UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH
PRÍRODOVEDECKÁ FAKULTA

DETEKCIA SKÓRE V ŠÍPKACH POMOCOU ALGORITMOV POČÍTAČOVÉHO VIDENIA

2023 **Matej Uhrin**

UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH
PRÍRODOVEDECKÁ FAKULTA

DETEKCIA SKÓRE V ŠÍPKACH POMOCOU ALGORITMOV POČÍTAČOVÉHO VIDENIA

VEDECKÁ PRÁCA

Študijný program:	Analýza dát a umelá inteligencia
Pracovisko:	Ústav informatiky
Školiteľ:	RNDr. Miroslav Opiela, PhD.

Košice 2023

Matej Uhrin

Abstrakt

V dnešnej dobe dokáže počítač vidieť viac, ako si myslíme. Počítačové videnie výrazne prispieva k automatizácii procesov, ktoré donedávna mohli vykonávať iba ľudia. Cieľom tejto práce je naučiť počítač lokalizovať šípku zapichnutú v šípkarskom terči a priradiť takejto lokalizácii príslušné skóre podľa pravidiel šípok. Využíva sa pri tom viacero techník počítačového videnia ako napr. geometrické transformácie, prahovanie, hľadanie aktívnych kontúr, detekcia hrán, čiar a rohov, ale aj bežné matematické operácie na obrázkoch. Správne a spoľahlivé fungovanie takejto detekcie môže byť základom softvérovej aplikácie, ktorá by mohla pomôcť šípkarom v ich tréningovom procese.

Kľúčové slová

šípky, terč, počítačové videnie, perspektívna geometrická transformácia

Obsah

Úvod

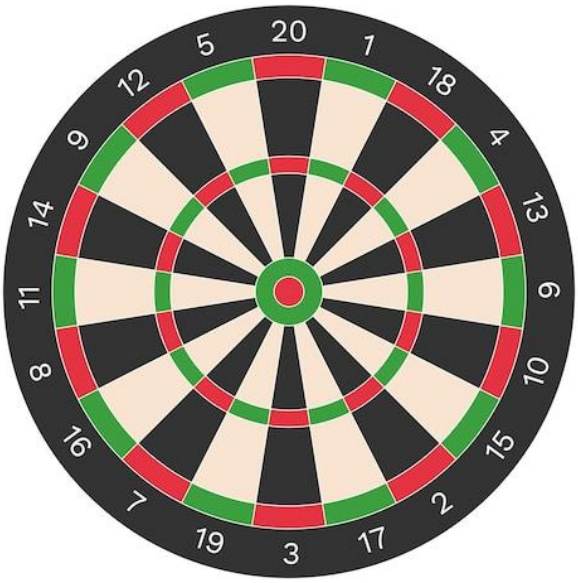
V poslednej dobe zaznamenávame nárast záujemcov o šípkarský šport. Komunita šípkarov sa za ostatné desaťročia výrazne znásobila. S príchodom pandémie v roku 2020 táto komunita prišla o možnosť hrať šípky naživo. A tak ako väčšina iných vecí, tak aj šípky sa istým spôsobom presunuli do digitálneho sveta. Preplnené turnajové haly či športové bary vystriedali vlastné izby vybavené šípkarským príslušenstvom. Hráči si postupne zvykli na iný štýl hrania zápasu. S využitím vlastného terča a vlastnej kamery sa stretávali medzi sebou, každý z pohodlia vlastného domova. I keď teraz, po uvoľnení pandemických opatrení, sa hráči opäť stretávajú v reálnom prostredí, stále využívajú online šípky ako súčasť svojho tréningového procesu. S tým, samozrejme, vznikajú rôzne vychytávky na zvýšenie komfortu pri hraní takýchto šípok. Jednou takou je aj automatická detekcia šípok na terči a s tým spojené počítanie skóre.

V tejto práci sa budeme venovať problému detekcie šípok na terči, ktorý je snímaný jednou kamerou z určitej vzdialenosti. Využijeme pritom viacero algoritmov počítačového videnia. Problém softvérovo implementujeme v programovacom jazyku Python[1] s využitím knižníc NumPy[2], Matplotlib[3] a OpenCV[4]. V niektorých prístupoch sa inšpirujeme podobným projektom[5], ktorý rieši rovnaký problém. Po teoretickej stránke nám pomáha kniha[6] s podrobným výkladom rôznych metód počítačového videnia.

1 Šípkarský terč a jeho segmenty

V tejto kapitole popíšeme vzhľad terča a princíp bodovania jednotlivých hodov.

Šípkarský terč je objekt okrúhleho tvaru, ktorý slúži na zapichovanie šípok. Na terči sa nachádza viacero oblastí, ktoré nazývame segmenty. Každý segment má svoje bodové ohodnotenie a pri zapichnutí šípky do daného segmentu je toto ohodnotenie pridelené hráčovi.



Obr. 1: Digitálna reprezentácia šípkarského terča

Popíšme segmenty terča podľa bodového ohodnotenia:

- Vonkajšie okružie čiernej farby – 0 bodov
- Vonkajšie okružie červeno-zelenej farby – dvojnásobok prislúchajúcej hodnoty
- Vnútorné okružie červeno-zelenej farby – trojnásobok prislúchajúcej hodnoty
- Zelená oblasť v strede terča – 25 bodov
- Červená oblasť v strede terča – 50 bodov
- Ostatné oblasti zodpovedajú jedennásobku prislúchajúcej hodnoty

2 Vstup a transformácia terča

V tejto kapitole popíšeme, ako vyzerá náš vstup a proces geometrickej transformácie obrázku terča.

2.1 Vstup

Na vstupe nášho problému sa nachádza obrázok šípkarského terča snímaný bežnou počítačovou kamerou (viď. *Obr. 2*). Zo zrejmých dôvodov je kamera umiestnená zo strany terča vo vzdialenosti približne 50 cm.

