Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

Year 2 - Mars Rover Group Project 2022

ELEC50008 - ENGINEERING DESIGN PROJECT 2 2021-2022

DR ADAM BOUCHAALA

DR EDWARD STOTT

MRS ESTHER PEREA BOROBIO

Authors:

Michal PALIČ (CID: 01942994, EIE Year 2)

Vaclav PAVLÍČEK (CID: 01868933, EIE Year 2)

Marco CHAN (CID: 01853132, EIE Year 2)

Nikhil NARAYANAN (CID: 01844915, EIE Year 2)

Timothy NEWMAN (CID: 01880918, EEE Year 2)

George ZHANG (CID: 01874168, EEE Year 2)

Prateek WAGLE (CID: 01956935, EEE Year 2)

June 22, 2022

expected frequency of 366Hz in order to confirm the target.

To solve this issue, a frequency detector, which converts frequency-to-voltage, is used. The signal is first passed into a comparator to generate square waves and then into the LM331 chip which outputs a PWM signal with the average voltage proportional to the frequency. The average voltage is obtained by using a low-pass filter. As seen in Fig. 7.2, the output voltage from the frequency detector increases linearly with frequency and the acceptable frequency range is mapped to 2.925-3.087V.

7.3 Integration with Rover

The rover ESP continuously checks if the frequency detector output indicates a frequency of 366 ± 25 Hz. If so, the radar subsystem communicates to the control subsystem to pause the alien mapping algorithm and instead run an algorithm for locating the bunker.

The first step of this algorithm utilises a perturb and observe algorithm where instructions are sent to the drive subsystem to rotate in small increments and uses averaged peak values to find the maximum signal strength. For the averaging function, 50 samples from the peak detector are taken at 1kHz frequency which is greater than the Nyquist frequency required to sample a ripple of approximately 366Hz. A similar algorithm is subsequently used for straight line distance to target. Once the location of maximum signal strength is found, the algorithm rotates the rover both clockwise and anti-clockwise until a second similar strength peak is found. The midpoint of the peaks is determined to be the center of the fan. This is to account for the lack of motion at the center of the fan as found earlier.

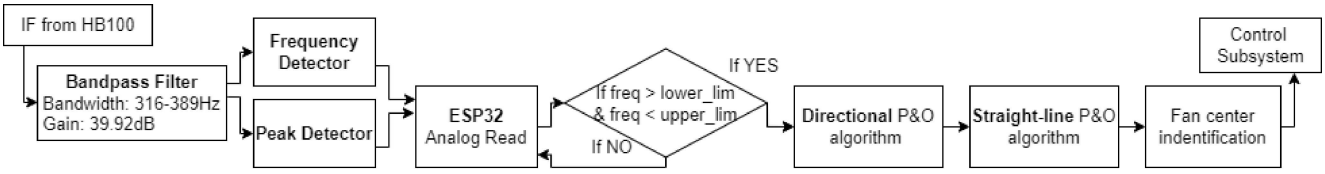


Figure 7.3: Block diagram and flowchart of radar subsystem

8 Vision Subsystem

8.1 Design Overview

The vision subsystem consists of three run-time re-configurable image processing pipelines implemented on the Intel DE-10 Lite FPGA development board. These were synchronized to the camera stream which can be seen in Figure 8.1. Since the control system requires accurate data even when the rover is in motion, the latency of the vision subsystem was a priority. Pipeline 0 is more complex, as it also includes modules building detection. Additional filtering is a requirement for Pipeline 0 due to alien detection, but is also useful for processing for more noise prone alien colours such as yellow. Pipelines 1 and 2 are only equipped for alien detection to conserve resources.

