Computer Vision in *FIRST*

Sam Carlberg
Brad Miller

Agenda

- Overview
- What was new in 2018
- Camera hardware options
- cscore
  - CameraServer (NetworkTable integration)
- GRIP
  - Code generation
- WPILib - VisionThread/VisionRunner
- End-to-end Demo with Raspberry Pi 3

Overview

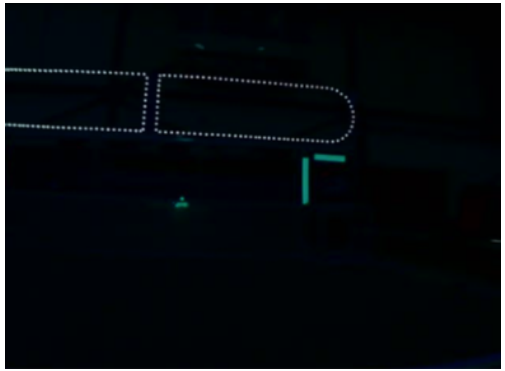
- What is computer vision?
- Photography + Computer Processing
- Start with good photos
  - Exposure & lighting
  - Composition & focal length
- Then process digitized values to measure, compare, and rate

Overview

- Math... not magic
- Example – What makes the target unique?
- Describe it to a friend on the phone/email
- Quantify that info by measuring
  - Color, size, area/CH-Area,
  - Aspect Ratio, Moment of Inertia
  - Limit Tests

FRC Workflow

- Goals on the field have often been marked with retro-reflective tape
- Make use of provided images on first day to design vision algorithms
- Create a vision program based on the design
- Use robot control loops based on gyros or other fast sensors – usually not on camera frames



The 2018 tools

- OpenCV included with Eclipse plugins
  - NIVision removed
- CameraServer rewrite
- WPILib VisionThread

OpenCV

- Popular computer vision library
- C, C#, C++, Java, Python, and LabVIEW interfaces
- Runs on Linux, Windows, Mac, iOS, and Android
  - Libraries included with WPILib for the roboRIO
- Designed for computational efficiency and can take advantage of accelerated hardware (GPUs)
- User community of 47,000 people and over 9M downloads

Cameras

IP $$

USB Webcam $

Cell phone $$$

USB 3.0 industrial $$$

Pixy Cam $

LimeLight $$$

IP Cameras

- Typically used for security
- Axis has low latency and good API
- Support multiple clients
- Easiest to stream to web page or other viewers
- Fixed lens, basic configuration

USB Web Cameras

- Typically used for chat/video conference
- Price and quality range is HUGE
  - HD? 4K? $30 to $40 seems sufficient
- With CameraServer, can make any USB camera into an IP camera
- Mounting issues
- Fixed lens

Cell Phone Cameras

- Used for selfies, ..., and robots
- Write an app
- Integrated image processor
- SW support and documentation are good
- How to mount on robot
- Lens adapters available

PixyCam

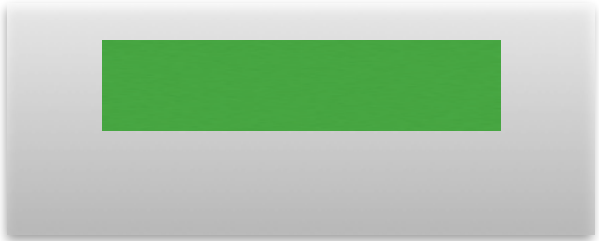
- Camera with built-in processor
- Limited capabilities, but has been used successfully by some teams (blob and color code tracking)
- Lots of output types (SPI, I2C, UART, analog/digital)
- Includes a cable to connect directly to an Arduino along with some libraries
- Very fast – 50 updates/sec

USB3 Industrial Cameras

- Used for machine vision – visual inspection
- Basler Dart and Point Grey Chameleon3 have recently become FRC-priced
- Operate at USB2 speeds on roboRIO
- SW support and documentation are excellent – GenICam
- 12mm S or CS lens/filter system
- Rich configuration, high performance

Basler Dart

- Sensitive sensor & Global Shutter
- 54fps color, 120 BW (USB3 full frame)
- Area scan – lower bw, even higher fps
- Hundreds of properties
- Digital lines for exposure control
- Lenses from companies like Edmunds



Hardware processing

- roboRIO – in an independent thread
- Driver station computer
- Onboard coprocessor
  - Raspberry PI
  - Kangaroo
  - NVIDIA TK1
  - Cell phone
  - LimeLight (Pi Compute Module 3)
  - ….

