HaViMo3.0

Computer Vision Module

October 13, 2015

Features

- Integrated 2MPix Color CMOS Camera
  - Frame Resolution: 160*120 Pixels
  - Color Depth: 12 bits YCrCb
  - Frame Rate: 20 Fps
  - Full Access to all CMOS Camera registers
    - Saving values in Flash, no need to reconfigure after power on
    - Auto / Manual Exposure, Gain and White balance
- Color-Based Image Processing
  - Integrated Color Look-up Table

    - Saved in FLASH, No need to recalibrate after power on
    - Up to 256 Objects can be defined
    - 3D viewing and editing tools
    - Real-time LUT overlay on the Camera Image
  - On-line Region-growing
    - Detection of up to 15 contiguous Regions per Frame
    - Reporting Color, CoG, Number of Pixels and Bounding box for each region
    - Adjustable Noise / small Region filtering
  - On-line Gridding
    - Reduces the Resolution of the Image to 32*24
    - Minimum Loss of Information using Object Priority
    - Reports Color and Number of Pixels for each 5x5 Cell
  - On-line GVG (Gradient Vector Gridding)
    - Reports image edges in a grid of 20*15
    - Position, direction and intensity for every edge candidate
    - Suitable for form-based object detection
  - Average Image
    - reports a grid of 32x24 of average intensities
    - Suitable for line following or low resource processing
  - Raw image output in both calibration and implementation modes
    - Full Frame Output at 0.5 FPS

- Supported Hardware
  - Half Duplex
    - ROBOTIS TTL bus. CM5, CM510, CM530, ROBOTIS OpenCM
  - Full Duplex
    - Arduino
    - Any microcontroller with UART

- Supported Software
  - HaViMoGUI0.4 and upwards

1 Introduction

HaViMo3.0 is a computer vision solution for low power microprocessors. It is equipped with a 2 Mega-pixel CMOS camera chip and a microcontroller which performs the image processing. The results are then accessed via serial port. In HaViMo3.0 several hardware and software features are improved.

Region growing algorithm is capable of recognizing colored blobs in the image. It reports the bounding box as well as the centroid and number of pixels in the blob.

Gridding algorithm is a suitable pre-processing step for many other applications such as shape based object recognition and self localization.

GVG algorithm is a fast and flexible edge detection algorithm, which facilitate shape based object recognition by reporting edge candidates.

HaViMo3.0 is compatible with all ROBOTIS controllers as well as microcontroller platforms capable of establishing a UART connection. Communication with Arduino is performed via soft-serial port.

2 Hardware Setup

Figure 1 shows the pin out of the module.

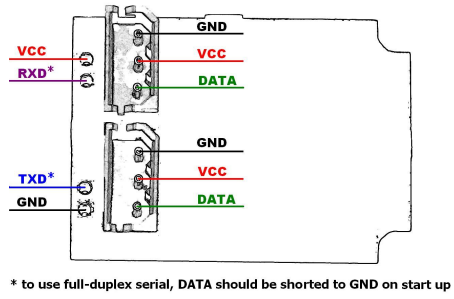


Figure 1. Pin out of the Module

The module can be used in two different types of configuration according to the hardware platform it is used in conjunction with.

Calibration Mode: The module is connected to a PC where a GUI facilitates the access to camera parameters as well as the color look-up table. in this mode, the camera chip can be configured and color to object associations are established by user according to the lighting conditions.

Implementation Mode: The Module is connected directly to the controller and sends computer vision results directly to it. No PC is needed in this mode.

3 Function Description

HaViMo3.0 is accessed on a serial bus. A communication protocol is designed for accessing the device which works on a command/response basis. A command packet contains an instruction which invokes a function of the device, reads or writes values or a combinations of these. In this section the function of the device is described.

3.1 Communication Protocol

HaViMo3.0 supports both half and full duplex communications in physical layer. The half duplex communication is compatible with *ROBOTIS* TTL bus. Full duplex communication can be used to access the module from all other platforms. Following diagram shows a summary of command and response packets in both operation modes.

