Digitális képfeldolgozás gyakorlat,
Levelező tagozat 2024 tavaszi félév,
Beadandó feladat

Készítsen egy **Python programot** a megfelelő csomagok (OpenCV, NumPy, stb.) segítségével, amely a következő feladatok közül **egyet valósít meg**!

A feladatokhoz tartozó teszt és eredményképek a gyakorlat Coospace színterének Dokumentumok mappájában találhatóak a `kf2024_beadando_tesztkepek.zip` nevű állományba tömörítve.

Amennyiben a gyakorlati anyagban nem szereplő csomagot használ a megvalósításhoz, akkor annak pontos helyét és verziószámát a `pluszcsomag.txt` állományban mellékelni kell.

A gyakorlati jegy a beadott program minősége és összetettsége alapján kerül megállapításra. A főbb értékelési szempontok:

- A programban megvalósított képfeldolgozási műveletsor a kiadott tesztképe(ke)n megfelelően működik.
- Zaj hatásának megfelelő vizsgálata, azaz a só-bors és a Gauss additív zaj hogyan befolyásolja az algoritmus működését. Javasolt a programban biztosítani a zaj hozzáadásának lehetőségét.
- A probléma általános megoldása felé irányuló, komplexebb megoldás, a megadottakhoz hasonló jellegű további tesztképek vizsgálata.

A megadottakon kívüli esetleges további tesztképeket a beadáshoz csatolni lehet. A megvalósított algoritmusnál a különböző teszteseteket érdemes a kódban elhelyezni, hogy a futtatás során egyszerre látható legyen az eredmény.

Alap elvárás: Python+OpenCV használat, felhasználói felület az alap OpenCV vezérlőelemekkel (képek megjelenítése billentyű lenyomásig, egyes feladatoknál egérkezelés).

Formai követelmények

Készítsen egy mappát a saját nevével és Neptun azonosítójával, ékezetek nélkül (pl.: KissTamas_C42R4M)!

Az elkészített mappába másolja be a program működéséhez szükséges összes fájlt és a program dokumentációját PDF formátuban! (Forráskód, esetleg további tesztképeket is. Amennyiben túl nagyok a program futásához szükséges teszt állományok, akkor használja a `teszt.link` fájl-t, amibe az összecsomagolt képeket tartalmazó web linket lehet elhelyezni.).

Csomagolja be a mappát úgy, hogy a tömörített állományban szerepeljen a mappa információ is (azaz kicsomagoláskor automatikusan létrejöjjön egy mappa és abban a kért fájlok). Tömörítéshez a zip programokat használja (pl.: zip -r KissTamas_ C42R4M.zip KissTamas_C42R4M /)!

Az elkészült tömörített állományt töltse fel a CooSpace-re a ***Beadandó feladat*** néven kiírt feladat alá!

A beadott feladat csak akkor értékelhető, ha megfelel a feladatkiírásban leírt tartalmi és formai specifikációnak, valamint a program működőképes! A beadandó feladat elkészítésekor tetszőleges szakirodalom felhasználható, de a beadott programnak **a hallgató saját munkáját kell tartalmaznia**!

A beadott programot be kell mutatni a gyakorlatvezetőnek egy online védés alkalmával az utolsó gyakorlaton vagy a gyakorlatvezetővel előre egyeztetett egyéb időpontban.

A védés opcionálisan kiváltható egy videó készítésével, amelyben a hallgató a képernyőképet megjelenítve bemutatja a megvalósított képfeldolgozó műveletsort futás közben, és a forráskód alapján elmagyarázza annak működését a műveletsor minden fázisára kitérve (az egyes fázisok hogyan lettek megvalósítva). A hallgatónak a magyarázatot el kell mondania a videóban, a szóban történő bemutatás nem helyettesíthető feliratozással! A videó feltölthető külső tárhelyre is, ez esetben annak elérési linkjét kell megadni egy `video.link` nevű szöveges fájlban vagy a beadási felületen megjegyzésként.

A védés kiváltására további opcióként – videó helyett – lehet készíteni egy dokumentációt **PDF** formátumban, amely a fent említett információkat tartalmazza (a program funkciója/funkciói és a forráskód egyes részeinek magyarázata). A dokumentációnak számítógéppel szerkesztettnek kell lennie, kézzel írt anyag szkennelt változata nem elfogadható.

Ha a gyakorlatvezető a programhoz opcionálisan mellékelt videót vagy dokumentációt nem találja kellően részletesnek, vagy nem saját munka gyanúja merülne fel, akkor a gyakorlatvezető kérheti a program online védését is.