CameraServer

cscore

- Makes cameras easy to use
- Source→sink model
  - Can switch sinks between sources
- Support for USB and IP cameras
- Multiplatform, except USB camera support is Linux only
  - USB camera support for Windows and Mac planned for 2019
- Set camera properties (if supported by the camera)
  - Brightness
  - Exposure
  - Frame rate
  - Frame size
- MJPEG streaming webserver (looks like an IP camera)
- Webpage access to camera properties for easy experimentation

CameraServer

- Wrapper around cscore
- Automatically creates MJPEG server when camera is created
- Publishes information about created cameras to NetworkTables
- Restreams IP cameras via roboRIO USB port

Code examples

- Single USB camera

```
CameraServer.getInstance().startAutomaticCapture();
```

- Multiple USB cameras

```
CameraServer.getInstance().startAutomaticCapture("Front", 0);
CameraServer.getInstance().startAutomaticCapture("Back", 1);
```

- Absolute USB camera path

```
CameraServer.getInstance().startAutomaticCapture("Front", "/
dev/v4l/by-path/platform-3f980000.usb-usb-0:1.2:1.0-video-
index0");
```

- Axis camera

```
CameraServer.getInstance().addAxisCamera("10.2.94.11");
```

Code examples

- Camera Settings

```
UsbCamera frontCamera =
    CameraServer.getInstance().startAutomaticCapture("Front",
    "/dev/v4l/by-path/platform-3f980000.usb-usb-0:1.2:1.0-video-
    index0");
frontCamera.setVideoMode(PixelFormat.kMJPEG, 320, 240, 30);
frontCamera.setBrightness(50);
frontCamera.setWhiteBalanceHoldCurrent();
frontCamera.setExposureManual(15);
```

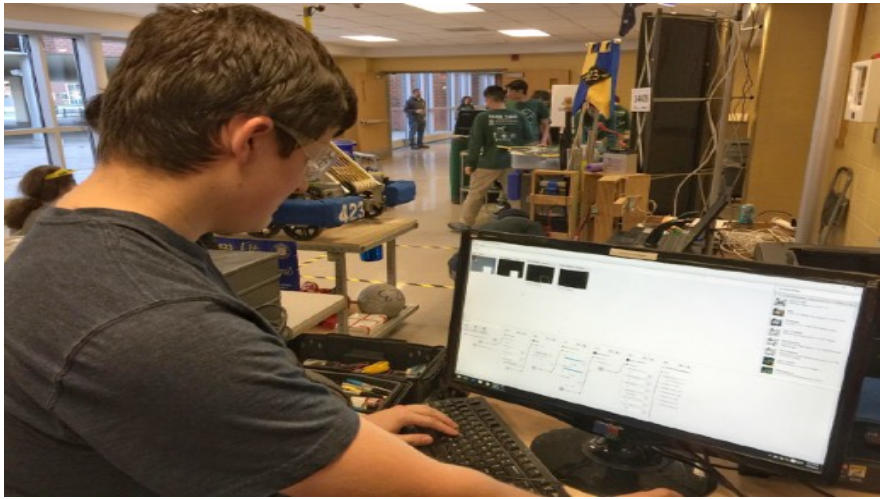
roboRIO CameraServer Demo

- Hook up a USB camera
- Add one line of code
- View camera in Shuffleboard and SmartDashboard
- View camera details in Outline Viewer
- View camera and camera settings in browser

