Palomar Aircraft Detection Laser Safety System

Alan Morrissett

California Institute of Technology
April 7, 2006

Introduction

An adaptive optics (AO) system is currently being developed and operated at the Palomar Observatory for the Hale 200-inch telescope. One of the operating modes for this AO system employs a sodium laser to create a laser guide star (LGS). The laser used for this purpose is Class V, with a current power of 5 to 8 watts and a prospective power of up to 15 watts. Such a laser is a potential hazard for over-flying aircraft, and must have a system that blocks it before it can illuminate any aircraft. A current solution to this need is to employ human spotters outside of the observatory dome; this solution, however, requires the recruitment, scheduling, management, and payment of the spotters and a more efficient solution is desired. This paper describes an automated aircraft detection safety system, employing computer-monitored visible and infrared cameras, which has been developed by Caltech to serve this need.

Camera Detection System Hardware

The aircraft detection system consists of a visible wavelength 'allsky' panoramic CCD camera, a bore-sighted mid-infrared CCD camera, a computer for each of the cameras to automate its operation and analyze images, Ethernet network connectivity, and a remote computer for a human operator to view images and operate the system.

Allsky Camera Hardware

- SBIG STL-1001E USB Camera (1024 X 1024 X 16 pixels), mounted on a permanent fixture on the outside perimeter of the Hale 200-inch telescope dome.
- Linux 2.4.21 computer 'allsky', rack-mounted in the enclosure under the dome of the Hale 200-inch telescope. Intel 3GHz Pentium processor, 2 GBytes memory, 140 GByte hard drive. USB 1.1 cable connection to the Allsky camera. Connected to Palomar LAN via slip-ring connection of the dome, with throughput of approximately 7 Mbits/sec.

Infrared Camera Hardware

- Indigo Merlin InSb MWIR Camera, 320 X 256 detector array, field of view approximately 15 degrees by 20 degrees, mounted in the prime focus cage of the 200-inch telescope. Camera power and external shutter controllable via switched power outlets, monitored by the Palomar telescope control system ('TCS').
- Linux 2.6.12 computer 'lwir', located in the west arm of the 200-inch telescope. Intel 2GHz Pentium processor, 512 MBytes memory, 60 GByte hard drive. A serial cable connection to the IR camera allows communication to the camera command interface.

- Linux Media Labs NTSC frame grabber card #LMLBT4M, connected via video cable to the IR camera.

Software Components

The Allsky and IR Camera ('Ircam') software has been built with the programs and libraries described in this section. An attempt has been made to use as much public domain free software as possible in order to leverage its existing capability.

- Language: C++ (Gnu g++ compiler).
- CFITSIO library (open source, Goddard Space Flight Center) for creation and reading of FITS image files.
- IRAF 'ds9' FITS file display program (IRAF open source) for displaying images.
- IRAF 'cdl' library ('Client Display Library', IRAF open source) for inter-process communication to the ds9 display processes.
- SBIG Universal Driver/Library Version 4.35 (proprietary binary) for control of the SBIG camera.
- Tcl/Tk libraries ('Tool Command Language', open source) for C language embeddable command interpretation and enabling of scripted GUIs which drive the applications via these commands.
- Videodev NTSC video device driver (open source)

Allsky Processes

The monitor and control of the Allsky camera is spread across several processes. These processes are:

1. LaserServer Process – Control of the laser is accomplished via a "dead-man" hardware switch which is one of a daisy chain of switches that enable the laser. This switch is controlled via pins 1 & 2 of a parallel port connection, and the signal on pin 1 must be toggled at 10 Hz or greater in order for the switch to remain closed. The motive for putting this function into a separate process is that there must be a single point of control that communicates to the parallel port, i.e. several processes cannot do it at the same time but a single server process such as this one can serve more than one client. Additionally, the Linux 'iomanip()' system function call is used to access the parallel port, and must be run with root privilege, so it was considered better to run a minimum amount of code that had this privilege.
2. AllskyMonitor Process – This is a threaded process that controls the allsky camera. It initiates image exposure, downloads and save images from the camera, performs analysis on the images to locate aircraft, issues commands to shutter the laser when aircraft are within a designated exclusion zone, optionally saves images to disk, and optionally serves images to a remote display process.

3. AllskyDisplay Process – An display and control process which is designed to run on a separate computer than allsky. The motive for putting this functionality in another process on a separate computer is to make the most efficient use of the bandwidth between allsky and an operator in the control room. It has proven much more efficient to copy the images via socket to the display computer (2 MByte images across a 7 Mbit/sec connection, every 7.5 seconds) rather than to run the display software on allsky and rely on the X display protocol to remotely show the image. This process creates a child process which executes ds9 to actually render the image; the CDL library is used to transfer these images. AllskyDisplay is also used to send commands to the AllskyMonitor process - it can accomplish this through either a command line interface or a scripted Tk GUI.

Ircam Processes

The monitor and control of the IR camera is similarly spread across several processes. These processes are:

1. LaserServer Process – This is an instance of the same laser control program that is run for the allsky computer, described above.
2. IrcamMonitor Process – A threaded process that controls the IR camera. It opens the IR camera external shutter, powers the camera using the TCS controlled power switch, captures images using the frame capture card, performs analysis on the images to locate aircraft, issues commands to shutter the laser when aircraft are detected in the image, optionally saves images to disk, and optionally serves images to a remote display process.
3. IrcamDisplay Process – A display and control program similar to the one used for allsky. Although there is not as much of a bandwidth constraint between lwir and the control room, it is still more efficient to copy the images over a socket and display them on the control room operator machine. This process creates a child process which executes ds9 to actually render the image; the CDL library is used to transfer these images. IrcamDisplay is also used to send commands to the IrcamMonitor process - it can accomplish this through either a command line interface or a scripted Tk GUI.

AllskyMonitor Program Architecture and Algorithms

The AllskyMonitor program is threaded in order to parallelize its processing and maximize throughput. The following threads run in parallel:

- Command (main) thread – starts the other service threads, then monitors for commands from the command line or from a dedicated socket opened by a remote process.

- Image acquisition thread – starts image exposure, monitors for completion of the exposure, then downloads the image from camera to memory. Notifies the image processing thread that an image is ready, then repeats the process.
- Image processing thread – takes a stack of recent images and runs the aircraft detection algorithm. Issues command to shutter the laser when an aircraft is detected with the exclusion zone. Logs detections to a log file.
- Image server thread – listens for image requests on a dedicated socket. When one is received it writes the most recent image to the image socket, then waits for the next image request.

The AllskyMonitor image acquisition thread algorithm is:

1. Issue command to AllskyCamera object to obtain an image. (This object encapsulates the low level commands needed to control the allsky camera via the SBIG driver library. The image is tagged with its start and end exposure times and the telescope and dome configurations obtained from TCS.)
2. On return of the new image, push a copy of it onto a size-bounded stack (i.e. the front of a circular queue) of recent images.
3. Increment the image ready semaphore.
4. Repeat to get the next image.

The AllskyMonitor image processing thread algorithm is:

1. On startup, shutter the laser.
2. Wait for the image ready semaphore to be incremented by the image acquisition thread. Decrement the image ready semaphore to 0.
3. If the dome is closed, shutter the laser. If the dome is open, examine the current and previous dome positions. If the difference is greater than 0.2 degrees, or TCS reports that the dome is slewing, then shutter the laser. The reason for this is that the timed exposure will cause image streaks of stationary objects as the dome is slewing, and the image detection algorithm will be invalid. Since we don't know the state of the sky, shutter the laser for safety.
4. Using the image stack, compute the background using the last N (=3) images. The background is considered to be the minimum value at each pixel location over the last N images.
5. Subtract the background from the current image. Compute the standard deviation of the current, background-subtracted image, using a clipping mask to eliminate the corners outside of the actual image and a small slice of the horizon. Using the standard deviation, compute a threshold for pixels differences by multiplying the standard deviation by a settable scaling factor (default 1.0). If this threshold value is above a maximum value (currently set at 200), then set the threshold to be this maximum value.
6. Subtract the previous image from the current image, yielding a difference image. On a bit-mapped mask of the image, tag every pixel position where the difference is greater than the threshold value.
7. Using the tag mask, aggregate adjacent pixels into groups. Every group that is within a window of sizes (default minimum = 10, maximum = 500) is considered to be a potential detection.

8. Compute statistics for each potential detection, including centroid, radius, slope, bounding values in the x & y directions, least squares linear fit, and the correlation constant for the linear fit.
9. In order to find spurious detections which are really saturation "bleed" lines from a bright moon, run a moon detection algorithm. Iterate over all pixels in the background image, marking pixels that are above a brightness threshold (default = 8000.) Aggregate adjacent pixels into groups. The largest group of sufficient size (default = 1000 pixels), if any, is considered to be the moon.
10. Iterate over the candidate detections to find moon saturation lines. A line which is sufficiently steep (slope greater than 6.0) and whose extension of its least squares linear fit intersects the moon centroid within a delta value, and which is within a minimum closeness to the moon (default = 5 pixels) is considered to be a spurious moon saturation line.
11. Under some weather conditions the laser produces a visible track in the image. Variations in intensity of this track can produce false detections. To find these, compute a predicted current laser path that goes from a fixed point on the perimeter of the image to the center of the current telescope bore sight. Iterate over all detections for ones that lie on this line, by checking for the same slope to within 10% and for a centroid that is within N=4 pixels of the predicted line.
12. Experimental cloud detection algorithms, for flagging false detections that are caused by bright, fast-moving clouds. This is currently under investigation; two different attempts thus far eliminate some false cloud detections but also eliminate some valid aircraft detections.
13. For all detections that are deemed to be aircraft, if any are within a computed exclusion zone around the current pointing position of the telescope, then shutter the laser.

Typical Image Acquisition Processing Times

SBIG Driver Command	Time
End Exposure	0.270
Start Exposure	0.965
Poll for Exposure Done	3.145
End Exposure	0.266
Image Exposure Subtotal	4.378
Freeze	0.001
Start Readout	0.001
Readout Time	2.769
End Readout	0.003
Image Readout Subtotal	2.777
Unfreeze	0.000
Total Image Acquisition Time	7.427

Image Analysis Processing Times

Processing Time	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5
Detection	.508	.506	.503	.489	.494
Moon Detection	.164	.159	.175	.171	.176
Cloud Detection	.000	.002	.000	.002	.010
Image Save	.038	.039	.038	.039	.037
Total Image Process Time	.717	.712	.726	.710	.730
Image Serving	.198	.276	.276	.275	.280
Image Display (remote process using ds9)	1.560	2.145	1.556	2.079	4.001

Sample Image 1 – Moonless, no aircraft
Sample Image 2 – Moonless, 2 aircraft
Sample Image 3 – Bright moon, no aircraft
Sample Image 4 – Bright moon, 2 aircraft
Sample Image 5 – Moon obscured, 10 cloud false detections

Typical Parallelism in Processing

Image Acquisition Thread	Image Processing Thread	Image Serving Thread	Remote Process Display Time
~7.5 sec	~.72 sec	~.275 sec	1.5 sec to >8 sec

AllskyDisplay Program Architecture and Algorithms

The purpose of the AllskyDisplay program is to display camera images and provide a control interface to an operator. It normally runs in a computer located in the observatory control room ('vulcan') rather than on allsky. This program is also threaded in order to parallelize the relatively slow rendering of images with image acquisition from AllskyMonitor program and with command execution.

The AllskyDisplay processing flow is:

1. Create a child process and execute ds9 in it. Open a cdl library connection to ds9 for sending images.
2. Spawn image acquisition thread. This attempts to open a socket to the AllskyMonitor image server socket on allsky. When a successful open occurs, it sends an image request, then reads the image from the socket and stores it locally. An image ready semaphore is raised, and the process repeated. If the socket is closed or broken, it attempts to re-open, then repeats the above loop.
3. Spawn image display thread. This waits for the image ready semaphore to be raised; when it is, a copy of the most recent image is made and a cdl library call

made to put that image to ds9. On completion of the image send to ds9, it again waits on the image ready semaphore. Because ds9 crashes after 1532 images have been displayed, this thread kills the ds9 process after 1500 images and re-starts a new one.

4. Spawn a status processing thread. This attempts to open a socket to the dedicated status socket on AllskyMonitor process. On successful open, it then listens for any status messages and then prints them to the standard output and displays them on the Tk GUI, if present (e.g. if the laser were shuttered because of a detection, the laser shuttered status and the reason for it would be displayed as a status.) If the status socket is closed or broken, this loops back to attempting to open the socket.
5. The main thread monitors for commands from either the command line or a Tk GUI command interface. Commands are immediately executed upon receipt. Commands intended for the remote AllskyMonitor program (the majority) are sent over a dedicated command socket and the command result read back.

IrcamMonitor Program Architecture and Algorithms

The IrcamMonitor program is threaded in order to parallelize its processing and maximize throughput. The following threads run in parallel:

- Command (main) thread – starts the other service threads, then monitors for commands from the command line or from a dedicated socket opened by a remote process.
- Image acquisition thread – issues commands to frame grabber card to capture an image, and copies this image into memory. Notifies the image processing thread that an image is ready, then repeats the process.
- Image processing thread – takes a stack of recent images and runs the aircraft detection algorithm. Issues command to shutter the laser when an aircraft is detected within the image. Logs detections to a log file.
- Image server thread – listens for image requests on a dedicated socket. When one is received it writes the most recent image to the image socket, then waits for the next image request.