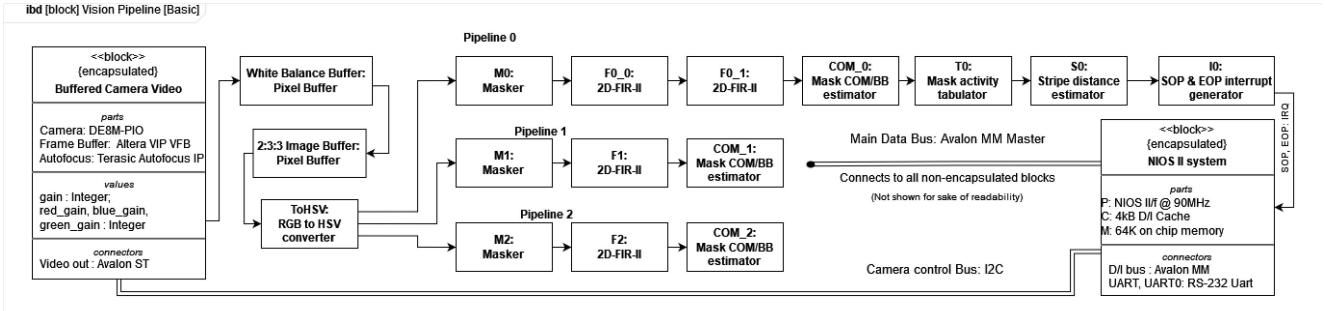


Figure 8.1: SysML internal block diagram of Vision Pipeline

8.2 Design Process

The HDL digital project management principles of problem decomposition, subsystem testing, and integration testing outlined in [18, p. 766] were applied. The system was first prototyped using Python and openCV in order to verify the feasability of the approaches undertaken. As part of the system design stage, a UART based data transmission protocol was designed and documented to assure easier integration. The full specification can be seen in appendix K.1. All of the blocks outlined in Fig. 8.1

were developed as independent and re-usable Avalon ST video compliant Platform Designer entities, which allowed for testing in parts and fast prototyping. This approach allowed for reusability which might be desireable for further client projects. Where possible one block at a time was tested first using SystemVerilog test benches to simulate data and control packet flow and then on the FPGA itself where the video stream was used to visually confirm correct behaviour.

8.3 Alien Detection Flow

The detection of aliens is done by converting the RGB video stream into the HSV colour space [19, p. 98]. A mask is generated from this HSV video stream (Block: Mask) depending on configurable upper and lower bounds. It is then passed through an average kernel (Block: 2D-FIR-II), and only retained if it is above a configurable threshold. The filtering and thresholding operation acts as a binary erosion/expansion operator [20, p. 74] depending on the gating threshold. For the purposes of this project, it was used to erode non-continuous areas of mask activation, which removed noise and small objects from the background. Once the mask was gated, the bounding box and the center of mass for the mask was found (Block:COM counter). The center of mass estimation approach only worked for objects with a known fixed size as it does not take the span of the object on screen into account. It was however very resilient to noise as a relatively small unwanted mask activation patch will only marginally change the center of mass. Keeping track of the pixel count of the mask allowed for noise below a threshold to be neglected when no alien was on camera. The information for all colours was then relayed using UART upon request from the control module.

8.3.1 Building Detection Flow

The building detection algorithm operated on a HSV value channel. A Prewitt horizontal gradient operator [21, p. 86] was applied to the image in order to determine the presence of stripe edges on the alien buildings. The Prewitt operator was chosen over the Sobel operator [21, p. 86] due to its greater directionality and thus better rejection of noise from diagonal edges in the background. Both can be seen in figures 8.2 and 8.3 respectively. After additional filtering to join the noisy edges vertically using kernel 8.4 and gating, the sum of mask activity in each $\frac{1}{20}$ horizontal part of the image is tabulated (Block: T0). This allows more complex command algorithm to distinguish multiple buildings on camera simultaneously. The shortest distance between two stripes (Block: S0) is found for distance estimation. Bounding box and center of mass information is also provided (Block: COM). Assuming a circular object, the distance and the bounding box are sufficient to estimate the size and position of the building.