Obr. 2: Vstupný obrázok

2.2 Geometrická transformácia

Prvým dôležitým krokom, aby sme mohli detekovať šípku z terča na obrázku (resp. videa) je transformácia terča. Kamera snímajúca terč nevníma bodovacie okružie terča ako kruh (teda tak intuitívne ako my), ale vníma ju ako elipsu. Ak chceme detekovať skóre, musíme nájsť pozíciu v rámci terča a následne túto pozíciu „ohodnotiť“ resp. priradiť jej skóre podľa pravidiel uvedených v prvej kapitole. Prirodzený fakt je, že pozícia sa v rámci kruhu zisťuje oveľa jednoduchšie ako priamo z elipsy. To je dôvodom, prečo chceme náš terč geometricky transformovať z elipsy na kruh.

V našom prípade volíme perspektívnu transformáciu. Perspektívna transformácia je tzv. 3x3 geometrická transformácia, teda na jej použitie potrebujeme nájsť transformačnú maticu veľkosti 3x3. Tento typ transformácie mení perspektívu obrázku

a zachováva rovné čiary. Na nájdenie transformačnej matice potrebujeme vybrať 4 body vstupného obrázka, z toho žiadne 3 body nesmú byť kolineárne. Následne aplikovaním tejto matice na vstupný obrázok dostaneme výstup v podobe transformovaného obrázka.

2.2.1 Manuálna geometrická transformácia

Na začiatok vykonáme tzv. manuálnu geometrickú transformáciu. Manuálnu v tom zmysle, že štyri body potrebné na transformáciu zvolíme ručne. Aby sme sa vyhli prirodzených odchýlkam, body by mali byť navzájom vhodne rozmiestnené. Presnosť transformácie znižujú napr. 4 body blízko seba. Taktiež, je vhodné zvoliť význačné miesta terča, ktorých pozíciu v transformovanom kruhu vieme (ideálne jednoducho) vyjadriť.

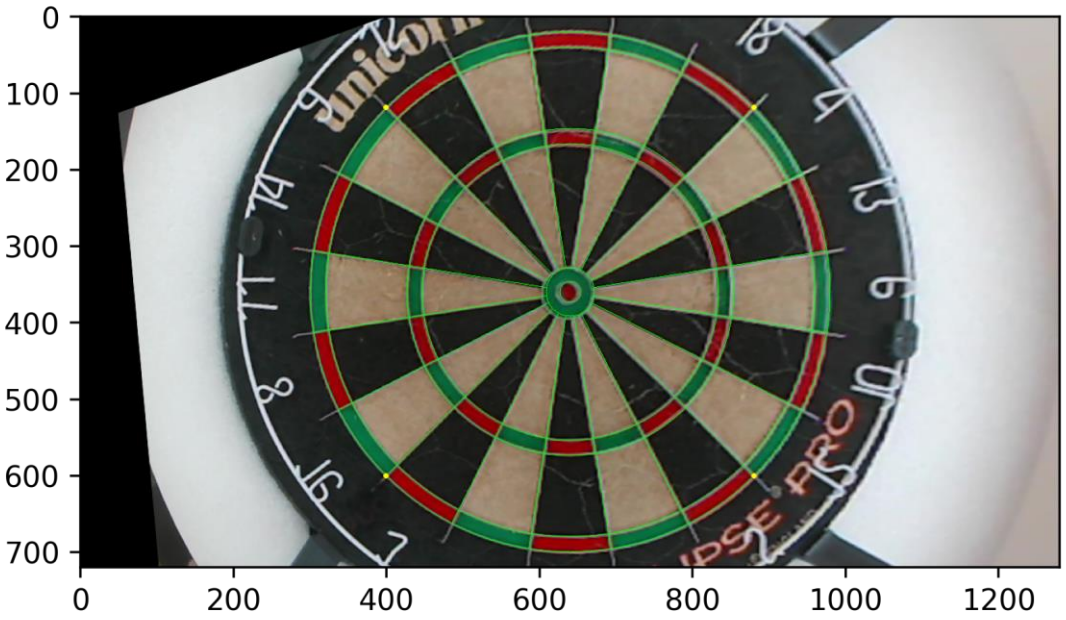
Na základe predchádzajúcich úvah prichádzame k priesečníkom elipsy oddeľujúca nulovú oblasť od tej nenulovej s hranami oddeľujúcimi jednotlivé číselné segmenty. Keďže ich je spolu 20, vyberme tú štvoricu s pravidelnými rozostupmi, napr. túto:

- priesečník oddeľujúci segmenty 18 a 4
- priesečník oddeľujúci segmenty 15 a 2
- priesečník oddeľujúci segmenty 16 a 7
- priesečník oddeľujúci segmenty 9 a 12

Na poradí záleží - uvažujeme teda aj také poradie.

Obr. 3: Pozície ručne zvolených bodov pre manuálnu transformáciu

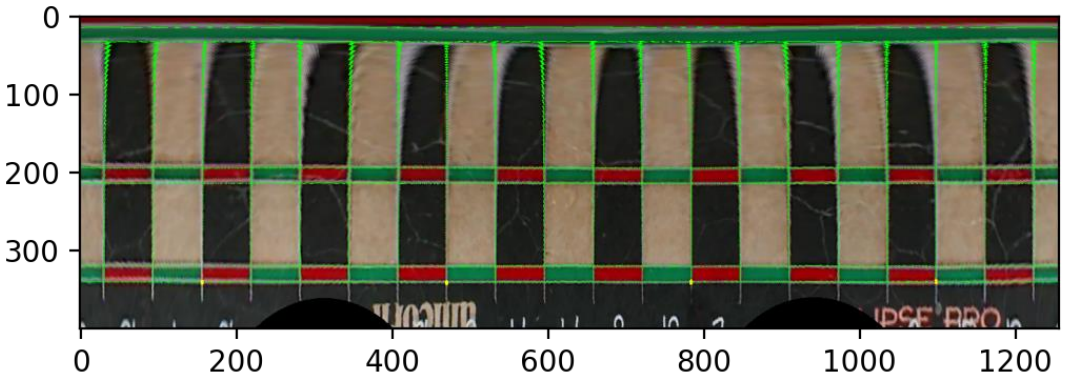
Po vykonaní perspektívnej transformácie na základe zvolených bodov dostávame transformovaný obrázok. Pre porovnanie presnosti transformácie na dokonalý kruh vykresľujeme aj „teoretické“ pozície segmentov.



