Önálló szakmai projekt III.

Ki nevet a végén játékról készült videó alapján a játékmenet követése

Domán Dániel

témavezető: Pataki Péter (ARH Zrt.), Tamaga István, Deli Gábor (ARH Zrt.)

Az utolsó félév célja: A ki nevet a végén nevű játékról készült videó alapján a játék követése. Azaz célunk, hogy folyamatában megállapítsuk, hogy hol van a játéktábla, hol helyezkednek el a bábuk, hol van a dobókocka, illetve mennyi az aktuális dobás értéke, és ezt lehetőség szerint valós időben.

Programnyelv: *C++*

Képfeldolgozó szoftverkönyvtár: *OpenCV*

Fejlesztői környezet: *Qt Creator*

Detektor: *Haar Cascade*

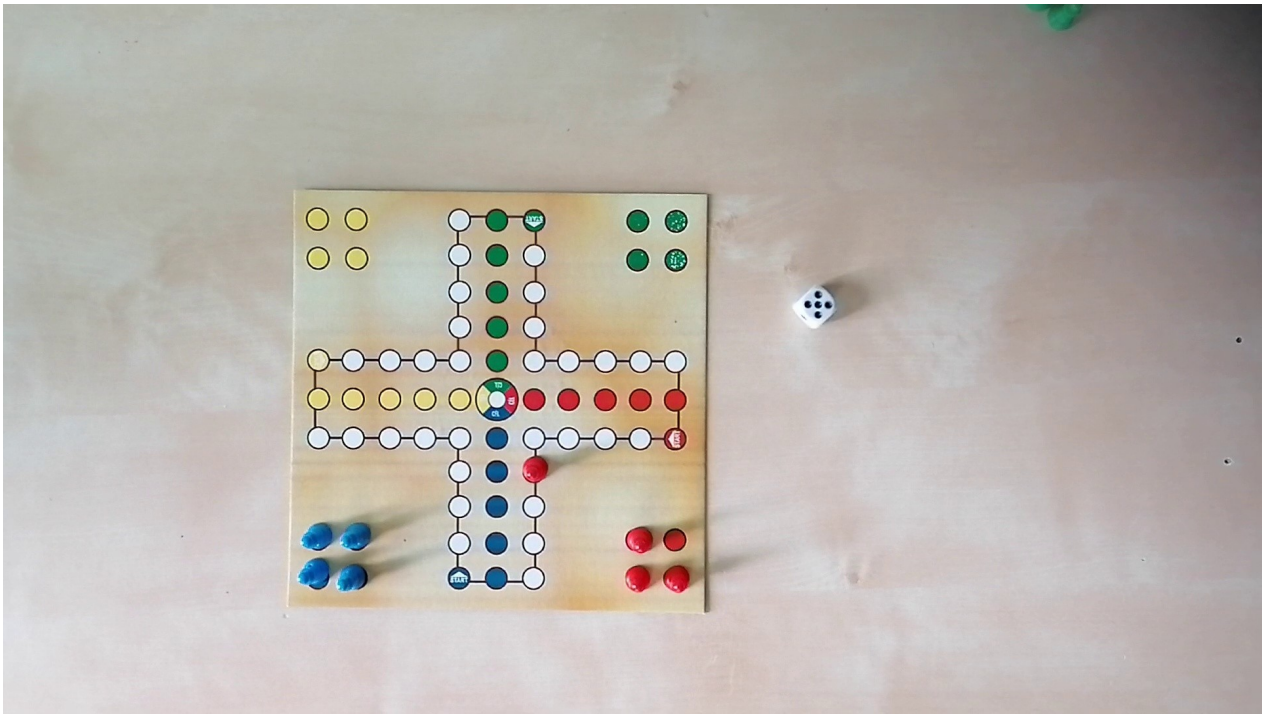
1. A tábla megtalálása

1.1. Rövid emlékeztető az első félév munkájáról

Az első félév során megkerestük a Ki nevet a végén játéktábla mezőit egy fényképen, és eltároltuk a középpontjait. A módszer lényege, hogy először a sarokmezőket keressük meg. Ha a négy sarokmezőből már legalább 3 megvan, akkor már elméletben az összes többi mező pozícióját ki tudjuk találni. Azonban mi megpróbáljuk az összes többi mezőt is a kép alapján megtalálni és azonosítani.