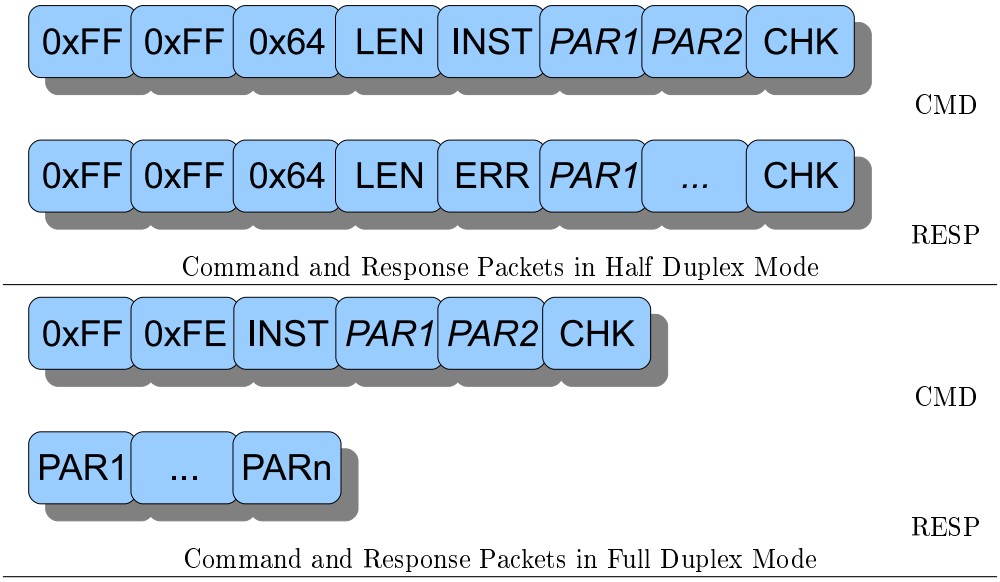


Figure 3. Command and Response Packets Half and Full Duplex Modes

3.1.1 Communication Protocol in *ROBOTIS* Mode

To understand the communication protocol in *ROBOTIS* mode, it is recommended to read *Dynamixel AX-12* data sheet. The Command packet is structured as follows.

Header 2 times 0xFF.

ID Fixed on 0x64 = 100.

LEN Number of bytes to be further transmitted.

INST Instruction code described in table 1.

PARn Optional parameters passed to the instruction.

CHK Check sum is calculated as complement of the lowest 8 bits of the sum of all bytes in the packet excluding the header.

The response packet has a similar structure as the command packet, however the instruction is replaced with an error indicator and parameters are filled with the results of running the instruction.

3.1.2 Communication Protocol in *full duplex* Mode

The command packet is structured as follows.

Header The sequence 0xFF 0xFE, which also includes the fixed ID = 30

INST Instruction code described in table 1.

PAR1,PAR2 Parameters associated with the instruction. Note that The number of parameters MUST always be 2

CHK Check sum is the lowest 7 bits of the exclusive or of the instruction and the parameters.

3.2 Instructions

Following table shows available instructions in HaViMo3.0.

Instruction	Value	Params.	Function
PING	0x01	0,0	No action. Used for obtaining a Status Packet
READ	0x02	addr,cnt	Read Results of Region Detection
WRITE	0x03	addr,data	Equivalent to CAP_REGION for Compatibility
READ_REG	0x0C	addr,cnt	Read Camera Chip Registers
WRITE_REG	0x0D	addr,data	Write Camera Chip Registers (1)
CAP_REGION	0x0E	0,0	Capture and Find Color Regions (1)
RAW_SAMPLE	0x0F	0,0	Sample the Raw Image (used by GUI) (2)
LUT_MANAGE	0x10	0,0	Enter LUT Manage Mode (used by GUI) (2)
CAP_GRID	0x15	0,0	Capture and Compress using Gridding algorithm (1)
READX16	0x16	addr,data	Read Results of the Algorithms (see note 3)
RAW_INTERL	0x17	0,0	Sample Raw Image Interlaced (skip 5 lines)
CAP_AVG	0x1C	0,0	Capture image and run Average Algorithm
CAP_GVG	0x1D	0,0	Capture image and run GVG Algorithm

(1) No return packets are generated for these instructions.
(2) Response is different from standard packets.
(3) The given address is internally multiplied by 16

Table 1. Available Instruction in HaViMo3.0

PING This instruction is used to check whether the device exists and is ready to receive the next instruction. The instruction returns an empty status packet.

READ This instruction is used to read the results of image processing algorithms. This command accepts multi byte read but can only access 255 bytes. See READX16 for extensions.

WRITE A write to an arbitrary address simulates a *CAP_REGION* instruction.

READ_REG This instruction is to read the content of camera registers. It is the same as *READ* instruction. This command accepts multi byte read.

WRITE_REG This instruction is to write the content of camera registers. It is the same as the *WRITE* instruction but it accepts only single byte write.

CAP_REGION This instruction starts capturing and processing of the next available frame. The processing algorithm used in this instruction is *Region Growing.* It takes approximately 50 ms to process a full frame. The main CPU should pole the functionality of the device using the *PING* command before sending the next instruction. The results can be then accessed using *READ_REGION* instruction.