A beadott megoldás csak akkor értékelhető, ha az teljesíti az alábbi kritériumok mindegyikét:

- a megoldás megfelel a feladatkiírásban leírt tartalmi és formai specifikációnak,
- a program működőképes, továbbá
- a program meg lett védve, vagy a védés kiváltására készült hozzá a fenti feltételeknek eleget tevő videó vagy dokumentáció.

1. Mennyi az idő?

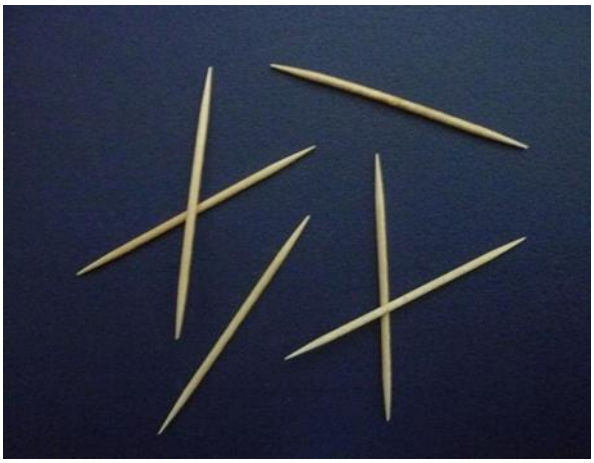
A feladat a faliora.jpg képen látható órán az idő meghatározása. Első lépésként a mutatókat kell szegmentálni, majd meghatározni az időt a mutató állásából.

Komplexebb megoldás: díszes mutatókkal rendelkező faliórák vizsgálata.

2. Pálcikák számolása

Írjunk olyan programot, amely a palcika1.jpg ill. a palcika2.jpg képen levő pálcikákat detektálja, megszámolja, és a kapott számot kiírja! A detektált pálcikákat jelenítse is meg jól kivehető - például vörös színű – vonalak berajzolásával! A könnyebb ellenőrízhetőség végett javasolt ezeket a vonalakat egyesével megjeleníteni, például úgy, hogy a program egy-egy újabb vonal rajzolása előtt egy-egy billentyű lenyomásáig várakozzon. A programnak egyszerre csak az egyik képet kell vizsgálnia, de mindkét képre helyesen kell működnie. (A palcika1.jpg képen 6, a palcika2.jpg képen pedig 9 db pálcika található.)

Komplexebb megoldás: különböző hosszúságú és színű pálcikák számolása.



3. Autópályán sávok színezése

A feladat a highway4.jpg képen levő, szaggatott záróvonallal jelölt úton a sávok címkézése és megjelenítése különböző színekkel (lásd például a lenti illusztrációt).

Komplexebb megoldás: több sáv (M0) szegmentálása, autót (bármilyen takarást) tartalmazó aszfaltrész jelölése.

(forrás: http://commons.wikimedia.org/wiki/File:J%C3%B5hvi-Tartu_highway_near_Tartu,_2007-12.jpg)

4. Gyalogátkelő detektálása

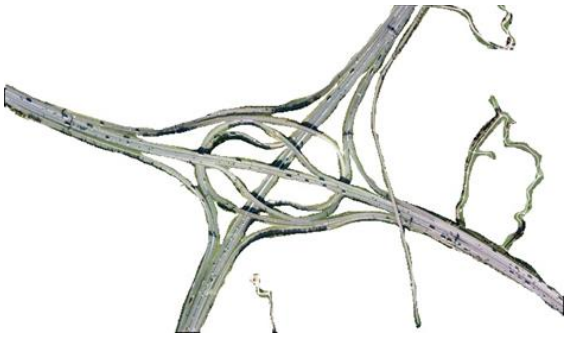
A feladat a crosswalk.jpg képen található zebra csíkjainak szegmentálása illetve kiszínezése. A program jelenítsen meg egy vagy több részeredmény képet. Egy lehetséges eredményt mutat a crosswalk_marked.jpg kép.

Komlexebb megoldás: a detektált területek egymással nagyjából párhuzamosak, az oldalsó csúcspontjai nagyjából egy egyenesre esnek.

5. Légi felvételen utak keresése

A program szegmentálja a roads.jpg képen található főbb utakat. Egy lehetséges eredményt mutat a roads_segmented.jpg kép.

Komplexebb megoldás: úttípusok elkülönítése vastagság alapján.