The IrcamMonitor image acquisition thread algorithm is:

1. Issue command to IRCamera object to obtain an image. (This object encapsulates the low level commands needed to control the IR camera via the videodev driver. The image is tagged with its start and end exposure times and the telescope and dome configurations obtained from TCS.)
2. On return of the new image, push a copy of it onto a size-bounded stack (i.e. the front of a circular queue) of recent images.
3. Increment the image ready semaphore.
4. Repeat to get the next image.

The IrcamMonitor image processing thread algorithm is:

1. On startup, shutter the laser.

2. Wait for the image ready semaphore to be incremented by the image acquisition thread. Decrement the image ready semaphore to 0.
3. If the dome is closed, shutter the laser. If the dome is open, examine the current and previous dome positions. If the difference is greater than 0.2 degrees, or TCS reports that the dome is slewing, then shutter the laser. The reason for this is that the timed exposure will cause image streaks of stationary objects as the dome is slewing, and the image detection algorithm will be invalid. Since we don't know the state of the sky, shutter the laser for safety.
4. Using the image stack, compute the background using the last N (=3) images, using a clipping mask to eliminate the noisy borders of the image. The background is considered to be the minimum value at each pixel location over the last N images.
5. Subtract the background from the current image. Compute the standard deviation of the current, background-subtracted image, using a clipping mask to eliminate a slice around the perimeter of the rectangular image where there is a lot of noise or no real image. Using the standard deviation, compute a threshold for pixels differences by multiplying the standard deviation by a settable scaling factor (default 1.0). If this threshold value is above a maximum value (default 10), then set the threshold to be this maximum value.
6. Subtract the previous image from the current image, yielding a difference image. On a bit-mapped mask of the image, tag every pixel position where the difference is greater than the threshold value.
7. Using the tag mask, aggregate adjacent pixels into groups. Every group that is within a window of sizes (default minimum = 10, maximum = 200) is considered to be a potential detection.
8. Compute a bounding box for each candidate detection. Compute the detection density of each potential detection by the ratio of marked vs. unmarked pixels within the bounding box. Groups which are below the cutoff density (default 0.35) are considered to be noise patterns.
9. If there are any detections that are deemed to be aircraft within the clipped region of the image, then shutter the laser.

IrcamDisplay Program Architecture and Algorithms

The IrcamDisplay program is very similar to AllskyDisplay program – its purpose is to display IR camera images and provide a control interface to an operator. It normally runs in the same computer as AllskyDisplay, located in the observatory control room ('vulcan').

The IrcamDisplay processing flow is:

1. Create a child process and execute ds9 in it. Open a cdl library connection to ds9 for sending images.
2. Spawn image acquisition thread. This attempts to open a socket to the IrcamMonitor image server socket on lwir. When a successful open occurs, it

sends an image request and then reads the image from the socket and stores it locally. An image ready semaphore is raised, and the process repeated. If the socket is closed or broken, it attempts to re-open, then repeats the above loop.

3. Spawn image display thread. This waits for the image ready semaphore to be raised; when it is, a copy of the most recent image is made and a cdl library call made to put that image to ds9. On completion of the image send to ds9, it again waits on the image ready semaphore. Because ds9 crashes after 1000+ images have been displayed, this thread kills the ds9 process after 1000 images and re-starts a new one.
4. Spawn a status processing thread. This attempts to open a socket to the dedicated status socket on IrcamMonitor process. On successful open, it then listens for any status messages and then prints them to the standard output and displays them on the Tk GUI, if present (e.g. if the laser were shuttered because of a detection, the laser shuttered status and the reason for it would be displayed as a status.) If the status socket is closed or broken, this loops back to attempting to open the socket.
5. The main thread monitors for commands from either the command line or a Tk GUI command interface. Commands are immediately executed upon receipt. Commands intended for the remote IrcamMonitor program (the majority) are sent over a dedicated command socket and the command result read back.

LaserServer Program Algorithm

The LaserServer uses the following algorithm:

1. Start a thread which sets the level for pin 2 (switch value) and toggles pin 1 of the parallel port at 10 Hz or greater. This thread micro-sleeps for 50 msec, then simultaneously writes the on/off switch value to pin 2 and the NOT value of its previous value to pin 1, and loops.
2. The main thread starts by shuttering the laser. It then opens a socket and accepts incoming connections. When a connection is opened, it loops on reading commands from the socket. If the socket connect is broken (the client process closes the socket or exits), the read function returns immediately and the laser is shuttered.

AllskyMonitor Tcl Commands

```
help
quit
run         [<n>]          (run for n frames, or indefinitely)
arm                        (open laser shutter)
stop                       (stop running)
lsopen                     (open laser shutter)
lsclose                    (close laser shutter)
lslock                     (lock laser shutter to current position)
lsunlock                   (unlock laser shutter from locked position)
lsstatus                   (get status of laser shutter)
exptime     [<t>]          (set/get exposure time, default 3 sec)
```

```
radius      [<r>]          (set/get exclusion zone radius, in degrees)
temp                       (get camera temperature)
gain        [<g>]          (set/get camera gain)
threshold   [<t>]          (set/get maximum detection threshold)
minpix      [<n>]          (set/get minimum pixels for a detection)
maxpix      [<n>]          (set/get maximum pixels for a detection)
saveimages  [on|off]       (turn on/off image save)
display     [on|off]       (turn image display on/off)
detect      [on|off]       (turn plane detection on/off)
detectstop  [on|off]       (stop run if detection occurs)
lasersim    [on|off]       (control of laser is/isn't simulated)
imgsource   [<dir>|camera] (specify source of images)
back        [<n>]          (move backward n frames)
forward     [<n>]          (move forward n frames)
delay       [<n>]          (insert n msec delay between frames)
imgsync     [on|off]       (force all images to be displayed)
testsight  {<az><alt>|off} (set test bore sight)
```

AllskyDisplay Tcl Commands

```
help
quit
run                        (start running)
stop                       (stop running)
startds9                   (start a ds9)
stopds9                    (stop ds9)
saveimages  [on|off]       (turn on/off image save)
display     [on|off]       (turn image display on/off)
detect      [on|off]       (turn plane detection on/off)
detectstop  [on|off]       (stop run if detection occurs)
forward     [<n>]          (move forward n frames)
back        [<n>]          (move backward n frames)
delay       [<n>]          (insert n msec delay between frames)
imgsync     [on|off]       (force all images to be displayed)
testsight  {<az><alt>|off} (set test bore sight)
auxdisp     [on|off]       (turn on/off difference image)
```

IrcamMonitor Tcl Commands

```
help\n"
quit\n"
run         [<n>]          (run for n frames, or indefinitely)
arm                        (open laser shutter)
block                      (close laser shutter)
stop                       (stop running)
shutter     [on|off]       (IR camera external shutter status)
camera      [on|off]       (IR camera power status)
lsopen                     (open laser shutter)
lsclose                    (close laser shutter)
lslock                     (lock laser shutter to current position)
lsunlock                   (unlock laser shutter)
lsstatus                   (get laser shutter status)
saveimages  [on|off]       (turn on/off image save)
display     [on|off]       (turn image display on/off)
detect      [on|off]       (turn plane detection on/off)
detectstop  [on|off]       (stop run if detection occurs)
```

```
lasersim    [on|off]       (laser control is/isn't simulated)
imgsource   [<dir>|camera] (specify source of images)
back        [<num>]        (move backward n frames)
forward     [<num>]        (move forward n frames)
delay       [<num>]        (insert n msec delay between frames)
imgsync     [on|off]       (force all images to be displayed)
ignoremove  [on|off]       (don’t shutter laser on dome slew)
```

IrcamDisplay Tcl Commands

```
help
quit
run                        (start running)
stop                       (stop running)
startds9                   (start ds9)
stopds9                    (stop ds9)\n"
saveimages  [on|off]       (turn on/off image save)\n"
display     [on|off]       (turn image display on/off)\n"
detect      [on|off]       (turn plane detection on/off)\n"
detectstop  [on|off]       (stop run if detection occurs)\n"
forward     [<n>]          (move forward n frames)\n"
back        [<n>]          (move backward n frames)\n"
delay       [<n>]          (insert n msec delay between frames)\n"
imgsync     [on|off]       (force all images to be displayed)\n"
```

Sample Allsky Detection Log File Entries (Top Entries From 9/23/2005)

20050923031627ut X=506 Y=443 ALT=77.0571 AZ=304.16 DOMEAZ=200.0 TELALT=85.7 TELAZ=205.9 RA=19:28:13.83 DEC=+29:26:57.1 RAD=15.0 SHUTDOWN=YES

20050923054306ut X=525 Y=552 ALT=81.5292 AZ=294.701 DOMEAZ=39.9 TELALT=90.0 TELAZ=269.8 RA=22:03:50.66 DEC=+33:19:40.4 RAD=15.0 SHUTDOWN=YES

20050923054313ut X=520 Y=547 ALT=82.7843 AZ=297.833 DOMEAZ=39.9 TELALT=90.0 TELAZ=269.8 RA=22:03:58.11 DEC=+33:19:40.4 RAD=15.0 SHUTDOWN=YES

20050923054320ut X=514 Y=542 ALT=84.0707 AZ=303.965 DOMEAZ=39.9 TELALT=90.0 TELAZ=269.8 RA=22:04:05.57 DEC=+33:19:40.3 RAD=15.0 SHUTDOWN=YES

20050923054328ut X=508 Y=538 ALT=85.0676 AZ=313.654 DOMEAZ=39.9 TELALT=90.0 TELAZ=269.8 RA=22:04:13.05 DEC=+33:19:40.3 RAD=15.0 SHUTDOWN=YES

20050923054335ut X=503 Y=533 ALT=86.058 AZ=-34.8737 DOMEAZ=39.9 TELALT=90.0 TELAZ=269.8 RA=22:04:20.52 DEC=+33:19:40.3 RAD=15.0 SHUTDOWN=YES

20050923054343ut X=498 Y=528 ALT=86.796 AZ=-17.044 DOMEAZ=39.9 TELALT=90.0 TELAZ=269.8 RA=22:04:27.99 DEC=+33:19:40.2 RAD=15.0 SHUTDOWN=YES

20050923054350ut X=493 Y=523 ALT=87.0832 AZ=7.4 DOMEAZ=39.9 TELALT=90.0 TELAZ=269.8 RA=22:04:35.45 DEC=+33:19:40.2 RAD=15.0 SHUTDOWN=YES

20050923054358ut X=485 Y=519 ALT=86.2034 AZ=32.1751 DOMEAZ=39.9 TELALT=90.0 TELAZ=269.8 RA=22:04:42.90 DEC=+33:19:40.2 RAD=15.0 SHUTDOWN=YES

20050923054405ut X=479 Y=514 ALT=85.2975 AZ=47.8261 DOMEAZ=39.9 TELALT=90.0 TELAZ=269.8 RA=22:04:50.36 DEC=+33:19:40.2 RAD=15.0 SHUTDOWN=YES

