

Zoltán Tomori, Matej Nikorovič

Počítačové videnie

Prednášky,
návody na cvičenia,
demonštračné príbehy

Počítačové videnie - prednášky

1 Počítačové videnie – úvod

Zrak hrá nezastupiteľnú úlohu pri poznávaní a v každodennom živote človeka. Je preto logické, že s rozvojom výpočtovej techniky silnela aj snaha po nahradení ľudského videnia strojovým a to hlavne pri rutinnej činnosti akou je kontrola kvality a robotika, prácou v nebezpečnom alebo neobvyklom prostredí (kozmický výskum, vojenská technika, diaľkový prieskum Zeme) alebo napr. v oblasti medicínskej diagnostiky (tomografia, ultrazvuk, mikroskopia a pod).

Ako sa výpočtová technika stáva dostupnejšou, aj metódy číslicového spracovania obrazu nachádzajú čoraz širšie uplatnenie nielen na pracoviskách, ale aj v domácnostach. Explozívne rozšírenie smartfónov, digitálnych fotoaparátov a kamier spolu s dostupnosťou internetu stimulujú nové aplikácie v tejto oblasti a absencia znalosti o podstate a možnostiach digitálneho obrazu môžu viesť k jednej z foriem počítačovej negramotnosti.

V poslednej dobe sa intenzívne presadzuje analýza a vizualizácia v 3D, čím sa zoznam populárnych aplikácií rozšíril o 3D televíziu, hračie konzoly založené na snímaní polohy hráča kamerou, viac-pohľadové (multiview) snímanie ktoré umožní vytvárať 3D modely a kombinovať ich s reálnymi obrazmi (virtuálna resp. pridaná realita) a pod.

1.1 Motivácia učebných textov, zdroje a ich organizácia

Učebné texty sú určené študentom a pracovníkom vedeckovýskumných organizácií u ktorých sa predpokladá viac než len bežná „gramotnosť“ v oblasti číslicového spracovania obrazu. Fyzika, umelá inteligencia, robotika, spotrebna elektronika atď. sú odbormi poskytujúcimi najvýraznejšie stimuly posúvajúce číslicové spracovanie obrazu na kvalitatívne vyššiu úroveň - počítačové videnie, ako je to popísané v nasledujúcej podkapitole.

Zdrojov z ktorých sa vychádzalo pri koncipovaní týchto učebných textov bolo niekoľko. Dominantným zdrojom je medzinárodne uznávaná učebnica [1]. Okrem spomínanej učebnice je niektoré pasáže v zjednodušenej, ale obsahovo takmer identickej podobe nájsť aj v staršej literatúre od spomínaných autorov [2]. Ďalším zdrojom informácií je kniha [3], ktorá je k dispozícii zdarma v elektronickej podobe [4]. Filozofia bezplatného poskytovania študijnej literatúry ako aj OpenSource softvéru bola mohutným impulzom pre rozvoj tohto vedného odboru.

Doplňkovou literatúrou, z ktorých sme vychádzali pri koncipovaní cvičení je [5] pre prácu s použitím jazyka MatLab ale hlavne [6] resp. [7] s popisom využitím prostredia OpenCV. Práve posledné dva zdroje umožňujú prístup k sofistikovaným metódam z oblasti spracovania obrazu v podobe knižničných funkcií. Aj keď ľahká dostupnosť takéhoto zdroja zvádzia k uniformným riešeniam často bez pochopenia problému, na druhej strane to je nástroj, ktorý umožňuje rýchly vývoj vlastných originálnych aplikácií.

1.2 Organizácia a forma dokumentu

Vychádzame z názoru, že by bolo mrhaním síl meniť koncepciu celosvetovo osvedčených učebníck vďaka malichernej snahe po originalite a preto sa menil iba obsah jednotlivých častí podľa špecifických požiadaviek poslucháčov.

Matematický aparát sa snažíme používať iba v miere nevyhnutnej pre exaktné pochopenie problému (podobne ako napr. [8] ktorej autor je vzdelaním matematik).

Naopak, za veľmi dôležité pokladáme možnosť overiť si činnosť algoritmu pomocou hotových knižníck. Preto je na konci kapitol vždy uvedený aj odkaz na príslušnú kapitolu z návodov na cvičenia s využitím OpenCV [9], ktoré sú integrovanou súčasťou tejto elektronickej knihy.

Okrem toho sú pri niektorých kapitolách uvedené aj odkazy na jednotlivé „príbehy“, ktoré majú čitateľovi zjednodušíť prvý kontakt s problematikou a tiež poukázať na jej praktickú využiteľnosť. Všetky príbehy sa odohrávajú v prostredí vojenského útvaru počas základnej vojenskej služby v predposlednej dekáde minulého storočia. Dôvodom výberu tohto prostredia je hlavne fakt, že zahrnuje mnohé analógie z oblasti algoritmov (hierarchická štruktúra, interpretácia a vykonávanie príkazov, paralelizmus atď.).

1.3 Počítačové videnie a vzťah k iným vedným odborom

Ked' vychádzame zo všeobecnej definície počítačového videnia ako „vednej disciplíny, ktorá sa snaží počítačovými prostriedkami napodobniť ľudské videnie“, tak je zrejmá jej prirodzená väzba na príbuzné vedné disciplíny. Základom je spracovanie 2D obrazu (Image Processing). Túto etapu väčšinou predstavujú algoritmy na nízkej úrovni abstrakcie (low-level), kym vyššiu úroveň (high-level) predstavujú algoritmy, ktoré extrahujú z nižzej úrovne znalosti, dávajú ich do súvislostí a snažia sa im porozumieť (Image understanding, machine learning). To by nebolo možné bez znalosti fyzikálnej podstaty senzorov, optiky, okolitého prostredia a tiež matematických a štatistických metód na ich hodnotenie. Pritom sa samozrejme vychádza zo znalostí získaných pri spracovaní 1-rozmerných dát (Signal Processing). Veľmi dôležitou vednou disciplínou sú aj Neurovedy ktoré získavajú poznatky o špecifických biologického vizuálneho systému a integrujú ich v podobe modelov do systému.

1.3.1 Prečo je počítačové videnie tak zložité?

Dôvodov je veľké množstvo, medzi najdôležitejšie patria:

1. **Strata informácie pri premietaní z 3D do 2D.** Človek totiž vníma okolity svet ako 3-rozmerný, pričom väčšina senzorov a snímacích zariadení je schopná registrovať iba 2 rozmery. Táto projekcia 3-rozmerného objektu na plochu nedokáže odlišiť či sa jedná o veľký vzdialený objekt alebo blízky menší objekt rovnakého tvaru. Táto problematika je komplexne popísaná v [10].
2. **Interpretácia v ľudskom mozgu nie je vždy správna.** Pretože aj sietnica ľudského mozga je v princípe 2-rozmerná, je to práve ľudský mozog, ktorý vytvára 3D predstavu na základe doterajších znalostí (modelu). Tento proces sa snaží napodobniť aj umelý systém, pričom na interpretáciu potrebuje vhodný matematický model nielen pre jednotlivé videné objekty, ale aj pre vzájomné vzťahy medzi nimi.
3. **Šum,** ktorý je prítomný v každom reálnom obraze komplikuje vytváranie predstavy na základe modelu. Preto namiesto deterministických modelov je potrebné používať

- stochastické modely, ktoré predpokladajú určitý stupeň neurčitosti a dokážu túto neurčitosť aj zahrnúť do matematického modelu.
4. **Obrovský tok dát** generovaný snímačmi obrazov, ktorý by sa mal spracovať v reálnom čase, kladie extrémne nároky na výpočtovú kapacitu a táto exponenciálne rastie s každým ďalším stupňom voľnosti (napr. pri optimalizácii).
 5. **Nejednoznačná interpretácia jasu** jednotlivých častí objektu. Bez poznania fyzikálnej povahy sledovaného objektu nie sme schopní jednoznačne určiť či napr. výrazne svetlé škvarky na obraze sú dôsledkom extrémne svetlých častí objektu alebo iba dôsledkom odrazu zdroja svetla v lesklej (ale nie svetlej) časti povrchu.
 6. **Rozpor medzi lokálnym pohľadom cez kameru a potrebou globálneho pohľadu** na scénu zobrazuje Obrázok 1 z ktorého je jasné, že séria obmedzených pohľadov neumožní získať dostatočné znalosti pozorovanej scény.



Obrázok 1. Znázormenie problému straty globálnej informácie pri videní pomocou viacerých lokálnych pohľadov. Zdroj [11]

1.4 Etapy spracovania obrazu vo väzbe na počítačové videnie.

Jednotlivé etapy spracovania obrazu môžeme vyjadriť nasledovnou schémou:

1. Snímanie obrazu
2. Predspracovanie
3. Segmentácia

4. Detekcia príznakov
5. Klasifikácia a rozpoznávanie objektov
6. Porozumenie scény
7. Rekonštrukcia a 3D zobrazenie

Prvé tri etapy predstavujú **nízkoúrovňové spracovanie obrazu** za ktorými nasledujú etapy predstavujúce **vyššiu úroveň spracovania**. Tu sa potom algoritmus na základe extrahovaných znalostí snaží riešiť základné problémy PV akými sú klasifikácia, porozumenie scény a 3D rekonštrukcia. Prechod na vyššiu úroveň znamená zároveň aj výraznú redukciu množstva spracovávaných dát, ktoré ale používajú zložitejší matematický formalizmus, rôzne modely a interpretácie dát, čiže prostriedky ktorých používanie je obvyklé vo vedných disciplínach z oblasti umelej inteligencie.

Samozrejme, uvedené rozdelenie nie je jediné možné, avšak predstavuje tradičné delenie kapitol vo viacerých významných monografiách a preto aj v týchto učebných textoch budú nasledovné kapitoly korešpondovať s takýmto delením.

2 Obraz, jeho snímanie a reprezentácia v počítači

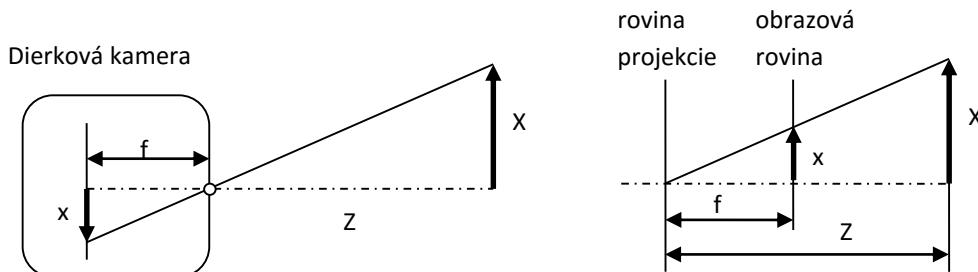
2.1 Čo je to 2D obraz, ako ho získať

Základným obmedzením pri vnímaní okolitého 3D sveta je skutočnosť, že naše zrakové orgány sú schopné vnímať obraz iba v 2D, ktorý takto predstavuje zjednodušený model zložitého 3D okolia.

2.2 Model dierkovej kamery.

Dvojrozmerný obraz vytvorený pomocou tohto najjednoduchšieho modelu je analógiou toho, čo vzniká na sietniči ľudského oka a predstavuje dvojrozmernú funkciu, ktorej hodnoty zodpovedajú intenzite jasu na určitom mieste obrazu, teda množstvu svetla naň dopadajúceho. Takýto obraz nazývame **intenzitný**, alebo **šedotónový**, keďže neuvažujeme farebnú citlivosť receptorov. Je výsledkom **perspektívneho zobrazenia** (ktoré znázorňuje Obrázok 2) a je výsledkom snímania obrazu komorou s dierkou. Obraz sa pritom premieta do obrazovej roviny vzdialenej od dierky o ohniskovú vzdialenosť f . To predstavuje najjednoduchšiu kameru (pinhole camera), ktorá je založená na reálnom fyzikálnom zariadení schopném získať obraz.

Príbeh „*Fotokomora (model dierkovej kamery)*“ demonštruje možnosť jej využitia.



Obrázok 2. Model dierkovej kamery. Vľavo - fyzický model predstavuje uzavretú krabiciu s dierkou, v ktorej je v ohniskovej vzdialosti od dierky vložený film. Vpravo - zjednodušený geometrický model pri ktorom sa obrazová rovina zrkadlovo presunie pred rovinu projekcie (rovina s dierkou).

Z podobnosti trojuholníkov vyplýva:

$$x = \frac{Xf}{Z} \quad (1)$$

Tento výraz predstavuje najjednoduchší model dierkovej kamery, ktorý je základom zložitejších modelov slúžiacich na výpočty napr. pri 3D projekcii, ako to bude prezentované neskôr.

Pre veľmi veľkú ohniskovú vzdialenosť (konvergujúcu do nekonečna) nazývame projekciu lineárne paralelnou alebo aj **ortografickou projekciou**.

Dôležitým pojmom v procese zberu dát je obrazová funkcia $f(i,j)$, kde (i,j) sú súradnice v rovine a hodnota funkcie predstavuje výstup snímacích senzorov (zvyčajne **jas**). Pokiaľ funkcia obsahuje aj

ďalší parameter, jedná sa o sekvenciu obrazov. Napr. $f(i,j,t)$ predstavuje časovú sekvenciu obrazov (video). V prípade, že je tretím parametrom súradnica z , jedná sa o priestorovú sekvenciu – napr. $f(i,j,z)$ predstavuje sériové (po sebe nasledujúce) rezy telesom. V závislosti na type snímacieho zariadenia a charaktere úlohy môže byť hodnotou obrazovej funkcie aj iná fyzikálna veličina (teplota, tlak a pod).

Obrazová funkcia môže byť:

- spojité (má spojity definičný obor aj obor hodnôt),
- diskrétna (má diskrétny definičný obor a spojity obor hodnôt),
- digitálna (má diskrétny definičný obor aj obor hodnôt)

Hodnoty dvojrozmernej diskrétnej obrazovej funkcie majú podobu **pixelov**, v prípade trojrozmernej funkcie sú to objemové elementy – **voxely**.

Z hľadiska dynamiky delíme obrazovú funkciu na:

- statickú - keď zobrazuje statickú scénu,
- dynamickú - keď zobrazuje dynamický dej (videosekvencie).