Összesen 72 mező van a játéktáblán, ha ezek közül legalább 40-et sikerül megtalálni, akkor a többit az ismereteink alapján kipótoljuk, ha kevesebbet, akkor pedig jelezzük, hogy nem találtuk meg a táblát. A hiányzó mezők lehetőségére már csak azért is fel kell készülni, mert a játék közben a játékosok lehetséges, hogy a kezükkel eltakarják a tábla akár jelentős részét is, azonban ekkor is fontos megtalálni a táblát a rendelkezésre álló információkból. Már három sarokmező önmagában

elég lenne, de ennél első sorban azért követelünk többet, mert előfordulhat, hogy azokat hibásan detektáljuk. Ekkor pedig jobb az, ha inkább jelezzük, hogy nincs meg a tábla.



A táblán, amin dolgozunk, a mezők mind kör alakúak, ezért a detektálásukat az OpenCV beépített körkereső algoritmusával végezzük, melynek neve *HoughCircles*

A megtalált köröket csoportosítjuk. Ehhez létrehozunk egy *tábla* struktúrát. A tagváltozói a mezők koordinátái különböző tömbökben: 4·4 „*startmező*”, 40 „*külső mező*”, 4·4 „*célegyenes*”.

A sarokmezők azonosításához feltesszük, hogy a tábla oldalai lényegében párhuzamosak a kép megfelelő széleivel. Egyelőre azt is feltesszük, hogy se a táblán, se mellette nincsenek egyéb tárgyak (pl. bábuk, dobókocka). Ezáltal például a bal alsó sarkot úgy azonosítjuk, hogy keressük azt a kört, amely középpontjának az első és második koordinátája is kisebb, az összes többinél (gyakorlatban akkor is kisebbnek tekintjük, ha csak kicsivel nagyobb). Ha ilyen kör nincs, akkor a bal alsó sarkot nem találtuk meg. Hasonló logikával keressük a többi sarkot is.

Ha már legalább három sarok megvan, akkor a tábla helyzetét ismerjük, amennyiben nem teljesen függőleges/vízszintes, akkor elforgatjuk a képet, ez a későbbiekben sokat segít. A sarokokból azt is meg tudjuk mondani, hogy milyen távol van egymástól két szomszédos mező, és innentől kezdve megadott sorrendben azonosítjuk a mezőket: ha az aktuális mező feltételezett helyének közelében van kör, akkor azonosítjuk, ha nincs, akkor kipótoljuk, és jelezzük, hogy itt egy kör nem volt meg.

1.2. Új fejlesztések a tábla megtalálásával kapcsolatban

Az eddigi munkánk során feltételeztük, hogy a táblán nincsenek „zavaró tényezők”, például bábu, dobókocka. Ez azonban nem lesz mindig így, és ha a kördetektáló talál egy kört a táblán kívül, az a sarokmezők detektálásában igen nagy gondot tud okozni. Éppen ezért, a kép mérete alapján megbecsülünk egy d korlátot a szomszédos mezők távolságára. Amennyiben a körkereső detektált egy olyan kört, hogy a középpontjának d sugarú környezetében nincs másik középpont, akkor azt a kört kitöröljük.