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Figure 8.2: Prewitt kernel ∂X

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

Figure 8.3: Sobel kernel ∂X

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Figure 8.4: Edge smoothing kernel

8.4 Highlighted Features and Approaches

8.4.1 HSV Converter Implementation

The presence of real numbered multiplication in the HSV conversion equations [19, p. 98] as seen in appendix K.2 was approached using a 15 bit (12 bit fractional part) fixed precision arithmetic type. This type was selected based on the intermediate result range for the integer part and experimental results for the fractional part. An early functionally identical but non-pipelined version of the block was parameterised with regards to the fixed point type and a SystemVerilog test bench with a reference floating point implementation was used to characterize the mean squared error across 4096 uniform samples of the RGB colour space. Empirical data from the Python prototype tuning showed an acceptable tolerance of ca. 3 hue units ($6°$). Saturation an value showed more tolerance. Fig. 8.5 presents the root mean square error as a function of fractional part and shows that 12 fractional bits were sufficient to meet this criterion. Further analysis showed that the main source of error came from the multiplication by $1/max(R, G, B)$ for bright objects in the saturation field. Since this error only occurred for bright objects ($s > 240$) this was deemed acceptable. The HSV transformation equations require multiple divisions by an 0-255 integer dividend. Inspired by the work of Liu et al [22], this operation was performed as a

multiplication using a 256 entry lookup table of the relevant fixed point expressions for $\frac{1}{x}$.

The inclusion of this block on the main image pipeline set the f_{max} requirements to 100MHz. When implemented as a combinatorial block with I/O registering the maximum operating frequency was not sufficient. Therefore, a 3 stage pipelined approach was taken and then tuned. In stage one, look-up table data was fetched and simple functions such as RGB channel minima, maxima and differences were calculated in preparation for the main multiplications. In stage two, main 8x15 bit multiplication was executed in parts as two 8x8 bit multiplications. This was not broken down further because it was expected that any sub 9x9bit multiplications will be efficiently mapped onto DSP blocks. In stage 3 these results were merged, the fixed point stripped and scaling constant multiplications were computed efficiently using Booth's recoding. Synthesis and fitter settings were set for higher effort (HE) to improve performance. Fig. 8.6 shows the improvements in f_{max} over the design process. The uncertainty due to fitter seed choice is expected to be ca. 3% according to Intel [23, p. 27]. Ultimately the f_{max} requirements were relaxed to 80MHz by operating the custom IP on a separate clock domain and transferring data in and out using dual-clock FIFOs.

