GoCV/openCV

An application of Go

Admin

-
- Lets look at the news

Open CV

- OpenCV
  - Is a c++ (with some C) library for doing real time computer vision.
  - Runs on all major platforms
    - Linux,
    - Windows (with a lot of work),
    - MacOS (except maybe apple silicon)
    - Android
  - Need gcc/g++
    - Minggw for you windows users without WSL

OpenCV

- OpenCV is 20+ years old
  - First released in 2000.
  - So very mature, and provides lots of functionality
    - So we can’t possibly cover it all
- wrappers/interface in many programming languages.
  - Python, java, Matlab, javascript
  - And of course GO!!

GoCV

- GoCV is the go wrapper around goCV
  - https://gocv.io/
  - Its a bit of a bear to make, which is why I asked you to do it over the week.

Simple Tutorial

- Lets dig through the first tutorial

```
import (
    "gocv.io/x/gocv"
)

func main() {
    webcam, _ := gocv.VideoCaptureDevice(0)
    window := gocv.NewWindow("Hello")
    img := gocv.NewMat()

    for {
    webcam.Read(&img)
    window.IMShow(img)
    window.WaitKey(1)
    }
}
```

VideoCapture

- First
- gocv.VideoCaptureDevice
  - Open the camera and get ready to stream
  - Second return val is error if if fails to open the camera

The window

- Easiest way to see the results is to use the gocv window
- gocv.NewWindow(<name>)
- Two params,
  - First is name, if there is already a window with this name, then it won't open a second
  - Second param is flags
  - Only default flag is autoresize
    - First window pops up small. Close it and window reopens correct size

Bit Flags

- How many of you have used bit flags before?
  - ?

Bit Flags

- How many of you have used bit flags before?
  - ?
  - Based on our experiences this semester I'll guess about 20% of you
    - Common in lower level CE/embedded programming where efficiency is very important.
- lets take a quick diversion and come back to bit flags

What is the biggest cost?

- What is the biggest expense today in computer programs?
  - Lucky volunteer?

What is the biggest cost?

- What is the biggest expense today in computer programs?
  - Lucky volunteer?
- What was the biggest expense 40+ years ago?
  - Maybe even 25-30 years ago

What is the biggest cost?

- What is the biggest expense today in computer programs?
  - Programmer time
- What was the biggest expense 40+ years ago?
  - Maybe even 25-30 years ago
  - Computing equipment
  - What and why of the “y2k” issue

Developers vs Programs

- So today you can get a linode https://www.linode.com/pricing/ for $5/month at the low end
- While a go developer is often making $100k or more in the US
  - https://www.indeed.com/salaries/golang-developer-Salaries
  -

So what do we optimize for?

- So given the expense of developers and inexpensive tools
  - For what should we optimize in our development?
  - Lucky volunteer?

So what do we optimize for?

- So given the expense of developers and inexpensive tools
  - For what should we optimize in our development?
  - Lucky volunteer?
  - Yup – definitely developer time.
  - And what do developers spend the most time on in any sort of “real” software?
    - Lucky volunteer again?

So what do we optimize for?

- So given the expense of developers and inexpensive tools
  - For what should we optimize in our development?
  - Lucky volunteer?
  - Yup – definitely developer time.
  - And what do developers spend the most time on in any sort of "real" software?
    - Lucky volunteer again?
  - Reading the existing code to understand where and what changes to make

Go philosophy

- The Go language is built for enterprise development
  - Google heritage of course
  - Small set of keywords
  - Slow pace of changes and major features
    - Good go code from 5 years ago still very readable today.
    - Java from 5 years ago?
    - Even python?
  - So go long emphasis on readability
    - Gofmt helps

Optimize for what?

- So when do we not want to optimize for developer time (readability)?
  - Lucky volunteer

Optimize for what?

- So when do we not want to optimize for developer time (readability)?
  - Those rare cases where performance is critical and worth the extra development time
    - Real time systems (rockets, autonomous robots, self-driving cars)
    - Anything else which continuous performance needs
      - Deep internals of an operating system
      - graphics/game loop
      - Highly optimized libraries used by developers whose internals most people will not see.

And now back....