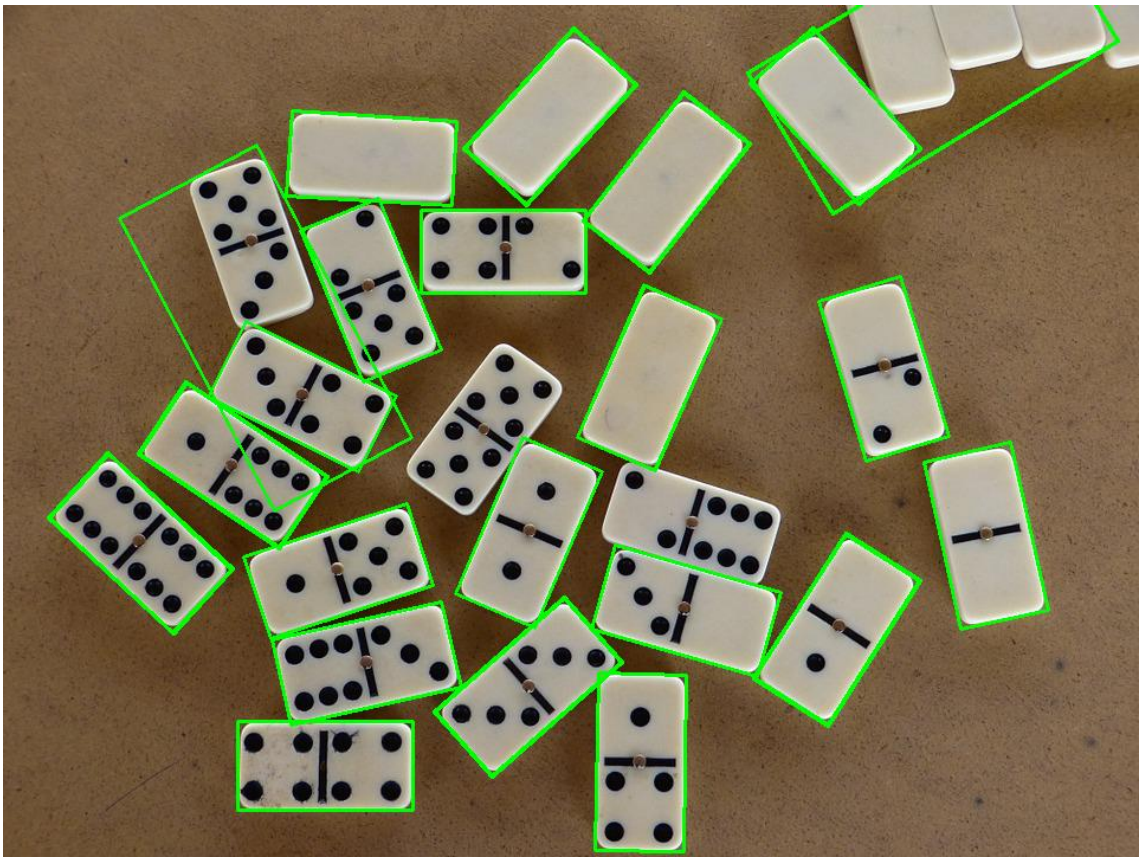
6. Dominó detekció

A feladat egy dominó kockákat tartalmazó képen a kockák detektálása. Feltételezhetjük, hogy a fotó az asztal síkjára nagyjából merőleges irányból készül, vagyis a perspektív torzulás elhanyagolható mértékű, valamint hogy a kockák nem fedik egymást. A detektált kockák befoglaló téglalapja kerüljön berajzolásra a képre, valamint az egyes kockák fekvő helyzetű téglalapként kerüljenek egyenként megjelenítésre.

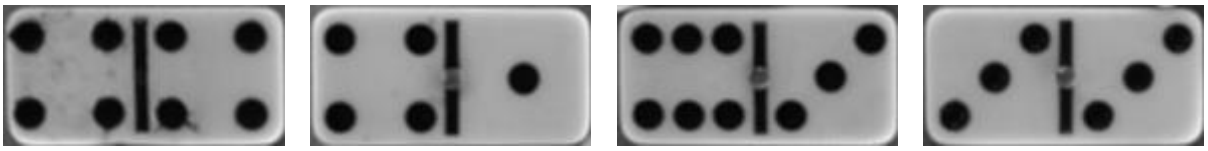
Nem elvárás, hogy minden dominó kocka megtalálásra kerüljön. Opcionálisan megvalósítható a dominó kockák pöttyei alapján a számértékük leolvasása is, ami növeli a megoldás értékét.

Komplexebb megoldás: a téglalap alakú területek egyenkénti vízszintes irányba forgatása.

