Signal Flags

TechmandanCZ

Úloha o programování! To je něco pro mě (zatím jediný speed bonus z minulých let byl za analýzu grafu) :)

Minulý rok (myslím) byla obdobná úloha, avšak u té na řešení stačilo pouze porovnávat barvy 2 pixelů, byly pouze dvě sekce které byly vždy jednobarevné. To byly časy…
Tentokrát se na obrázcích musely rozpoznat vlajky. A navíc námořní.

Analýza obrázků? To zní jako python věc. S tím já ale moc neumím, a po tom co jsem se marně snažil obrázky vůbec načíst do opencv jsem si řekl že to určitě půjde jednodušeji.

Začal jsem tím že jsem našel stránku která převáděla text na námořní vlajky, a všiml jsem si že pouze načte různé obrázky typu gif (avšak neanimované), a tak jsem si v rychlosti napsal script který všechny vlajky stáhnul. Zároveň jsem se podíval na obrázky, a všiml si že všechny vlajky stejného typu jsou stejně velké.

A začal jsem si psát script (tentokrát node.js). Nejdřív jsem začal tím že najdu podle zelených vlajek všechny vlajky a jejich velikosti:

```
let detected = [];

for(let x = 0; x < width; x++) {
  for(let y = 0; y < height; y++) {
    const [r, g, b, a] = getPixel(x, y)
    if(detected.find(([startX, startY, endX, endY]) ⇒ x ⩾ startX && x ⩽ endX && y ⩾ startY && y ⩽ endY)) {
        continue
    }
    let isgreen = r==0 && g > 50 && b==0
    if(isgreen) {
        // detect rectangle size
        let startY = y;
        let endY = y;
        while(endY < height) {
            const [r, g, b, a] = getPixel(x, endY)
            if(r==0 && g > 50 && b==0) {
                endY++
            } else {
                endY--;
                break
            }
        }
        let startX = x;
        let endX = x;
        while(endX < width) {
            const [r, g, b, a] = getPixel(endX, startY)
            if(r==0 && g > 50 && b==0) {
                endX++
            } else {
                endX--;
                break
            }
        }
        if(endX - startX < 5 || endY - startY < 5)
            continue
        detected.push([startX, startY, endX, endY])
    }
  }
}
```

Daný algoritmus skenuje obrázek zleva doprava, ze shora dolů, a hledá zelené obdélníky. Kvůli malým “tykadlům” je kontrola že jsou obdélníky větší jak 5 pixelů do obou stran.

A následně už jsem se pustil do detekce vlajek.
Začal jsem čísly, těch je pár, to natluču do kódu za chvilku. No, asi tak po půl hodině jsem měl tuto nádheru:

```
    // number
    const mid = getPixel(x + w/2, y + h/2)
    const leftmid = getPixel(x + w/4 + w/10, y + h/2)
    const rightmid = getPixel(x + w/4*3 - w/7, y + h/2)
    const topleft = getPixel(x + w/4 + w/7, y + h/4)
    const bottomleft = getPixel(x + w/4 + w/7, y + h/4*3)
    const midIsRed = mid[0] > 50 && mid[1] < 10 && mid[2] < 10
    const leftmidIsYellow = leftmid[0] > 50 && leftmid[1] > 50 && leftmid[2] < 10
    const rightmidIsYellow = rightmid[0] > 50 && rightmid[1] > 50 && rightmid[2] < 10
    const leftmidIsRed = leftmid[0] > 50 && leftmid[1] < 10 && leftmid[2] < 10
    const rightmidIsRed = rightmid[0] > 50 && rightmid[1] < 10 && rightmid[2] < 10
    const rightmidIsBlue = rightmid[0] < 10 && rightmid[1] < 10 && rightmid[2] > 50
    const midIsBlue = mid[0] < 10 && mid[1] < 10 && mid[2] > 50
    const leftmidIsWhite = leftmid[0] > 250 && leftmid[1] > 250 && leftmid[2] > 250
    const rightmidIsWhite = rightmid[0] > 250 && rightmid[1] > 250 && rightmid[2] > 250
    const midIsWhite = mid[0] > 250 && mid[1] > 250 && mid[2] > 250
    const topleftisBlack = topleft[0] < 10 && topleft[1] < 10 && topleft[2] < 10
    const bottomleftisBlack = bottomleft[0] < 10 && bottomleft[1] < 10 && bottomleft[2] < 10
    const bottomleftisWhite = bottomleft[0] > 50 && bottomleft[1] > 50 && bottomleft[2] > 50

    let redyellowmix = [ 232, 19, 0, 255 ]
    let blackwhitemix = [ 236, 236, 236, 255 ]

    if(leftmidIsYellow && midIsRed && rightmidIsYellow) {
        return 0
    } else if(leftmidIsRed && midIsWhite && rightmidIsWhite) {
        return 1
    } else if(leftmidIsWhite && midIsBlue && rightmidIsBlue) {
        return 2
    } else if(leftmidIsRed && midIsWhite && rightmidIsBlue) {
        return 3
    } else if(leftmidIsWhite && midIsWhite && rightmidIsWhite) {
        return 4
    } else if(leftmidIsYellow && rightmidIsBlue) {
        return 5
    } else if(carr(leftmid, blackwhitemix) && carr(mid, blackwhitemix) && carr(rightmid, blackwhitemix)) {
        return 6
    } else if(carr(leftmid, redyellowmix) && carr(mid, redyellowmix) && carr(rightmid, redyellowmix)) {
        return 7
    } else if(leftmidIsRed && midIsRed && rightmidIsRed) {
        return 8
    } else if(carr(leftmid, [232,20,20,255]) && carr(mid, [236,219,2,255])) {
        return 9
    } else console.log(leftmid, mid, rightmid)
```