Obr. 4: (Manuálna) perspektívna transformácia terča a porovnanie jej presnosti

Pri pohľade na transformovaný obrázok môžeme povedať, že takáto transformácia sa podarila. Najviac viditeľná chyba je v oblasti stredového kruhu. Spôsobuje to zrejme príliš šikmý pohľad kamery na terč resp. kombinácia výrobnej chyby terča spolu s veľkou vzdialenosťou od všetkých štyroch bodov transformácie (vysoká citlivosť).

Pre zaujímavosť – keď už máme terč dokonale okrúhly, vieme na ňom vykonať ešte jednu zaujímavú transformáciu. Ide o tzv. polárnu transformáciu, ktorá dvojicu súradníc x, y pretransformuje na dvojicu vzdialenosť a uhol od zadaného stredu, v našom prípade stredu terča. Výslednú transformáciu je možné vidieť na *Obr. 5*. Tento výstup ďalej nebudeme využívať, ale zobrazuje trochu iný pohľad na náš problém.

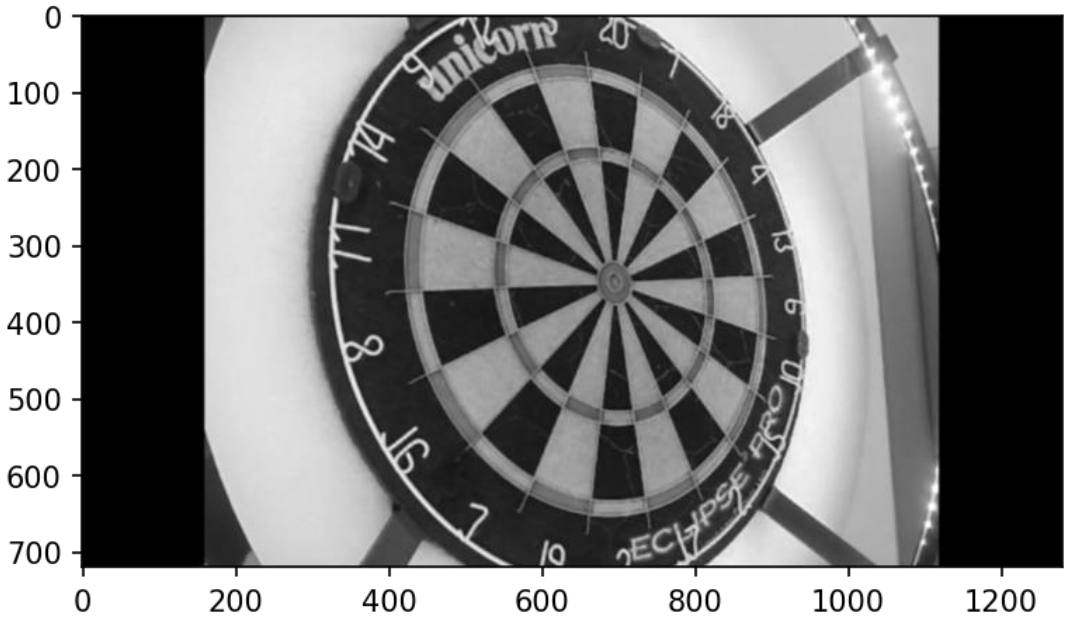


Obr. 5: Polárna transformácia terča

2.2.2 Automatická geometrická transformácia

Jedinou nevýhodou manuálnej transformácie je ručné nájdenie presných súradníc jednotlivých priesečníkov. V tejto podkapitole sa to pokúsime zmeniť a vymyslieť algoritmus na približné nájdenie takýchto bodov (využitím mnohých metód počítačového videnia). Konkrétnejšie, našim cieľom bude nájsť nejaké 4 význačné body, ktoré budeme neskôr vedieť použiť ako zdrojové body do perspektívnej transformácie, ktorú sme si ukázali vyššie. Rozdielom je, že body nájde počítač automaticky, nie my.

Na začiatku uvažujeme vstupný obrázok vo veľkosti 1280x720. Na ďalšie použitie v algoritmoch budeme využívať šedotónový vstupný obrázok a taktiež nám pomôže aj „value“ kanál HSV vstupného obrázka.



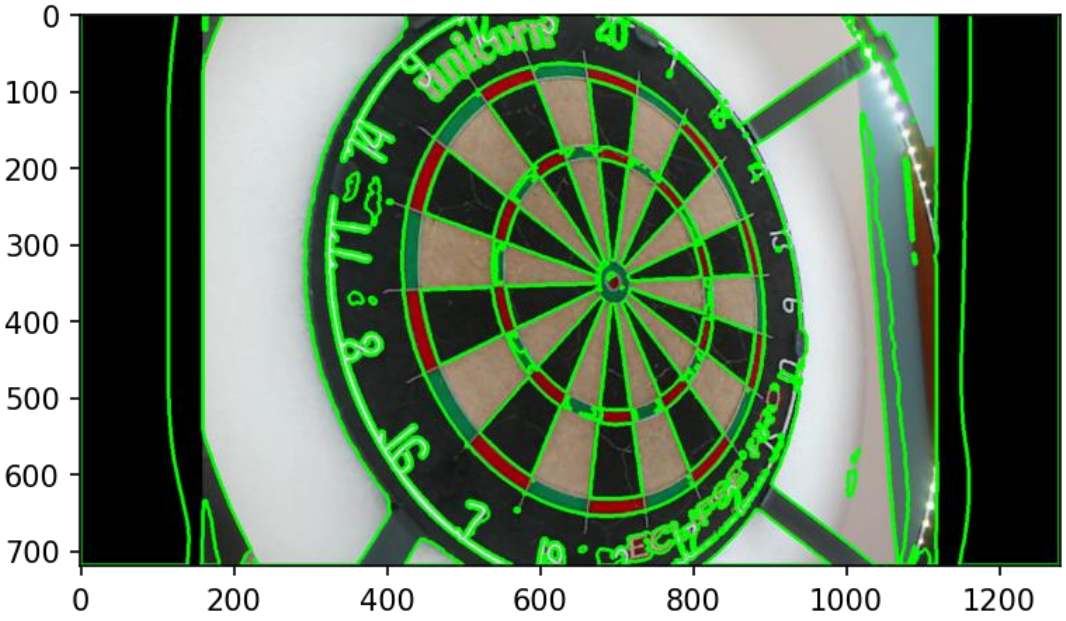
Obr. 6: Vstupný obrázok - value kanál HSV