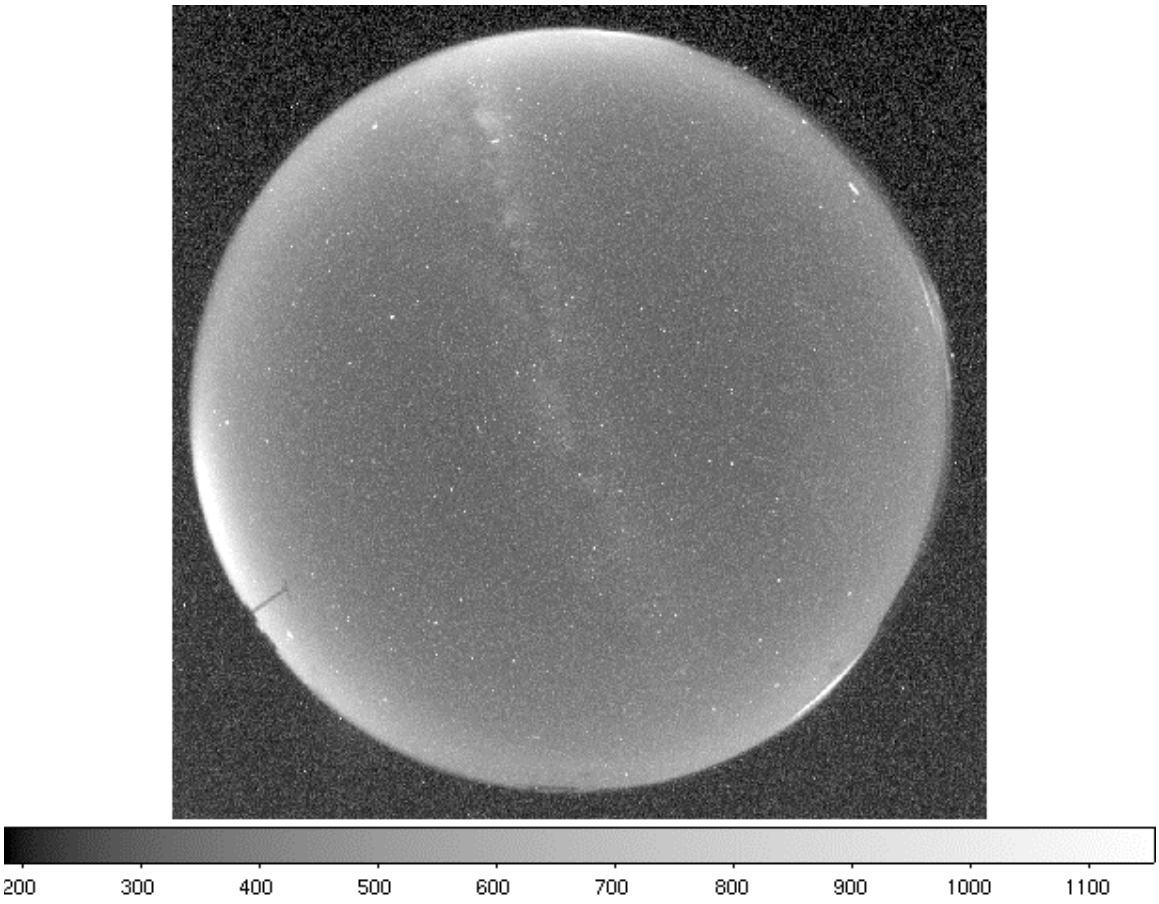

Image 1 – Aircraft Detections, Unprocessed Image

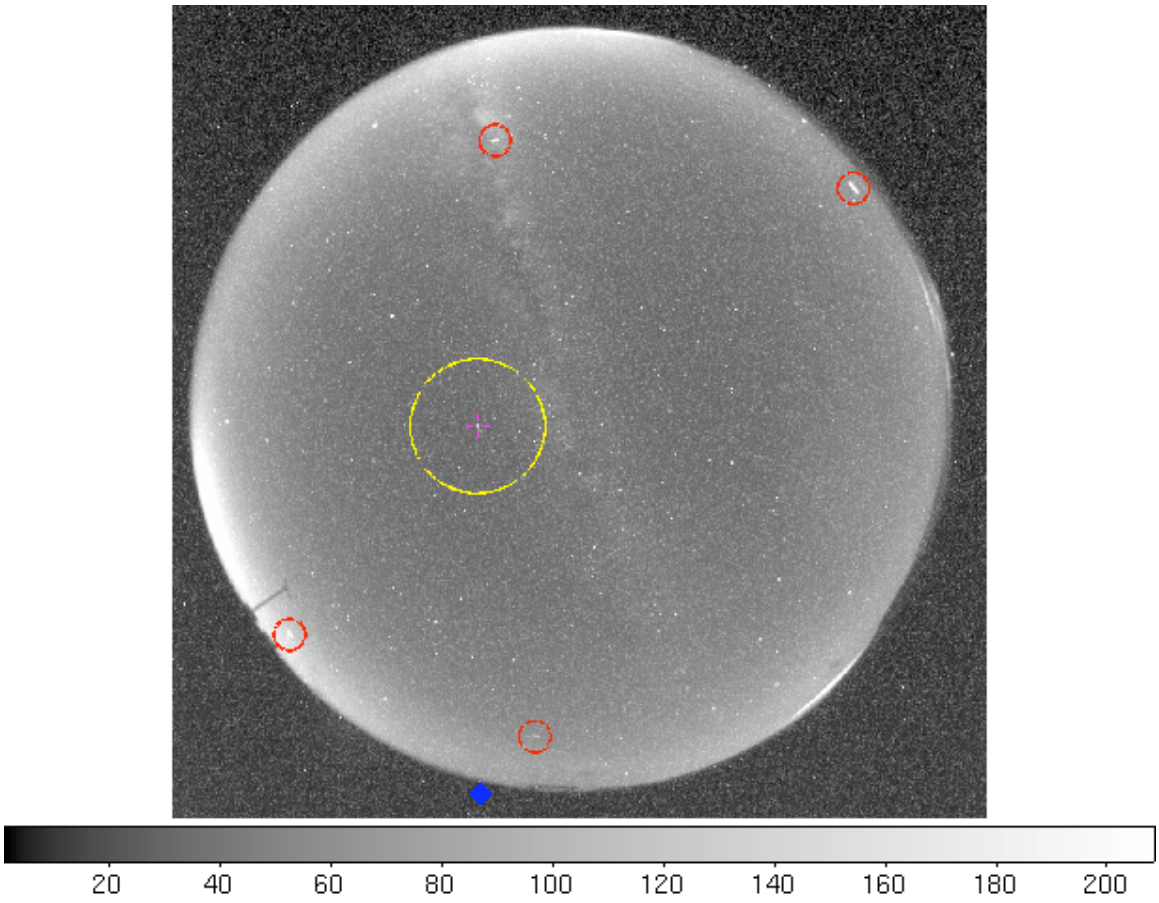

Image 2 – Aircraft Detections, Processed Image

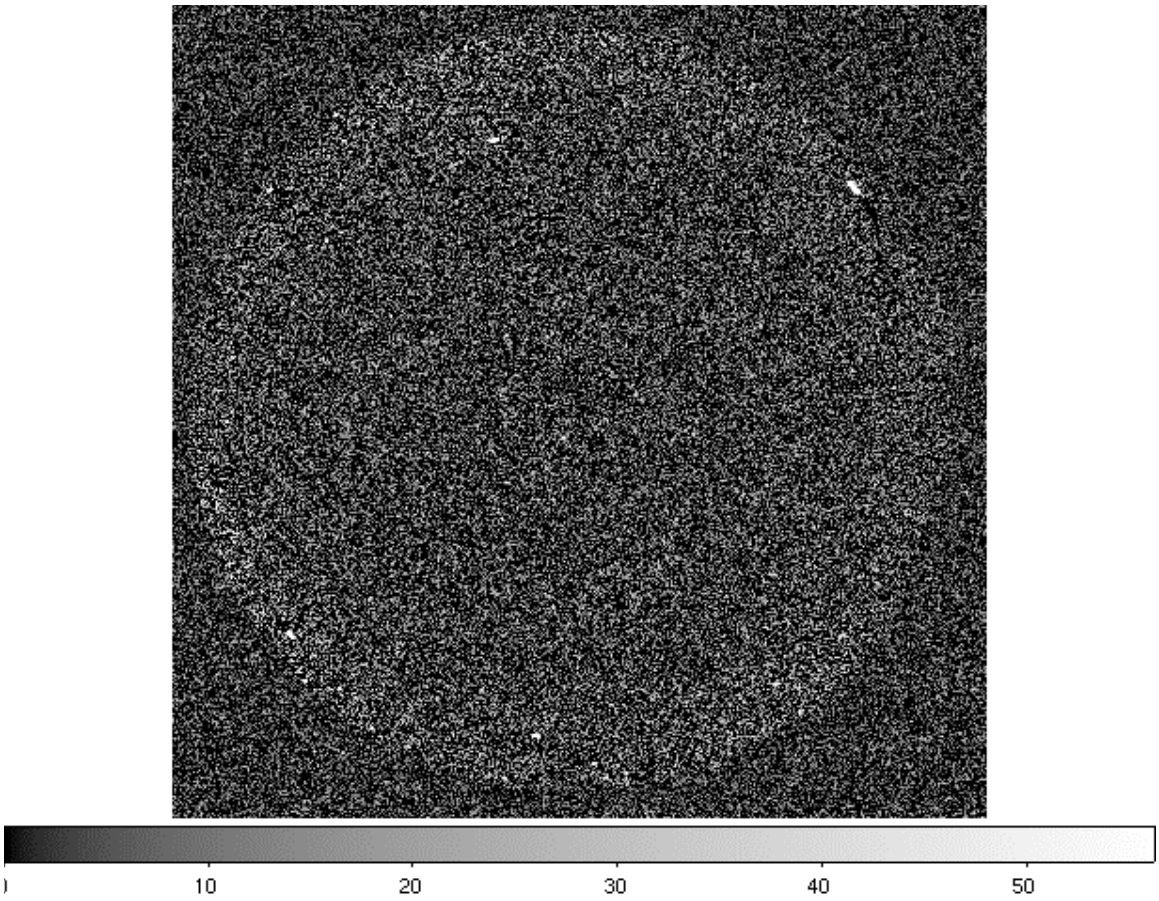

Image 3 – Aircraft Detections, Difference Image

Image 4 – Aircraft Detections, Detection mask

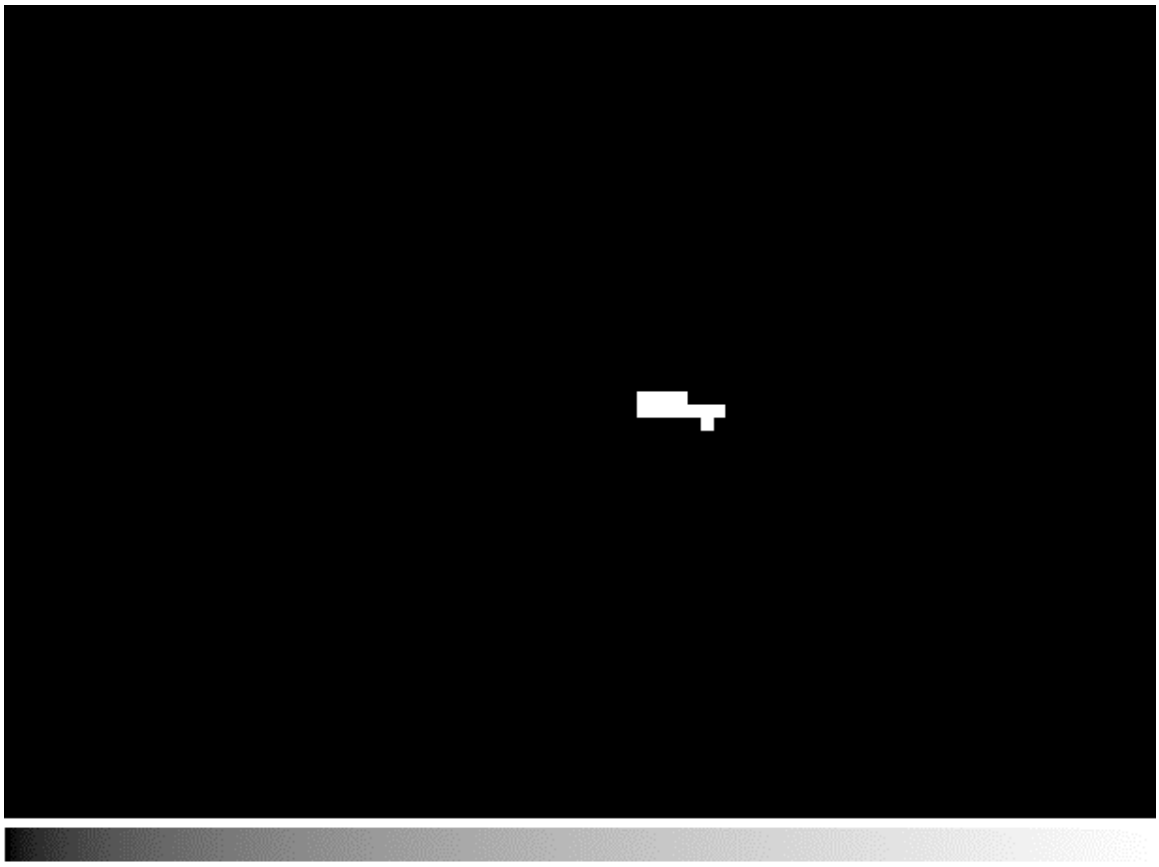