RAW_SAMPLE With this instruction the camera module transmits a full frame of raw image data. This instruction is used when the GUI receives a request to sample a raw image. This instruction does not use the Standard packet protocol.

LUT_MANAGE After receiving this instruction, the module enters the programming mode, in this mode the device accepts no more packets, but other instructions assigned to manage the look-up table, such as erasing, reading and writing into it. This instruction is used by User interface during calibration phase and should not be used in implementation phase.

CAP_GRID This instruction invokes the gridding algorithm. The algorithm compresses the image into a 32*24 cell grid and reports the number of pixels and the color observed in the 5x5 window related to the cell. Only one color is accepted for each cell. Lower color codes dominate the higher ones but Unknown = 0 has the lowest priority.

READX16 This instruction reads the results of image processing algorithms. It has a similar structure to READ instruction but the given address is internally multiplied by 16. This gives access to a wider range of addresses, however at least 16 bytes should be read to cover the whole space.

RAW_INTERL With this instruction the camera module transmits a full frame of raw image data using interlacing. This instruction is used when the GUI receives a request to sample a raw image and a higher bandwidth is available. This instruction does not use the Standard packet protocol.

CAP_AVG The instruction invokes the averaging algorithm. It reports an average intensity for every 5x5 pixel. The results can be accessed via READX16.

CAP_GVG Runs the GVG algorithm on the camera image. The algorithm provides a 20x15 grid of cells, each presenting an edge point with its position, direction and strength.

3.3 Image Processing Algorithms

HaViMo2.5 is equipped with two image processing algorithms, which are described in this section. In the first step both algorithms translate color values to object codes using the built-in look-up table. Therefore an exact calibration of the colors should have a great impact on the results of the recognition.

3.3.1 On-line Region Growing Algorithm

The goal of the region growing algorithm is to detect contiguous color blobs in the image. A 4-pixel neighborhood is used to determine connections. The function is invoked using the instruction *CAP_REGION*. The detected regions are summarized using the following parameters:

Value	Bytes	Description
Index	1	Contains zero if the region is invalid and nonzero otherwise.
Color	1	Color code of the detected region (0 = Unknown, 1 = Ball , ...)
Pixels	2	Number of detected pixels inside the region
SumX	4	Sum of the X coordinates of the detected pixels (*)
SumY	4	Sum of the Y coordinates of the detected pixels (*)
MaxX	1	Bounding box right margin
MinX	1	Bounding box left margin
MaxY	1	Bounding box bottom margin
MinY	1	Bounding box top margin

(*) Values can be divided by the number pixels to obtain the center point.

Table 2. Data Format of the Results of Region Growing Algorithm

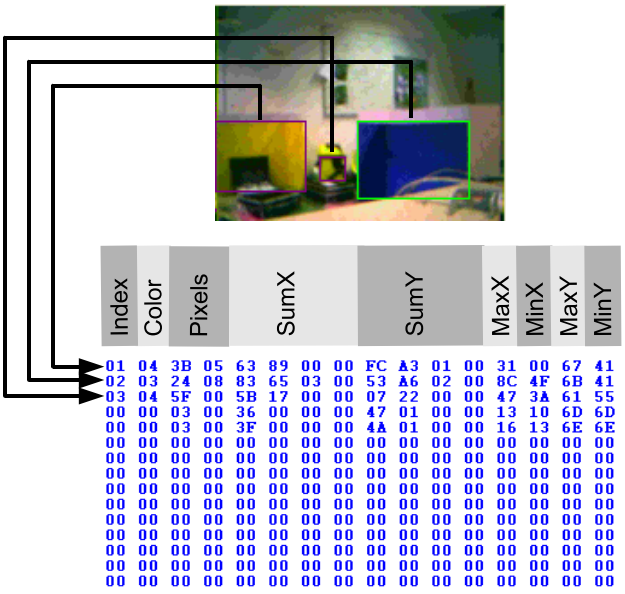
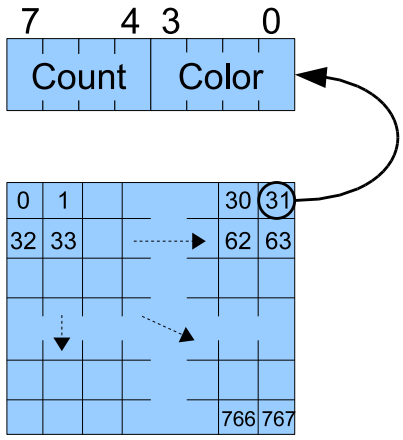


Figure 5. Results of the Region Growing Algorithm

To access the results, the instruction *READ* should be used. Up to 15 regions can be read from the address range 0x10 to 0xFF using the *READ* (0x02) instruction. Following example shows the produced output of the module according to a given image.