Prvou podúlohou je v istom zmysle nájsť terč na obrázku. Presnejšie sa pokúsime lokalizovať elipsu, ktorá oddeľuje bodovaciu oblasť od nebodovacej. Popíšeme postupnosť použitých metód:

- Na (HS)V obrázku vykonáme adaptívny thresholding.
- Potom aplikujeme morfologickú operáciu otvorenia.
- Na záver spustíme algoritmus hľadania aktívnych kontúr.

Výstupom tejto postupnosti algoritmov je *Obr. 7*. Všimnime si celkom dobre viditeľnú elipsu, ktorú hľadáme.

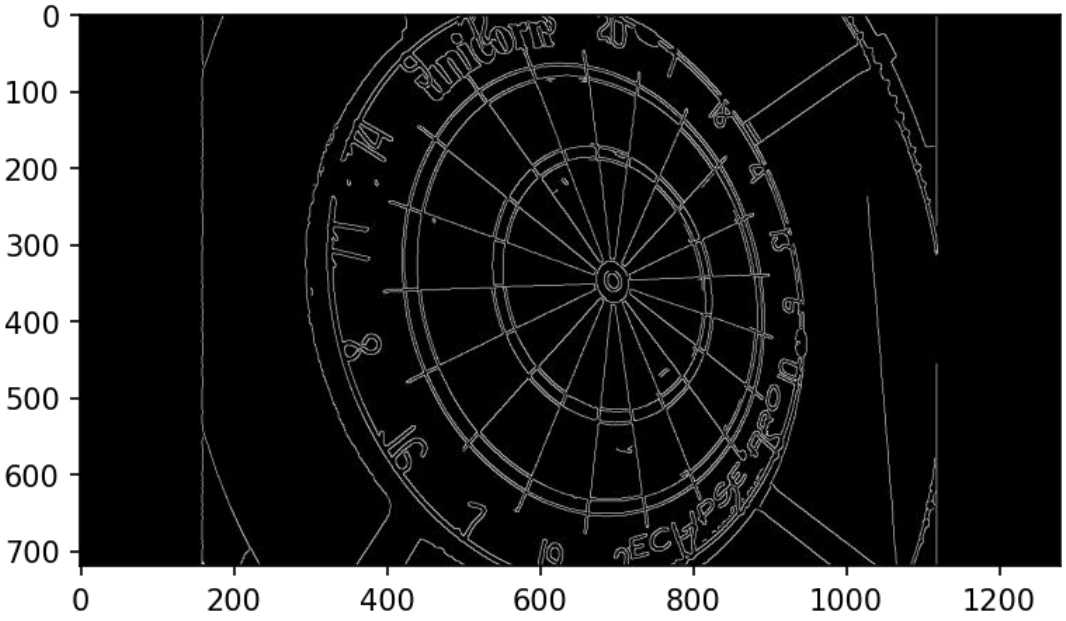
Následne sa pokúsime nájsť túto kontúru a čo najpresnejšie odhadnúť parametre tejto elipsy, aby sme s ňou neskôr mohli pracovať. Hľadanú kontúru nájdeme podľa jej veľkosti. Keďže obrázok má fixnú veľkosť a máme predpoklad, že kamera sa na terč pozerá vždy z podobného uhla a vzdialenosti. Na základe nájdených parametrov vieme elipsu do obrázka vykresliť (*Obr. 8*).



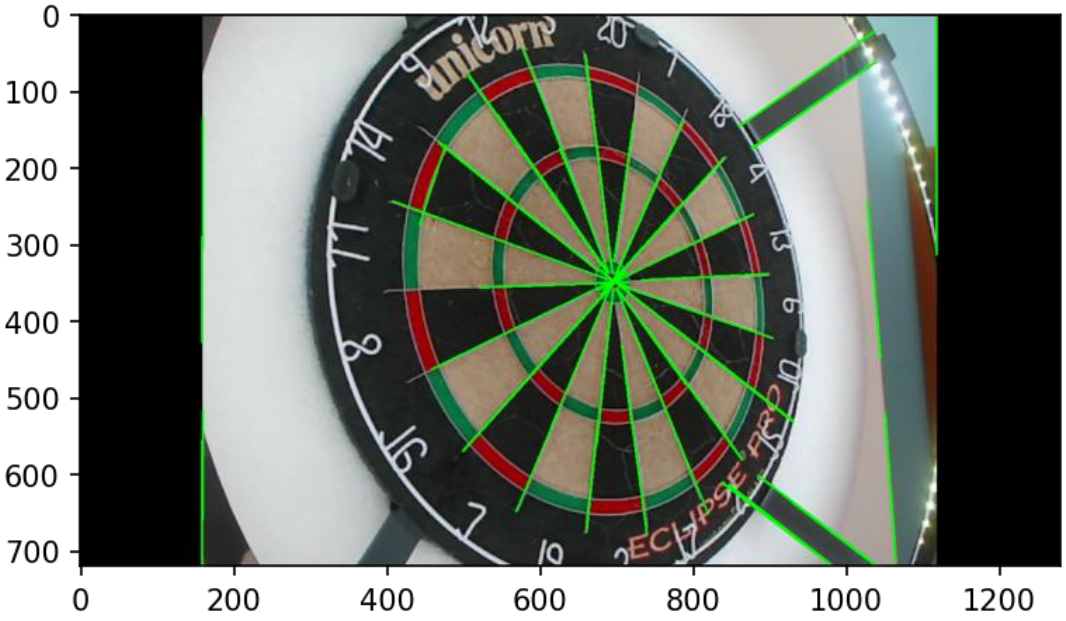
Obr. 7: Nájdené aktívne kontúry na vstupnom obrázku