Image 5 – Aircraft Detections, Detection Mask
Zoomed to Bottom Detection

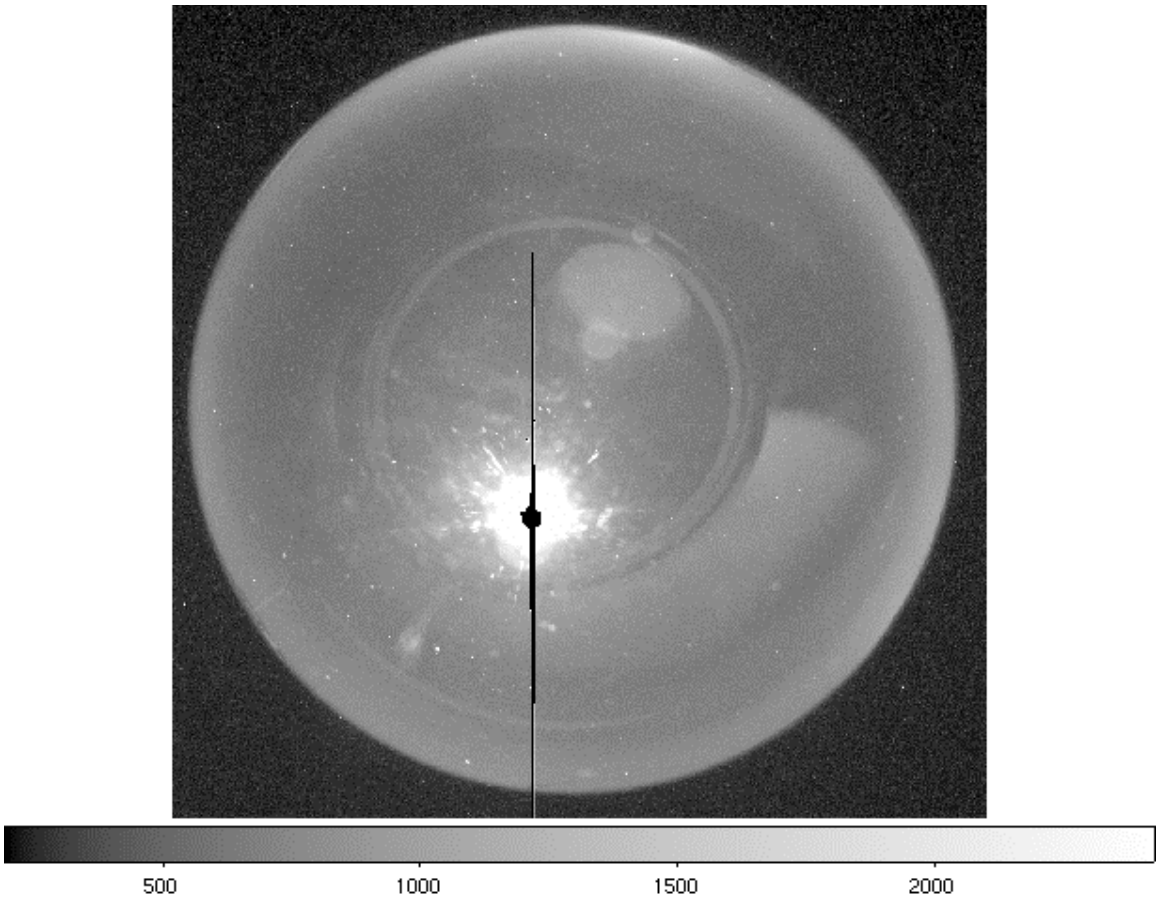

Image 6 – Bright Moon, Unprocessed Image

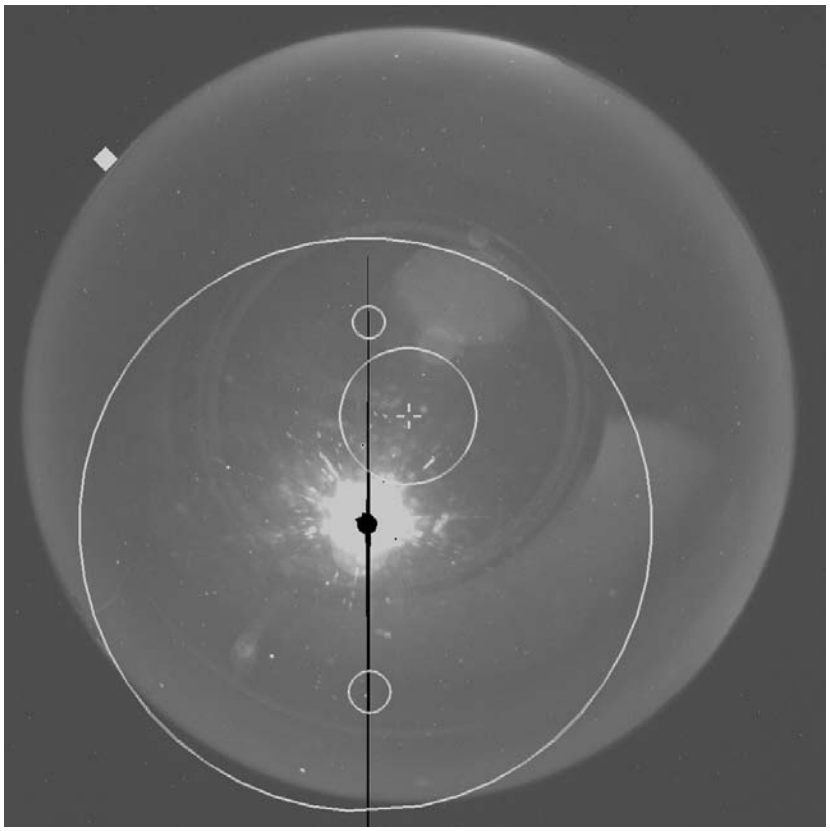

Image 7 – Bright Moon, Processed Image

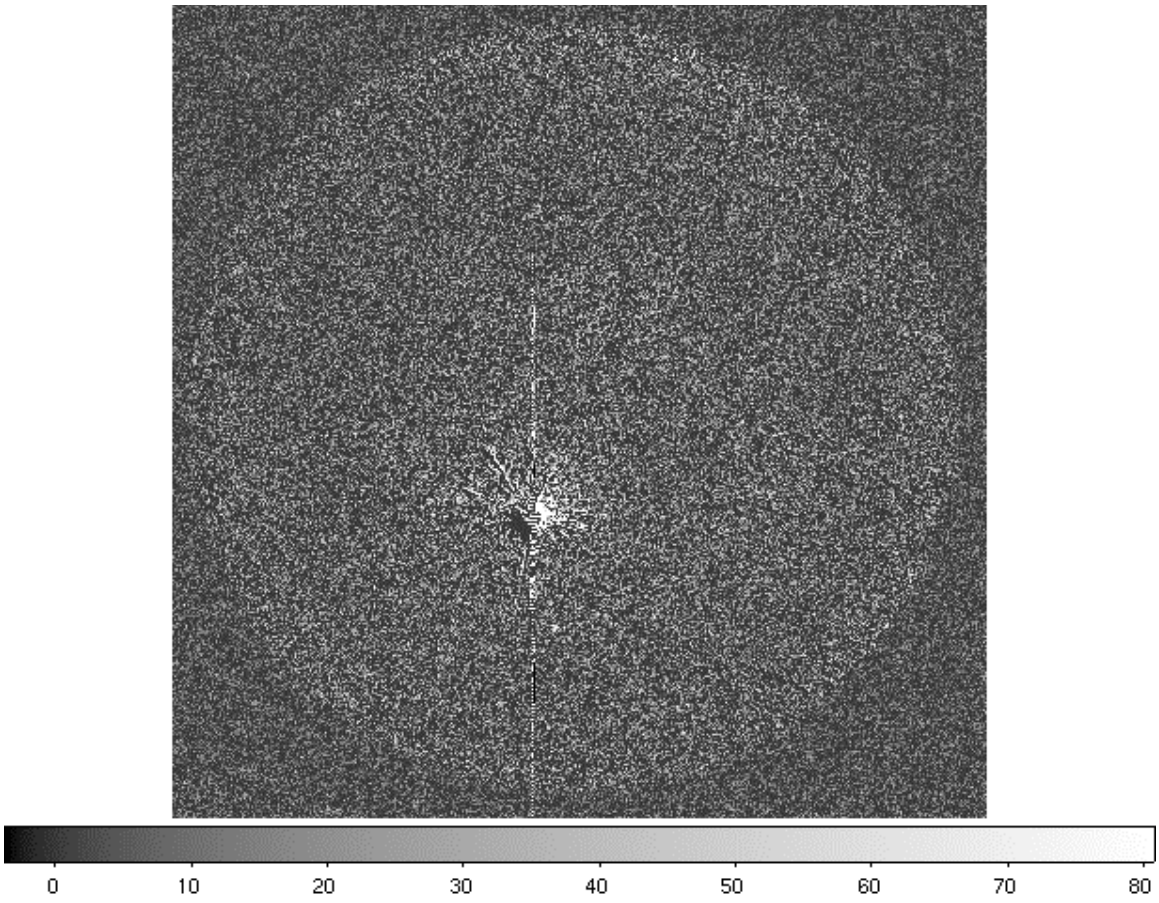

Image 8 – Bright Moon, Difference Image

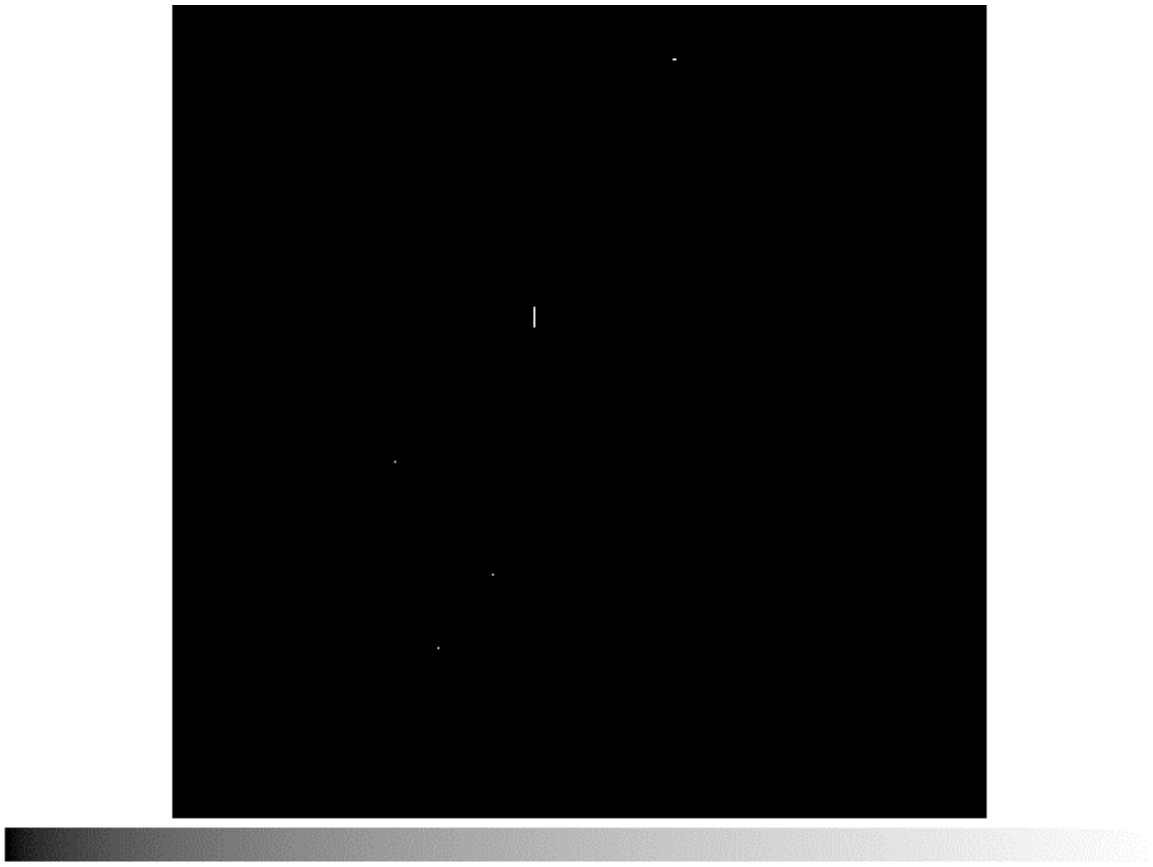

Image 9 – Bright Moon, Detection Mask