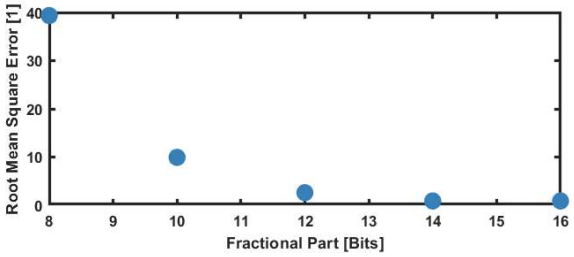


Figure 8.5: Root mean square error as a function of fractional precision

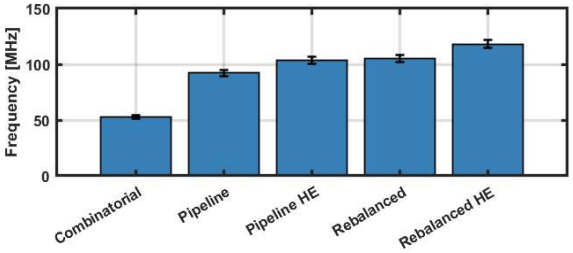


Figure 8.6: Evolution of HSV f_{max}

8.4.2 Image Transmission

For a real life scenario, it is believed that imaging to manually resolve an uncertain navigation state would be desireable as such it was implemented. Extensive attempts were made to interface with the existing frame buffer or use DMA controller IP blocks, but this failed due to incompatibility with the VIP streaming standard or incorrect control packet forwarding behaviour. Custom IP was developed to route control packets around the block, but this failed similarly and was not debuggable. Ultimately, a lower resolution approach using a custom buffer based around 2 port RAM IP was developed. This allowed a sub-sampled version of the Image to be retained and accessed by the NIOS system. For our design the resolution was 80x60 pixels at 8 bits per pixel (2:3:3 RBG) due to memory and UART bandwidth constraints. The image was transmitted at 1 pixel per packet over the UART protocol. A sample image taken can be seen in Fig. 8.9.

8.4.3 Gain and White Balance Estimation

During testing, the camera colour rendering proved very sensitive to the ambient light spectrum. A white balance algorithm was developed from the grey world assumption approach presented by Ebner [19]. The traditional grey world white balance algorithm assumes the average colour of the frame to be gray and then uses the observed average to estimate the emission spectrum of the global light source. This is then applied as a correction factor to the image. The issue is that this requires the image to be gamma corrected so that there is a linear relationship between the light intensity and the observed RGB value. This gamma correction on any representative sample was deemed too computationally expensive on the NIOS II system as an exponential weighting would have to be calculated for each pixel. Instead, an iterative approach in the style of a proportional controller was developed. The camera provided registers for setting the red, green, blue channel gains separately. A small full colour depth 20x15 pixel image buffer was developed from the Image buffer in order to provide feedback for the gain adjustments. Here the channel gains were individually adjusted until the buffer average came within small difference of ϵ from the target image average. This assured that on average the colour was gray. The common gain register and the average luminance register provided by the camera were used to achieve an average target luminance in a very similar manner.

For improved performance an auto-calibration procedure using grey calibration card was developed as the coloured walls of the lab did not fulfil the grey world assumption well. First, the gain was

adjusted to give an average luminance of 128 across the card, then the white balance was corrected until the channel averages were R:128, G:128, B:128. Finally the gain was adjusted again for the expected operating environment to account for the lightness difference between the card and the environment. This procedure notably improved the robustness of the system when testing in varying lighting conditions across the lab. The results of the procedure can be seen in Figs. 8.7 and 8.8 where varying light temperatures were simulated using a smart bulb. It is believed that this would be a very important feature for the rover to operate robustly in an unknown environment as lighting variations over the course of a day, possibly due to atmospheric scattering or airborne particulates could otherwise affect the function of the rover.

Figure 8.7: Performance at 2700K with default settings and after white balance

Figure 8.8: Performance at 5000K with default settings and after white balance

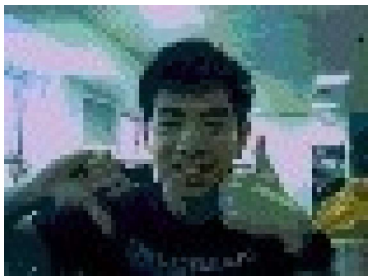


Figure 8.9: Gesturing human imaged

9 Command Subsystem

9.1 Design Decisions

Fast and reliable data transfer from the rover to the ground station was chosen as the primary requirement for the system. For the frontend, React was chosen due to its ability to quickly render UI elements, efficient data fetching [24] and efficient handling of the DOM [25]. For the backend, Express.js framework was chosen due its simplicity and ability to service HTTP requests efficiently and robustly [26]. NoSQL database MongoDB was chosen to store persistent data because of its scalability compared to SQL.

9.2 Network Architecture

The system is interconnected with a REST API running on the server that provides different services through uniform interfaces as [27] suggests. The rover periodically sends telemetry updates via the POST request to the endpoint `/rover/telemetry`, and as a response it receives commands to execute. The frontend part of the application receives updated telemetry from the `/controller/telemetry` endpoint and it can send POST requests with commands for the rover to the `/controller/command` endpoint.