Obr. 8: Elipsa (žltou farbou) oddeľujúca bodovaciu oblasť terča

Paralelnou úlohou je detekovať čiary na terči vychádzajúce zo stredu terča, ktoré oddeľujú jednotlivé číselné segmenty. Na šedotónovom obrázku najprv nájdeme hrany pomocou Cannyho detekcie hrán. Výstupom je binárny obrázok (*Obr. 9*), na ktorom môžeme hľadať spomínané čiary. Najlepšie výsledky vykazovala pravdepodobnostná verzia Houghovej transformácie, ktorú teda použijeme s vhodnými parametrami.



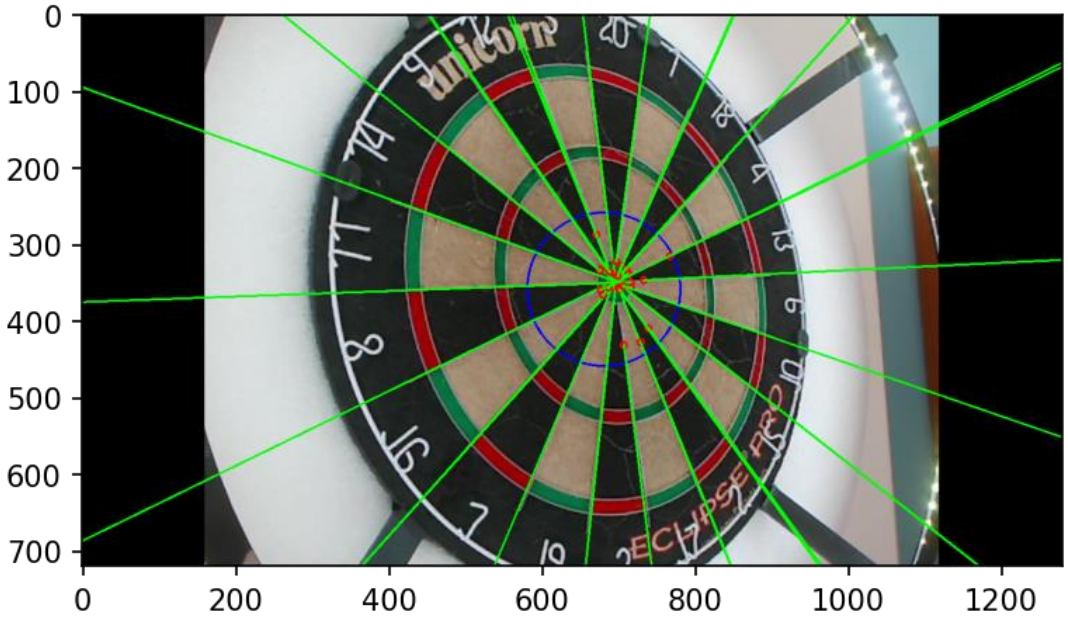
Obr. 9: Vstupný obrázok po aplikovaní Cannyho detekcie hrán



Obr. 10: Čiary detekované algoritmom Houghovej transformácie

Vidíme, že výsledné čiary identifikovalo dobre (na tomto obrázku dokonca všetkých 20 prechodov). Aby sme mohli tieto čiary použiť, musíme najprv odstrániť

nepotrebné čiary, ktoré sa vyskytli aj mimo okružia terča. Princíp riešenia tohto problému je jednoduchý - ponecháme iba čiary, v ktorých sa jeden z koncov nachádza v určitej blízkosti stredu terča. Stred terča pre počítač nie je detekovaný, ale použijeme korigovanú pozíciu stredu elipsy, ktorú sme našli vyššie. Zároveň ponechané čiary predlžíme smerom von z terča.



Obr. 11: Ponechané relevantné čiary s predĺžením smerom von

Popis Obr. 11: modrá kružnica znázorňuje oblasť, ktorá sa brala pre výber koncových bodov v blízkosti stredu terča, tieto body sú vyznačené červenou farbou. Ak čiara nemala ani jeden z koncových bodov v tejto oblasti, bola vyradená z ďalšieho procesu.

Máme teda obrázok s detekovanými čiarami. Teraz pre každú z týchto čiar nájdeme jej priesečník s elipsou detekovanou v *Obr. 8*. Nájdenie takéhoto priesečníka nie je jednoduchá vec. K dispozícii máme čiaru vyjadrenú ako dvojicu bodov a parametre elipsy, ktorá je transformovaná v priestore a neexistuje jej pekné analytické vyjadrenie. Čo ale vieme z parametrov elipsy spraviť, je transformovať ju na kruh so stredom v bode [0,0]. V transformovanom priestore nájdeme priesečník a potom jeho súradnice transformujeme inverznou transformáciou naspäť do pôvodného priestoru.

Na *Obr. 11* môžeme vidieť, že v niektorých prípadoch našlo pre jednu hranu v terči viacero čiar, ktoré sa líšia len nepatrne, avšak vytvorili viacero priesečníkov pre rovnaký hľadaný bod. Takéto zdvojené priesečníky nahradíme jedným, pričom ich spriemerujeme. Na záver teda dostaneme množinu (v ideálnom prípade 20) priesečníkov (*Obr. 12*).