Každému pixelu resp. voxelu môže byť priradená jedna hodnota (napr. jas) alebo to môže byť **vektor** hodnôt (farebné RGB resp. multispektrálne obrazy). Digitálna obrazová funkcia je vlastne **maticou**, ktorej súradnice i hodnoty sú celočíselné a konečné, kde vo všeobecnosti,

$i=0,1 \dots M-1$; M - počet vzoriek v smere osi x ,

$j=0,1 \dots N-1$; N - počet vzoriek v smere osi y ,

$0 \leq f(i,j) < G$, kde G – počet jasových úrovní

2.3 Ako dostať obraz do počítača

Zber obrazu (image acquisition) je prvou etapou v procese jeho číslicového spracovania. Cieľom tejto etapy je transformovať obraz v intuitívnom (fyzikálnom) slova zmysle do podoby vhodnej na spracovanie počítačom (zvyčajne matica pixelov). Jednotlivé etapy tohto procesu sú:

- snímanie (scanning)
- vzorkovanie a kvantovanie (sampling and quantisation)
- reprezentácia, kódovanie, uloženie (representation, coding, storing)

2.3.1 Snímanie obrazu (scanning)

Je prvou etapou zberu obrazu a jej cieľom je previesť optické veličiny na elektrické. Tejto etape je treba venovať veľkú pozornosť, nakoľko môže v rozhodujúcej miere ovplyvniť kvalitu signálu. Tak ako nekvalitná anténa znehodnotí aj najcítlivejší rozhlasový príjmač, podobne aj chyby spôsobené nekvalitnou kamerou alebo zlým usporiadaním snímacieho reťazca je potom veľmi ťažké korigovať.

2.3.2 Zdroje obrazu

Snímače konvertujú optický signál na signál jasu alebo farebný RGB signál.

Skener slúži na snímanie statických predlôh z filmu alebo papiera. Snímačom býva fotocitlivá dióda alebo fotonásobič. Najčastejšie skenery sú rovinné alebo bubnové. Niektoré špeciálne typy sú

založené na báze čítacieho lúča usmerňovaného sústavou zrkadiel, pričom predloha i snímač sú nepohyblivé.

Družicové snímače sa používajú na diaľkový prieskum Zeme. Rozklad obrazu v horizontálnom smere zabezpečuje rotujúce zrkadlo, vo vertikálnom smere je to samotný pohyb družice. Sníma sa odrazené žiarenie odrazené zo Zeme (napr. infračervené svetlo). Príkladom je družica LANDSAT, ktorá sníma v 4 viditeľných a 1 IČ spektrálnom pásme.

Televízne kamery. Staršie typy sú na báze snímacích elektróniek, novšie na báze CCD prvkov. Zvyčajne poskytujú výstup v TV norme. Najkvalitnejšie farebné CCD kamery rozdelia opticky dopadajúce svetlo do 3 pásem, každá spektrálna zložka je snímaná samostatným snímačom.

2.3.3 Spektrum a farba.

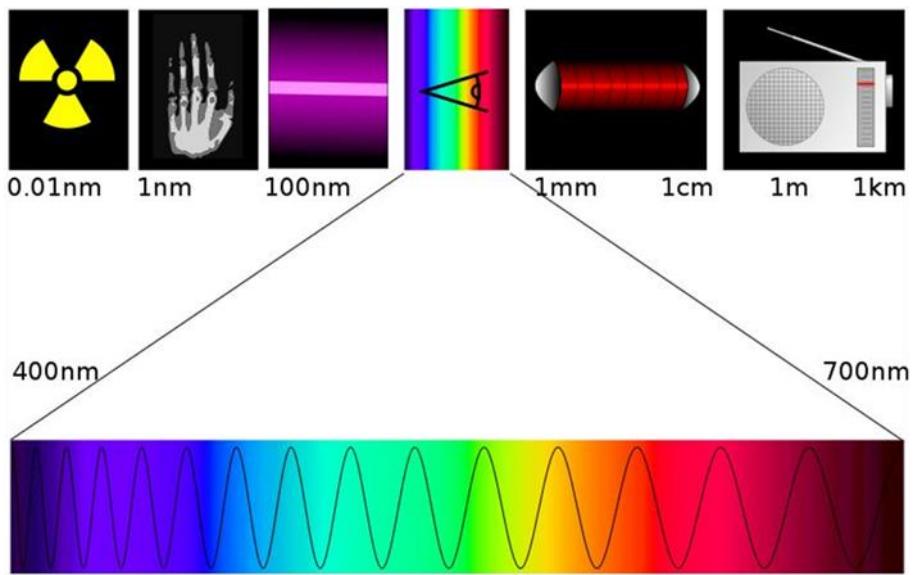
Princíp videnia (včítane počítačového) je založený na interakcii senzora (oko, filmová emulzia, CCD čip) so svetlom prichádzajúcim od pozorovaného objektu.

Pozorovaný objekt mení svetlo ktoré na neho dopadá zo zdroja svetla (slnko, lampa) a to nasledovne:

1. Odraz dopadajúceho svetla od povrchu. V tomto prípade sa všetko dopadajúce svetlo odrazí od povrchu pod rovnakým uhlom a jeho spektrum sa nemení (lesklý povrch nemá farbu).
2. Odraz svetla vnútra objektu. Svetlo prenikne do vnútra objektu a pod jeho povrhom reaguje s časticami pigmentu. Časť svetla určitej vlnovej dĺžky je pohltiená a zvyšok vyžiari v náhodnom smere do okolitého priestoru. Senzor potom vníma spektrum odrazeného svetla ako určitú farbu.

V súčasnosti dokážu detektory zachytiť celú oblasť elektromagnetického spektra a previesť ho na oblasť viditeľného spektra (viď Obrázok 3), ktoré je možné vidieť ľudským okom. Metódy číslicového spracovania obrazu sú zamerané hlavne na úlohy, pri ktorých človek dokáže okom vidieť vstupný obraz a tiež vyhodnotiť výsledok počítačového spracovania. Treba tiež zdôrazniť, že na ľudskom vnímaní farby sa okrem samotného oka podieľa aj celý kognitívny systém a teda vnem farby môže byť individuálny. Dôležitou vednou disciplínou je „Color constancy“, ktorej úlohou je rovnaké vnímanie farby povrchu bez ohľadu od uhla nasvietenia, tieňa, orientácie objektu a pod. To má význam nie len pri vizuálnom vnímaní (verná reprodukcia farieb pri tlači a displejoch), ale aj v robotike pri rozpoznávaní objektov určitej farby.

Človek rozpoznáva asi 400 000 odtieňov – t.j. stačí zhruba 100 jasových úrovní pre každú farebnú zložku. V praxi sa používa zvyčajne 256 úrovní (8 bitov).



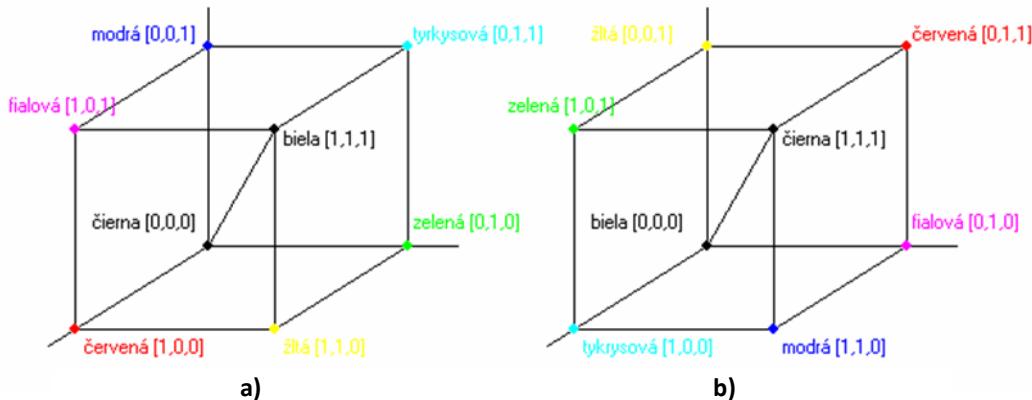
Obrázok 3. Oblast' elektromagnetického spektra predstavujúca viditeľné svetlo.
zdroj: Wikipédia: <http://cs.wikipedia.org/wiki/Soubor:Spectre.svg>

Vhodným výberom typu a smeru osvetlenia dokážeme v mnohých aplikáciach výrazne ovplyvniť kvalitu obrazu a ušetriť tak množstvo času pri výpočte. Napríklad vhodným osvetlením objektov na výrobnej linke pri kontrole kvality dostaneme vysoko kontrastný obraz s nízkym šumom, ktorý sa ľahko segmentuje prahovaním a je možné ho spracovať v reálnom čase alebo v pripade 2,5D(3D) kamier vhodným osvetlením dokazeme získať geometriu snimaneho objektu.

2.3.4 Miešanie farieb

Obrázok 4 ukazuje, ako sa jednotlivé farebné zložky premietajú do tzv. farebného priestoru – v danom prípade 3-rozmerného pre farebné zložky R, G, B. Tento farebný priestor sa využíva hlavne v televíznej technike, kde jednotlivé farebné zložky prestavujú červenú (**Red**), zelenú (**Green**), resp. modrú (**Blue**) farbu.

Ked' intenzita každej farebnej zložky je kódovaná 8-mi bitmi (256 úrovní), potom pomocou nich vieme namiešať ľubovoľnú farbu z $256^3 = 16777216$ kombinácií, čiže farieb.



Obrázok 4. Miešanie farieb. a) – Adatívne, b) Subtraktívne.

Okrem spomínaného RGB farebného priestoru existujú aj iné modely ako napr. YIQ, kde Y predstavuje intenzitu a I, Q farbu. Podobne CMY (Cyan, Magenta, Yellow) využíva 3 farby a uplatňuje sa hlavne pri farebných tlačiarňach. HSV (Hue, Saturation, Value) a ich modifikácie (HLS, HSI) majú špecifické užitočné vlastnosti. Podrobnejší popis jednotlivých farebných modelov je vo všetkých citovaných učebniciach a konverzii medzi formátmi vykonáva väčšina knižníc na spracovanie obrazov včítane OpenCV.

2.3.5 Digitalizácia obrazu

Väčšina snímačov obrazu poskytuje na výstupe obrazovú funkciu v spojitej tvare, čo znamená spojitosť v definičnom obore i v obore funkčných hodnôt. Jej spracovanie na číslicovom počítači vyžaduje prevod do diskrétnej podoby, čiže **digitalizáciu**. Tá pozostáva z 2 častí :

1. **vzorkovania** obrazu v matici MxN bodov
2. **kvantovania** jasovej úrovne na L intervalov.

2.3.6 Vzorkovanie

Je jasné, že čím je **hustota vzorkovania** vyššia, tým presnejšie bude diskrétna funkcia approximovať spojité. Je ale zrejmé, že z technického hľadiska je nezmyslom vzorkovať hustejšie, než je s ohľadom na charakter obrazu potrebné. Otázkou je aká je najnižšia potrebná vzorkovacia frekvencia, aby nedošlo ku strate informácie. S tým súvisí aj otázka zamedzenia stroboskopického efektu ako to popisuje príbeh „Vrtule (vzorkovanie)“.

Odpoveď dáva známa **Shanonova veta o vzorkovaní**, ktorá hovorí, že vzorkovacia frekvencia musí byť aspoň dvakrát väčšia ako maximálna frekvencia ktorú obsahuje vzorkovací signál.

$$f_{vzor} \geq 2f_{\max} \quad (2)$$

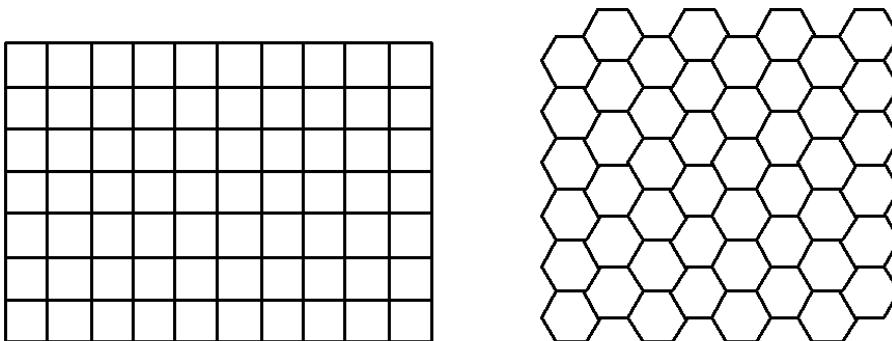
V oblasti spracovania obrazu má Shanonova veta prirodzenú fyzikálnu interpretáciu ktorá hovorí, že interval vzorkovania musí byť menší alebo rovný polovici rozmerov najmenších detailov na obraze, ktoré chceme ešte vidieť.

Vo frekvenčnej oblasti je interpretácia nasledovná: Hodnotu obrazovej funkcie na určitom mieste získame konvolúciou obrazovej funkcie a posunutej impulznej funkcie (tzv. Diracovej delta popísanej

v nasledujúcej kapitole). Táto má teoreticky nekonečne malú šírku, avšak v praxi je šírka daná rozmermi CCD elementu snímacieho senzora. Je známe, že Fourierova transformácia snímaného obrazu je sumou periodicky opakovaných Fourierovských transformácií. Shanonova veta vo frekvenčnej oblasti je interpretovaná tak, že frekvencia obsiahnutá v týchto $F(u,v)$ oblasti nemá byť vyššia ako je maximálna, v opačnom prípade dochádza k rušivému efektu zvanému **aliasing**. Príbeh „*Košela. (Aliasing)*“ popisuje možné dôsledky a možnosti zamedzenia aliasingu.

2.3.7 Usporiadanie vzoriek pri vzorkovaní.

Elementárne body, ktoré dostaneme po vzorkovaní, sa nazývajú pixely a môžeme ich usporiadať rôznym spôsobom a to podľa tvaru **vzorkovacej mriežky** (viď Obrázok 5).



Obrázok 5. Štvorcová a hexagonálna mriežka.