A későbbi vizualizációhoz szükségünk van színekre. Ötféle színt különböztetünk meg: piros, kék, zöld, sárga és egyéb. Miután csoportosítottuk a mezőket, már tudjuk, hogy melyek lesznek azonos színűek. Ha mindegyiket külön-külön szeretnénk detektálni, az nagyon sok hibázási lehetőséget adna, ezért a következőt csináljuk: megállapítjuk a startmezők színét, és az alapján színezzük ki a többi mezőt. Ha a sarokmező színe „*egyéb*”, akkor a hozzá tartozó célegyenes utolsó mezője alapján állapítjuk meg az adott színcsoportba tartozó mezők színét.

Egy mező színét úgy állapítjuk meg, hogy megnézzük a kör középpontjának az RGB értékét, és kikísérletezett határok alapján eldöntjük, hogy milyen színű. Ha az „*egyéb*” szín kerül megállapításra, akkor egy kis környezetében tovább vizsgáljuk a pontokat: 10 képponttal jobbra, balra, stb., amíg nem kapunk „egyéb”-től eltérő színt. Erre többek közt akkor lehet szükség, ha a mező fénylik. A fényviszonyok jelentős megváltozása nehézséget tud jelenteni, de valamennyi toleranciát sikerült beleépíteni a modellbe.

Ha minden mezőnek külön kellene megállapítani a színét, az jelentős hibázási lehetőséggel járna, valamint az időnként benyúló kezek miatt igen sokszor hiányozna is az adat, és a vizualizációnál se előnyös, ha „tarka” a tábla. Éppen ezért kihasználjuk, hogy a mezők már csoportosítva vannak, tehát tudjuk, hogy melyek kerülnek azonos színosztályba. Ezek közül kiválasztunk egyet, esetünkben a sarokmezőt, és annak a színére színezzük a többit. Amennyiben a sarokra „egyéb” szín jönne ki, ami akár hibás detekció, akár egy közbenyúló kéz miatt megeshet, akkor az utolsó célegyenes mező alapján állapítjuk meg a színt. Ez van a legtávolabb az azonos osztálybeli startmezőtől, ezért gyakorlatban elég ritkán fordul elő, hogy mindkettő el van takarva.

2. Dobókocka megtalálása és leolvasása

2.1. Rövid emlékeztető a második félév munkájáról

Sok társasjátéknak fontos eleme a dobókocka. Ennek a megtalálására két módszert is kifejlesztettünk.

Az első módszer az OpenCV éldetektálóját használja. Az detektált élek között keresünk olyan zárt élsorozatot, melynek (legalább) 4 csúcsa van, konvex, az átmérő négyzetének fele egyenlő az alakzat területével, az átmérő hossza bizonyos értékek közé esik, és van rajta 1,2,...6 fekete pötty.

A pöttyöket úgy azonosítjuk, hogy végigmegyünk a megtalált kockajelölteken, és csak ezeken belül keressünk pöttyöket. A pötty esetünkben egy olyan zárt élsorozatot jelent, amely konvex, van legalább 4 csúcsa, ha az átmérő nagysága d, akkor a terület $d^2\pi/4$, az átmérő hossza bizonyos értékek közé esik, és fekete, vagyis a középpontjának a színére kötöttünk ki korlátot. Akkor találtuk meg a kockát, ha valamely kocka belsejében találtunk 1...6 pöttyöt. A tesztelt képek 86%-ánál sikerült ezzel a módszerrel azonosítani a kockát, és 74%-ban volt helyes a leolvasása.

A másik módszer gépi tanulást alkalmaz. A kockát a Haar Cascade detektorral keressük. Ehhez szükség van pozitív és negatív képekre, azaz olyanokra, amiken szerepel a kocka, és olyanokra, amiken nem, de feltehetően a detektor találkozni fog vele.

A Haar Cascade a tanulás során legelőször átméretezi a keresett objektumot egészen kicsi méretűre esetünkben 16x16 pixelesre. A működés elve, hogy kidobja azt, ami biztosan nem a keresett objektum. Ennek az az értelme, hogy villámgyorsan eldobjuk a vizsgálandó kép területének 90-95%-át. A potenciális objektumok további szűrését már csak az első lépésben “reménykeltőnek” jelzett területeken kell lefuttatni ugyanúgy, mint az előbb, csak egyre finomabb szűrőkkel. Minden szinten további „nem-objektumokat” hajítunk ki. A mi esetünkben 15-20 szintet használunk.