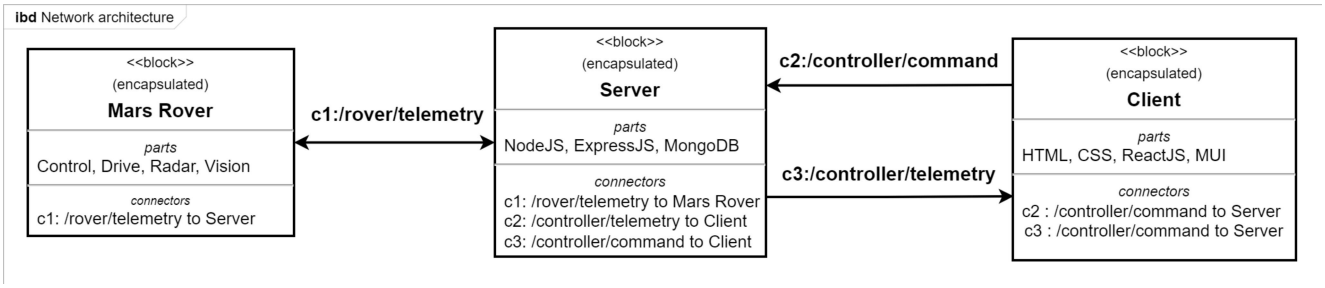


Figure 9.1: High level architecture of the command communications.

9.3 Communication Protocol

After exploring a variety of communication protocols, HTTP was chosen as the communication protocol. Alternatives to HTTP include MQTT (the OASIS standard for IoT messaging) and WebSockets due to their prominence in real time applications. Despite MQTT's popularity, it did not seem to fulfill the speed requirements as [28] showed that RTT latency drops for MQTT when travelling long distances across low power networks (which would be the case on Earth to Mars communication). The real-time bidirectional nature of WebSockets was investigated, but the lack of flow control semantics makes their use less ideal as the rover should have the capability of managing priority streams in order to quickly relay emergency

K Vision

K.1 Rover Nios Protocol Design Document V3

This protocol is a duplex transmission protocol with acknowledgements. A message under this protocol shall take the form of a variable number of comma delimited data entries ending with a new line. No space follows the delimiting comma. All fields are of variable length hex. The transmission is initiated by the master by the R character. The data is sent one entry at a time. Each entry must be acknowledged by the A character.

Each entry shall take the form:

Note: COM – center of mass

Field index	0	1	2	3	4	5	6	7
Field data	Object code	COM pixel X coordinate	COM pixel Y coordinate	COM mass	Bounding box (BB) left X coordinate	BB top Y coordinate	BB right X coordinate	BB bottom Y coordinate

Example Session:

Master: R

Slave: O 123 456 789 1 2 60 70,

Master: A

Slave: C 123 456 789 1 2 60 70,

Master: A

Slave: Z 123 456 789 1 2 60 70,

Master: A

The above message would indicate an object of type O with COM coordinates (0x123, 0x456), a mask mass of 789 highlighted pixels and would be bounded by by a rectangle with top left corner (0x1, 0x2) and bottom right corner (0x60, 0x70). It is followed by another object of type C/Z which can be decoded with the same key. Observed objects shall be identified by type, where the information shall be encoded by a single character object code. This code shall generally correspond to the observed color or obstacle type via the table below

Code	Object type/color
R	Red
L	Lime
B	Dark Blue
Y	Yellow
G	Dark Green
P	Pink
W	Building

Protocol Extensions:

Building Histograms: When ‘H’ Shall be requested by the master, 20 space delimited hexadecimal values shall be transmitted ending in a new line. These values correspond to the mask activation in the nth 10 horizontal segment of the image for the sum of the Prewitt kernel and -1* Prewitt kernel

For example:

Master: ‘H’

Slave: “0 0 0 0 0 12 ab aaa aa bb cc 99 0 0 0 0 0 0 0 0”

This communication shows significant sharp edges in the 5/20- 12/20 segments of the image. Based on this heat map it is expected that this is where the building can be found. After some low pass filtering and gating to account for the background noise, this approach can be used to identify and coarsely estimate the position of buildings on screen.