Obr. 12: Lokalizované priesečníky čiar s elipsou

Z nájdených priesečníkov potrebujeme vybrať 4 reprezentatívne, ktoré použijeme ako vstup do perspektívnej transformácie. Empiricky sme zistili, že ideálne priesečníky sú na hraniciach týchto číselných segmentov: 14/9, 15/10, 16/7 a 18/4. Otázkou zostáva ako ich nájsť, ideálne v každej situácii. Priesečníky nájdeme pomocou uhla, ktorý zviera ich priamka so stredou s inou, pevne zadanou, priamkou (viď. *Obr. 13*). Empirickým prístupom sme zvolili vhodný interval na vyfiltrovanie daných štyroch priesečníkov.

Obr. 13: Ohodnotenie priesečníkov podľa zvieraného uhla

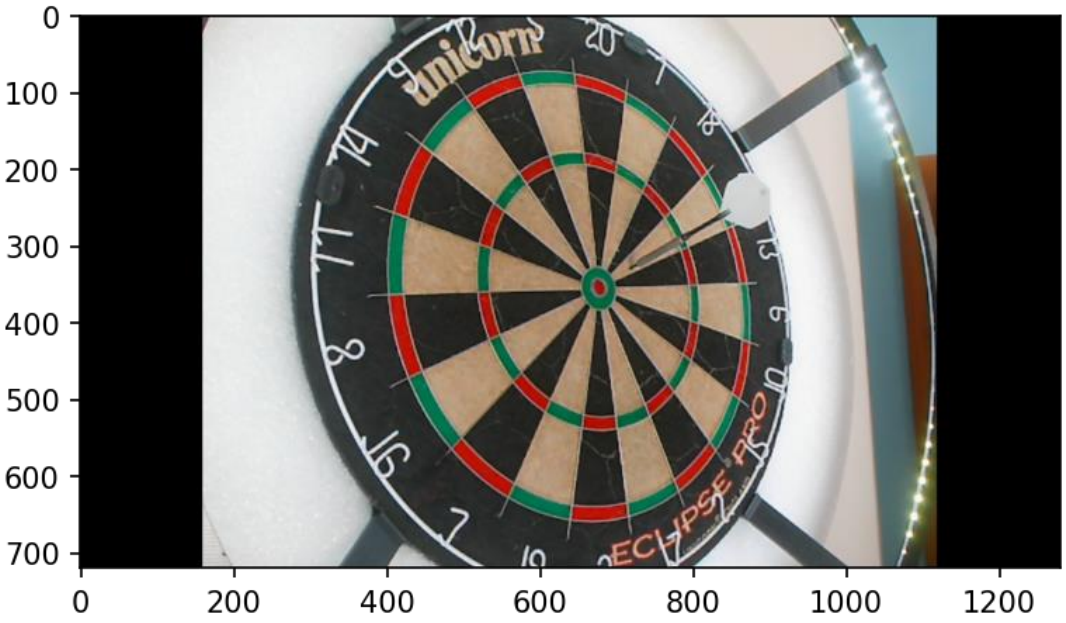
Bez toho, aby sme si manuálne vybrali priesečníky (tak, ako to bolo pri manuálnej transformácii), máme 4 body potrebné na perspektívnu transformáciu. Po transformovaní dostaneme *Obr. 15*. Vzhľadom na konkrétny obrázok, sú zmeny oproti manuálnej transformácii minimálne, teda môžeme povedať že automatická transformácia sa vydarila.

Obr. 14: Automaticky vybrané priesečníky na transformáciu

Obr. 15: Výstup automatickej transformácie s porovnaním jej presnosti

3 Lokalizácia šípky v terči

V tejto kapitole bude našou úlohou nájsť pozíciu šípky v terči na obrázku. Na začiatku sa pozrieme na nájdenie hrotu šípky, neskôr sa pokúsime pre lokalizovaný bod nájsť príslušné skóre. Pre jednoduchosť sa v tejto práci obmedzíme iba na jednu šípku v danom momente. V skutočnosti sa hádžu tri šípky na jedno kolo – tento prípad je náročnejší na rozpoznávanie, najmä ak sú šípky blízko seba.