A Haar Cascade detektor egyik legnagyobb előnye, hogy nagyon gyors, tapasztalatok szerint 4-szer – 5-ször gyorsabb, mint egy hasonló feladatra kialakított neurális háló. A hátránya pedig az, hogy egy neurális hálóhoz képest nem elég pontos.

Az algoritmus kimenete egy a tengelyekkel párhuzamos, rögzített oldalarányú határoló téglalap, mely sikeres detekció esetén minimális méretű és tartalmazza a dobókockát.

A pozitív képek gyártásához nem egyesével rögzítjük a kocka helyét a képen, hanem segítségül hívjuk az első módszert, a hibás detekciókat „kézzel” megszűrjük. A negatív képeket üres táblából, annak tükörképéből, illetve különböző darabjaiból készítjük el.

Ahhoz, hogy a tanulás eredményes legyen, nem elég 30-50 kép, szükség lenne legalább ezerre. Azonban van más megoldás is, mint ezer fotó elkészítése. A képeken apró módosításokat végrehajtva, igen sok példányt lehet generálni. Ilyen apró módosítás például a fényerő változtatása, vagy a képeken különböző lineáris transzformációk végrehajtása, ügyelve arra, hogy ne torzuljanak túlságosan.

A tesztelt képek 89%-ánál sikerült ezzel a módszerrel azonosítani a kockát, és 78%-ban volt helyes a leolvasása. Ez valamivel hatékonyabb, mint a másik módszer, a másik nagy előnye, hogy gyors, ami fontos tényező, ha videón szeretnénk kockát detektálni, ezért a videón ezt a technikát használjuk.

2.2 Fejlesztések a dobókockával kapcsolatban

Kicsit előreugorva, a dobókocka leolvasásának további fejlesztése akkor vált szükségessé, amikor már nem csak álló képeken, hanem videón kellett működésre bírni. Fontos megjegyezni, hogy a tanító- és tesztképek is állóképek voltak, nem videóból kivágott képkockák. Gyakorlatban sajnos a videónál a hatékonyság lényegesen rosszabb, és nem is a kocka azonosítása, hanem a pöttyök megtalálása az, aminek a hatékonysága jelentősen csökken.

Eredetileg abban bízván, hogy a leolvasás a videón is hatékony, az volt a terv, hogy egy adott pillanatban a dobásértéket úgy határozzuk meg, hogy az előző 10 képkockán leolvassuk az eredményt, és az adathalmaz móduszát fogjuk eredményként visszaadni. A tesztelések során azonban észrevettük, hogy sokszor kevesebb pöttyöt sikerül csak megtalálni, mint amennyi valójában volt, többet viszont soha. Éppen ezért annyi módosítást eszközöltünk, hogy nem az értékek móduszát, hanem a maximumát tekintjük eredménynek.

3. Bábukeresés

A játék követéséhez elengedhetetlen, hogy megtaláljuk a bábukat. Erre a célra is a Haar Cascade detektort használjuk. A tanító adatokat a következő módon állítjuk elő: Felhasználjuk a tábla megtalálásához használt kördetektáló algoritmust. Ennek segítségével a táblát fel tudjuk darabolni oly módon, hogy minden kör egy külön kép legyen. Ezt megtesszük nagyjából 20 különböző fotóval. Ezekből több, mint 1300 kis kép készül, ezeknek nagyjából az ötödén szerepel valamilyen figura. „Kézi erővel” kettéválasztjuk őket, így keletkeznek a pozitív és negatív képek.