Image transmission:

Upon a ‘P’ request, a 80*60px image is transmitted in scan-line order. Each pixel corresponds to one character transmitted. The image is transmitted 60 pixels at a time, before requiring an

acknowledgement ‘A’. The character stream will not contain the control characters ‘’ and ‘
n’, but any other data should be expected.

K.2 RGB to HSV Equations

$$H = \begin{cases} \frac{1}{6}\frac{G-B}{max-min} & max = R \\ \frac{1}{6}(2 + \frac{B-R}{max-min}) & max = G \\ \frac{1}{6}(4 + \frac{R-G}{max-min}) & max = B \end{cases}$$

Figure K.1: Hue equation

$$S = \frac{max - min}{max}$$

Figure K.2: Saturation equation

$$V = max\{R, G, B\}$$

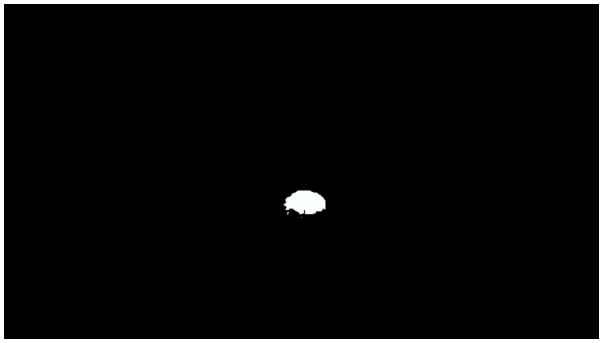
Figure K.3: Value equation

K.3 Colour Mask Performance

It can be seen in figure K.4a that the achieved performance varied greatly depending on the color. This occurred because some colors were more closely spaced together in hue relative to themselves and the background. Most notably this affected yellow, where due to its light color, the top of the ball was excessively bright, leading to incorrect classification and the bottom picked up reddish hues from the martian surface. This was prominent because the separation in hue between red and yellow was as little as 4 degrees from our testing. Therefore tighter bounds on saturation and value had to be imposed. It should be noted that mask activation above a configurable height was set to be ignored by the pipeline for both buildings and aliens. This threshold was set to approximately emulate the horizon.

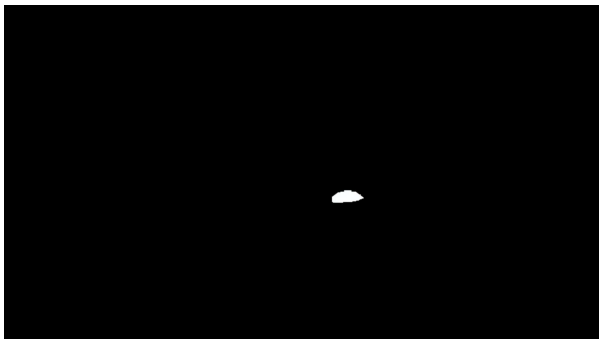
(a) Scene from which aliens were extracted