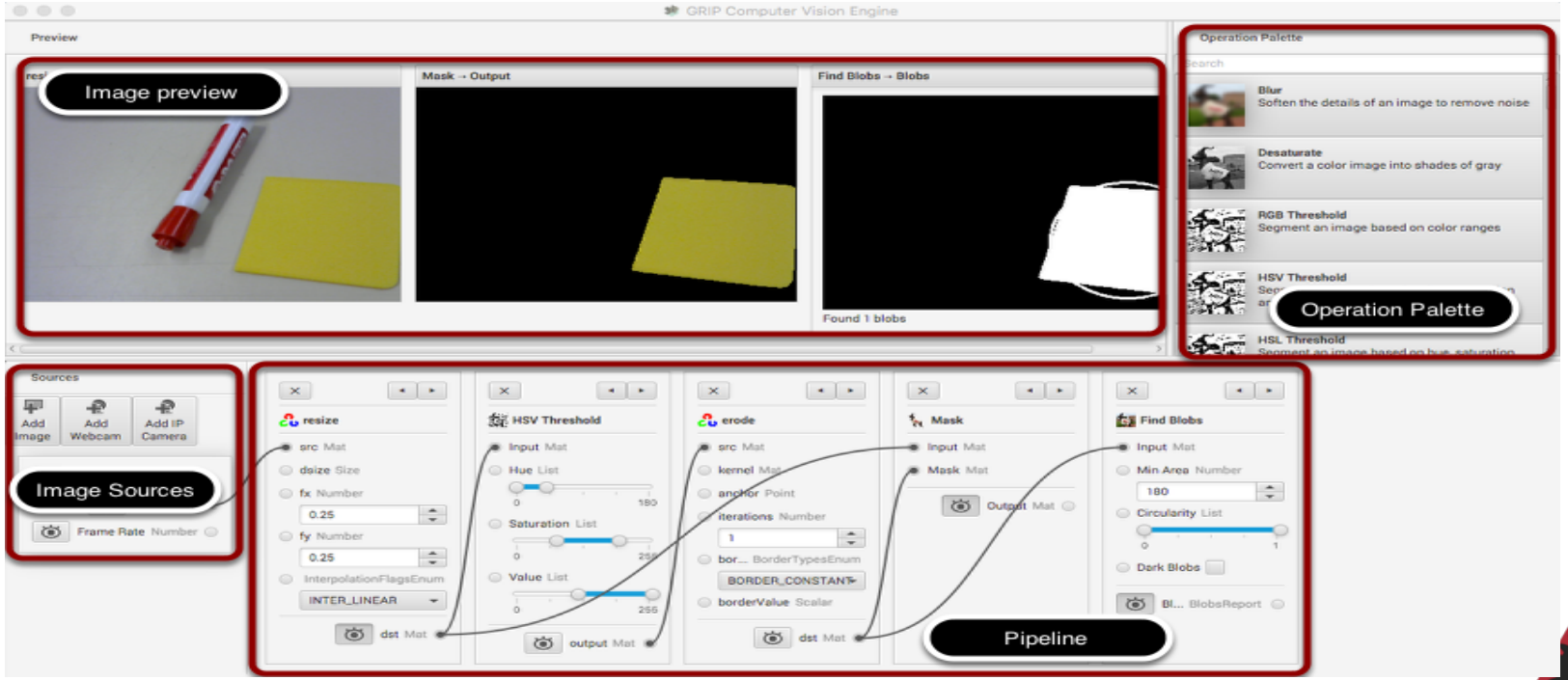
GRIP

GRIP

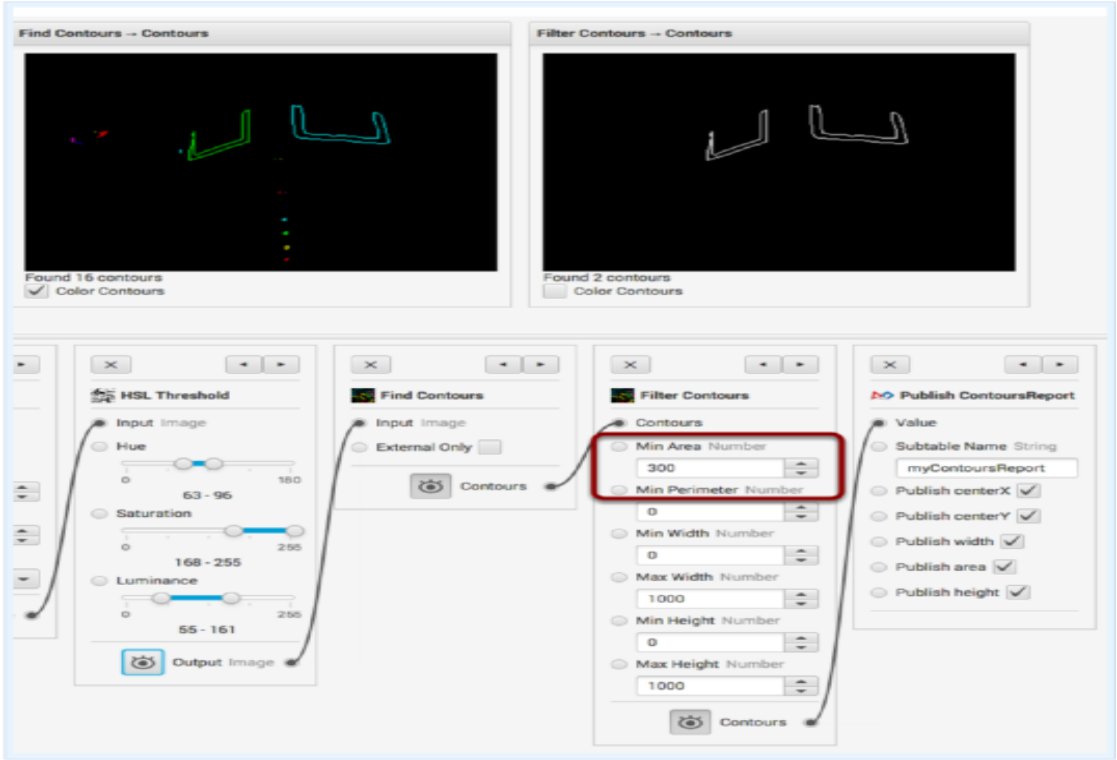
- Developed as a WPI RBE/CS senior capstone project
- Designed to make vision processing easier
  - FRC teams and researchers
- Based on OpenCV
- Works equally well with any robot programming language



GRIP User Interface



GRIP


Find Contours → Contours
Found 16 contours
Color Contours
Filter Contours → Contours
Found 2 contours
Color Contours
HSL Threshold
Input Image
Hue
0
180
63 - 96
Saturation
0
255
168 - 255
Luminance
0
255
55 - 161
Output Image
Find Contours
Input Image
External Only
Contours
Filter Contours
Contours
Min Area Number
300
Min Perimeter Number
0
Min Width Number
0
Max Width Number
1000
Min Height Number
0
Max Height Number
1000
Contours
Publish ContoursReport
Value
Subtable Name String
myContoursReport
Publish centerX
Publish centerY
Publish width
Publish area
Publish height