Azonban az alig 200 pozitív kép kevésnek tűnik ahhoz, hogy a tanítás eredményes legyen. Ennek ellenére az első tesztek azt igazolták, hogy sikeres a tanítás már ilyen kevés tanító adattal is, ami némileg meglepő, hiszen ha például a kör alakú bábu pontosan a kör alakú mező közepén áll, akkor az még szabad szemmel se mindig nyilvánvaló, hogy ott van-e a figura vagy sem. Jobban megfigyelve viszont rájöhetünk, hogy a detektor se a figurára tanult rá, hanem az árnyékára, hiszen a fotók nagy százalékban ugyanolyan fényviszonyok mellett készültek, ugyanolyan szögből. Erre azt a megoldást találtuk ki, hogy a bábukat tükröztük, forgattuk, ezzel a tananyag méretét is növelve.

Fontos még megemlíteni, hogy a kimenet itt is egy tengelyekkel párhuzamos határoló téglalap.

Végül pedig a bábu színének megállapítására ugyanazt az algoritmust használjuk, amit a mezők kiszínezésére, ahogy azt az 1.2 szakaszban részleteztük.

4. A játék követése videófelvétel alapján

4.1 Algoritmus, javítások

A videót képkockánként olvassuk be. Ezen futtatjuk az eddig megírt, képekre kitalált függvényeket. Ezek azonban sok hibával működnek. Ezeknek a kiküszöbölésére kihasználjuk, hogy a képek egy videóból származnak és két egymást követő képkocka között csak kevés eltérés lehetséges.

Először is megállapítjuk, hogy nem szükséges minden képkockát kiértékelni. A modellünk jelenlegi formájában minden negyedik képen keresi a bábukat, és minden nyolcadik képen értékeli ki a tábla pozícióját, illetve a dobókocka helyzetét és értékét.

A bábuk kapcsán végrehajtunk egy korrekciós műveletet, mely első sorban a vizualizáción javít sokat. Mindig eltároljuk az előző detekcióból származó bábuk helyét és színét is. Ha az aktuálisak között találunk egy olyat, amely nagyon közel van valamely előzőhöz és a színük is megegyezik, akkor az aktuális bábut pontosan az előző helyére tesszük.

A táblát minden nyolcadik képen értékeljük ki újra, azonban az is előfordulhat, hogy nem sikerül megtalálni, mivel túl sok a hibás kép, vagy nincs meg legalább 3 sarokmező. Ebben az esetben az előző kiértékelésre hagyatkozunk.

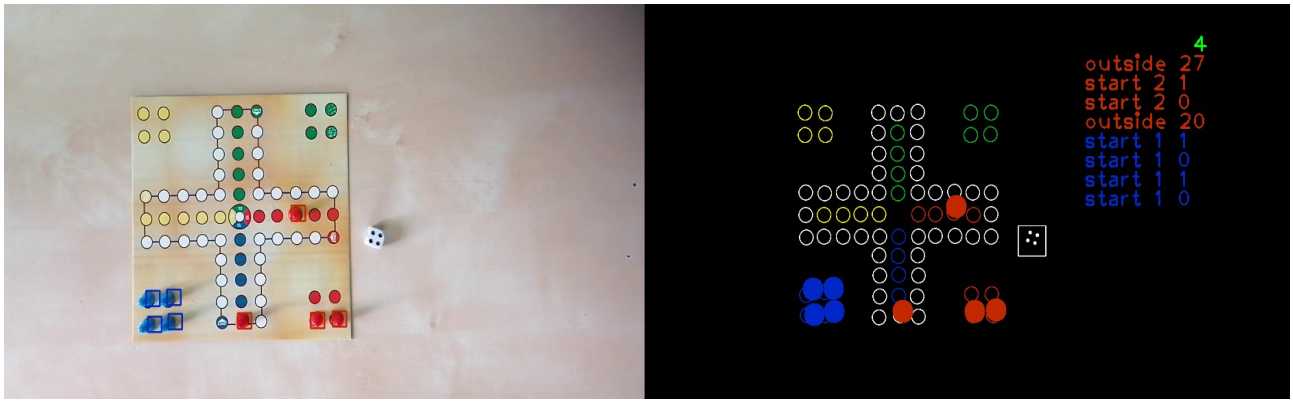
A dobókockát a táblával együtt értékeljük ki, egy aktuális pillanatban a dobásértéket pedig nem úgy határozzuk meg, hogy az éppen mit látunk a kockán, hanem az előző 10 kiértékeléskor látott dobásoknak vesszük a maximumát, mint ahogy a 2.2 szakaszban már részleteztük.