Hexagonálna mriežka oproti štvorcovej poskytuje výhodu rovnakej vzdialenosťi bodu od všetkých svojich susedov. Aj napriek tejto výhode sa nepresadila a dnes sa už používa iba sporadicky.

2.3.8 Kvantovanie

Kvantovanie je prevod hodnoty obrazovej funkcie do číslicovej podoby. Počet kvantovacích úrovní musí byť dostatočne vysoký na to, aby nevznikali falošné obrysy, a aby citlivosť zariadenia sa blížila citlivosti ľudského oka. Najčastejšie sa kvantuje na 256 úrovní jasu, čo zodpovedá kódovaniu 8 bitmi. **Binárne obrazy** vystačia iba s jediným bitom, ktorý odlišuje pixely objektov (jednotka) od pozadia (nula). V niektorých aplikáciách (napr. denzitometrii) sa kvantuje aj na oveľa vyšší počet úrovní než je schopné ľudské oko zachytiť (zvyčajne 16 bitov). Tu však nie je cieľom vizuálny vnem, ale meranie úrovne jasu, ktoré je v denzitometrii zvyčajne proporcionálne inej fyzikálnej veličine.

Najväčším problémom kvantovania na nedostatočný počet úrovní je vnímanie falošných hrán. Je to možné odstrániť napr. nelineárnym kvantovaním, kde sa zväčšuje rozsah tých intervalov jasu, ktoré sú v obraze málo zastúpené). Ďalším „trikom“ je pridanie umelého tzv. bieleho šumu, ktorý (paradoxne) zlepšuje vizuálny vnem tým, že rozruší subjektívne vnímané umelé hranice.

2.3.8.1 OpenCV

Umožňuje pracovať s obrazmi reprezentovanými rôznym počtom bitov (8/16/32/64). Tieto bity môžete interpretovať ako celočíselné (signed alebo unsigned), resp. ako reálne čísla (float alebo double). Tak isto umožňuje priradiť obrazu určitý počet kanálov(1 - 4).

Mat Img(480, 640, CV_8UC3) vytvorí obraz Img rozmerov 640x480 kde každý pixel je reprezentovaný troma hodnotami (RGB), kde každý kanál je kódovaný 8 bitmi (unsigned).

`Img.convertTo(Img, CV_32FC3)` konvertuje obraz `Img` na obraz iného typu (každý kanál je 32 bitové reálne číslo).

2.4 Vlastnosti digitálnych obrazov

2.4.1 Metrické a topologické vlastnosti

2.4.1.1 Vzdialenosť

Je základným pojmom v oblasti topológie a môže byť definovaná rôzne.

Euklidovská vzdialenosť medzi bodmi (i,j) a (h,k) najviac súvisí s predstavou vzdialenosť používanej v bežnom živote. Bohužiaľ neceločíselný výsledok a zložitý algoritmus počítania pre odmocniny je vážnym hendikepom v porovnaní s inými definíciami.

$$D_E = \sqrt{(i-h)^2 + (j-k)^2} \quad (3)$$

Vzdialosť v mestských blokoch (city block distance, Manhattan distance). Je založená na predstave pohybu v meste s pravouhlou sieťou ulíc, kde prekonanie vzdialenosť „vzdušnou čiarou“ nie je možné.

$$D_4 = |i - h| + |j - k| \quad (4)$$

Šachovnicová vzdialenosť (chessboard distance) si môžeme predstaviť ako najmenší počet krokov, ktorý musí urobiť kráľ na šachovnici, aby prekonal túto vzdialenosť (môže sa pohybovať v horizontálnom, vertikálnom i uhlopriečnom smere).

$$D_8 = \max \{|i - h|; |j - k|\} \quad (5)$$

So vzdialenosťou úzko súvisí aj pojem Vzdialenosťná transformácia (Distance transform), ktorá je podrobnejšie rozoberaná v časti Matematická morfológia.

2.4.1.2 Susedstvo, okolie.

Tieto pojmy definujú, ktoré body je možné považovať za susedné. V prípade diskrétnej mriežky predstavuje **8-susedstvo** určitého bodu všetky okolité body, včítane uhlopriečnych, čiže obe súradnice sa môžu lísiť o 1.

Naproti tomu **4-susedstvo** nepovažuje za susedov body v uhlopriečnej vzdialenosťi, čiže susedné body sa môžu lísiť iba jednou súradnicou.

2.4.1.3 Cesta, oblasť, objekty, pozadie.

Cesta z bodu P do bodu Q sa nazýva postupnosť obrazových elementov A_1, A_2, \dots, A_n pre ktoré platí:

$$A_1 = P, \quad A_n = Q, \quad A_{i+1} \text{ je susedom bodu } A_i \quad (6)$$

Oblasť je taká množina obrazových elementov (pixelov), že medzi ľubovoľnou dvojicou z nich existuje cesta, ktorá patrí celá do tejto množiny. Hovoríme tiež, že je to **súvislá množina**.

Predpokladajme, že R_i sú oblasti obrazu organizované tak, že sa nedotýkajú okrajov obrazu. Nech R je oblasť vytvorená zjednotením všetkých R_i . Množina R^C je komplementárna (doplňková) oblasť ku R . Tú časť R^C , ktorá je súvislá s okrajmi obrazu voláme **pozadím**, zvyšok sú **diery**. Pokiaľ oblasť R_i neobsahuje diery, hovoríme, že je **jednoducho súvislá**. V prípade, že obsahuje diery je **viacnásobne súvislá**.

Na rozdiel od oblasti, **objekt** je časť obrazu, ktorá koreluje s objektmi reálneho sveta. Proces získavania objektov z obrazu voláme segmentácia a bude jej venovaná náležitá pozornosť v ďalších častiach.

2.4.1.4 Hranice oblastí.

Hranica oblasti je množina všetkých obrazových elementov z oblasti, z ktorých každý má aspoň jedného suseda, ktorý nepatrí do oblasti.

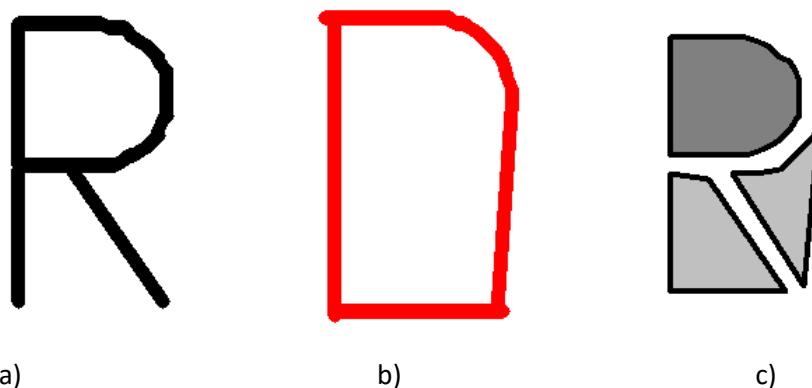
Naproti tomu **hrana** je vlastnosťou obrazového elementu a jeho lokálneho okolia. Je určená veľkosťou a smerom. **Veľkosť hrany** zodpovedá gradientu spojitej obrazovej funkcie v príslušnom obrazovom elemente, jeho smer je určený smerom najväčšieho rastu obrazovej funkcie.

Smer hrany je kolmý na smer gradientu (pootočený o -90 stupňov, je teda dotyčnicou hranice oblasti, čo zjednodušuje jej vyhľadávanie a reťazenie.

V niektorých aplikáciách sa strelneme aj s pojmom **medzibunečná hrana**. Každý element má teda okrem hodnoty jasu aj 4 hodnoty zodpovedajúce absolútnej hodnote rozdielov jasov medzi elementom a jeho okolitými pixelmi v zmysle 4-susedstva.

2.4.1.5 Konvexný obal.

Je to najmenšia oblasť obsahujúca objekt taká, že každé dva body oblasti môžu byť spojené úsečkou, ktorej všetky body patria do oblasti (Obrázok 6). Vhodným modelom takéhoto konvexného obalu je gumička, ktorú natiahneme na oblasť. Vnútri konvexného obalu ostanú aj pixely, ktoré nepatria do objektu, voláme ich **deficit konvexnosti**. Tento sa ďalej môže rozdeľovať na oblasti plne ohraničené objektom (**jazerá** – silnejšie vyfarbené na c), a potom **zálivy**, ktoré ležia medzi konvexným obalom a objektom (slabšie vyfarbené).



Obrázok 6. a) objekt, b) konvexný obal objektu c) deficit konvexnosti (jazera a zálivy).

2.4.1.6 OpenCV

`void convexHull(InputArray points, OutputArray hull, bool clockwise=false)`

Pre množinou bodov `points` reprezentujúcich obrysové body oblasti nájde inú množinu bodov reprezentujúcich jej konvexný obal.

`convexityDefects(InputArray contour, InputArray convexhull, OutputArray convexityDef)` - vytvorí zoznam oblastí reprezentujúcich deficit konvexnosti medzi kontúrou a jej konvexnou obálkou.

2.4.2 Histogram jasových úrovní

Histogram jasu je vektor H s počtom zložiek rovným počtu jasových úrovní. Hodnota každej zložky zodpovedá počtu bodov, ktoré majú príslušnú jasovú úroveň. Na Obrázok 7 je obraz pozostávajúci zo šedotónových pixelov. Graf pod ním zobrazuje histogram tohto obrázku. Je na ňom viditeľných niekoľko vrcholov. Prvý vrchol zľava reprezentuje najtmavšie pixely (kalkulačka, kábel, nápis v pozadí, hrana stola). Postupne smerom doprava rastie počet sivých pixelov (steny tlačiarne a skenera). Posledný vrchol vpravo patrí najsvetlejším pixelom reprezentujúcich biely list papiera v popredí.

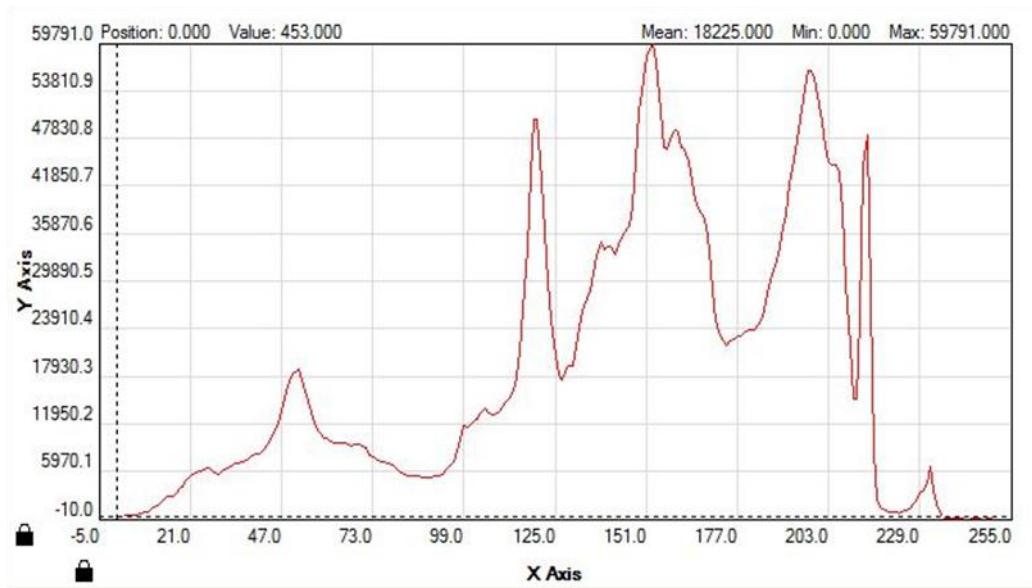
Histogram jasových úrovní predstavuje globálnu informáciu o obraze, ktorú je možné využiť napr. na nastavenie vhodných podmienok snímania a pozorovania obrazov, porovnanie podobnosti obrazov, rozpoznavanie, alebo na segmentáciu prahovaním, o ktorej ešte bude reč. Jednému obrazu môže byť priradený iba jediný histogram, avšak jeden histogram môže byť priradený viacerým obrazom. Napr. posunutím objektu na obraze, ktorého pozadie má konštantný jas, sa histogram nemení.

Mnoho algoritmov hľadá extrémne hodnoty grafu zodpovedajúce jednotlivým objektom s rozličným jasom. V tom prípade je dôležité filtrovaním grafu odstrániť lokálne extrémy, ktoré môžu byť spôsobené napr. šumom.

2.4.2.1 OpenCV

`calcHist(...)`. Táto funkcia nájde histogram jedného alebo viacerých obrazov rôzneho typu. Obsahuje viacero parametrov špecifikujúcich počet a typ obrazu - vid' dokumentácia.

`double compareHist(InputArray H1, InputArray H2, int method)` - porovnávanie histogramov rôznymi metódami



Obrázok 7. Histogram šedotónového obrazu, na x-ovej osi je jas, na osi y je počet pixelov obrázku s daným jasom.

2.4.3 Šum

Vzniká pri snímaní, prenose i spracovaní obrazu. Môže byť závislý i nezávislý od úrovne signálu.. Existujú rôzne typy šumu. Podľa pravdepodobnostnej charakteristiky alebo podľa formy výstupu ich ich môžeme deliť na:

Biely šum – je idealizovaný šum, ktorý má vo svojom výkonovom spektre rovnomerne zastúpené všetky frekvencie.

Gaussov šum – jeho pravdepodobnostná charakteristika je 1-rozmerná náhodná veličina s Gaussovým (normálnym) rozdelením a má hustotu pravdepodobnosti danú vzťahom

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \quad (7)$$

kde μ je stredná hodnota a σ je smerodajná odchýlka.

Aditívny šum je obvykle nezávislý od signálu (vidikónová kamera). Dá sa popísať nasledovne:

$$f(x, y) = g(x, y) + v(x, y) \quad (8)$$

kde šum v a vstupný obraz g sú neznáme veličiny.

Multiplikatívny šum je taký, pri ktorom úroveň šumu je závislá od úrovne signálu (napr. šum TV rastra, ktorý ma maximum v čiernej a minimum v bielej oblasti. Pri dostatočne veľkej úrovni šumu platí:

$$f = g + vg = g(1 + v) \approx gv \quad (9)$$

Kvantizačný šum – vzniká pri digitalizácii obraz – bol už spomínaný v časti „Digitalizácia obrazu“.