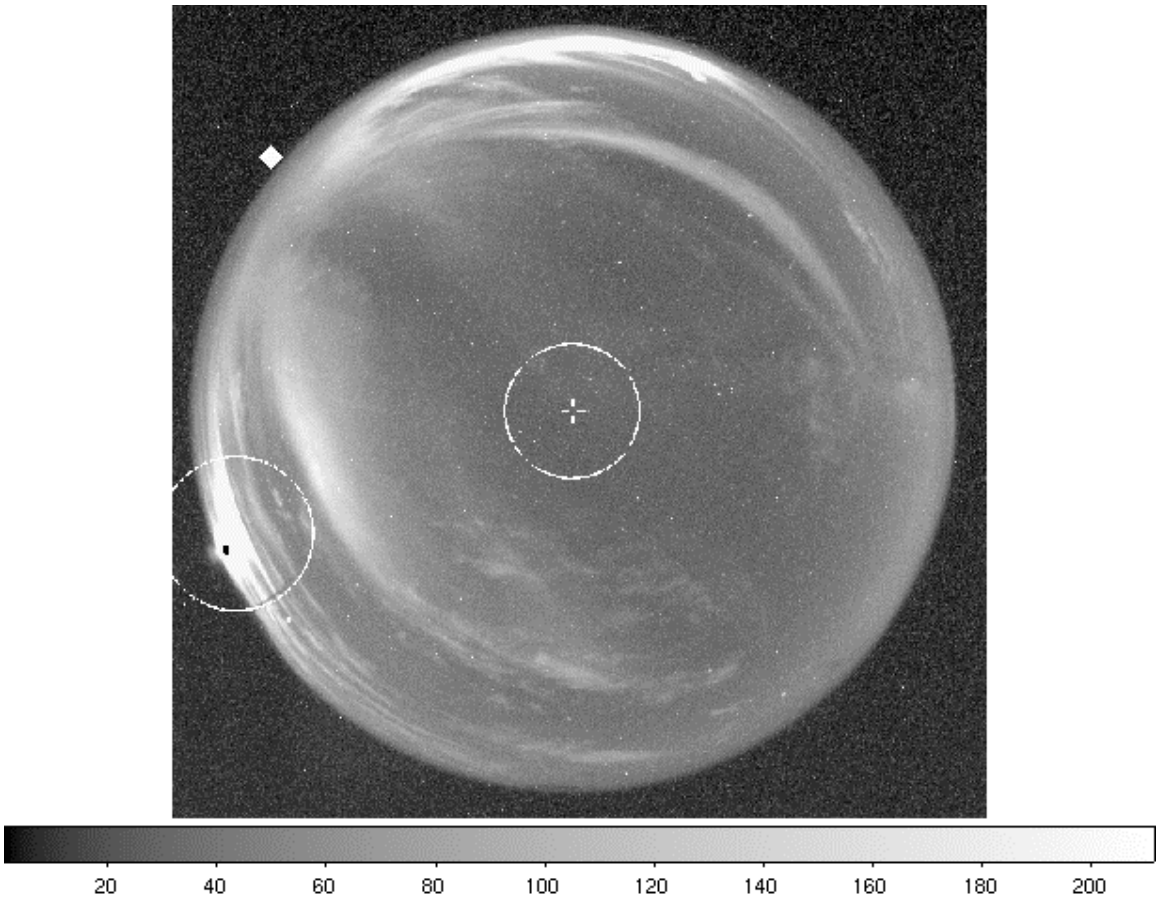

Image 10 – Before Laser Image, Frame 1

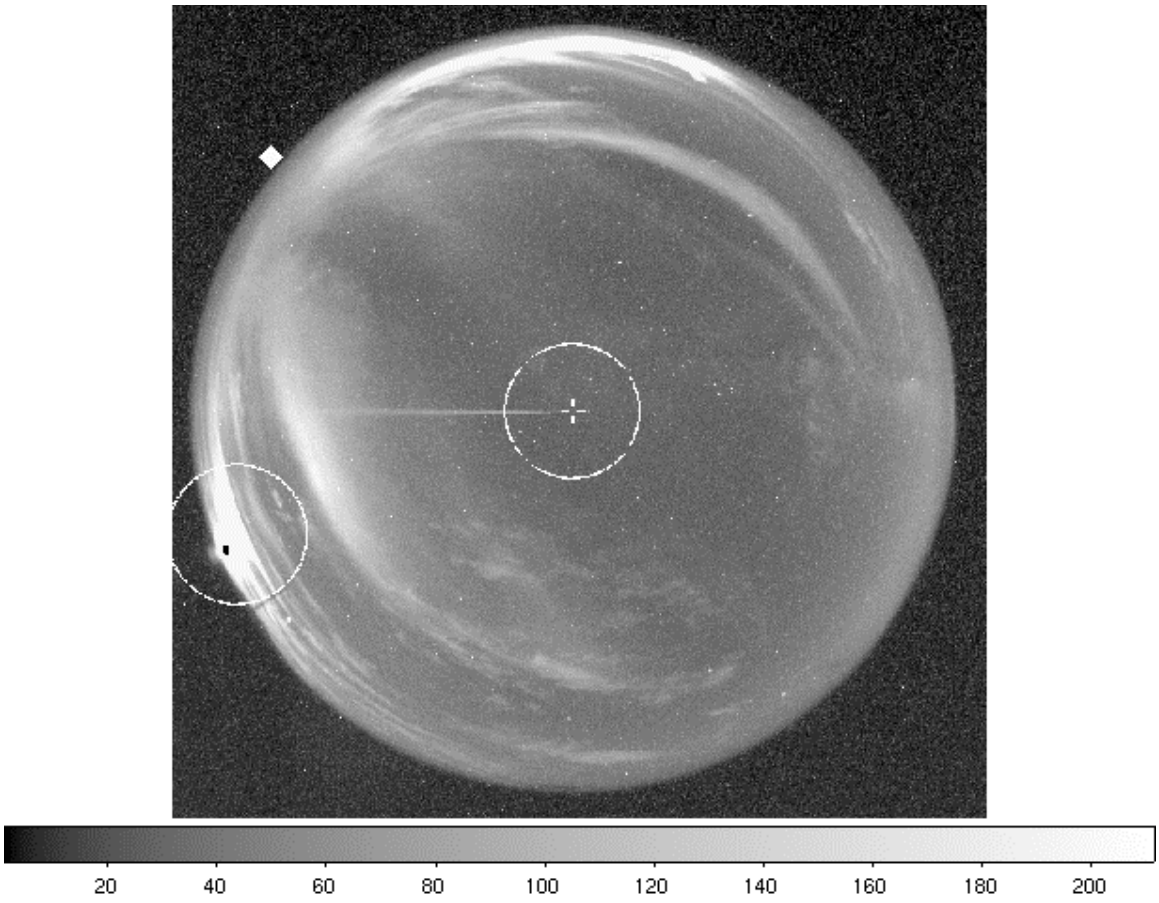

Image 11 – Laser Image, Frame 2

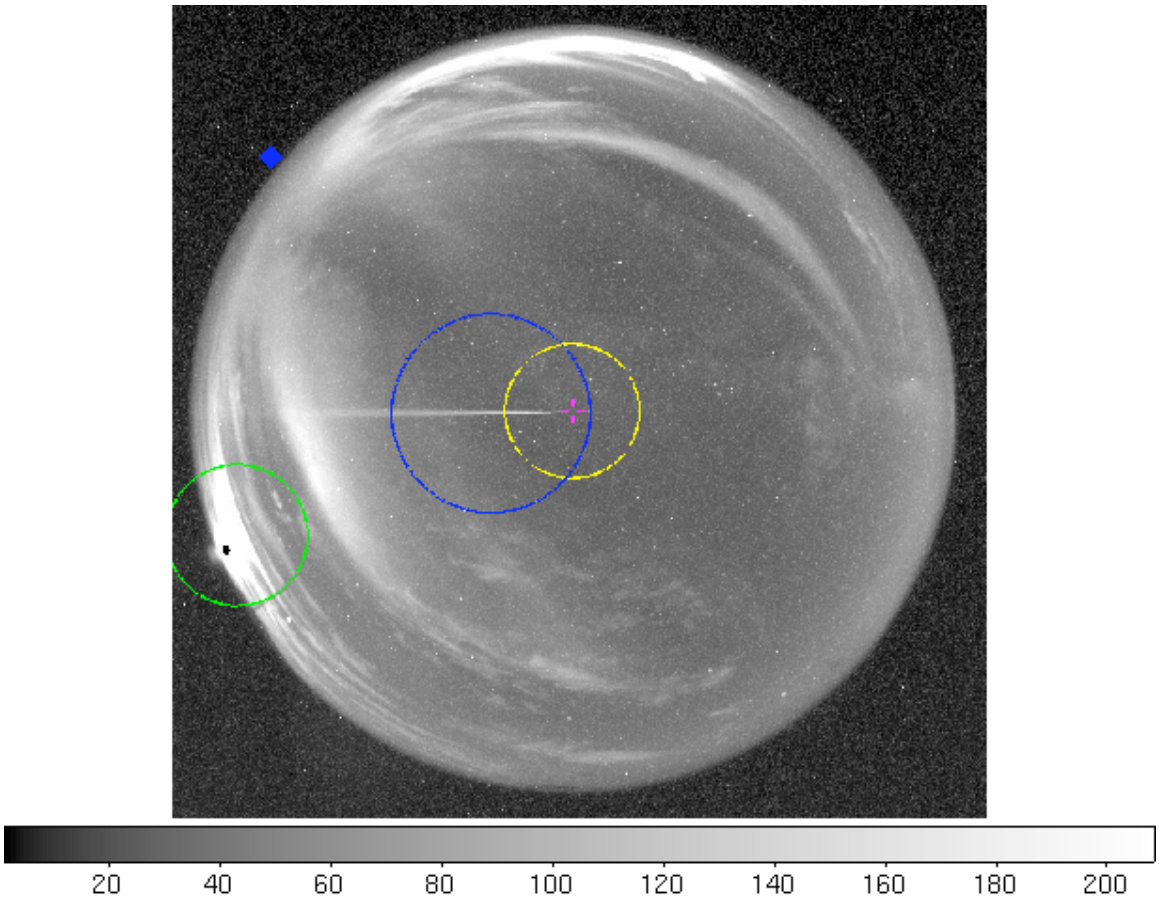

Image 12 – Laser Image, Frame 3

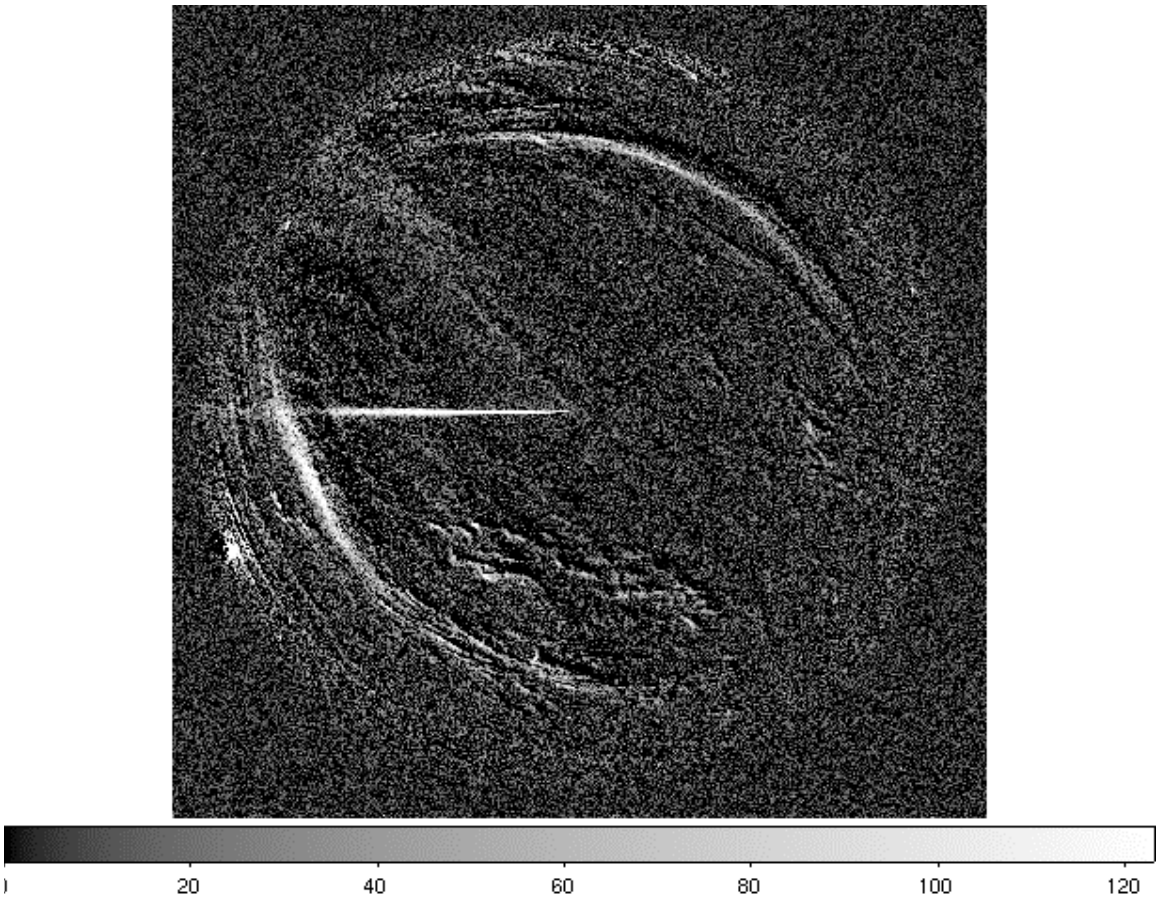

Image 13 – Laser Image Diff, Frame 2

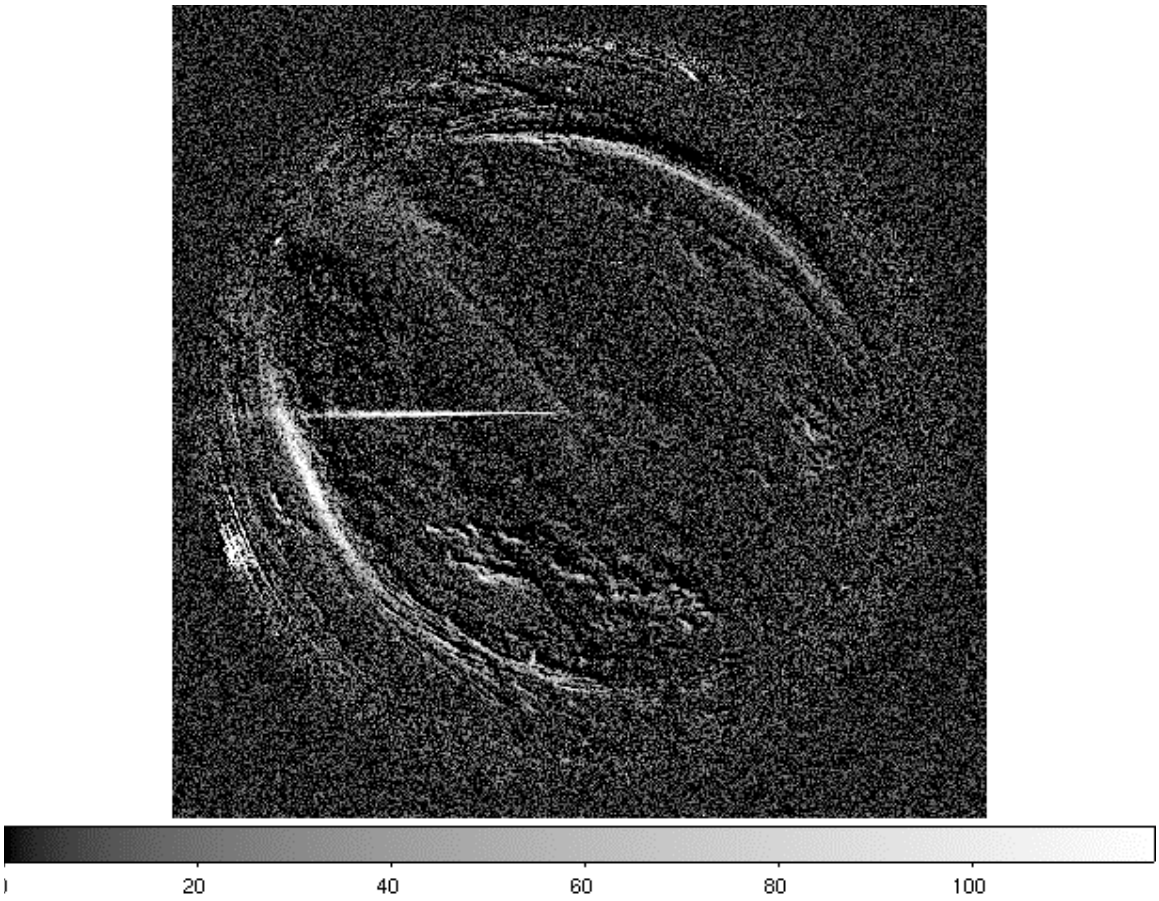