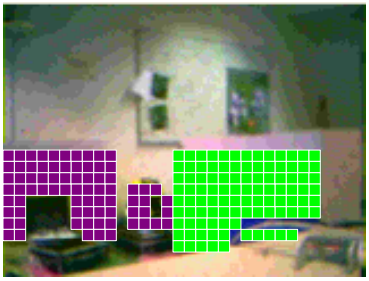
3.3.2 On-line Gridding Algorithm

After invoking the gridding algorithm with *CAP_ GRID* instruction, it compresses the image into a grid of 32*24. Each cell is a representative for a 5x5 square on the original image. For each cell one byte is calculated which contains the flowing information. The lower 4 bits determine which color was the most prior detected in the square. The higher 4 bits are the number of pixels occupied with this color. If over 15 pixels are detected the count remains 15.



The algorithms results 768 bytes of data. These can be read using the *READX16* instruction. Note that the given address is internally multiplied by 16 to increase the access range. It is therefore required that a multi-byte read operation with at least 16 bytes be used.

Following example shows the result of a the gridding algorithm. Note that only the color is visualized.



3.3.3 Gradient Vector Gridding (GVG)

The GVG algorithm calculates a 20*15 grid of image edges. The algorithm is invoked with *CAP_GVG* instruction, The result is a grid of 20*15 of 8 bit cells. Each cell is a representative for the dominant edge in an 8x8 window of the original image. The lower 3 bits of each cell determine the orientation of the most prior detected edge in the cell. The next 3 bits contain the displacement of the edge from the center of the cell as a signed value. The higher most 2 bits decode the strength of the edge.

The algorithm results 300 bytes of data. These can be read using the *READX16* instruction. Note that the given address is internally multiplied by 16 to increase the access range. It is therefore required that a multi-byte read operation with at least 16 bytes be used.

3.4 Calibration Functions

In order to calibrate the device, there are several instructions reserved for making a raw sample form the camera image, and accessing the look-up table. These section describes how these instructions are used.

3.4.1 Downloading a Raw Image Sample

The instruction RAW_SAMPLE (0x0F) is used to get a complete frame of camera image form the module. As the result, 19200 bytes of data are sent back over the bus. The amount of data corresponds a full frame of 160*120 pixels in YCrCb 422 format, however it is derived form several consecutive frames. This is to reduce the data rate and avoid a buffer over flow in systems with low speed serial communications. In *RAW_SAMPLE* instruction the image sample is obtained from 40 consecutive frames. The correspondence is described in Figure 6.

0,0	1,0	2,0	3,0	
0,1	1,1	2,1	3,1	
0,2	1,2	2,2	3,2	1st Frame
...	...	...	...	
0,119	1,119	2,119	3,119	
4,0	5,0	6,0	7,0	2nd
4,1	5,1	6,1	7,1	Frame
...	...	...	...	...
156,0	157,0	158,0	159,0	
...	...	...	...	40th Frame
156,119	157,119	158,119	159,119	

Figure 6. Downloaded data stream and its correspondence to the image using *RAW_SAMPLE*.

bits	7-4	3-0
Even Addresses	Y	Cr
Odd Addresses	Y	Cb

Figure 7. YCrCb 422 data format

3.4.2 Color Look-up Table Access

By invoking a LUT_MANAGE command, the module enters its look-up table access mode. The request is replied with a '*' character. In this mode normal commands are disabled and a new set of commands are activated to access a part of the flash, where the color look-up table resides.

Instruction	Val	Par. #	Function	Returns
ERASE	'e'	0	Erase the color look-up table	0x0D
SET_ADDR	'a'	2	Set the current address pointer	0x0D
WRITE_WORD	'w'	2	Write 16 bits of data in the page buffer	0x0D
WRITE_PAGE	'm'	0	Write page buffer into flash	0x0D
READ_WORD	'r'	0	Read 16 bits of data from the color look-up table	B1 B2 0x0D
EXIT_LUTMAN	'x'	0	Exit from LUT manage mode	Status Packet

Table 1. Instruction in LUT_MANAGE mode

Note that due to the access to the flash, some commands may need several of milliseconds to complete. It is therefore necessary that the client waits for an acknowledge form the module.

Valid address range is between 0x0100 and 0x10FF. Values outside this range are capped internally.