Šum typu „pepor a sol“ (impulzný šum), ktorý sa prejavuje výrazne odlišou hodnotou oproti hodnote signálu.

2.4.3.1 OpenCV

void randn(InputOutputArray dst, InputArray mean, InputArray stddev) - generuje obraz obsahujúci pixely ktorých hodnoty sú normálne rozdelenie s parametrami mean a stddev. Takto je možné pridať Gaussovský šum k ľubovoľnému obrazu a študovať účinky rôznych filtrov.

funkcia randu generuje šum rovnomerného rozdelenia. Iné typy rozdelenia je možné generovať napr. funkciami RNG::fill, kde RGN je generátor náhodných čísel s definovaným rozdelením.

2.5 Dátové štruktúry používané pri spracovaní

Okrem bežných (zvyčajne maticových) foriem reprezentácie celého obrazu sa používajú aj také formy, ktoré kódujú kontúry segmentovaných objektov. Uzavretá hranica objektu sa nazýva kontúra, môže slúžiť ako reprezentant objektu a zdroj informácií o ňom. Je to oveľa úspornejšia forma reprezentácie v porovnaní s pixelovou reprezentáciu oblasti. Princíp hierarchie sa využíva na rôznych úrovniach abstrakcie objektov.

2.5.1 Matica

Je najobvyklejšou formou reprezentácie na nízkej úrovni. Pozícia prvku v matici zodpovedá pozícii na obraze, jeho hodnota zodpovedá zvyčajne jasu. Matica môže byť interpretovaná pomocou štvorcovej mriežky, alebo pomocou hexagonálnej mriežky kde sú jednotlivé riadky voči sebe o polovicu fázy posunuté.

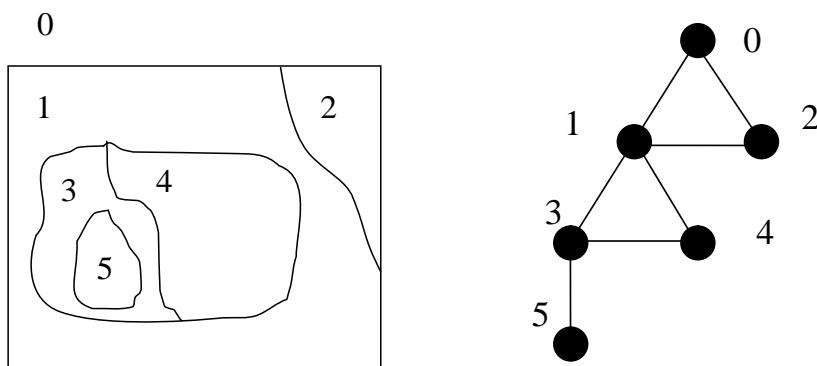
Zvláštnou formou reprezentácie pomocou matíc je tzv. „kookurenčná matica“, ktorá je popísaná v kapitole o textúrach.

2.5.2 Kódovanie úsekov riadkov (Run-length coding).

Je to tiež forma reťazového kódu, používaná hlavne v binárnych obrazoch. Každý riadok obsahuje na začiatku číslo riadku a potom nasleduje zoznam dvojíc bodov, z ktorých prvý predstavuje začiatok oblasti v riadku a druhý jej koniec. Výsledný kód potom reprezentuje zoznam kódov jednotlivých riadkov. Kódovanie na podobnom princípe používajú hlavne faxové prístroje. Použitie tejto metódy na šedotónové obrazy vyžaduje kódovanie úsekov, ktorých jas môžeme považovať za konštantný.

2.5.3 Kódovanie topologických vlastností obrazu.

Na kódovanie použijeme graf $G=(V,E)$, čo je algebraická štruktúra pozostávajúca s množinou uzlov $V=\{v_1, \dots, v_n\}$ a množinou spojníc $E=\{e_1, \dots, e_m\}$. Aby sme sa vyhli zámene s hranou objektov, nebudeme používať obvyklý termín hranu grafu, ale miesto toho použijeme termín spojnica grafu. Každá spojnica e_k je incidentná s neusporiadaným párom uzlov $\{v_i, v_j\}$. Stupeň uzla je rovný počtu spojníc, ktoré sú s nimi incidentné. Pomocou týchto poznatkov vieme vytvoriť graf susedstva oblastí (Obrázok 8), kde uzly grafu zodpovedajú oblastiam, a susediace oblasti sú spojené spojnicou. Oblast 0 označuje oblasti mimo obraz, v grafe sa označenie 0 využíva k vyjadreniu oblastí, ktoré sa dotýkajú okraja obrazu.



Obrázok 8. Graf susedností oblastí.

2.5.3.1 Relačné štruktúry.

Všetky informácie sú sústredené v sémanticky významných častiach obrazu, teda v objektoch. Relačná tabuľka zaznamenáva jednotlivé objekty, ich hlavné vlastnosti (farba), polohu a vzájomné vzťahy (viď Obrázok 9).

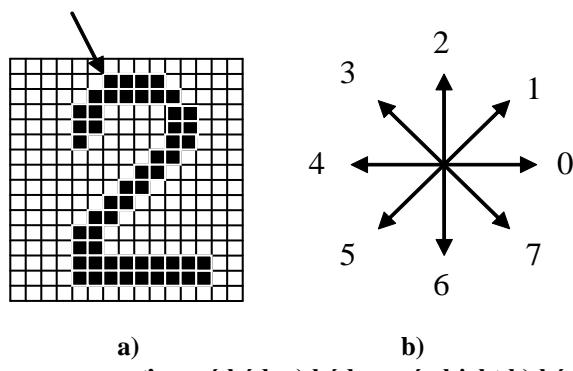
č. meno	farba	poloha	vnútri
1 slnko	biela	(50,50)	2
2 obloha	modrá	(0,0)	-
3 mrak	šedá	(20,150)	2
4 kmeň	hnedá	(100,75)	6

Obrázok 9. Scéna predstavujúca krajinu a relačná tabuľka objektov, ktoré obsahuje.

2.5.4 Freemanov (reťazcový) kód

Je určený počiatočným bodom a postupnosťou symbolov zodpovedajúcich úsečkám jednotkovej dĺžky v niekoľkých vopred stanovených orientáciách. Na príslušnom obrázku je zobrazené kódovanie smerov. Reťazový kód pre číslicu 2 so začiatkom v mieste na ktorá ukazuje šípka bude:

0000776655555660000000644444444222111112234445652211



Obrázok 10. Freemanov reťazový kód. a) kódovaný objekt b) kódovanie smerov.

Freemanov reťazcový kód je vhodný aj na syntaktickú formu rozpoznávania.

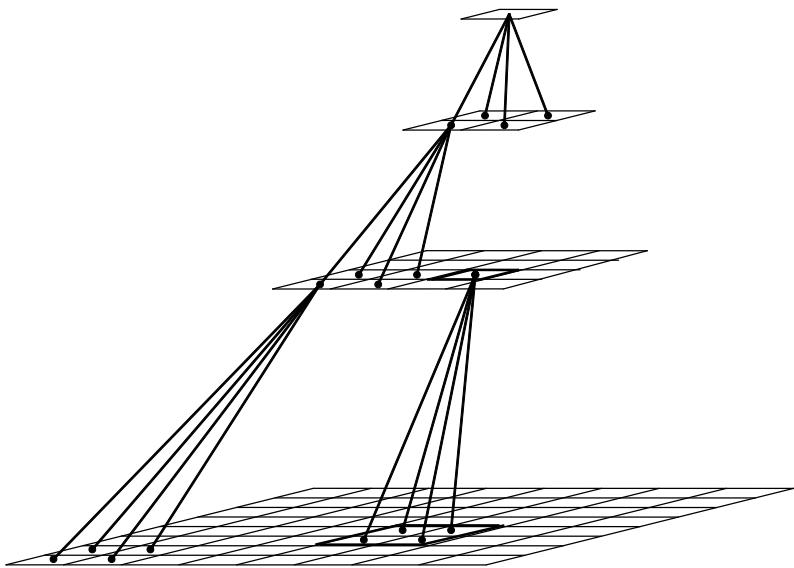
2.5.5 Polygonálna reprezentácia hranice

Je to reprezentácia, ktorá approximuje oblasť mnohouholníkom. Keďže mnohouholník je jednoznačne určený súradnicami jeho vrcholov, bude aj polygonálna reprezentácia hranice objektov reprezentovaná postupnosťou súradníc hraničných bodov. Úseky medzi týmito bodmi sú approximované úsečkou. Podľa stupňa presnosti s akou sa approximuje reálny obrys úsečkou je možné aj voliť presnosť approximácie podľa požiadaviek. Pokiaľ vystačíme s nižšou presnosťou, bude polygonálna reprezentácia veľmi úsporná.

Existujú aj alternatívne metódy, ktoré namiesto priamky používajú na approximáciu polynómy 2. rádu, čím dostaneme hladší obrys pri zachovaní úspornosti reprezentácie.

2.5.6 Hierarchické dátové štruktúry

Obsahujú okrem obrazu v plnom rozlíšení aj nejakú formu zjednodušenej verzie obrazu, ktorá má typicky menšie množstvo dát. To pomáha riešiť základný problém spracovania obrazu a to je výpočtová náročnosť. Princíp spočíva v tom, že najprv sa algoritmus aplikuje na hrubšiu approximáciu obrazu a cieľom vtipovať na obraze v plnom rozlíšení perspektívne oblasti. Algoritmus sa potom sústredí iba na ne, čím sa značne redukuje objem spracovávaných dát. Medzi základné hierarchické reprezentácie patria pyramídy (Obrázok 11) a kvadrantové stromy.



Obrázok 11. Pyramída obrazov (M-pyramída)

Každý bod nadradenej hladiny sa vypočíta ako aritmetický priemer zodpovedajúcej štvorice bodov z hladiny nižzej. Takáto pyramída predstavujúca postupnosť obrazov v maticovom tvare sa niekedy volá **M-pyramída** (Matrix pyramid). Existuje viacero spôsobov ako počítať hodnotu bodu vyšej hladiny z množiny bodov hladiny nižzej, najčastejšie používaná je forma keď váhy prispievateľov approximujú Gaussovskú krivku (Gaussovská pyramída).

2.5.6.1 *T-pyramída, Kvadrantové stromy*

Je to reprezentácia podobná M-pyramíde, avšak nepoužíva maticový zápis pixelov obrazu ale **stromovú štruktúru** s cieľom úspornejšieho zápisu. Preto sa volá T-pyramída (Tree pyramid). Pokiaľ štvorica pixelov tvoriaci pixel nadradenej hladiny je homogénna, stáva sa ich reprezentantom v hierarchickom strome. **Kvadrantové stromy** sú špeciálnym prípadom T-pyramídy, kde sa nekódujú všetky úrovne stromu, ale keď je určitý kvadrant homogénny, označí sa daný uzol stromu ako terminálny. Takže ak by sme mali napr. štvorcový obraz 256x256 a prvý kvadrant by bol homogénny, potom na zakódovanie celej podliehajúcej oblasti 128x128 bodov stačí zakódovať jedený pixel na hladine bezprostredne pod vrcholom. Príklad kvadrantového stromu je uvedený v kapitole o segmentácii. Bolo publikovaných množstvo algoritmov, ktoré spracovávajú obraz priamo v tvare kvadrantového stromu.

2.5.6.2 *OpenCV*

`void buildPyramid(InputArray src, OutputArrayOfArrays dst, int maxlevel,...0)`

Táto funkcia vytvorí Gaussovskú pyramídu za použitia pomocných funkcií (pyrDown a pyrUp).

3 Diskrétné lineárne integrálne transformácie

V tejto časti sa budeme zaoberať hlavne problematikou konvolúcie, Fourierovej transformácie a ich vzájomnej súvislosti a praktického významu. Táto oblasť je však ďaleko širšia a okrem spomínaných do nej patria napr. Diskrétna kosínusová transformácia, Wavelety, Wienerove filtre a ďalšie, ktoré je možné nájsť napr. v (Szeliski, Computer Vision Algorithms and Applications, 2011).

3.1 Linearita a konvolúcia

Linearita znamená, že počítame v priestore, kde sa dá použiť maticová algebra.

Lineárna kombinácia znamená, že nový element výstupného obrazu sa získa sčítaním niekoľkých elementov vstupného obrazu násobených príslušnými koeficientami. V prípade lokálnych operátorov k hodnote určitého pixelu prispievajú svojimi hodnotami aj jeho susedia.

3.1.1 Konvolúcia

Je to metóda, ktorá systematicky prechádza celý obraz a na výpočet novej hodnoty bodu využíva malé okolie O reprezentatívneho bodu. Príspevok každého pixelu z okolia O na výsledok je vážený príslušným koeficientom ktorý udáva obraz h , ktorý sa zvyčajne volá **konvolučná maska**. Táto má zvyčajne nepárný počet riadkov a stĺpcov (napr. 3x3, 5x5 a pod.), čo umožňuje stotožnenie vyšetrovaného bodu s centrálnym bodom masky.

Funkcia g je **konvolúcia** dvoch funkcií f a h :

$$g = f * h \quad (10)$$

Konvolúcia sa počíta nasledovne:

$$g(i, j) = \sum_{k,l} f(i-k, j-l)h(k, l) = \sum_{k,l} f(k, l)h(i-k, j-l) \quad (11)$$

Najjednoduchší príklad je napr. konvolučná maska 3x3, ktorej každý prvok má hodnotu 1. Nakreslime takúto maticu na priesvitnú fóliu a posúvajme ju postupne po obraze. Pixel výstupného obrazu (pod centrálnym bodom konvolučnej masky) bude daný súčtom všetkých pixelov pod konvolučnou maskou (v tomto prípade násobených jednotkou). Tento súčet normujeme, teda vydelíme súčtom všäk masky, teda hodnotou 9. Takáto konvolúcia je vlastne analógia neváhovaného klízavého aritmetického spriemerňovania v okolí O.