Körvonal rávetítés:



Néhány beforgatott és kivágott dominó téglalap eredménykép:

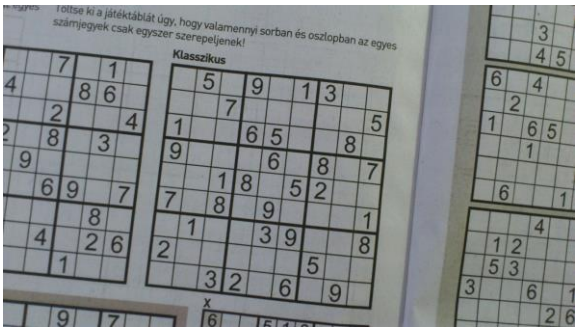


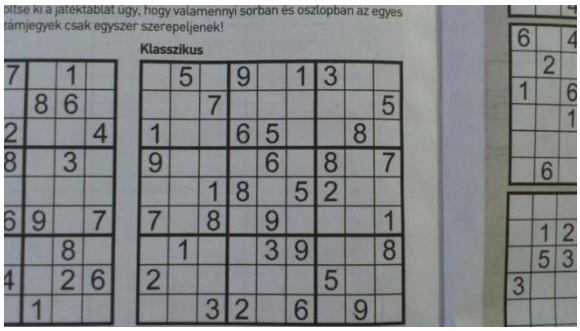
7. Fotó kiegyenesítés

Az elkészítendő program feladata a ferde helyzetben készült fotók kiegyenesítése, hasonlóan a Google Fotók funkciójához. A program a képből kinyert élek alapján automatikusan állapítson meg egy olyan forgatási szöget, amely a hosszú, egyenes szakaszokat vízszintes vagy függőleges közeli irányba forgatja. A kiindulási és az eredményképet jelenítse meg. Utólagosan interaktívan, egy egyenes berajzolásával lehessen az eredményen finomítani: a berajzolt egyenest forgassa be a program vízszintes vagy függőleges irányba, amelyikhez közelebb van.

Komplexebb megoldás: az interaktív javítás megvalósítása.

Eredeti és forgatott fotók:



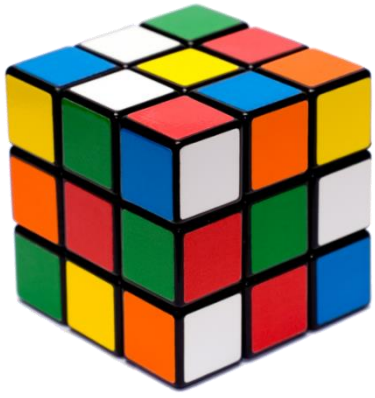



8. Rubik kocka szín detekció

A program feladata egy klasszikus, 3x3x3 méretű Rubik kockáról készült fotón egy lap megkeresése, és a lapon szereplő színek felismerése, mátrix alakban kiírása. A lap detekcióhoz automatikus esetben párhuzamos éleket keressünk, de legyen lehetőség interaktívan is megadni a lap 4 sarokpontját.

Kiíráskor a színeket (fehér, zöld, piros, kék, narancssárga, sárga) kezdőbetűjükkel rövidítsük, például:

S	Z	K
N	P	Z
Z	S	N



Komplexebb megoldás: kocka lap detektálás párhuzamos élek alapján, amennyiben a képen a terület nagyobb részét egy lap tölti ki.

9. Járművek szegmentálása autópálya képeken

Szegmentálja autópályán készült fényképeken a járműveket nappal készült felvételeken! A túl közel lévő autók együttes szegmentálását nem tekintjük hibának. A feladat megoldásakor először próbálja az aszfaltot elkülöníteni a háttértől, majd az aszfalton lévő objektumokat különítse el. Nem elvárás, de a megoldás értékét növeli, ha a járművek árnyékát el tudja különíteni (leválasztani) a járművektől.

Komplexebb megoldás: különböző fényviszonyok vizsgálata, automatikus/félautomatikus szegmentálás paramétereinekmeghatározása.


10. Hány gyertya van a képen?

Írjon egy eljárást, amely bemenő paramétere a betöltött kép. Az eljárás adja vissza a képen látható gyertyák számát. A programban több teszt képre is hívja meg az eljárást, ami a visszaadott értékeket kiírja a konzolra. Az eljárásnak több paramétere is lehet, ha az algoritmus azt megköveteli.

Komplexebb megoldás: fényviszonyok hatásainak kiküszöbölése, egymáshoz közel lévő fények szétválasztása.