

# A photogrammetry system

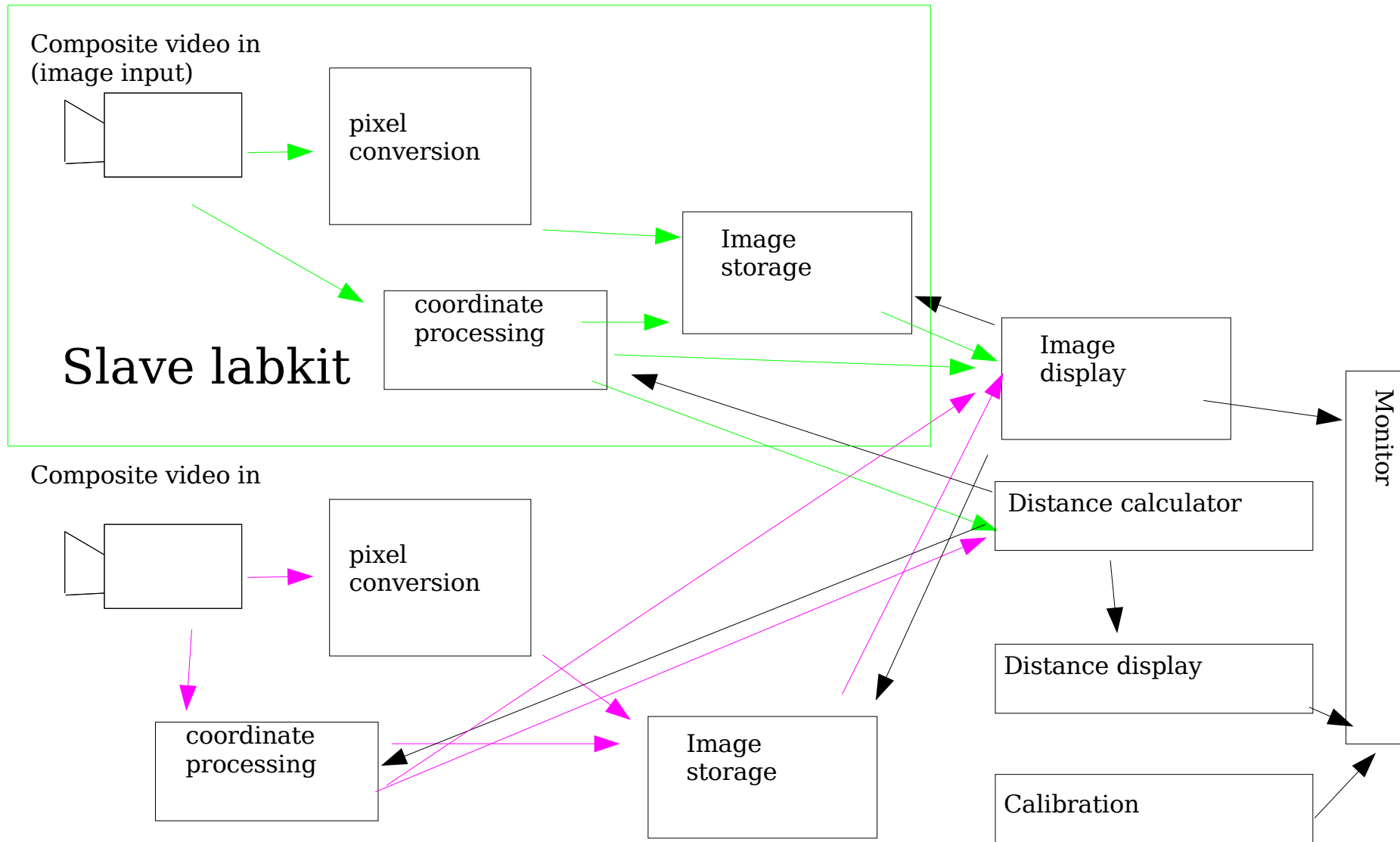
Miles Colman- 11/14/05

Objective: determine the distance to an object from two photographs of the same object

parts that a working system will have:

- Image input (x2)
- pixel conversion from YUV to RGB (x2)
- coordinate processing (x2)
- Image storage (x2)
- Distance calculation
- Display of distance
- Display of images
- Calibration

# Higher level overview



implementation: input

Will use Javier's module for system inputs:

- h sync
- v sync
- 24-bit YUV

implementation : pixel conversion

Will produce 24 b RGB data from 24 b YUV data

Multiply YUV data by constants to get RGB data:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

(from wikipedia.org)

## Lower level : coordinate processing

### Inputs:

- h sync
- v sync
- field
- 8 bit luminance (Y)
- used (from distance calculator)

### Outputs:

- x coordinate
- y coordinate
- averaged x coordinate
- averaged y coordinate
- new

### Implementation:

- x coordinate counts with clock (every other cycle)
- x coordinate resets on h sync
- y coordinate counts h sync pulses
- Y coordinate resets on v sync
- coordinate processing unit stores x and y values for pixels exceeding a threshold luminance, and keeps track of how many pixels exceed the threshold
- on a v sync, the coordinate processing unit calculates the centroid of the x and y values
- new pulses high when a new averaged coordinate is calculated
- Used pulses high when the new averaged coordinate is used by the distance calculator

## Lower level : coordinate processing : centroid calculation

- Store the **n** x and y coordinate values that exceed luminance threshold in a frame
- During v sync pulse, calculate average x and y values:
  - 1)Add first n x and y values from memory
  - 2)Divide x and y sums by **n**
- Storing 720 x 520 pixels 30 times per second requires a bit rate of 11 Mb/s
- storage required (worst case) is  $720 \times 520 \times 20 \text{ bits} = 7 \text{ Mb}$  (requires SIMM)

## Lower level : image storage

### Inputs:

- 10 bit x coord (from coord proc)
- 10 bit y coord (from coord proc)
- 24 bit RGB data (from converter)
- 10 bit x coord (from image display)
- 10 bit y coord (from image display)
- pixel request (from image display)

### Output:

- 24 bit RGB data (to image display)

- Storage unit uses x and y coordinate values to address 24 bit pixel RGB data
- size requirement is  $2 \times 520 \times 720 \times 24 \text{ b} = 18 \text{ Mb}$
- bitrate is  $2 \times 520 \times 720 \times 30 \text{ fps} = 22 \text{ Mb/s}$
- will using SIMMs on labkit

## Lower level: distance calculator

Inputs:

10 bit x av pixel for Image1

10 bit y av pixel for Image1

10 bit x av pixel for Image2

10 bit y av pixel for Image2

outputs:

X [10 bit position]

Y [10 bit position]

Z [10 bit position]

Distance is calculated from average x and y pixel values for two images and a matrix defining an affine transform

Distance calculation requires knowledge of 12 constants defining transformation from image to 3D space

constants are determined by using calibration feature to

- get pixel locations in image
- using Matlab script to determine coefficients
- coding coefficients into verilog

## Lower level: distance calculator: input timing

- On reset, distance calculator checks the interval between the two camera's done signals
- the shorter interval is used to pick the first camera
- the 3D position from the second camera's previous 'done' signal is stored
- when a new position becomes available from the second camera, a new 3D position is calculated
- the two 3D position values are averaged weighted according to how much time passed between the two frames from the second camera

## Lower level: calibration module

Calibration mode shows the position of a user-selected point in the image coordinate system

inputs are:

switch and direction buttons

outputs are:

10 bit cursor y (to monitor)

10 bit cursor x (to monitor)

show calibration flag

If calibration mode is on (controlled by user with switch):

- a + is displayed on the image along with the +'s coordinates in the image coordinate system
- the position of the + is displayed below the image
- the position of the + is controlled with the arrow keys

## Image display : lower level

Image display is a wrapper that accesses the stored images and outputs the rgb value for a given pixel input:

- 11 bit x position (from monitor)

- 10 bit y position (from monitor)

- 24 bit pixel value (from image storage) (x2)

output:

- 10 bit x position (to image storage) (x2)

- 10 bit y position (to image storage) (x2)

- pixel request (x2)

## Position display: lower level

input:

- 10 bit {X,Y,Z} data

- 11 bit x position

- 10 bit y position

output:

- 24 bit pixel that uses text display module

## Monitor: lower level

### Inputs:

- 24 bit RGB pixel for image display (x2)
- 24 bit RGB pixel for text display
- 10 bit x value for calibration pixel
- 10 bit y value for calibration pixel
- show-calibration flag

### Outputs:

- 24 bit RGB signal to monitor
- h sync
- v sync