(b) Lime alien filtered mask

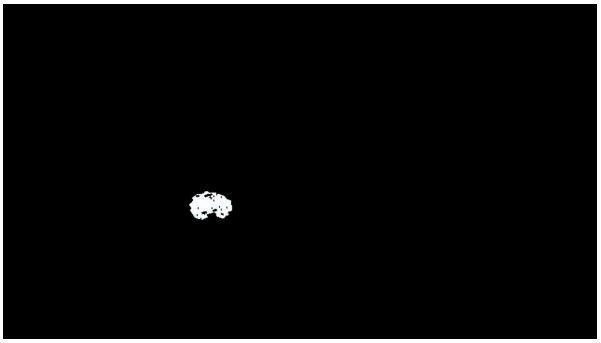


(c) Blue alien filtered mask

(d) Yellow alien filtered mask



(e) Red alien filtered mask



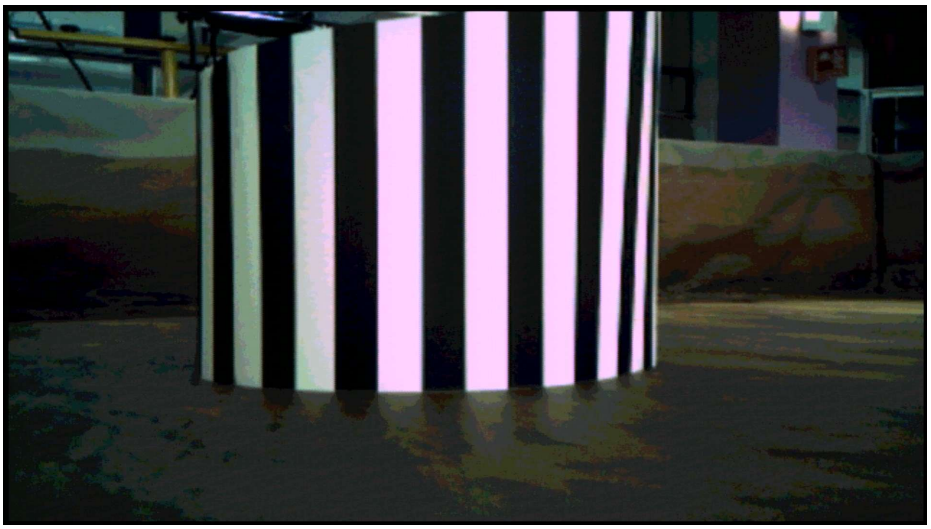
(f) Pink alien filtered mask

(g) Green alien filtered mask

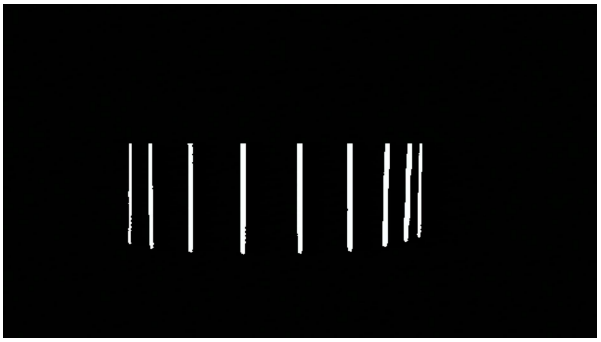
Figure K.4: Alien scene and resulting color masks

K.4 Building Mask Performance

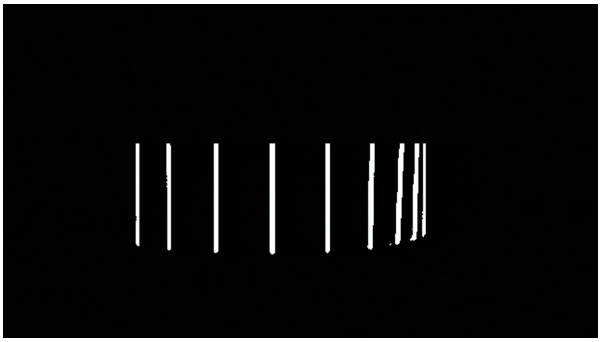
It can be seen that in isolation the signal from the edges of the buildings worked very well for detection using bounding boxes. However the sharp edges of the alien stand are also normally picked up by the filter. As such, the secondary approach using a heat map over portions of the images serves as a fallback. The presence of multiple objects can be inferred from multiple maxima in the heat map even once smoothing is applied.



(a) Scene from which building features were extracted



(b) Filtered high positive gradient areas



(c) Detected high to low brightness transitions

Figure K.5: Building scene and resulting edges