Která kontrolovala různé hodnoty na různých pixelech (pozn. getPixel bere x,y a vrátí [r,g,b]; carr porovná že mají 2 array stejné hodnoty).

Tak jsem si řekl, a dost, tohle se mi nechce psát pro všech 26 písmen, a nedejbože všech možných vlajek (neprojel jsem obrázky abych věděl kolik je vlajek, a ani se mi nechtělo). Během hledání online jsem narazil na repozitář který porovnával rozdíl barev, a to jsem si řekl že bude dobrý nápad na jednoduché řešení, za podmínek že obrázky mají podobné barvy (což mají) a stejnou velikost (to jsem zařídil svým prvním scriptem).

```
function numCutoff(num) {
    if(num < 15) return 0
    if(num > 240) return 255
    return num
}

/**
 * @param {import("pngjs").PNGWithMetadata} src
 * @param {import("pngjs").PNGWithMetadata} dst
 * @param {number} sx Source X coordinate
 * @param {number} sy Source Y coordinate
 */
function imageDiff(src, dst, sx, sy) {
    let diff = 0;
    const width = dst.width;
    const height = dst.height;

    for(let x = sx; x < sx + width; x++) {
        for(let y = sy; y < sy + height; y++) {
            const srcIndex = (src.width * y + x) << 2
            const dstIndex = (width * (y - sy) + (x - sx)) << 2
            diff += numCutoff( num: Math.abs(src.data[srcIndex] - dst.data[dstIndex]))
            diff += numCutoff( num: Math.abs(src.data[srcIndex + 1] - dst.data[dstIndex + 1]))
            diff += numCutoff( num: Math.abs(src.data[srcIndex + 2] - dst.data[dstIndex + 2]))
        }
    }

    return diff
}
```

A to se ukázalo že fungovalo ještě lépe než zamýšleno, dělalo to chybu pouze v jednom případě, a to u písmen “s” a “x”.
S:

X:

Které, dle mého laického pohledu, nevypadají na to že by měli dělat chybu, avšak dělají, hlavně kvůli tomu že na lodích je proužek lehce užší, a ty rozdíly se nasčítají…
Avšak, jelikož už jsem něco věděl o porovnávání pixelů, jsem přidal jednoduchou kontrolu toho, zda-li u písmena “s” není modrý pixel vlevo, uprostřed a vpravo, a jestliže ano, tak to není písmeno “s” ale “x”.

```
  // letter
  const mid = getPixel(x + w/2, y + h/2)
  const leftmid = getPixel(x + w/4 + w/7, y + h/2)
  const rightmid = getPixel(x + w/4*3 - w/7, y + h/2)
  const leftmidIsBlue = leftmid[0] < 10 && leftmid[1] < 10 && leftmid[2] > 50
  const rightmidIsBlue = rightmid[0] < 10 && rightmid[1] < 10 && rightmid[2] > 50
  const midIsBlue = mid[0] < 10 && mid[1] < 10 && mid[2] > 50
  let minDiff = Infinity
  let minDiffLetter = null
  for(const letter of alphabetMap) {
      const diff = imageDiff( PNGWithMetadata src: png, PNGWithMetadata dst: letter[1], Number sx: x, Number sy: y)
      // console.log(diff, letter[0])
      if(diff < minDiff) {
          minDiff = diff
          minDiffLetter = letter[0]
      }
  }
  // console.log(minDiff, minDiffLetter)
  if(minDiffLetter === "s" && leftmidIsBlue && midIsBlue && rightmidIsBlue) {
      minDiffLetter = "x"
  }
  return minDiffLetter
```

A skoro hotovo, už jen něco s tím textem, který nedává smysl.
Tam už stačí pouze roztřídit podle hlavních vlajek, ale tam už jsem věděl co dělat - stačilo mít seznam existujících vlajek, a zkontrolovat, zda-li se vlajka lodi v tomto seznamu nenachází.