GRIP

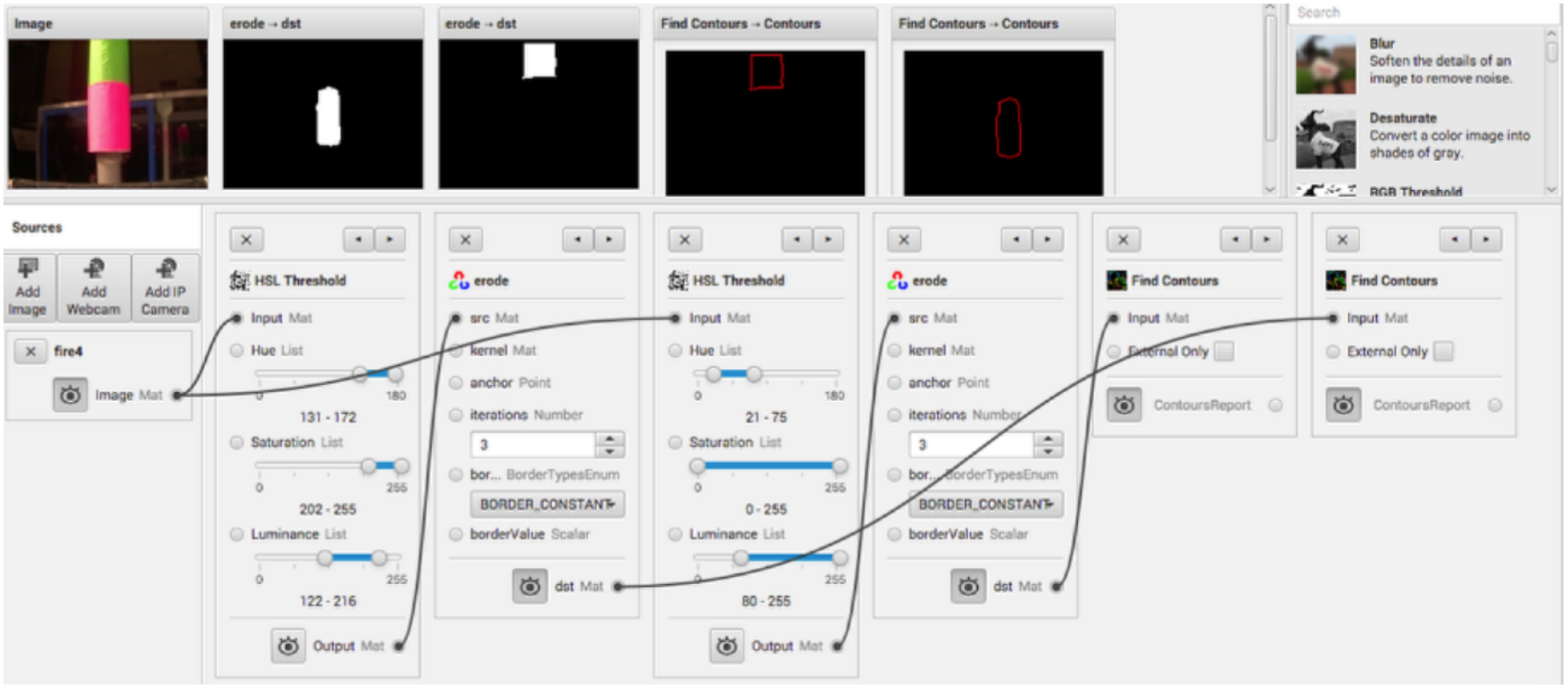


Image
erode → dst
erode → dst
Find Contours → Contours
Find Contours → Contours
Search
Blur
Soften the details of an image to remove noise.
Desaturate
Convert a color image into shades of gray.
RGB Threshold
Sources
Add Image
Add Webcam
Add IP Camera
fire4
Image Mat
HSL Threshold
Input Mat
Hue List
0
180
131 - 172
Saturation List
0
255
202 - 255
Luminance List
0
255
122 - 216
Output Mat
erode
src Mat
kernel Mat
anchor Point
iterations Number
3
bor... BorderTypesEnum
BORDER_CONSTANT
borderValue Scalar
dst Mat
HSL Threshold
Input Mat
Hue List
0
180
21 - 75
Saturation List
0
255
0 - 255
Luminance List
0
255
80 - 255
Output Mat
erode
src Mat
kernel Mat
anchor Point
iterations Number
3
bor... BorderTypesEnum
BORDER_CONSTANT
borderValue Scalar
dst Mat
Find Contours
Input Mat
External Only
ContoursReport
Find Contours
Input Mat
External Only
ContoursReport

GRIP Workflows

- Select a source – camera, video, or saved images
- Create the processing pipeline while viewing intermediate results
- Set outputs via network tables variables
- Robot program gets values and drives/turns/aims at target

Three workflows:

1. GRIP on DS laptop
2. GRIP generated code on roboRIO
3. GRIP generated code on co-processor

GRIP Results

- What happened in 2017
  - 16,000 downloads!
  - Code generation
  - HTTP REST API
  - Many teams running on roboRIO, driver station and co-processors
- Future features
  - GPU acceleration
  - Improved FRC usage
  - Limelight will support Grip in 2019