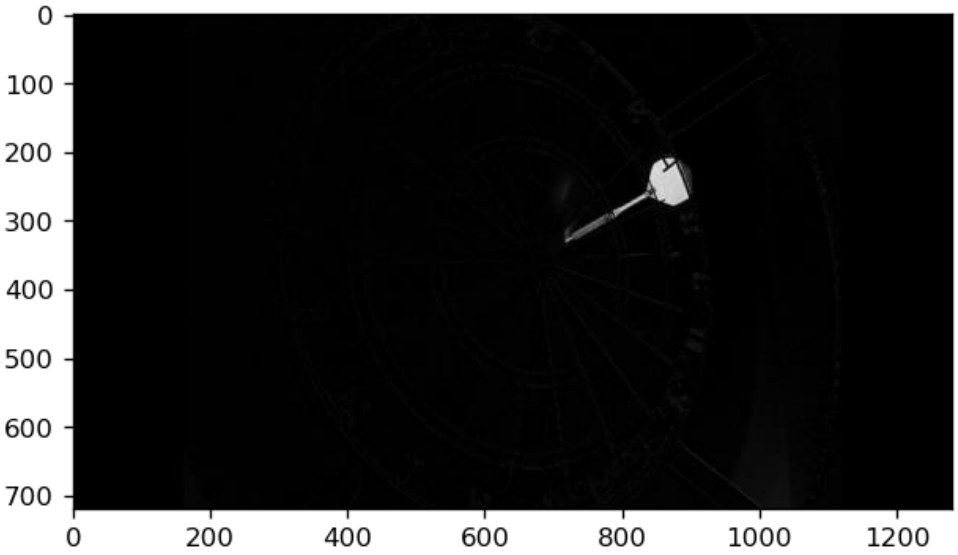


Obr. 16: Príklad šípky zapichnutej v terči

3.1 Detekcia hrotu

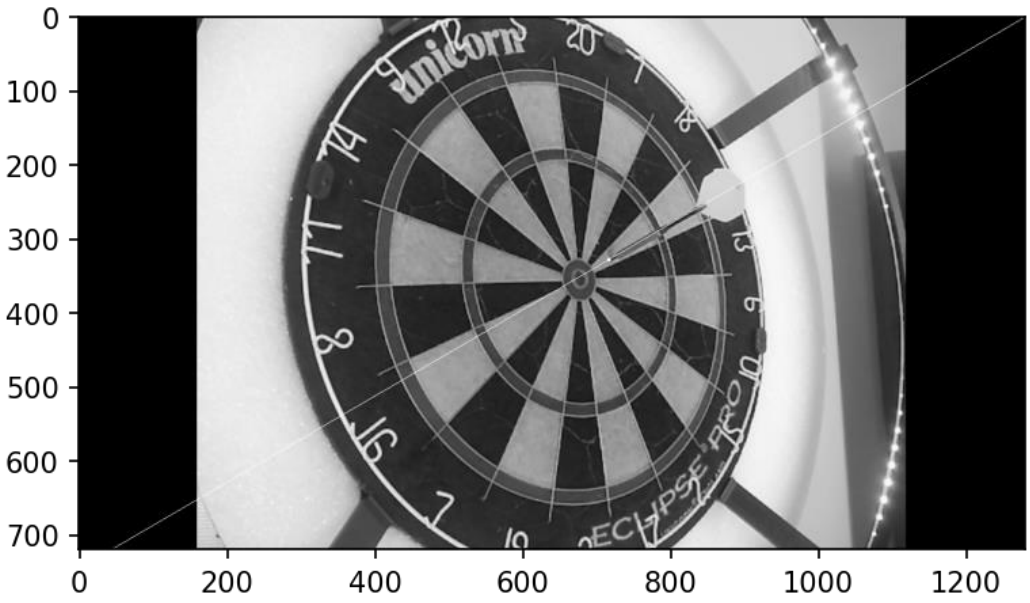
Ak by sme sa pozreli na zopár obrázkov, na ktorých je viditeľná šípka zapichnutá v terči, uvedomili by sme si viacero skutočností. Žiaľ, terč obsahuje farebné segmenty, na ktorých je terč vidno rôzne – azda najhorší je čierny podklad, na ktorom hrot nevidíme, v prípade svetlých segmentov je situácia pomerne priaznivá. Ďalej je šípka v terči zapichnutá pod podobným uhlom a nie je veľmi nutné uvažovať úplne iný sklon. Z toho v našom prípade vyplýva, že hrot šípky sa nachádza vždy v ľavej dolnej časti výrezu šípky na obrázku. V tejto práci využijeme tento fakt a pokúsime sa zostrojiť prístup založený práve na hľadaní ľavého dolného rohu zmeny obrázka oproti základnému obrázku (pozadiu).

Rozoberme si postup hľadania hrotu šípky na *Obr. 16*. Ak máme k dispozícii obrázok pozadia (ten môže byť vyhotovený pár sekúnd pred okamihom zapichnutia), potom na týchto dvoch obrázkoch vykonáme operáciu rozdielu.



Obr. 17: Operácia rozdielu pozadia a *Obr. 16*

Obrázok rozdielu jemne rozostríme, aby sme odstránili prípadný svetelný šum. Potom sa pokúsime nájsť všetky rohy v takomto obrázku. Využili sme algoritmus *Good features to track*. Následne týmito rohmi preložíme priamku a pre každý nájdený roh v blízkosti preloženej priamky urobíme projekciu na danú priamku. Na záver, najľavejšiu projekciu zvolíme ako bod hrotu šípky.

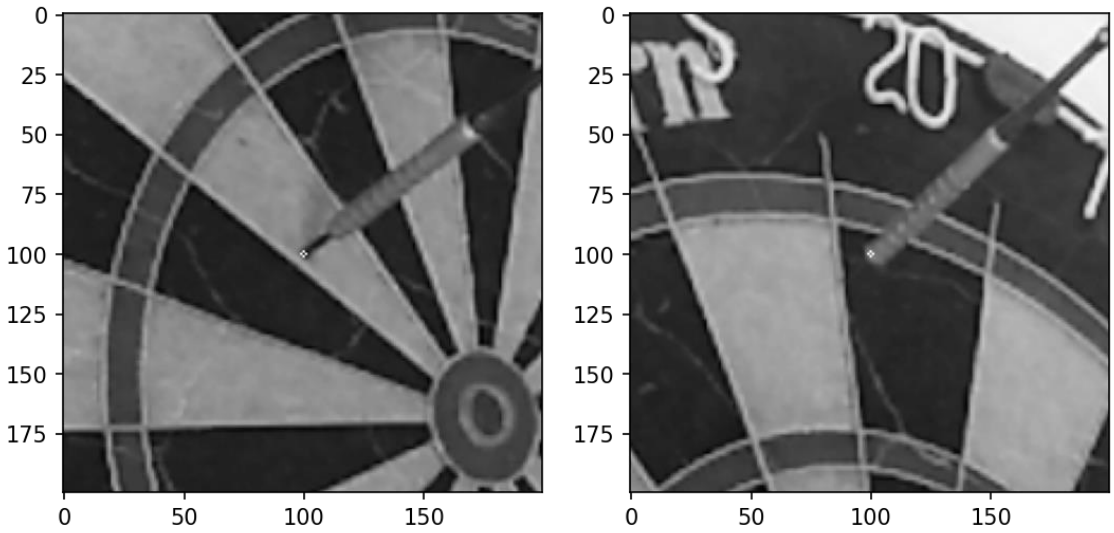


Obr. 18: Priamka preložená rohovými bodmi a lokalizovaný hrot šípky

Vo všeobecnosti sú výsledky hľadania hrotu na obrázku podobné ako sme čakali. Príklady vieme rozlíšiť do troch kategórií:

- Šípka zapichnutá v svetlom segmente – lokalizácia hrotu takmer presná.
- Šípka zapichnutá v „sivom“ segmente – lokalizácia menej presná, ale viacmenej vyhovujúca.
- Šípka zapichnutá v čiernom segmente – lokalizácia nepresná, často sa lokalizuje iba telo šípky nie hrot. Avšak treba povedať, že pri týchto segmentoch má aj človek problém lokalizovať hrot a odhaduje ho iba podľa vzhľadu šípky.