Fontos célkitűzésünk volt még, hogy a bábukról megmondjuk, hogy épp hol tartózkodnak. Ezt úgy állapítjuk meg, hogy minden bábu esetén végigmegyünk a csoportosított mezőkön, és amelyiknek a középpontjához a legközelebb volt a bábu középpontja, azt adjuk meg pozícióként.

A célmezőt eredetileg nem rögzítettük, egyrészt azért, mert jelentősen más tulajdonságokkal rendelkezik, mint a többi, például jóval nagyobb, másrészt több olyan tábla is létezik, ami ugyanolyan struktúrájú, mint a miénk, csak nincs rajta célmező, így jó eséllyel azokon is működik a modellünk. Jelezni azonban mégis kell, ha egy bábu beért, ezért lényegében egy virtuális mezőt létrehozunk (csak a középpontot), amelyet két szemközti sarok felezőpontjára teszünk. Ha egy bábu ehhez van a legközelebb, akkor ezt adjuk meg pozícióként.

4.2 Vizualizáció

A vizualizáció is képkockánként fog történni. Az eredeti videóból sem fogunk minden képkockát megjeleníteni, hanem ugyanakkor frissül, mint a bábukeresés, azaz a jelenlegi modellben minden negyedik képkockánál. Egymás mellé rakjuk az eredeti videót, és a modell által alkotott képeket. Utóbbin kirajzoljuk a táblát, a bábukat és a dobókockát is, amennyire lehetséges. A dobás eredményét kiírjuk, valamint a bábuk pozícióját szín szerint rendezve. A videóban használt táblától eltérően mi az összes külső mezőt fehérre fogjuk színezni.



5. További fejlesztési lehetőségek

Sok fejlesztési lehetőség van még a projektben, az állapotokat lehetne gazdagabbá, precízebbé tenni. Például meg lehetne állapítani, hogy éppen melyik játékos dob, és hozzákapcsolni, hogy

melyik játékos lép. Össze lehetne vetni a lépéseket a dobás értékével. Meg lehetne állapítani, hogy egy lépés szabályos volt-e, azaz annyit lépett-e a játékos, amennyit dobott.

Más irányból is meg lehet közelíteni a kérdést, ha tudjuk, hogy a játékos mennyit dobott, és feltesszük, hogy szabályosan lép, akkor lehet következtetni arra, hogy hova léphetett, és a potenciális helyeken a bábukereső paramétereket úgy állítani, hogy könnyebben elfogadja, ha bábut talál ott.

Ezekhez azonban se a bábudetektálás, se a kockaleolvasás nem elég pontos. Neurális hálókkal lehetséges, hogy el lehetne menni ebbe az irányba, csak az kérdéses, hogy egy kellően pontos háló elég gyors-e ahhoz, hogy valós időben ki tudjon értékelni.

6. Köszönetnyilvánítás

Szeretnék köszönetet nyilvánítani mindhárom témavezetőmnek: Pataki Péternek, aki a projektet megálmodta, koordinálta, és azóta munkalehetőséget is adott, Deli Gábornak, akivel a mostani félévben tudtam konzultálni, és Tamaga Istvánnak, aki az előző két félévben segített nagyon sokat különösen a számomra új eszközök megismerésében.