Image 14 – Laser Image Diff, Frame 3

Image 15 – Laser Image Mask Frame 2

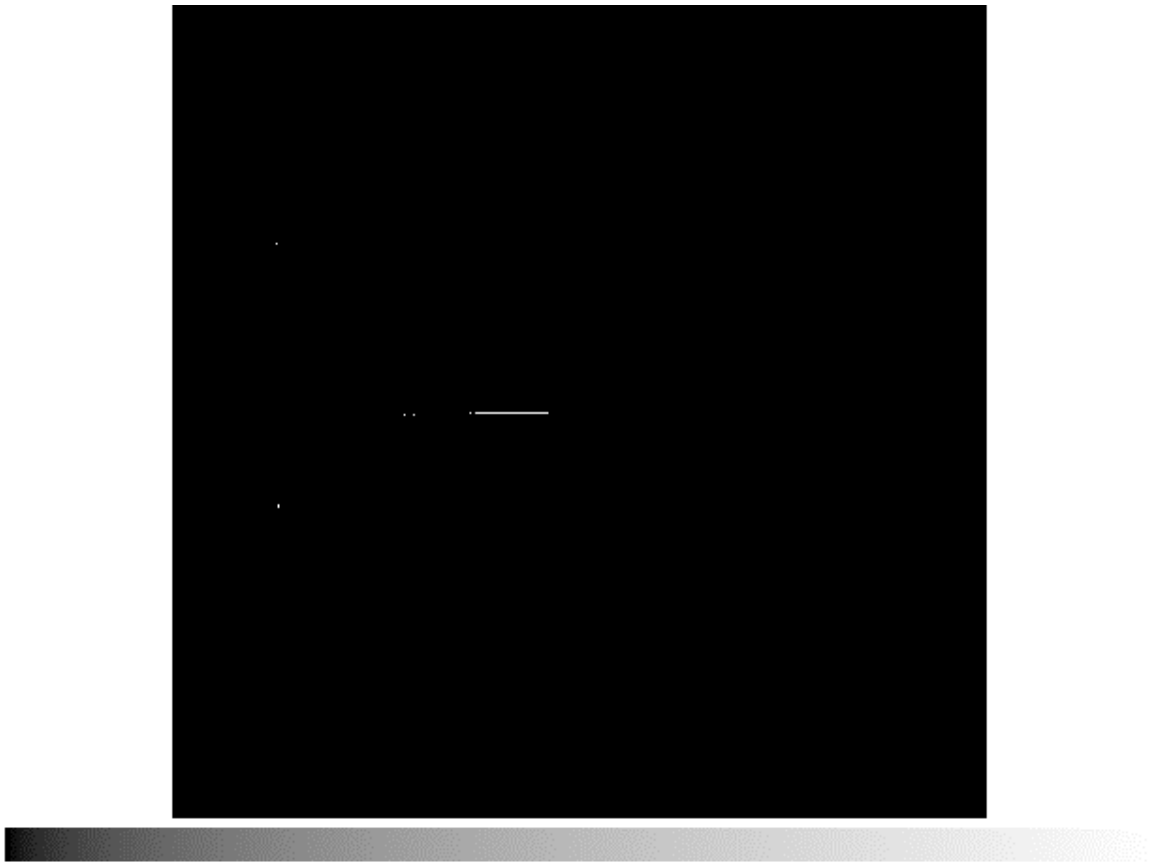

Image 16 – Laser Image Mask Frame 3

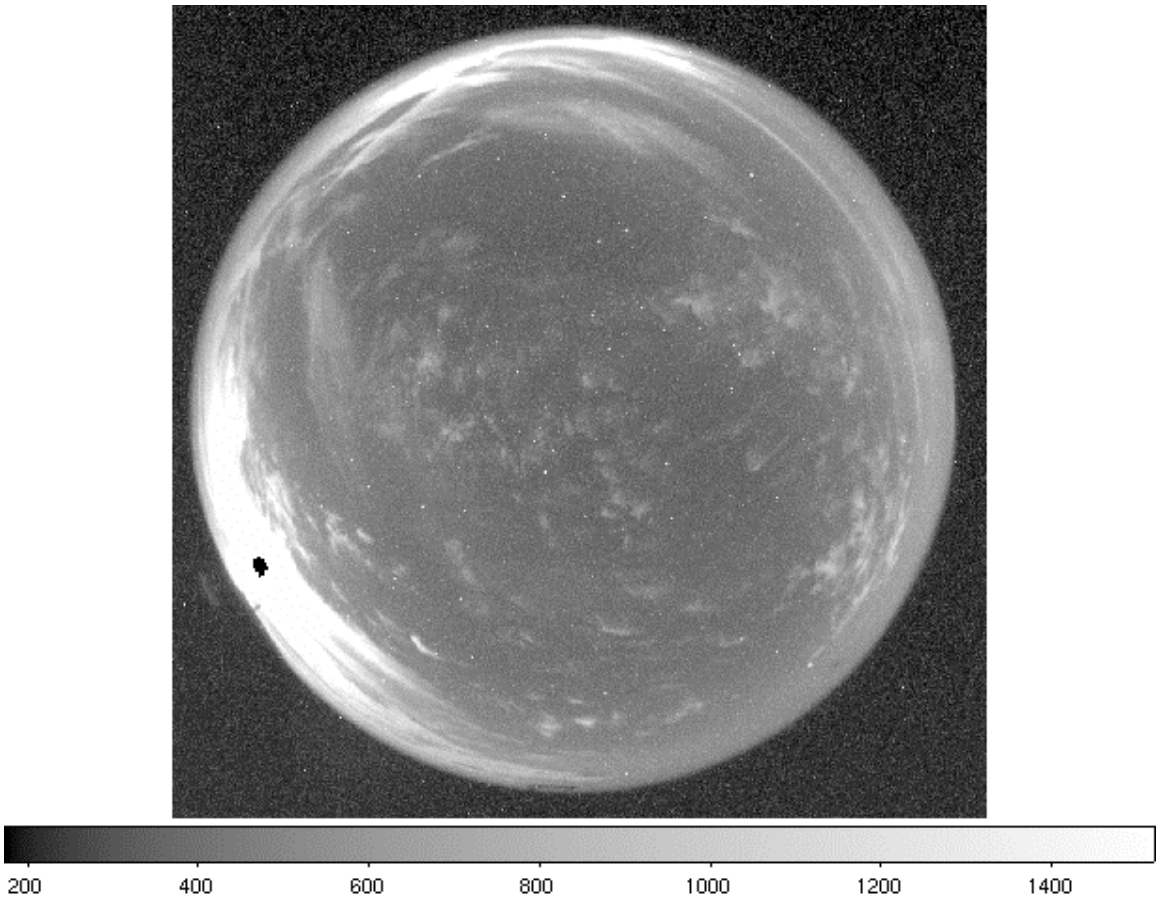

Image 17 – Clouds Image, Unprocessed

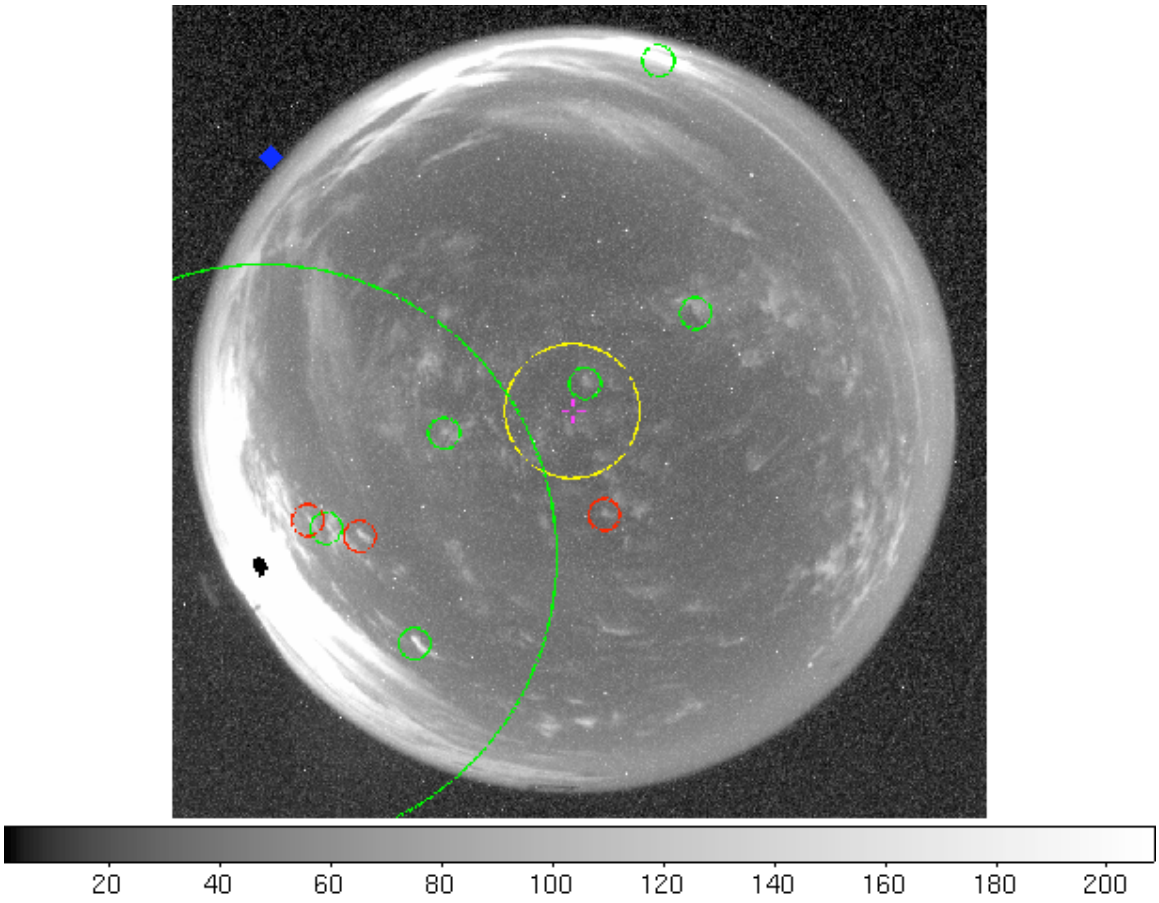

Image 18 – Clouds Image, Processed

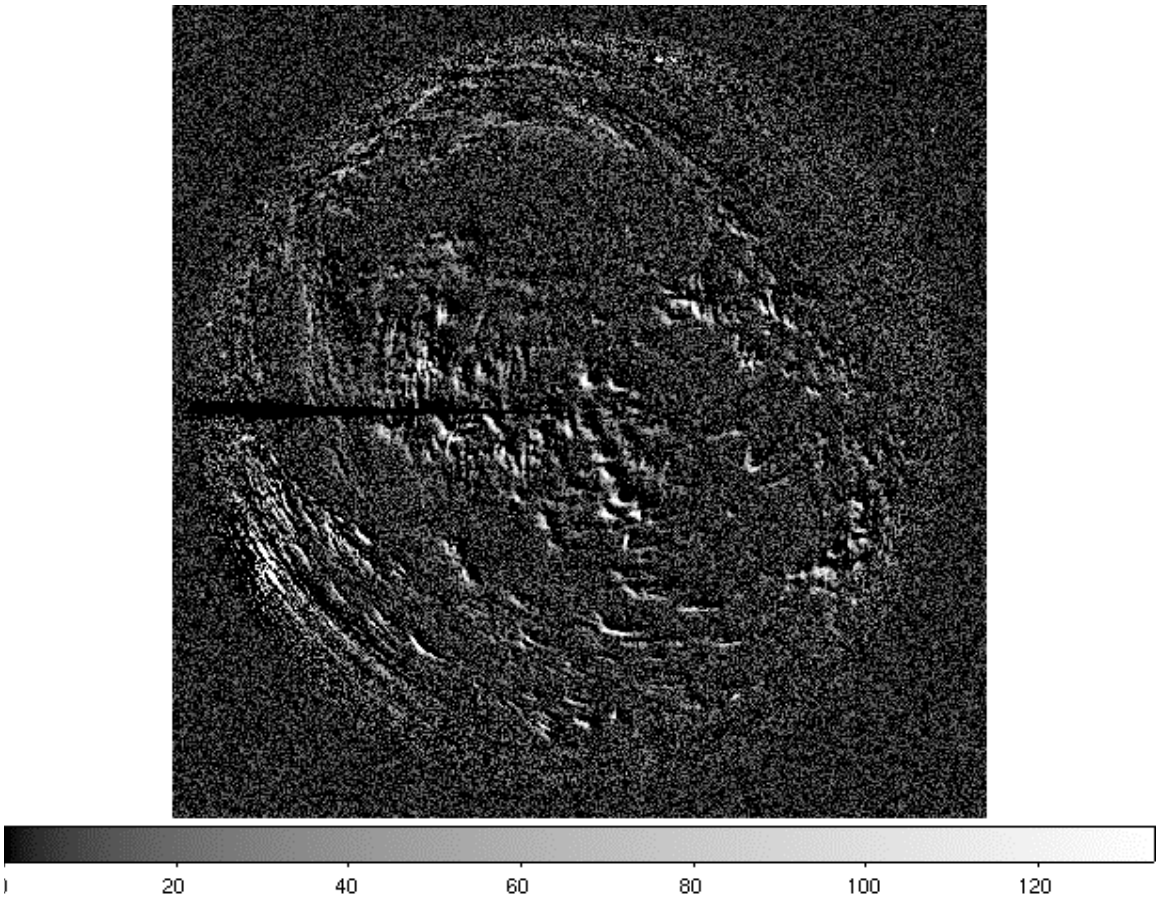

Image 19 – Clouds Image Difference