3.2 Lineárne integrálne transformácie

Spracovanie obrazov, ktoré využíva spojité alebo diskrétné integrálne transformácie, ako napr. Fourierovu transformáciu, je možné považovať za klasické techniky. Základné postupy sa rozvíjali najmä v 60. rokoch, inšpirovali sa najmä v lineárnej teórii signálov

Existujú dva spôsoby filtrácie:

1. Filtrácia v priestorovej oblasti - najjednoduchším príkladom je spriemerovanie. Transformuje vstupný (zašumený) obraz priamo na výstupný s redukovaným šumom.
2. Filtrácia vo frekvenčnej oblasti (priama transformácia do frekvenčnej oblasti, potom aplikácia frekvenčného filtra a spätná transformácia)

3.2.1 Jednorozmerná Fourierova transformácia

Je založená na jednoduchej základnej myšlienke. Majme filter s neznámou prenosovou charakteristikou. Na jeho vstup pripojíme generátor sínusového signálu a meriame jeho charakteristiku na výstupe.

$$s(x) = \sin(2\pi f x + \phi_i) = \sin(\omega x + \phi_i) \quad (12)$$

kde ω je uhlová rýchlosť daná frekvenciou f a ϕ_i je fáza vstupného signálu. Na výstupe filtra dostaneme

$$o(x) = h(x) * s(x) = A \sin(\omega x + \phi_o) \quad (13)$$

čiže výstupný signál má rovnakú frekvenciu, avšak rozdielnu amplitúdu A a fázu ϕ_o . Podobne ako pri lineárnom filtri kde výsledná hodnota bola váženým súčtom pixelov posunutých voči stredu konvolučnej masky, aj tu bude výstupný signál váženým súčtom posunutých sínusoviek o rovnakej frekvencii.

Ľubovoľnú funkciu $f(x)$ môžeme zapísť vo forme súčtu sínusových a kosínusových výrazov rastúcej frekvencie. Ľubovoľné priestorové alebo časom sa meniace dátá môžu byť transformované do frekvenčnej oblasti.

Vychádzame z Eulerovej formuly pre komplexné čísla, ktorá upraví goniometrické funkcie do tvare exponentu.

$$s(x) = e^{j\omega x} = \cos \omega x + j \sin \omega x \quad (14)$$

čiže

$$o(x) = h(x) * s(x) = A e^{j\omega x + \phi} \quad (15)$$

kde A je amplitúda a ϕ je rozdiel fázového posunu medzi vstupným a výstupným signálom. Priama spojitá jednorozmerná Fourierova transformácia môže byť vyjadrená vzťahom:

$$H(\omega) = \int_{-\infty}^{\infty} h(x) e^{-j\omega x} dx \quad (16)$$

v diskrétnnej oblasti:

$$H(k) = \frac{1}{N} \sum_{x=0}^{N-1} h(x) e^{-j2\pi k x / N} \quad (17)$$

3.2.2 Dvojrozmerná Fourierová transformácia

V dvojrozmernnej oblasti miesto jednorozmernej funkcie - sínusoidy vytvoríme orientovaný sinusoid:

$$s(x, y) = \sin(\omega_x x + \omega_y y) \quad (18)$$

Potom dvojrozmerná Fourierová transformácia bude v spojitej oblasti

$$H(\omega_x, \omega_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(x, y) e^{-j(\omega_x x + \omega_y y)} dx dy \quad (19)$$

v diskrétnej oblasti:

$$H(k_x, k_y) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} h(x, y) e^{-j2\pi(k_x x + k_y y) / MN} \quad (20)$$

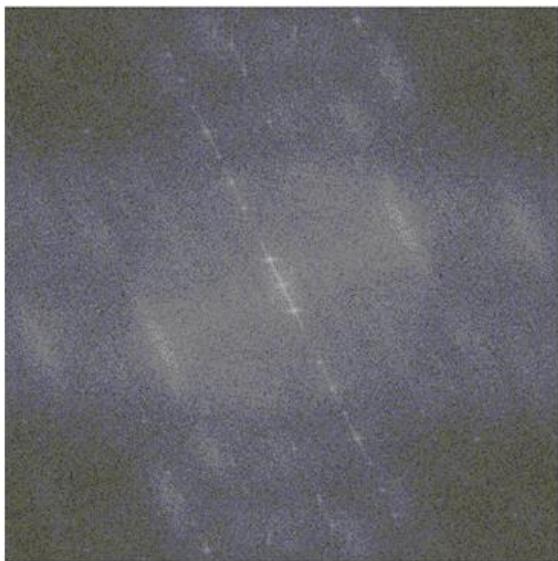
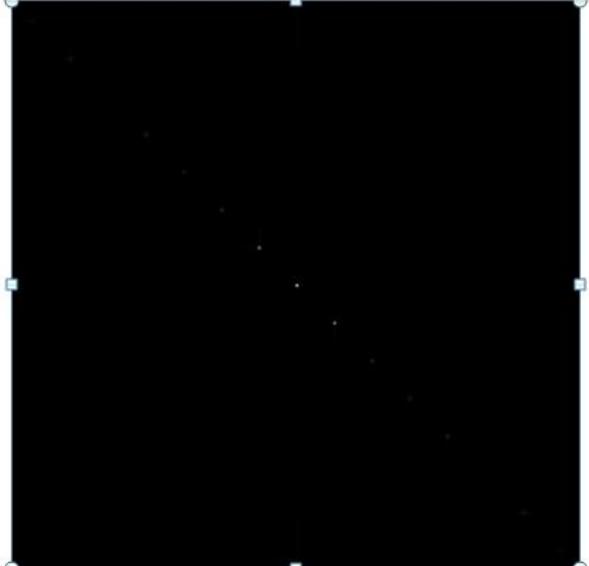
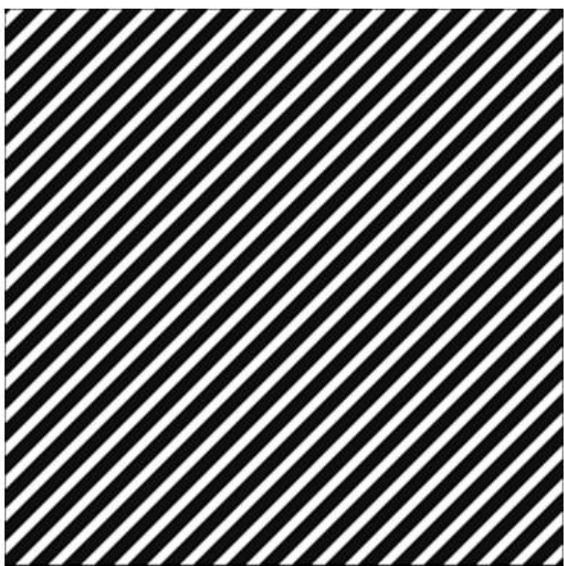
V súčasnosti pri praktických úlohách užívateľ mállokedy programuje výpočet podľa hore uvedených vzorcov, miesto toho sa využívajú optimalizované knižnice. Preto neuvádzame vzorec pre spätnú Fourierovú transformáciu, ktorý je možné nájsť v akejkoľvek literatúre zaoberejúcej sa touto problematikou (napr. (Sonka, Hlavac, & Boyle, 2008)). V knižniciach sa s obľubou využíva algoritmus výpočtu pomocou rýchlej Fourierovej transformácie (FFT). Hoci je tento algoritmus limitovaný na rozmery matíc $2^N \times 2^N$, úspora výpočtového času je značná. Knižnice (ako napr. OpenCV) majú tieto funkcie implementované ešte efektívnejšie.

3.2.2.1 OpenCV:

`void dft(InputArray src, OutputArray dst, int flags=0, int nonzeroRows=0)`

Vykoná diskrétnu Fourierovu transformáciu (priamu, spätnú, jednorozmernú, dvojrozmernú). Typ transformácie sa špecifikuje pomocou flags.

pomocné funkcie: `getOptimalDFTSize()`, `mulSpectrums()`, `filter2D()`, `matchTemplate()`, `flip()`, `cartToPolar()`, `magnitude()`, `phase()`.



Obrázok 12. Obrázky v priestorových súradničiach (vľavo) a ich reprezentácia vo frekvenčnej oblasti (vpravo).

3.2.3 Využitie Fourierovej transformácie

Na Obrázok 12 sú znázornené obrázky v priestorových súradničiach a ich fourierovské obrazy. Horná dvojica znázorňuje, že periodický signál sa transformuje na trojicu bodov. V strede je bod zodpovedajúci signálu s nulovou frekvenciou predstavujúca jednosmernú zložku. V smere vlnenia je potom dvojica "bodiekov", ktorých vzdialenosť od stredu predstavuje frekvenciu (čím vyššia je frekvencia vlnenia, tým sú ich vzdialenosť väčšie). Ich smer udáva smer z ktorého vlnenie prichádza a intenzita (jas) zodpovedá výške vlny.

Praktické využitie tohto princípu predstavuje spodná dvojica obrázkov. Tlačený text sa transformuje do frekvenčnej oblasti tak, že sú viditeľné výrazné extrémy z ktorých vieme zistiť nielen orientáciu textu, ale aj vzdialenosť medzi riadkami, prípadne menej výrazné periodicity (odstavce).

3.2.4 Aliasing

Aliasing v preklade znamená „falšovanie“ a vyjadruje situáciu, keď nesprávnym vzorkovaním dochádza k falošnému vnemu. Častým falošným úkazom tohto typu sú Moire prúžky, ktoré sú pozornému pozorovateľovi televízie známe ako falošné pruhy objavujúce sa pri snímaní pravidelne textúrovaných objektov (Obrázok 13).



Obrázok 13. Vľavo obraz snímaný tak, že vzdialenosť prúžkov je menšia než polovica vzdialenosťi medzi senzormi. Vpravo je ten istý obraz snímaný tak, že je dodržané Shanonove kritérium.

Zdroj: Wikipédia - Aliasing

Na odstránenie aliasingu sa používajú rôzne techniky – napr. zaradenie dolnopriepustného filtra, ktorý odfiltruje vysokofrekvenčné zložky spôsobujúce aliasing. V optických sústavách tento účel plní sklenený filter, ktorý jemne rozostri detaily.

3.2.5 Vlastnosti Fourierovej transformácie

1. linearita
2. invariantnosť voči posuvu
3. obraz konvolúcie, vzťah ku korelácií
4. periodicitá

$$\begin{aligned} F(u, -v) &= F(u, N - v), & f(-m, n) &= f(M - m, n) \\ F(-u, v) &= F(M - u, v), & f(m, -n) &= f(m, N - n) \end{aligned} \quad (21)$$

$$F(aM + u, bN + v) = F(u, v) \quad f(aM + m, bN + n) = f(m, n) \quad (22)$$

3.2.6 Konvolučný teorém

Z predošej časti je zrejmá úzka späťosť konvolúcie a Fourierovej transformácie. Táto podobnosť dovoľuje použitie Fourierovej transformácie pre výpočet konvolúcie dvoch signálov (obrazov), čo má veľký praktický význam.

Konvolučná veta – hovorí o tom, že medzi konvolúciou a násobením existuje dualita, čiže konvolúciu dvoch obrazov môžeme nahradieť súčinom ich fourierovských spektier s následnou spätnou transformáciou. Analogicky, konvolúcia vo Fourierovej oblasti je ekvivalentom súčinu obrazov.

$$\begin{aligned} F\{f * h\}(x, y) &= F(u, v)H(u, v) \\ F\{f(x, y)h(x, y)\} &= (F * H)(u, v) \end{aligned} \quad (23)$$

Túto vlastnosť vieme s výhodou využiť na zefektívnenie výpočtov, keďže násobenie je výpočtovo oveľa jednoduchšou operáciou ako konvolúcia. Výhodné je to hlavne v prípade, keď oba obrazy majú porovnateľnú veľkosť a keď máme k dispozícii efektívny algoritmus Fourierovej transformácie (priamej i spätej).

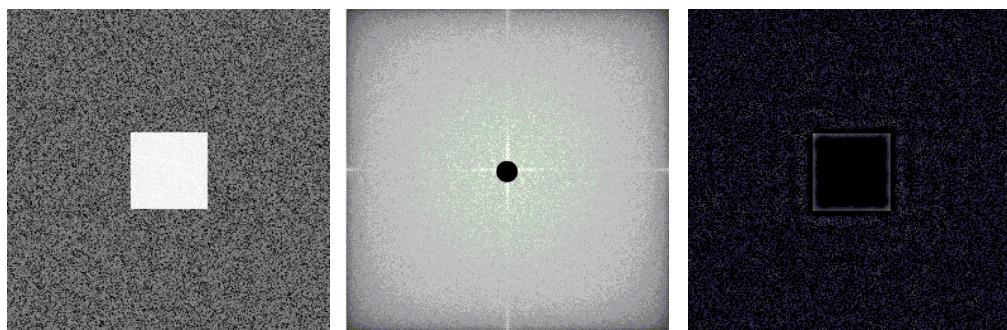
3.2.7 Filtrovanie pomocou Fourierovských spektier

Kedže každý obraz vieme transformovať do Fourierovej oblasti a späť, môžeme to s výhodou využiť na úlohy filtrovania. Vieme, že v spektrálnej oblasti je stred obrazu tvorený jednosmernou zložkou a čím ďalej od stredu pixel je, tým vyššiu frekvenčnú zložku predstavuje. Teda ak napr. vo Fourierovej oblasti ponecháme hodnotu iba tých pixelov, ktoré sú v určitej vzdialosti od stredu a všetky ostatné nastavíme na nulu, bude to predstavovať analógiu dolnopriepustného filtra.

Na základe tohto môžeme vytvoriť nasledovné typy filtrov

- Dolná pásmová priepust
- Horná pásmová priepust
- Pásmová priepust

Obrázok 14 znázorňuje aplikáciu nízkofrekvenčného filtra, kde sa vo výstupnom obraze nachádzajú iba vysoké frekvencie, čo sa prejaví zvýraznením hrán a potlačením nízkych frekvencií.