Riešenie na tento problém sa doposiaľ nepodarilo nájsť, a tak v procese priraďovania skóre budeme pracovať s týmto algoritmom.



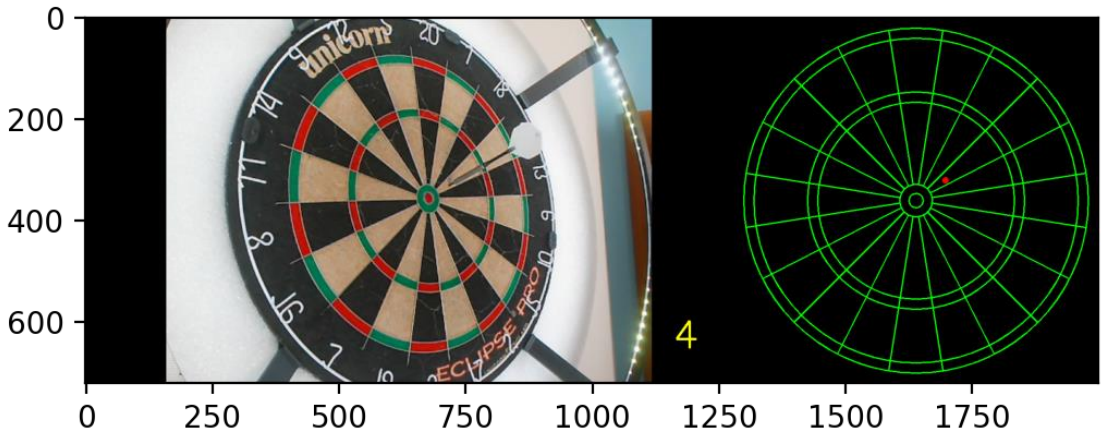
Obr. 19: Presná (vľavo) vs. menej presná (vpravo) lokalizácia hrotu

3.2 Identifikácia skóre

V tejto časti vykonáme pomerne jednoduchú úlohu – na základe pozície hrotu šípky identifikujeme skóre, ktoré bolo zaznamenané hodením konkrétnej šípky. Využijeme takmer všetko, čo sme popísali v tejto práci. Postup je nasledovný:

- Algoritmom v kapitole 3.1 nájdeme pozíciu šípky (jej hrot).
- Tento bod perspektívne transformujeme do priestoru s dokonale kruhovým terčom. Transformačnú maticu dostaneme automatickým prístupom.
- Z transformovanej pozície v rámci kruhu získame zasiahnuté skóre.

Ukážeme si 2 vzorové výstupy:



Obr. 20: Lokalizácia hrotu šípky (červenou farbou) a identifikácia skóre (žltou farbou)

Obr. 21: Lokalizácia hrotu šípky (červenou farbou) a identifikácia skóre (žltou farbou)

Záver

V rámci tejto práce sa nám podarilo:

- pripraviť manuálnu perspektívnu transformáciu terča (z elipsy na kruh),
- popasovať sa tiež s automatickou perspektívnou transformáciou, ktorá je v 90 % prípadoch dostatočne úspešná,
- lokalizovať pozíciu hrotu jedinej šípky v terči,
- prepojiť pozíciu hrotu s perspektívnou transformáciou za účelom získania skóre.

Povenujme sa teraz úspešnosti vyhodnocovania takýchto pozícii. Počas testovania algoritmu sme vytvorili viacero vzorových výstupov. Medzi nimi sa nachádzalo:

- pomerne veľa správnych lokalizácií,
- niekoľko tesne zlých lokalizácií, kde rozhodovali milimetre – pri týchto prípadoch bola tiež dôležitá presnosť transformácie či hľadania hrotu šípky,
- a tiež aj niekoľko „failov“ spôsobených najmä nerobustnosťou algoritmu hľadania šípky.

Práve hľadanie hrotu šípky je najväčšia výzva tohto problému. Či už to je konkrétne hľadanie hrotu na rôznych farbách segmentu alebo aj porovnávanie obrázku pozadia s obrázkom šípky – pri väčšom šume algoritmus zlyháva.

Stále je to však jednoduchý prípad – jedna šípka v terči. Už v prípade dvoch šípok často nastáva prekrytie hrotov a situácia sa výrazne komplikuje, vtedy veľakrát nepomôže ani ľudské oko. Riešením je dvojica (alebo aj trojica) kamier snímajúca terč z rôznych uhlov.

Subjektívne, považujem výsledok tohto snaženia za pomerne dobrý a určite splnil moje minimálne očakávania. V prípade vylepšovania existuje viacero možných vetiev, ktoré by stáli za otestovanie. Určite ma zaujal aj prístup použitia viacerých kamier (ktorý sa využíva v komerčných produktoch), kde sa otvára viacero ďalších možností riešenia problému identifikácie skóre z terča.

Zoznam použitej literatúry

[1] VAN ROSSUM, G., DRAKE, F.L. 2009. *Python 3 Reference Manual*, Scotts Valley, CA: CreateSpace.

[2] HARRIS, C.R. et al. 2020. *Array programming with NumPy*. Nature, 585, s. 357–362.

[3] HUNTER, J.D. 2007. *Matplotlib: A 2D graphics environment*. Computing in science & engineering, 9(3), s. 90–95.

[4] BRADSKI, G. 2000. *The OpenCV Library*. Dr. Dobb‘s Journal of Software Tools.

[5] HOETTINGER, H. 2017. *opencv-steel-darts.* https://github.com/hanneshoettinger/opencv-steel-darts.

[6] SZELISKI, R. 2022. *Computer Vision: Algorithms and Applications*. 2nd ed. 2022 edition. Cham: Springer.