GRIP Demo

LimeLight camera

Simple to start

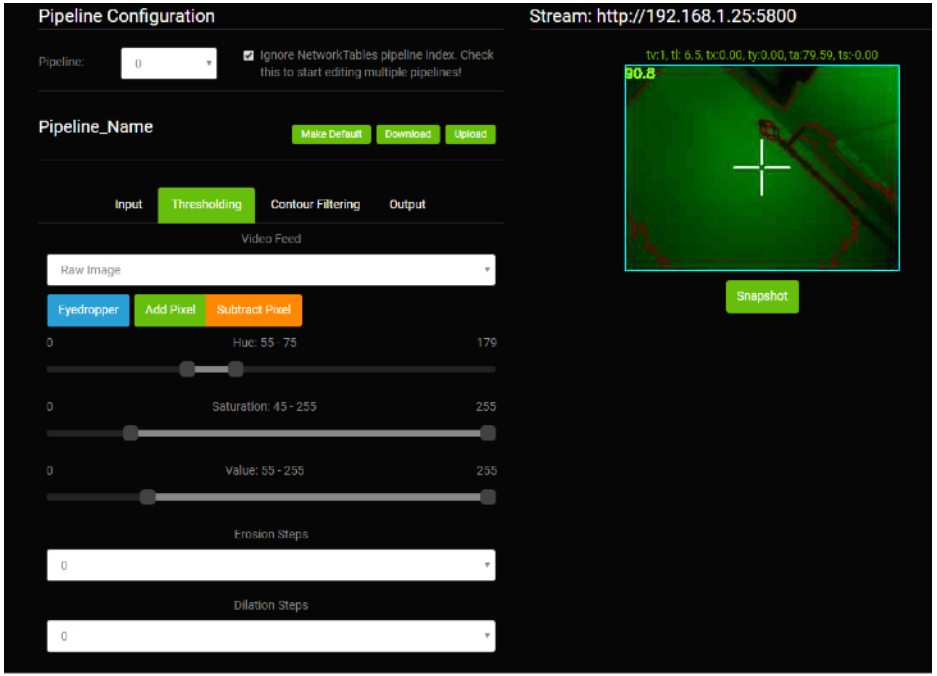
- Designed for FRC camera system that works with Java, C++, LabVIEW and Python
- Includes camera, LED lights, microprocessor with network tables, and a web server
- Easy to mount, easy to power, built in vision-pipeline
- Tune the pipeline parameters using web-browser
- Robot reads data from the “LimeLight” NetworkTables table

```
NetworkTable table = NetworkTable.getTable("limelight");
double targetOffsetX = table.getNumber("tx", 0);
double targetOffsetY = table.getNumber("ty", 0);
```

LimeLight camera 2

- Has built-in “Fixed Function” OpenCV vision pipeline
  - Threshold
  - Erode
  - Dilate
  - Filter contours
  - Targeting logic
- Up to 10 pipeline configs
- Second USB camera can be added to the feed

- Soon: Upload your custom GRIP pipeline to LimeLight!



LimeLight Camera Demo

Raspberry Pi 3 Demo

Raspberry Pi 3 Demo

- 2 cameras hooked up to Pi3 (both USB)
  - Pi Camera also supported but not demo'ed
- Pi configured with read only file system
- For demo purposes, building with Eclipse project
  - Could use gradle, etc
- Deploy GRIP pipeline with generated code

Uses Raspberry Pi 3 libraries from here: https://www.chiefdelphi.com/forums/showthread.php?t=161767

WPILib C++/Java Vision Future Plans

- Easier deployment mechanisms for vision code to coprocessors
- Prepackaged image for Raspberry Pi 3
- CameraServer USB camera support for Windows and Mac
- CameraServer web dashboard improvements
  - Set video mode
  - Change visible stream resolution / compression / FPS
  - Display stats on active streams
- CameraServer support for changing JPEG compression level without also changing resolution (with MJPEG source)
- CameraServer persistent settings (save/load from web and software)

Ultimate Simplicity

- Like Jeff Goldblum said in Jurassic Park, “life finds a way”
- No camera – just a flashlight for aiming with the vision sensor being the drivers at the driver station