- And now back to our regularly scheduled topic
  - So opencv and the Qt window library fall into some of those highly optimized back end libraries
  - QT runs the Tesla in-car GUI and other places with limited compute.
  - So they use optimizations that might be overkill elsewhere

Bitflags

- In much of our computing
  - Boolean flags
  - In c++ and go booleans are 1 byte
    - In python and java even more
  - Not too much for 8 gigs of RAM
  - But if you need 16 flags and open hundreds of windows – can be a lot
    - What is a window discussion

Bitflags II

- So solution use a byte or int16 to store a bunch of values each is a power of two
- Example from open cv cpp headers
  - enum cv::MouseEventFlags {
  - cv::EVENT_FLAG_LBUTTON = 1,
  - cv::EVENT_FLAG_RBUTTON = 2,
  - cv::EVENT_FLAG_MBUTTON = 4,
  - cv::EVENT_FLAG_CTRLKEY = 8,
  - cv::EVENT_FLAG_SHIFTKEY = 16,
  - cv::EVENT_FLAG_ALTKEY = 32
  - }

Bitflags III

- Use bitwise or to use two flags together
  - flags:= EVENT_FLAG_RBUTTON | EVENT_FLAG_CTRLKEY
- Then use bitwise and to check to see if a flag is set
  - if flags & EVENT_FLAG_RBUTTON {
    - This code only reached of the EVENT_FLAG_RBUTTON value was set in the flag
  - flags : 00001010
  - EVENT_FLAG_RBUTTON 00000010
  -

Remember that program

- Lets dig through the first tutorial

```
import (
    "gocv.io/x/gocv"
)

func main() {
    webcam, _ := gocv.VideoCaptureDevice(0)
    window := gocv.NewWindow("Hello")
    img := gocv.NewMat()

    for {
    webcam.Read(&img)
    window.IMShow(img)
    window.WaitKey(1)
    }
}
```

Mat

- img := gocv.NewMat()
- Creates a new material ‘Mat’
  - With zero value
- From docs: “Mat represents an n-dimensional dense numerical single-channel or multi-channel array. <and more>”
  - So this is going to be our image

Get this frame

- Get the current frame

```
webcam.Read(&img)
```

  - Pass pointer to a material which gets filled
  - Return value is boolean – false if can’t read

Show the image

- Now we show the image on the window

```
window.IMShow(img)
```

- Needs WaitKey next or will not show
- WaitKey takes param: minimum number of milliseconds to wait
- WaitKey also polls events – so event processing happens here

Face Finder

- New lets try a face finder demo
- Before we can find faces need to get a classifier
  - https://github.com/opencv/opencv/tree/master/data
  - Basically a trained neural net in xml format
  - haarcascade_frontalface_default.xml
    - Large, but seems to work for wide variety of faces
    - Grab it from github

The new main

```
func main() {
webcam, err := gocv.VideoCaptureDevice(0) //or 1
if err != nil {
fmt.Println(err)
return
}
defer webcam.Close()
displayWindow := gocv.NewWindow("Find a face")
classifier := gocv.NewCascadeClassifier()
success := classifier.Load("haarcascade_frontalface_default.xml")
if !success {
log.Fatal("Failed to load classifier - can't continue")
}
defer classifier.Close()
FindFaces(webcam, displayWindow, classifier)
}
```

Face Finder

- Lets build it
- Note the false positives
- On the mac in my office I’m ‘superman’ it can’t find me with glasses – but without finds me fine

```
func FindFaces(camera *gocv.VideoCapture, window *gocv.Window, faceFiindingNet gocv.CascadeClassifier) {
img := gocv.NewMat()
defer img.Close()
for {
if ok := camera.Read(&img); !ok {
fmt.Printf("cannot read from camera!")
continue
}
if img.Empty() {
continue
}
potentialFaces := faceFiindingNet.DetectMultiScale(img)
for _, rectangle := range potentialFaces {
gocv.Rectangle(&img, rectangle, colornames.Darkkhaki, 3)
}
window.IMShow(img)
if window.WaitKey(10) >= 0 {
break
```

Now lets extend that

- Lets extend the face finder to do a privacy blur
  - Lets try that now

Now lets extend that

- Lets extend the face finder to do a privacy blur
  - Lets try that now
  - faceRegion := img.Region(rectangle)
  - gocv.GaussianBlur(faceRegion, &faceRegion, image.Pt(55, 95), 0, 0, gocv.BorderDefault)
  - faceRegion.Close()

Now lets extend that

- Lets extend the face finder to do a privacy blur
  - Lets try that now
- One more if we have time
  - Lets write “redacted” over the blur