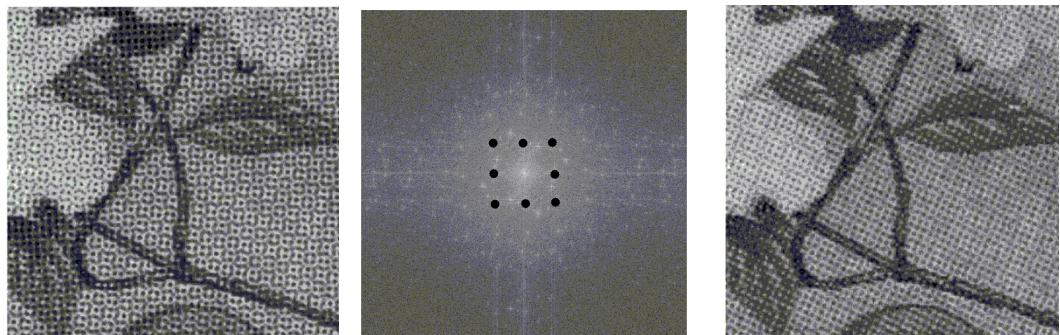


Obrázok 14. Filtrovanie vo frekvenčnej oblasti. Vstupný obraz (vľavo) sa transformuje do Fourierovej oblasti (vstrede). Odstránením (zamalovalaním) nízkych frekvencií okolo stredu sa po spätej transformácii do výstupného obrazu (vpravo) dostanú iba vyššie frekvencie zvýrazňujúce hrany.

3.2.8 Filtrovanie periodického šumu

Obzvlášť výhodné je filtrovanie v spektrálnej oblasti vtedy, keď potrebujeme odstrániť šum periodického charakteru, akým je napr. šum superponovaný na signál zo siete 50 Hz alebo napr. periodická vzorka papiera, ktorá ostáva viditeľná na tlačenom obraze a pod. Takýto periodický šum je v spektrálnej reprezentácii reprezentovaný bodom (v praxi malou oblasťou) s výrazne odlišným jasom. Potom stačí takýto objekt odstrániť, resp. priradiť mu rovnaký jas, ako má okolie a po spätej transformácii bude periodický šum odfiltrovaný. Na eliminovanie oblasti s odlišným jasom

v spektrálnej oblasti môžeme použiť interaktívnu techniku (premaľujeme svetlý bod na čierne) alebo akékoľvek techniky z oblasti spracovania obrazu (napr. top-hat filter).



Obrázok 15. Potlačenie periodického šumu. Na obrázku vľavo je obraz s výrazným periodickým šumom, ktorý sa premietne do výrazných bodov okolo stredu obrazov. Odstránením (zamaľovaním) týchto bodov (obrázok v strede) sa po spätej Fourierovej transformácii periodický šum zredukuje (vpravo).

4 Predspracovanie obrazu

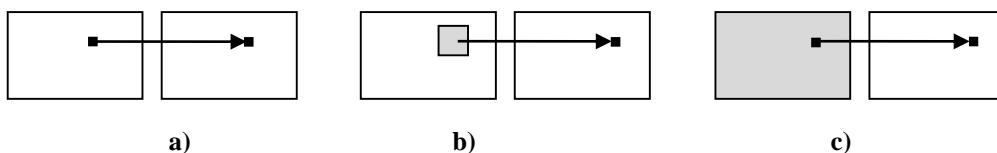
Slúži na zlepšenie obrazu z hľadiska ďalšieho spracovania. Vstupom aj výstupom je obraz na nízkej úrovni abstrakcie. Využíva sa pritom značná nadbytočnosť údajov v obraze. Napríklad môžeme opraviť pixel skreslený šumom na základe hodnôt jeho susedov. Predspracovanie nezvyšuje množstvo informácie z hľadiska Shannonovej teórie, môže iba nejakú informáciu zvýrazniť alebo potlačiť.

4.1 Rozdelenie metód predspracovania:

1. Bodové jasové transformácie
2. Geometrické transformácie
3. Lokálne predspracovanie (filtrácia, ostrenie a detekcia hrán)
4. Obnovenie obrazu pri známej degradácii
5. Matematická morfológia

Z hľadiska veľkosti okolia prispievajúceho k transformácii vstupného bodu na výstupný poznáme nasledovné typy operácií (Obrázok 16):

1. bodové (výstupný bod je transformáciu hodnoty jediného vstupného bodu)
2. lokálne (výstupný bod je tvorený hodnotou vstupného bodu a jeho lokálneho okolia)
3. globálne (na hodnote výstupného bodu sa podieľajú všetky body vstupného obrazu)



Obrázok 16. Typy operácií používaných pri predspracovaní obrazu.

- a) bodové: hodnota výstupného pixelu závisí iba od hodnoty vstupného pixelu,
b) lokálne: hodnota výstupného pixelu závisí od hodnoty vstupného pixelu a jeho okolia,
c) globálne: hodnota výstupného pixelu závisí od hodnôt všetkých pixelov vstupného obrazu.

4.2 Bodové jasové transformácie

1. jasová korekcia
2. zmena jasovej stupnice
3. rozsah
4. tvar
5. gamma korekcia
6. vyrovnávanie histogramu

4.2.1 Jasová korekcia

Používa sa v prípade, keď snímacie zariadenie má odlišnú citlivosť na rôznych miestach obrazu. Podobný efekt má i nerovnomerné osvetlenie snímaného obrazu. Pokiaľ je táto porucha systematická, poznáme odchýlku citlivosti každého bodu od ideálnej charakteristiky, a teda ju vieme aj korigovať.

Najčastejšie predpokladáme porušenie obrazu multiplikatívnym koeficientom $e(i,j)$. Pri nedokonalom snímaní potom pre každý bod $g(i,j)$ získame na výstupe skreslený bod $f(i,j)$.

$$f(i, j) = e(i, j) \cdot g(i, j) \quad (24)$$

Hodnoty transformácie $e(i, j)$ vieme získať tak, že pri stálych snímacích podmienkach zosnímame obraz, ktorý má známy priebeh jasovej funkcie $g(i, j)$. Čiže ak snímame obraz s konštantným jasom c , po jeho nasnímaní získame deformovaný obraz $f_c(i, j)$. Potom môžeme systematické chyby snímacieho reťazca korigovať podľa vzťahu:

$$g(i, j) = \frac{f(i, j)}{e(i, j)} = \frac{c \cdot f(i, j)}{f_c(i, j)} \quad (25)$$

Táto korekcia platí pre stále snímacie podmienky pokiaľ to nie je zabezpečené, je treba korekčnú transformačnú maticu získať kalibráciou pred každým snímaním.

Pokiaľ by predošlý výpočet poskytol hodnoty mimo povolený rozsah, buď použijeme najbližšiu povolenú krajnú hodnotu, alebo uskutočníme zhustenie jasovej funkcie tak, aby sa do nej vypočítaná hodnota zmestila.

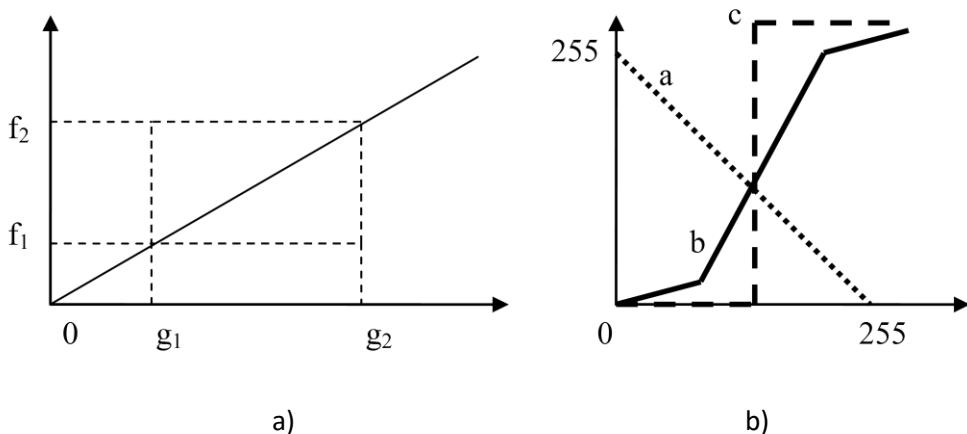
4.2.2 Modifikácia jasovej stupnice

Na rozdiel od jasovej korekcie, modifikácia stupnice nezávisí na polohe bodu v obraze. Táto modifikácia T prevedie vstupnú jasovú hodnotu na výstupnú a to pomocou známej funkcie (tabuľky).

4.2.2.1 Rozsah a tvar jasovej stupnice

Obrázok 17a zobrazuje transformáciu rozsahu jasových úrovní. Pritom g_1 a g_2 ohraňčujú rozsah vstupných hodnôt, f_1 a f_2 rozsah výstupných hodnôt. Jasová stupnica je pritom lineárna, pre ktorú platí:

$$f = \frac{g_2 - g_1}{f_2 - f_1} (g - g_1) + f_1 \quad (26)$$



Obrázok 17. Transformácia rozsahu jasovej stupnice. Niektoré typické transformácie jasovej funkcie a-negatív, b-zvyšovanie kontrastu, c-prahovanie.

Obrázok 17b zobrazuje niektoré typické funkcie používané na transformáciu jasu (negatív, zvyšovanie kontrastu, prahovanie).

V praxi sa často používajú nelineárne transformácie majúce tvar rôznych kriviek. Najčastejšie sa technicky realizujú pomocou vyhľadávacej tabuľky (**lookup table**), ktorá má toľko prvkov, koľko je hodnôt jasu. Vstupná hodnota je daná poradovým číslom a výstupná (transformovaná) hodnota je obsahom takto indexovaného pamäťového miesta.

Najčastejšie používanou nelineárnu funkciou je **Gama korekcia**, ktorá je daná rovnicou

$$f = g^{\frac{1}{\gamma}} \quad (27)$$

Podobným spôsobom je možné uskutočniť transformáciu farebných hodnôt v **palete farieb**.

Pseudofarebný obraz dostaneme vtedy, keď v takejto lineárnej šedotónovej palete nedodržíme zásadu R=G=B=index a niektoré zložky potlačíme. Tým môžeme napríklad farebne zvýrazniť určity interval jasov.

4.2.2.2 OpenCV

funkcia `addWeighted` umožní vážený súčet dvoch obrazov, čo umožní realizovať napr. jasové korekcie.

4.2.3 Vyrovnanie (ekvalizácia) histogramu a kumulatívny histogram

Nepriaznivá situácia z hľadiska vizuálneho vnemu nastáva vtedy, keď objekty nášho záujmu sú tvorené pixelmi s veľmi podobným jasom, čo je okom ľahko rozlíšiteľné. Preto je výhodné je transformovať pixely reprezentujúce lokálne maximum histogramu tak, aby sme využili viac jasových úrovní („rozťiahneme“ a „sploštíme“ pík). Naopak, viacero jasových úrovní, ktoré sú málo zastúpené, môžeme zlúčiť do jednej. Tento proces sa volá ekvalizácia (vyrovnanie) histogramu. Príbeh „Čudné terče. (Ekvalizácia histogramu)“ sa to snaží demonštrovať na praktickom príklade zo života.

Nech $\langle p_0, p_k \rangle$ je interval jasov vo vstupnom obraze a $H(p)$ nech je histogram vstupného obrazu. Hľadáme monotónnu jasovú transformáciu $q = T(p)$ takú, aby výstupný histogram $G(q)$ bol rovnomerný pre celý výstupný interval jasov $\langle q_0, q_k \rangle$.

Kedže T má byť monotónna, platí:

$$\sum_{i=0}^k G(q_i) = \sum_{i=0}^k H(p_i) \quad (28)$$

Obe strany tejto rovnice by bolo možné považovať za distribučné funkcie diskrétneho rozdelenia. Pre obraz NxN pixelov s histogramom $G(q)$ tomu zodpovedá konštantná hustota pravdepodobnosti s hodnotou g_c

$$g_c = \frac{N^2}{q_k - q_0} \quad (29)$$

Pravú stranu rovnice (29) dosadíme za ľavú stranu rovnice (28). Ďalej použijeme abstrakciu, že histogram má spojité rozdelenie (iba vtedy vieme totiž získať ideálny ekvalizovaný histogram), V tom prípade rovnicu (28) môžeme prepísať nasledovne:

$$N^2 \int_{q_0}^q \frac{1}{q_k - q_0} ds = \frac{N^2 (q - q_0)}{q_k - q_0} = \int_{p_0}^p H(s) ds \quad (30)$$

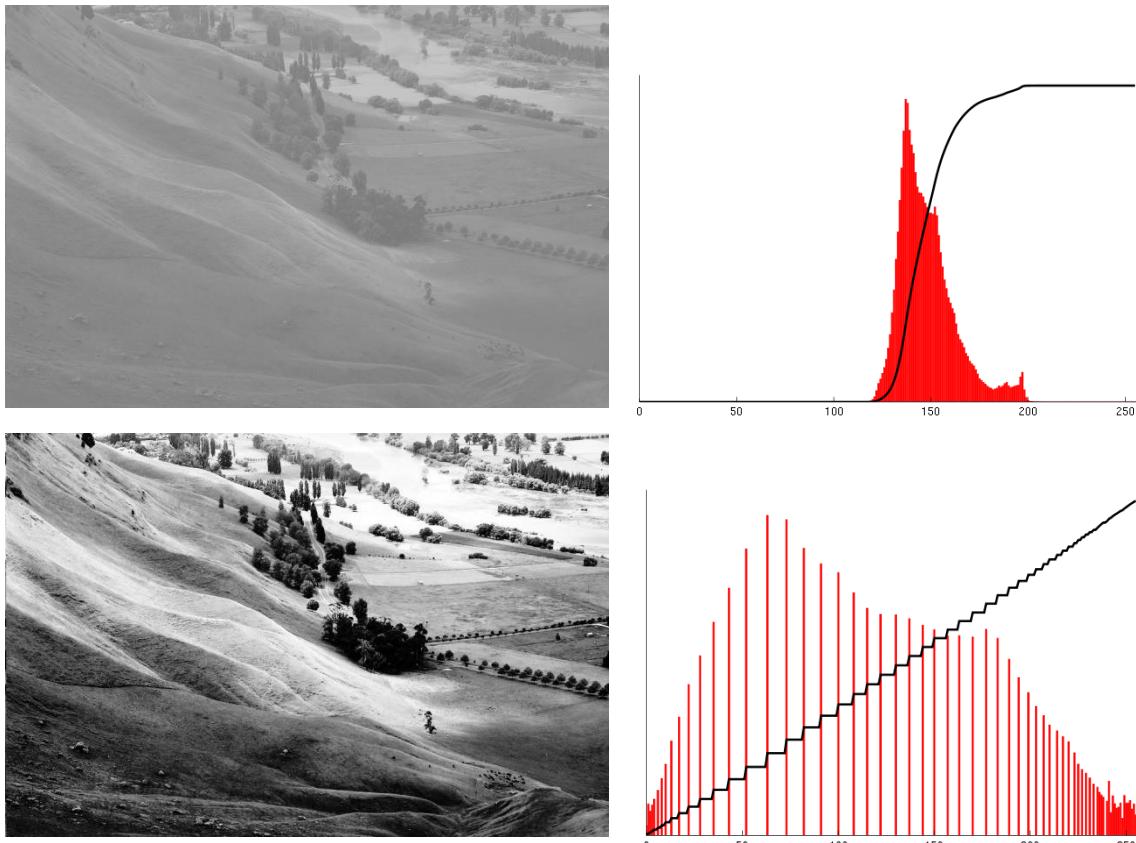
Potom výsledná jasová transformácia bude:

$$q = T(p) = \frac{q_k - q_0}{N^2} \int_{p_0}^p H(s)ds + q_0 \quad (31)$$

pričom integrál v tejto rovnici voláme **kumulatívny histogram**. Po prepísaní do diskrétneho tvaru:

$$q = T(p) = \frac{q_k - q_0}{N^2} \sum_{i=p_0}^p H(i) + q_0 \quad (32)$$

pričom samozrejme po nahradení integrálu sumou nikdy nedostaneme ideálne rovný histogram, iba jeho aproximáciu.