Image 20 – Clouds Image Mask

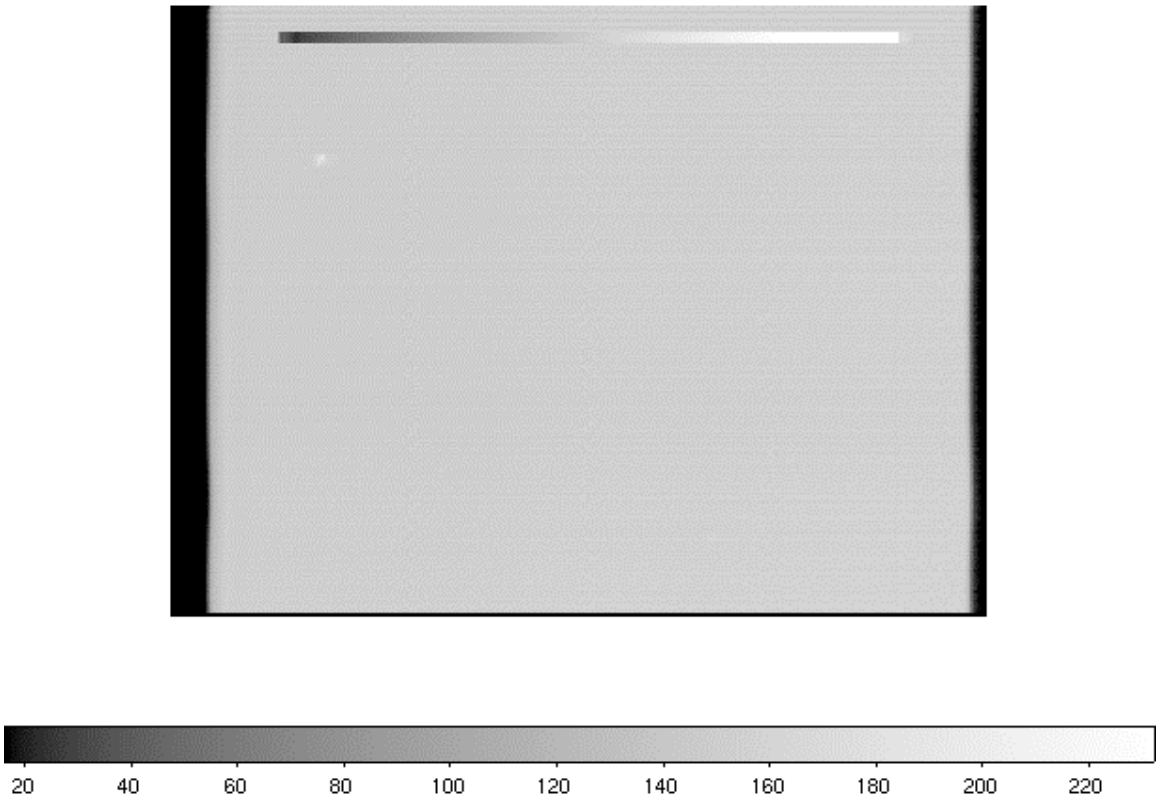

Image 21 – IR Aircraft Detection #1, Unprocessed

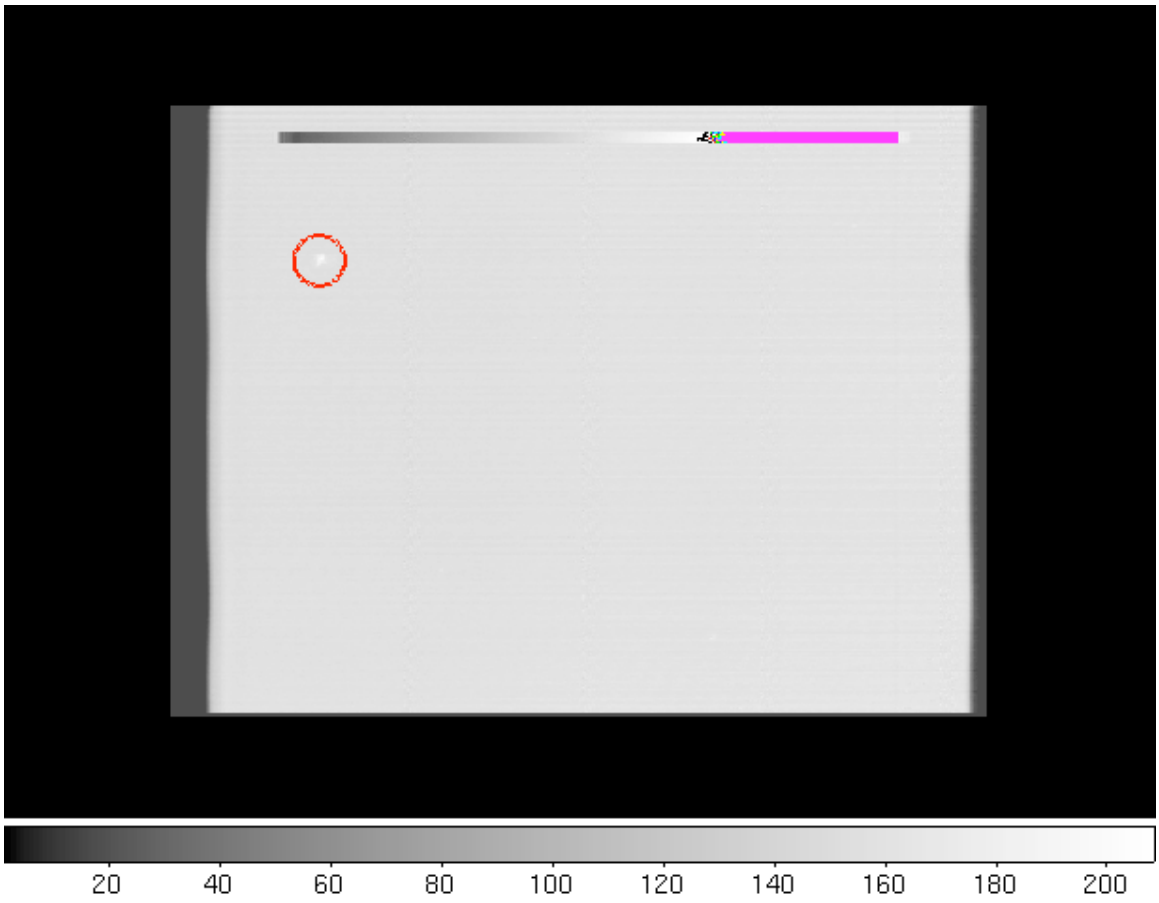

Image 22 – IR Aircraft Detection #1, Processed

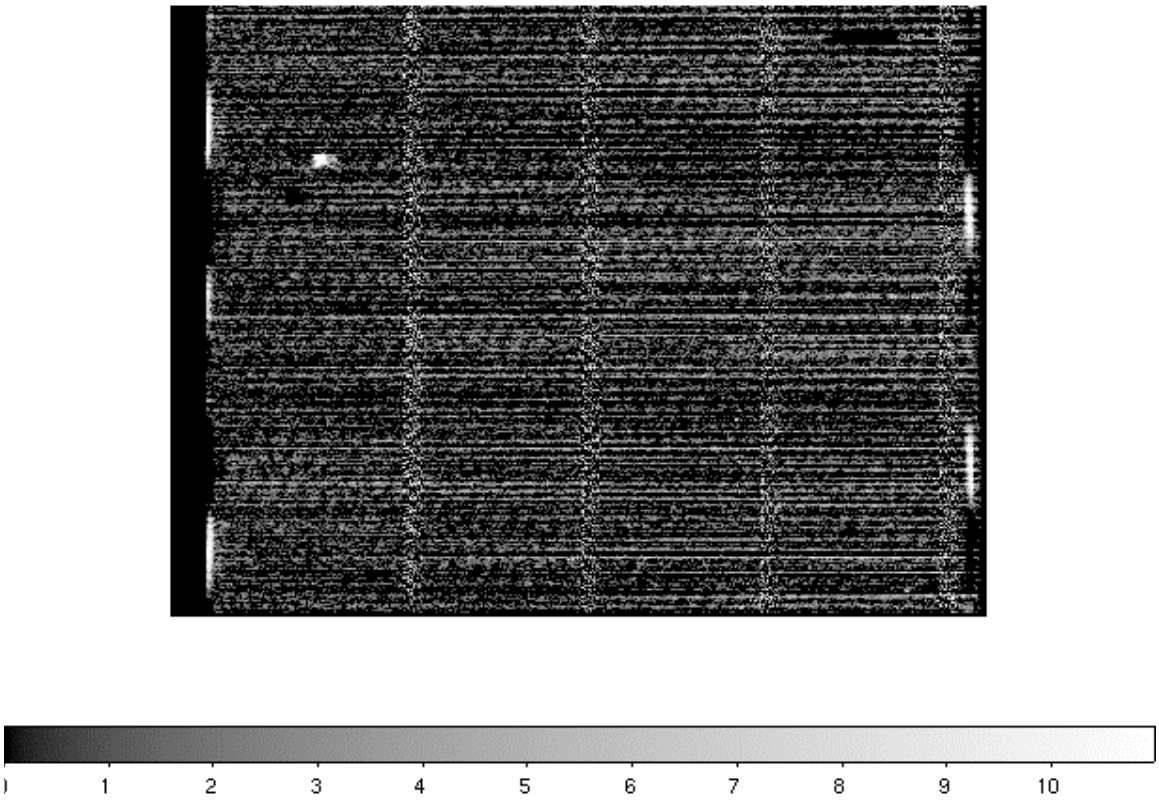

Image 23 – IR Aircraft Detection #1 Image Difference

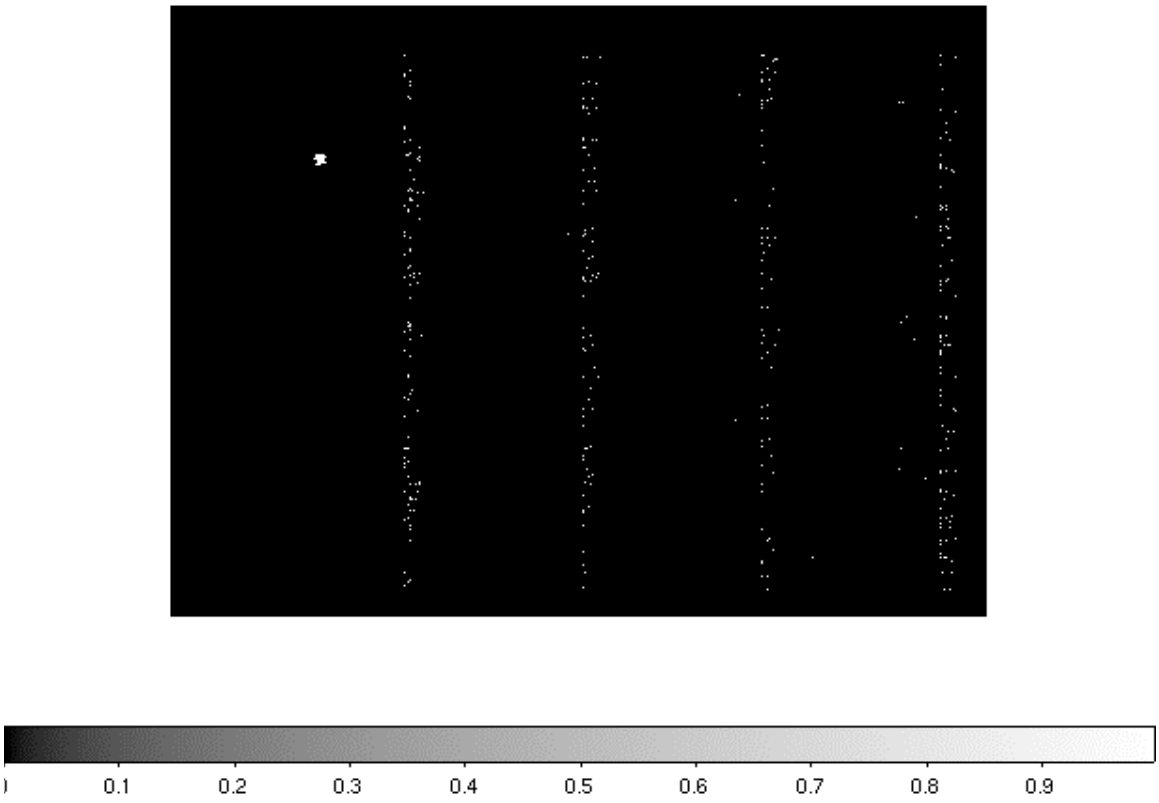

Image 24 – IR Aircraft #1 Detection Mask