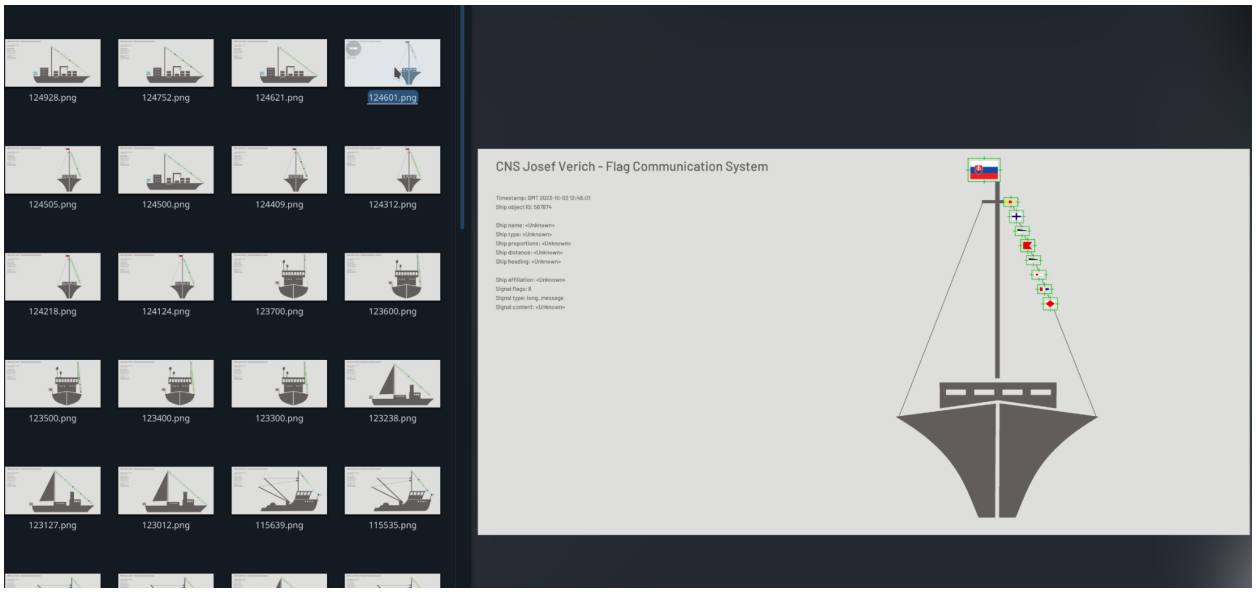
```
(w === 180) {
  // country code
  const mid = getPixel(x + w/2, y + h/2)
  const leftmid = getPixel(x + w/4 + w/7, y + h/2)
  const rightmid = getPixel(x + w/4*3 - w/7, y + h/2)
  const topmid = getPixel(x + w/2, y + h/4 + h/7)
  const bottommid = getPixel(x + w/2, y + h/4*3 - h/7)
  const countryCode = [mid,leftmid,rightmid,topmid,bottommid]
  const countryIndex = countries.findIndex(country => country.every((t, i) => carr(country[i], countryCode[i])))
  console.log("country", countryCode, countryIndex)
  if(countryIndex === -1) {
      countries.push(countryCode)
      return "country" + (countries.length - 1)
  } else {
      return "country" + countryIndex
  }
  return "country"
```

No, a nakonec vyřešit poslední můj přešlap - už vidím text který mi něco říká, občas text, občas HEX (0x…), a občas… reverse HEX (...x0)? Vlajky jsou vždy ze shora dolů avšak já upřednostňuji zleva doprava. A já než bych řešil prohození dvou písmenek ve for loop jsem radši přidal jednoduchý if:

```
if(flags[flags.length - 1] === 0 && flags[flags.length - 2] === "x") {
    flags.reverse()
}
if(flags[0] === 0 && flags[1] === "x") {
```

Pro moje řešení jsem každý text dal do .txt souboru podle názvu původního obrázku. A tady se našel další problém, ony obrázky nebyly seřazeny podle času. Číst text z obrázku? Tak blízko řešení, nechci řešit další problém s kódy, a tak otevírám průzkumníka souboru (delfína),

zvětšuju preview okénku a přepisuju manuálně název souboru na čas. Hezké? Ne. Rychlé? Možná?



A tím jsem měl úlohu vyřešenou, vlajka tam byla již hezky zmíněna v base64 (cyberchef pomohl).
A navíc jsem ani nemusel třídit podle vlajek zemí, nakonec se to akorát zobrazilo v terminálu ale neuložilo, a stejně fungovala odpověď.