Obrázok 18. Príklad vyrovnávania histogramu. Hore – pôvodný obraz a jeho histogram. Dole – upravený obraz a jeho histogram upravený vyrovnávaním.

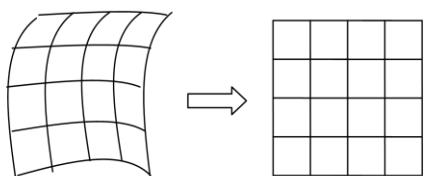
Zdroj: https://en.wikipedia.org/wiki/Histogram_equalization.

Obrázok 18 Chyba! Nenašiel sa žiadен zdroj odkazov. znázorňuje praktický dôsledok vyrovnávania histogramu pri ktorom došlo k roztiahnutiu jasovej škály najtmavších pixelov, čo viedie k lepšiemu vizuálnemu odlíšeniu detailov obrazu krajiny.

4.2.3.1 OpenCV

void equalizeHist(InputArray src, OutputArray dst) generuje obraz dst ktorý má oproti obrazu src vyrovnaný histogram.

4.3 Geometrické transformácie



Obrázok 19. Geometrická transformácia súradníc v rovine.

Geometrická transformácia 2D obrazu je vektorová funkcia T , ktorá zobrazí bod (x, y) do bodu (x_0, y_0) . Situáciu ilustruje Obrázok 19, kde je časť roviny transformovaná bod po bode. Môžeme si to predstaviť ako vyrovnávanie deformovanej mriežky nakreslenej na elastickej podložke. Hľadáme teda takú kombináciu pôsobiacich síl (t.j. parametrov transformačnej funkcie), aby sa kresba na elastickej podložke čo najviac podobala pravidelnej mriežke. Príbeh „Výhľad z basy. (Geometrické transformácie)“ je jedným z rukolapných príkladov.

Geometrické transformácie vypočítajú na základe súradníc bodov vo vstupnom obraze súradnice bodov vo výstupnom obraze. Nejedná sa teda o transformáciu hodnoty funkcie, ale jej súradníc. Vďaka tomu môžeme odstrániť geometrické skreslenie vzniknuté pri snímaní obrazu (napr. korekcie geometrických porúch objektív kamery, oprava skreslenia družicového snímku spôsobená zakrivením zemegule).

Transformácia T je definovaná dvoma vzťahmi:

$$x_0 = T_x(x, y) \quad y_0 = T_y(x, y) \quad (33)$$

Transformačné rovnice T_x a T_y môžu byť vopred známe (rotácia, posunutie) alebo môžu byť odvodené na základe znalosti vstupného a transformovaného obrazu. Za týmto účelom sa na oboch obrazoch nájdú dvojice zodpovedajúcich si bodov (lícovacie body), ktoré sa na obrázkoch ľahko hľadajú napr. priesčníky vláknitých štruktúr, rohy objektov a pod.

Samotná geometrická transformácia pozostáva z dvoch bodov:

1. Transformácia súradníc. Body vstupného obrazu, ktoré majú celočíselné súradnice sa transformujú do nových pozícií, ktoré nemusia zodpovedať celočíselnému rastru, ale majú podobu reálnych čísel.
2. Aproximácia jasovej funkcie. Hľadáme hodnotu jasu v celočíselných pozíciách a to interpoláciou na základe hodnôt transformovaných súradníc (reálnych čísel), ktoré sú najbližšie.

4.3.1.1 OpenCV

void remap(InputArray src, OutputArray dst, InputArray map1, ...) je najväčšou formou geometrickej transformácie kde map1 udáva transformované hodnoty súradníc src v subpixelovej presnosti (float). Môžeme si ich predstaviť ako deformovanú verziu vyrovnanej mriežky vstupného obrazu.

4.3.2 Transformácia súradníc

Nasledujúca rovnica reprezentuje všeobecný prípad, keď hľadáme na súradnice bodu na výstupnom (transformovanom) obraze. Tento vzťah je zvyčajne aproximovaný pomocou polynómu

$$x_0 = \sum_{r=0}^m \sum_{k=0}^{m-r} a_{rk} x^r y^k \quad y_0 = \sum_{r=0}^m \sum_{k=0}^{m-r} b_{rk} x^r y^k \quad (34)$$

Táto transformácia je lineárna z hľadiska činitelov a_{rk} a b_{rk} . Ak teda máme množinu dvojíc vzájomne si zodpovedajúcich „lícovacích“ bodov (x, y) a (x_0, y_0) , môžeme pomocou metódy najmenších štvorcov určiť koeficienty a_{rk} a b_{rk} . Pre transformácie, ktoré nevykazujú prudké zmeny pozície transformovaných súradníc vystačíme s aproximáciou pomocou polynómov nízkeho stupňa ($m=2$, resp. $m=3$).

Rovnicu (47) veľmi často nahradzujeme **bilineárnu transformáciu**, na ktorú postačujú 4 dvojice vzájomne si zodpovedajúcich lícovacích bodov na vstupnom resp. na transformovanom obraze.

$$\begin{aligned} x_0 &= a_0 + a_1 x + a_2 y + a_3 xy \\ y_0 &= b_0 + b_1 x + b_2 y + b_3 xy \end{aligned} \quad (35)$$

Zvláštnym prípadom bilineárnej transformácie je **afinná transformácia**, ktorá v sebe zahrnuje najčastejšie používané formy transformácie, ako je translácia, rotácia, a škálovanie. Pri tejto transformácii stačia 3 páry lícovacích bodov.

$$\begin{aligned} x_0 &= a_0 + a_1 x + a_2 y \\ y_0 &= b_0 + b_1 x + b_2 y \end{aligned} \quad (36)$$

Použitím iba niektorých koeficientov dostaneme základné najpoužívanejšie typy transformácií

Translácia (koeficienty a_0 a b_0 určujú posun)

$$x_0 = x + a_0 \quad y_0 = y + b_0 \quad (37)$$

Zmena škály (koeficienty a_1 a b_2 určujú koeficienty stlačenia)

$$x_0 = a_1 x \quad y_0 = b_2 y \quad (38)$$

Rotácia (koeficienty $a_2 = \sin \varphi$ a $b_1 = -\cos \varphi$, kde φ je uhol rotácie)

$$x_0 = x + a_2 y \quad y_0 = b_1 x + y \quad (39)$$

Uvedené transformácie je možné vzájomne kombinovať (napr. otočenie posunutého obrazu). V tom prípade je výhodné používať maticový zápis jednotlivých transformácií. Sériu postupných transformácií sa potom dá vyjadriť formou násobenia jednotlivých matíc.

4.3.2.1 OpenCV

`Mat getAffineTransform(InputArray src, InputArray dst)`. Vstupom sú dve trojice vzájomne si odpovedajúcich bodov src a dst . Výstupom je matica M rozmerov 2×3 obsahujúcu koeficienty $a_0, a_1, a_2, b_0, b_1, b_2$.

`void warpAffine(InputArray src, OutputArray dst, InputArray M,...)` realizuje geometrickú transformáciu vstupného obrazu na základe transformačnej maticy M .

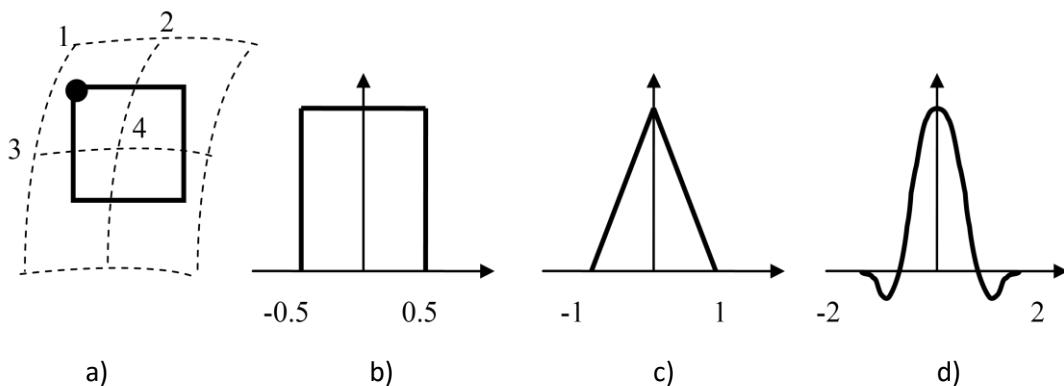
4.3.2.2 Príklady použitia:

Uvedené transformácie sa často používajú pri analýze obrazov z družíc (diaľkový prieskum Zeme), kde je možné takto korigovať napr. rotáciu Zeme počas snímania, alebo zmenu škály pri zmene vzdialenosť od povrchu.

Ďalšou oblasťou, kde je potrebné robiť spomínané operácie je tzv. registrácia obrazov v mikroskopii. Často je potrebné analyzovať častice v určitom objeme, pričom k dispozícii máme tzv. sériové rezy študovaného objemu. To znamená, že sledovaný objekt sa zmrazí, pokrája na tenké plátky a jednotlivé rezy sa sledujú pod mikroskopom. Problém je v tom, že nedokážeme vložiť po sebe idúce rezy presne na to isté miesto, vždy budú vzájomne posunuté a pootočené. Pomocou referenčných bodov potom softwarovo transformujeme snímaný obraz tak, aby presne lícoval s predchádzajúcim. Zvyčajne postačuje operácia rotácie a posuvu, sú však aj zložitejšie metódy registrácie s viacerými stupňami voľnosti.

4.3.3 Aproximácia jasovej funkcie

Geometrickou transformáciou sme vypočítali hodnoty (x_0, y_0) , ktoré zodpovedajú bodom (x, y) originálneho obrazu. Celočíselné koordináty vstupného obrazu však transformáciou nadobudnú hodnoty z oboru reálnych čísel. To znamená, že poznáme úroveň jasu v bodech s reálnymi súradnicami (priesečníky prerušovaných čiar na Obrázok 20a). My však potrebujeme poznáť jas v bodech predstavujúcich celočíselný raster. (štvorec nakreslený plnou čiarou na Obrázok 20a).



Obrázok 20. a) Transformované súradnice, ktorých poloha je daná reálnymi číslami (prerušovaná čiara). Plná čiara predstavuje celočíselný raster, hľadáme jas na mieste označenom krúžkom.

- b) interpolácia metodou najbližšieho suseda (prevzatie hodnoty jasu bodu 1),
c) lineárna interpolácia (hodnotu určuje plôška daná bodmi 1,2,3,4),
d) bi-kubická interpolácia (hodnotu určuje bikubický polynom daný širším okolím 16 bodov).

Je zrejmé, že hodnoty jasu v bodech predstavujúcich celočíselný raster získame interpoláciou susedných bodov, ktorých hodnoty poznáme. Je viacero spôsobov, ako túto hodnotu vypočítať.

Obrázok 20b zobrazuje interpoláciu **metódou najbližšieho suseda**, keď jas v hľadanom bode sa určí ako jas najbližšieho transformovaného suseda, ktorého jas je známy. Pri **lineárnej interpolácii** sa berie do úvahy lineárna kombinácia jasov štyroch najbližších transformovaných susedov (1,2,3,4 pre zvýraznený bod na Obrázok 20a). Bližší susedia majú väčší vplyv.

Pri **bikubickej interpolácii** je táto hodnota získaná aproximáciou 16 najbližších susedov, ich vplyv určuje interpolačné jadro v tvare „Mexického klobúka“ (Obrázok 20d).

4.4 Predspracovanie pomocou lokálnych operátorov

Lokálne operátory pracujú tak, že algoritmus systematicky (zvyčajne po riadkoch) prechádza celý obraz a upraví hodnotu každého pixelu na základe hodnoty pixelov v jeho bezprostrednom okolí. Podľa účelu môžeme prácu lokálnych operátorov rozdeliť do dvoch skupín:

Vyhľadzovanie (filtrácia). Účelom vyhľadzovania je odstrániť z obrazu šum a ostré hrany, čiže vysokofrekvenčné zložky v zmysle Fourierovského spektra. Neželaným sprievodným javom tohto procesu je aj rozmazanie hrán, ktoré boli na pôvodnom obraze ostré.

Detekcia hrán (ostrenie). Jeho účelom je, naopak, zvýrazniť vysokofrekvenčné časti spektra (hrany) a odfiltrovať tie časti obrazu, kde dochádza iba k pomalým zmenám. Neželaným sprievodným javom detekcie hrán je aj zvýraznenie šumu, ktoré má tiež vysokofrekvenčný charakter.

