the circular frame buffer.

3.6 Circular Frame Buffer

- Inputs - WE, Write, RAddr, WAddr,
- Output - Read
- Here we will store the last 5 to 10 image frames for processing. This will use the ZBT RAM and store the cropped low-pass filtered images.

3.7 Control Unit

- Outputs - REnable signals to X-shift, Y-shift, Intensity, Magnitude Change.
- The control unit synchronizes the magnitude-change amplification module with the translation amplification module. It accomplishes this by providing read enable signals to these modules. It also provides the select to the mux that inputs the Raddr to the ZBT RAM.

3.8 Magnitude-Change Amplifier

- Inputs - Intensity-in, REnable, Params
- Outputs - Intensity-amp, RAddr
- This module implements a temporal edge detector that performs discrete-time differentiation. The output pixel is calculated as the second term in equation (2). This will amplify changes in intensity in the single-pixel time-series.

3.9 Translation Amplifier

- Inputs - Intensity-in, REnable, Params
- Outputs - Intensity-out, WE,WAddr
- This module highlights spatial variations in the output by amplifying the coordinate translations of moving objects in the input image. The module amplifies motion in the video input according to the first term in discrete-time equation No. 2. There are three sub-modules:

  1. X shift: computes the shift in the x coordinate (computes the discrete-time derivative)

  2. Y shift: computes the shift in the y coordinate (computes the discrete-time derivative)

3. Intensity: Uses x,y shifts to compute a new coordinate and obtain its intensity from the ZBT frame buffer. Then it outputs this to the display buffer along with appropriate address and enable signals.

3.10 Parameter Encoder

- Inputs - Reprogram, Param-select, Param-value
- Outputs - Translation constants, Magnitude constants
- This stores our constants, such as gains and filtering window sizes. This will be initialized at RESET but can be reprogrammed by the user.

3.11 Display Buffer

- Input - Data-in, WE, WAddr, REnable,RAddr
- Output - Disp-OUT
- This serves a single frame buffer for constructing the output image on the VGA display.

IV. Timeline

- Nov 11th - ZBT RAM and preprocessing modules will be implemented.
- Nov 18th - Demonstrate temporal amplification on grayscale images (MATLAB functionality shown in this proposal
- Nov 29th - Demonstrate spatial amplification on grayscale images.
- The remaining time will be used for testing and debugging. If time permits we will modify our design for full color operation.

V. Individual Resposibility

We'll roughly divide the work into three parts as follows

- Akashnil - Translation amplification module, ZBT circular buffer
- Rishi - Temporal amplification module, Output Interface, Parameter encoder
- Pranav - Spatial LPF, Control Unit, Cropper, ZBT display buffer

VI. Related Work

We were motivated to take up this problem for our project by the work of Freeman et. al. from MIT CSAIL (*Eulerian Video Magnification for Revealing Subtle Changes in the World*)

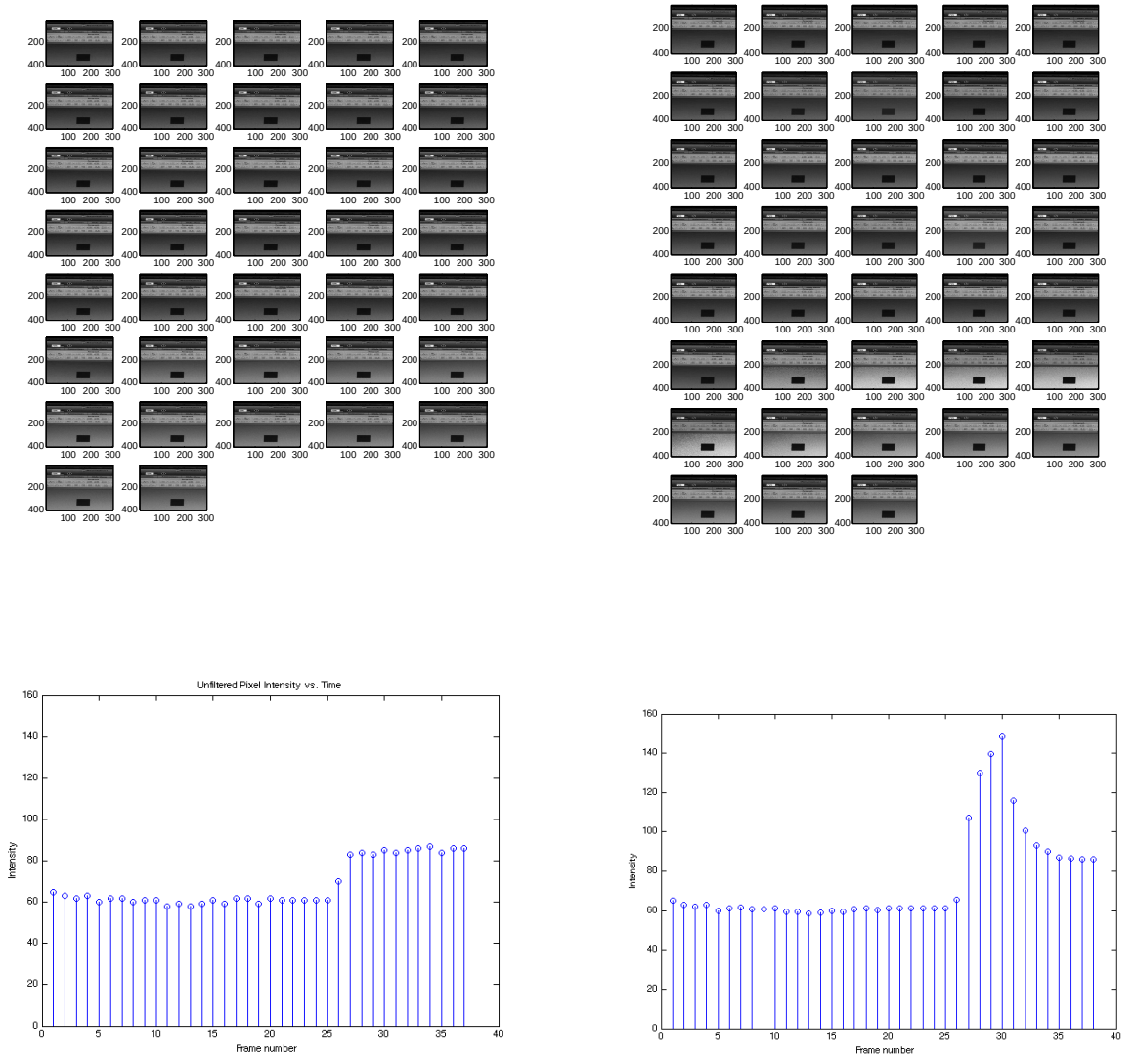


Figure 1: *In the top two figures the results from our matlab experiment are shown on a greyscale video stream. The left set of frames is the original video with small change in brightness. The signal is temporarily magnified in the corresponding frames of the right panel. The plots in the bottom depict the variation in intensity of a sample pixel.*