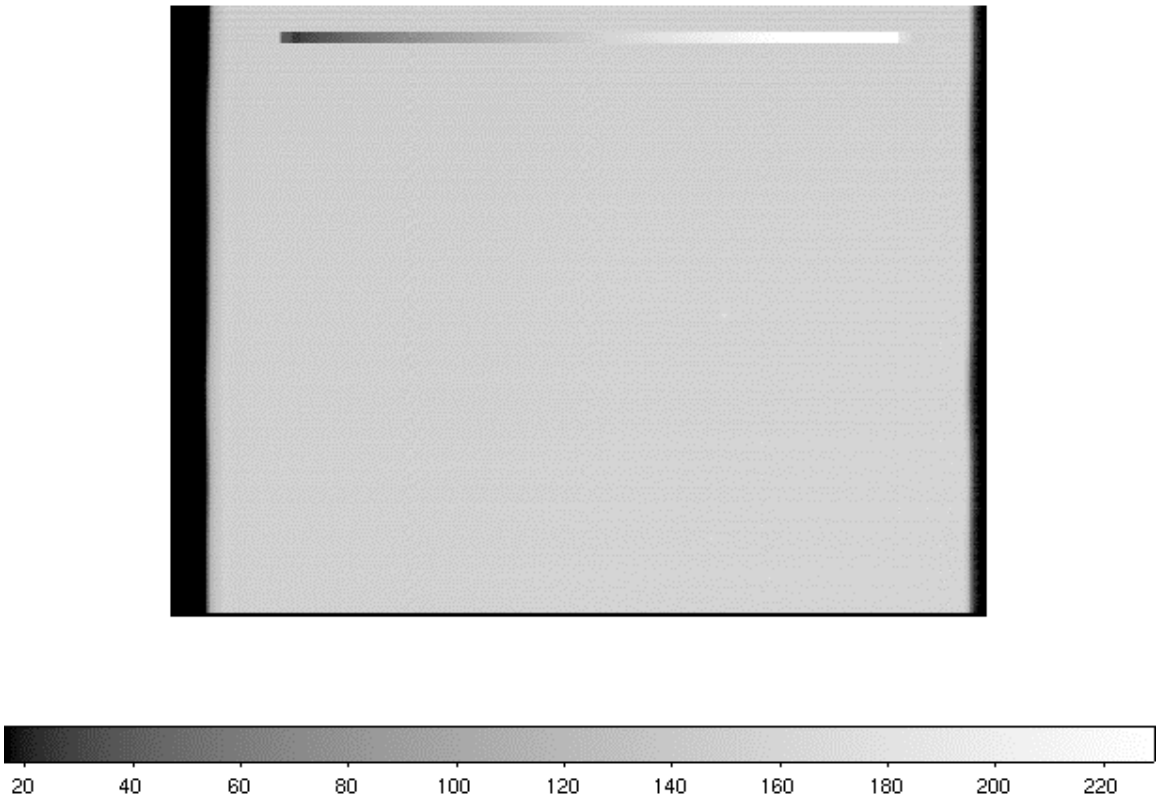

Image 25 – IR Aircraft Detection #2, Unprocessed

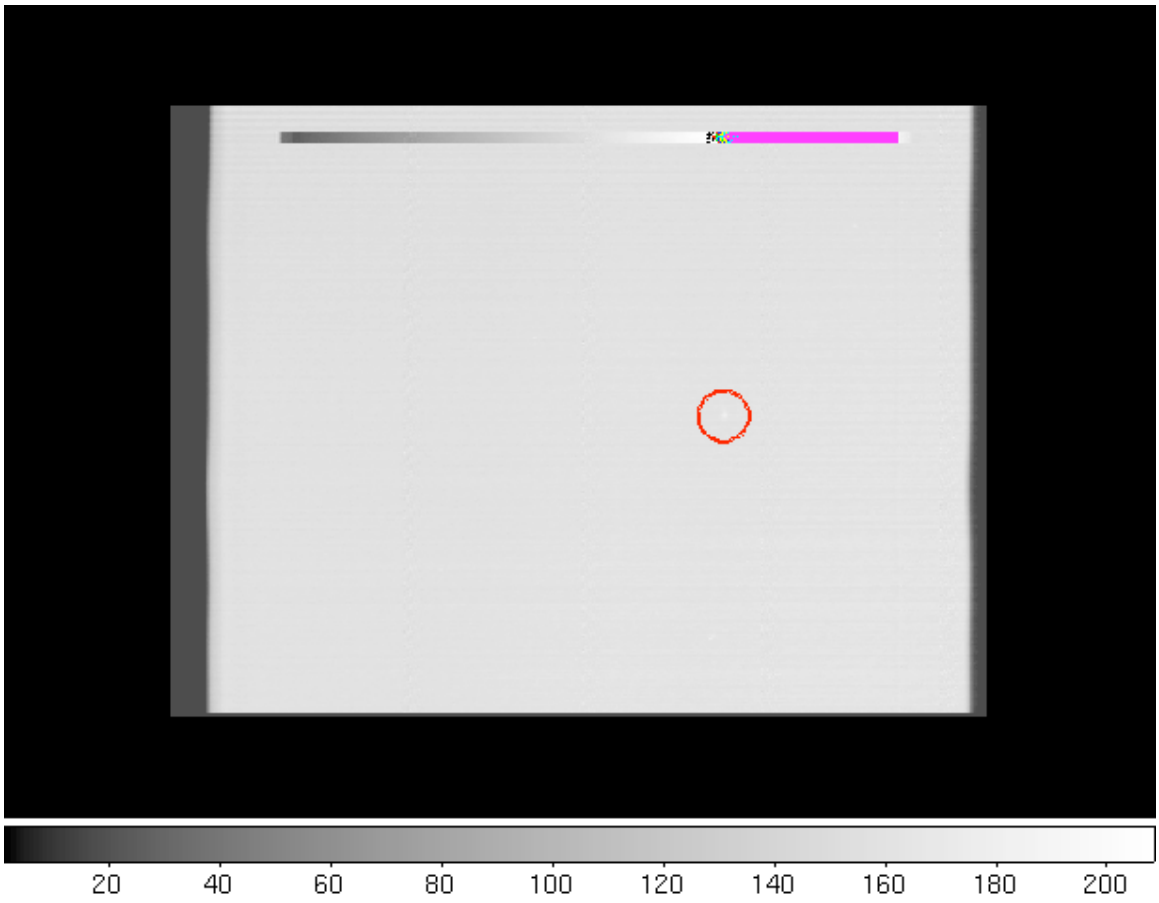

Image 26 – IR Aircraft Detection #2, Processed

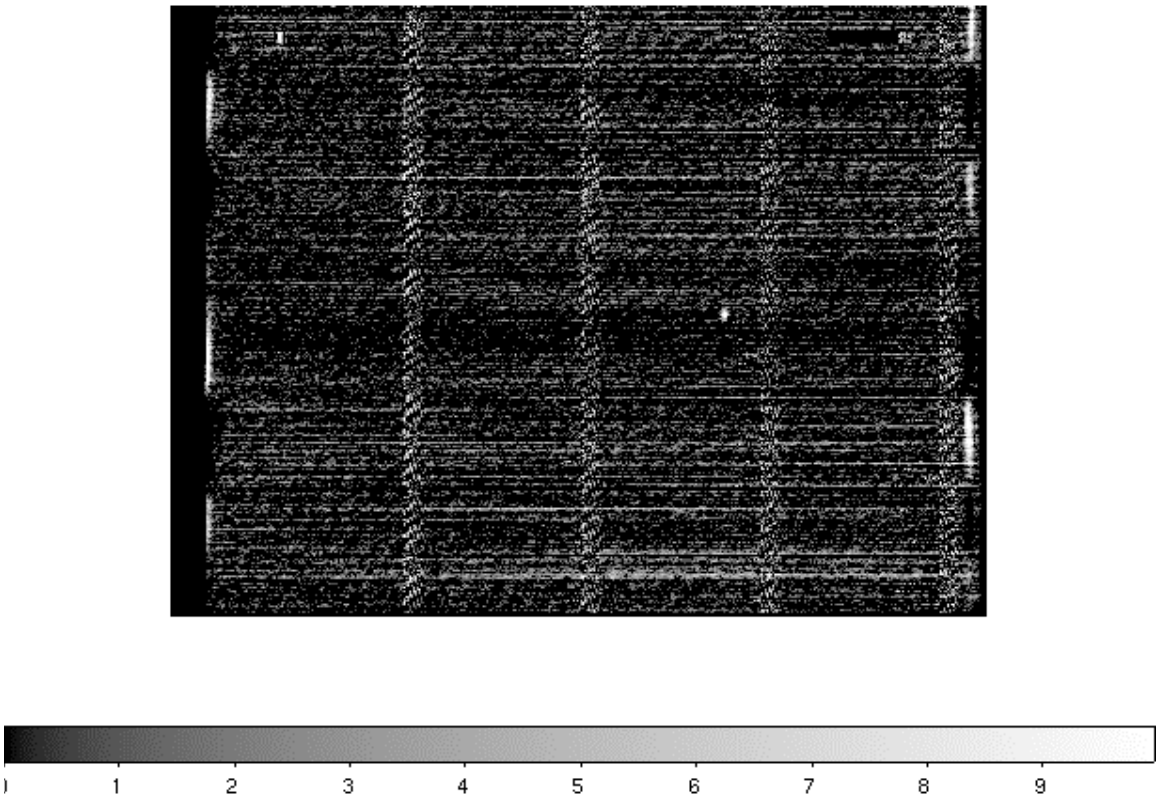

Image 27 – IR Aircraft Detection #2 Image Difference

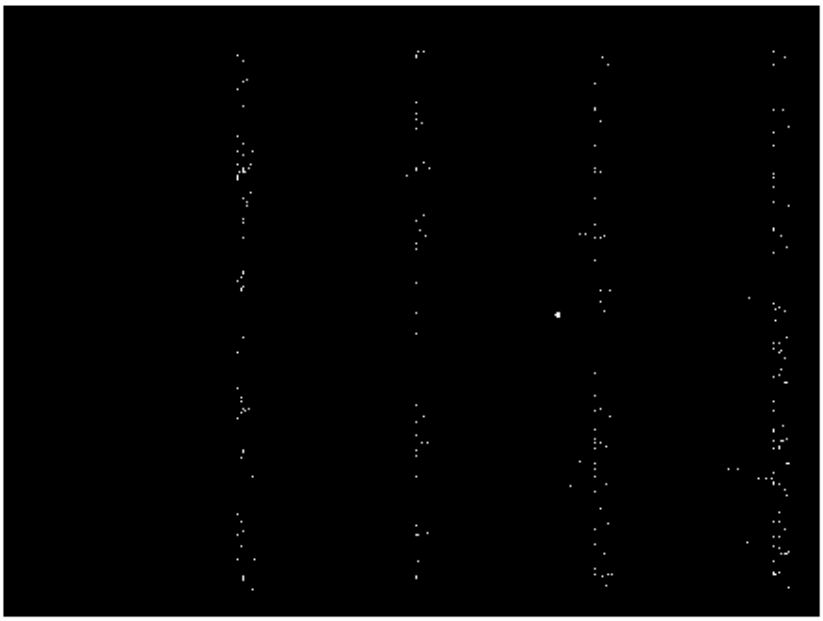

Image 28 – IR Aircraft #2 Detection Mask