Oba zdanivo protichodné postupy majú rovnakú matematickú podstatu založenú na konvolúcii. Príbeh „Konvolučné družstvo (lokálne operátory)“ v zjednodušenej podobe vysvetľuje podstatu.

Konvolúcia počíta hodnotu výstupného pixelu ako lineárnu kombináciu hodnôt pixelov v malom okolí zodpovedajúceho pixelu vstupného obrazu. Príspevok jednotlivých pixelov tohto okolia O je daný koeficientami, ktoré obsahuje konvolučná maska h . Táto má zvyčajne rovnaký a pritom nepárný počet riadkov i stĺpcov (3x3, 5x5 ... a pod.).

$$f(x, y) = \sum_{n,m \in O} h(x-m, y-n) g(m, n) \quad (40)$$

Okrem spomínaných lineárnych metód existujú aj nelineárne, ktoré sa snažia filtrovať obraz bez rozmazania hrán, čo však nie je triviálny problém.

Pretože malé okolie spracovávaného pixelu nedovoľuje jeho komplikovanejšiu analýzu, je predspracovanie iba zriedkavo založené na znalostiach obsahu obrazu.

Operácie založené na konvolúcii sú časovo pomerne náročné, existujú však postupy, ako ich urýchliť a to aj za pomoci špecializovaného hardvéru.

4.4.1 Filtrácia pomocou lokálnych operátorov

Cieľom filtrácie je odstránenie šumu. Vo frekvenčnej oblasti to predstavuje potlačenie vysokých frekvencií, keďže šum sa zvyčajne prejavuje ako vysokofrekvenčná zložka.

Predpokladáme, že náhodné veličiny popisujúce aditívny šum v jednotlivých bodoch sú nezávislé. Rozloženie hodnôt šumu má nulovú strednú hodnotu μ a smerodajnú odchýlku σ .

Zosnímajme ten istý obrázok viac krát po sebe. Vyberme si niektorý bod a označme ho indexom, podľa toho, do ktorého obrazu v sekvencii patrí. Dostaneme postupnosť g_1, g_2, \dots, g_n , pričom šum

v týchto bodech je popísaný náhodnými veličinami v_1, v_2, \dots, v_n . Odhad správnej hodnoty jasu bodu g potom môžeme vyjadriť aritmetickým priemerom

$$\frac{g_1 + g_2 + \dots + g_n}{n} + \frac{v_1 + v_2 + \dots + v_n}{n} \quad (41)$$

Druhý sčítanec vo výraze predstavuje náhodnú veličinu s nulovou strednou hodnotou a so smerodajnou odchýlkou rovnou σ/\sqrt{n} .

4.4.1.1 OpenCV

Operácie sumovania a delenie konštantou sa realizujú pre celé obrazy veľmi jednoducho pomocou operátorov "+" a "/". Vďaka optimalizovaným knižniciam sa tieto nízkoúrovňové operácie realizujú rýchlo vďaka čomu je minimálne spozdenie toku obrazov snímaných kamerou.

4.4.1.2 Spriemerovanie.

Je najjednoduchším prípadom vyhladzovania šumu na obrate. V tomto prípade nemáme k dispozícii postupnosť obrazov tej istej statickej scény, ale iba jeden obraz, spoľahneme sa ale na značnú redundanciu obrazu. Vďaka nej môžeme predpokladať, že susedné body majú ten istý, alebo veľmi podobný jas.

Pre každý bod obrazu sa jeho jas nahradí aritmetickým priemerom jasov jeho susedov. Zvyčajne sa za susedov považujú body zo štvorcového nepárneho okolia bodu (3x3, 5x5, ... a pod.). Filtrácia obyčajným spriemernením je vlastne špeciálnym prípadom konvolúcie, kde konvolučná maska má nasledovný tvar:

$$h = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (42)$$

4.4.1.3 Filtrovanie pomocou Gaussovej masky

Gaussovú masku dostaneme, keď v konvolučnej maske posilníme význam stredového bodu, prípadne aj jeho 4-susedov, čím docielime, aby maska lepšie aproximovala vlastnosti šumu s Gaussovým rozdelením.

$$h = \frac{1}{10} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad h = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad (43)$$

Rozmazávanie hrán je to prirodzeným sprievodným javom vyhladzovania pomocou obyčajného spriemerovania.

4.4.1.4 Mediánový filter

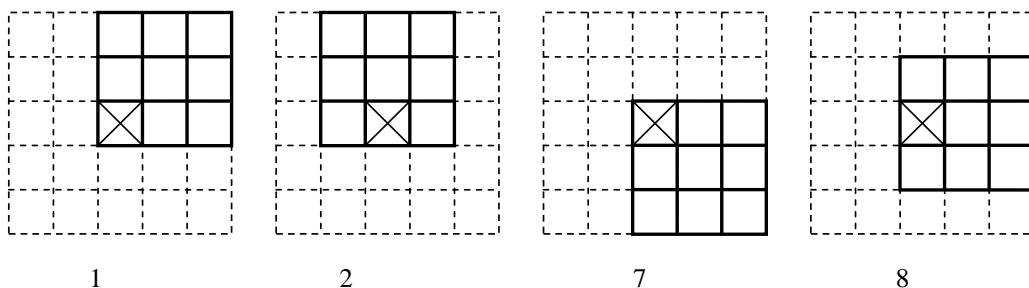
Medián ako alternatíva aritmetického priemeru je výhodná vtedy keď chceme eliminovať vplyv extrémov (viď príbeh „Vojak Median (Mediánový filter)“).

Majme náhodnú veličinu x. V teórií náhodných premenných je medián M definovaný ako hodnota, pre ktorú je pravdepodobnosť javu $x < M$ rovná $\frac{1}{2}$. Výpočet mediánu je veľmi jednoduchý – usporiadame jasové úrovne z lokálneho okolia vzostupne a ako mediánovú hodnotu vyberieme jas

v strede tejto postupnosti. To zabráni, aby extrémy na krajoch postupnosti ovplyvnili vybranú hodnotu, tak ako je tomu u spriemernenia. Touto hodnotou nahradíme jas filtrovaného pixelu. Táto metóda filtrácie veľmi dobre redukuje impulzný šum a je pomerne šetrná k hranám. Bohužiaľ, mediánová filtrácia sa nie celkom dobre hodí v prípade tenkých čiar a ostrých rohov v obraze, ktoré poruší.

4.4.1.5 Filtrácia metódou rotujúcej masky

Vychádza z predpokladu, že sa bude spriemerovať tá časť okolia bodu, ku ktorej bod pravdepodobne prináleží. Z okolia podielajúceho sa na spriemerovaní budú teda vylúčené body ležiace na hrane. Okolie teda musí byť homogénne, čo sa dá vyšetriť na základe rozptylu jasu pixelov.



Obrázok 21. Filtrovanie obrazu pomocou rotujúcej masky

Obrázok 21 zobrazuje 8 prípadov, keď maska 3×3 , ktorá rotuje na pozadí okna 5×5 . Bod označený krížikom nadobudne hodnotu jasu rovnú aritmetickému priemu „najlepšej“ masky 3×3 . Za najlepšiu masku považujeme tú, ktorá má v okne 3×3 najväčšiu homogenitu jasov.

4.4.1.6 OpenCV

knižnica obsahuje veľké množstvo preddefinovaných jednoúčelových filtrov `boxFilter`, `bilateralFilter`, `GaussianBlur`, `medianBlur`. Okrem toho je možné použiť užívateľom definované konvolučné masky (`Smooth`, `createGaussianFilter`, `createLinearFilter...`). Pre návrh špeciálnych typov filtrov slúži trieda `FilterEngine`.

4.4.2 Detekcia hrán a ostrenie obrazu

Hrana v obraze je vlastnosť obrazového elementu a jeho okolia. **Hrana** je vektorová veličina, ktorá je určená **veľkosťou a smerom**.

Je treba odlišiť detekciu hrán od hľadania hraníc. **Detekcia hrán** je jednoduchou operáciou založenou zvyčajne na lokálnych operátoroch. Naproti tomu **hľadanie hraníc** je sofistikovanejšou činnosťou (patriacou do oblasti segmentácie obrazov), ktorá môže byť založená na detekcii hrán, avšak využíva znalosti objektov na to, aby jednotlivé úseky hrán správne spojila do hranice.

Základnou myšlienkou detekcie hrán je hľadanie zmeny funkcie jasu $f(x,y)$ v obraze.

Mierou indikácie hrany je určenie **veľkosti (sily) hrany**. Smer hrany v určitom bode obrázku sa nazýva **orientácia hrany**. Tieto hodnoty sú vypočítané pomocou diskrétnych derivácií funkcie jasu. Pri

hranovej detekcii sa využíva vlastnosť gradientu, že hodnota gradientu funkcie dvoch premenných je v oblasti hrany najväčšia. Gradient dvojrozmernej skalárnej funkcie $f(x, y)$ v bode so súradnicami (x, y) je vektorová funkcia definovaná vzťahom

$$G[f(x, y)] = s_x \frac{\partial f(x, y)}{\partial x} + s_y \frac{\partial f(x, y)}{\partial y} = s_x G_x + s_y G_y \quad (44)$$

kde s_x je jednotkový vektor v smere osi x a s_y jednotkový vektor v smere osi y .

Vektor $G[f(x, y)]$ je orientovaný v smere maximálnej strnosti funkcie $f(x, y)$ v bode (x, y) , maximálna rýchlosť zmeny hodnoty funkcie $f(x, y)$ na jednotku vzdialosti je v smere vektora G v bode (x, y) a je daná absolútou hodnotou vektora, t.j.

$$\|G[f(x, y)]\| = \sqrt{G_x^2 + G_y^2} \quad (45)$$

Smer vektora gradientu v bode (x, y) vzhľadom na os x je:

$$\alpha(x, y) = \arctan \frac{G_y}{G_x} \quad (46)$$

Pri detekcii hrán sa najčastejšie uvažuje s absolútou hodnotou gradientu (6), ktorá sa pre jednoduchosť označuje ako gradient, pričom sa spravidla určuje jeho aproximácia v tvare

$$|G[f(x, y)]| = G_c(x, y) \approx |G_x| + |G_y| \quad (47)$$

Vzhľadom na výpočet parciálnych derivácií vyšších rádov možno využiť aj ďalšie princípy pre charakterizovanie dvojrozmernej skalárnej funkcie, a to predovšetkým Laplaceov operátor, pre ktorý platí:

$$L[f(x, y)] = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2} = G_{2x} + G_{2y} \quad (48)$$

Vzhľadom na diskrétnu realizáciu výpočtu gradientu, resp. Laplaceovho operátora je nutné príslušné parciálne derivácie approximovať **diferenciami** v určitom okolí bodu (i, j) digitalizovaného obrazu. Najčastejšie sa uvažuje okolie 3×3 bodov. Pomocou diferencií je potom možné approximácie jednotlivých operátorov zapísat v tvare konvolučných masiek.

4.4.2.1 Laplacov operátor

$$h_4 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad h_8 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (49)$$

tento operátor approximuje druhú deriváciu, udáva veľkosť hrany a je necitlivý na smer. Všimnime si, že súčet hodnôt v konvolučnej maske je rovný 0.

4.4.2.2 Robertsov operátor

$$h_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad (50)$$

pričom veľkosť gradientu sa vypočíta nasledovne:

$$|G[f(x, y)]| = |g(i, j) - g(i+1, j+1)| + |g(i, j+1) - g(i+1, j)| \quad (51)$$

Sobelov operátor approximuje prvé parciálne derivácie. Pre každý smer existuje zvláštna maska, napr. pre dva smery

$$h_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \quad h_2 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \quad (52)$$

Bez znalosti štatistických vlastností obrazu sa však nedá dopredu povedať, ktorá maska bude lepšia. Veľmi často sa vhodný výber masiek určuje pokusmi.

4.4.2.3 OpenCV

K dispozícii sú špecializované funkcie určené pre jednotlivé typy operátorov sú Sobel, Scharr, Laplacian, ... Okrem toho je možný návrh vlastných filtrov pomocou konvolučnej masky tak ako to bolo spomínané pri vyhľadzovaní.

5 Klasické metódy segmentácie obrazov

5.1 Úvodné poznámku ku segmentácii

Segmentáciu môžme chápať ako rozdelenie obrázku na časti, ktoré korelujú s objektmi reálneho sveta. Obrázok 22 zobrazuje situáciu, keď objektom môže byť napr. celá katedrála alebo aj jej časti (strecha, veža, steny). Z tohto hľadiska je pojmom objekt a teda aj samotný cieľ segmentácie **subjektívnym pojmom**.

Často sa to definuje ako závislosť na úlohe i obraze: „Segmentation is task dependent and image dependent“. Jedným zo sprievodných javov (cieľov) segmentácie je výrazná **redukcia objemu spracovávaných dát**.



Obrázok 22. Subjektívny charakter cieľa segmentácie.

Objektom je celý chrám, resp. lampa a nie jej jednotlivé časti. Ako objekty vnímame aj 2 veľké trojuholníky, z ktorých žiadny nie je reprezentovaný celistvou hranou.

5.1.1 Definícia segmentácie.

Napriek subjektívному charakteru segmentácie je zrejmé, že potrebujeme objektívnejšiu definíciu a preto budeme v ďalšom definovať segmentáciu nasledovne:

Nech R označuje oblasť (celý obraz) a nech H je predikát pre dvojhodnotové ohodnenie homogeneity oblasti. Potom *segmentácia* je definovaná ako rozdelenie oblasti R do M podoblastí R_1, R_2, \dots, R_M , takých, že:

$$\begin{aligned} \bigcup_{m=1}^M R_m &= R \\ R_m \cap R_l &= \emptyset \quad \text{pre } m \neq l \\ H(R_m) &= \text{TRUE} \\ H(R_m \cup R_l) &= \text{FALSE} \quad \text{pre } m \neq l \end{aligned} \tag{53}$$

Z tejto definícii vyplýva, že rozdelíme obraz na vzájomne disjuktné homogénne časti tak, že ich zjednotením je celý pôvodný obraz. Každá z podoblastí musí splňať kritérium homogeneity, pričom spojením ľubovoľných dvoch podoblastí by toto kritérium bolo porušené.

Kritérium homogeneity môže zohľadňovať úroveň jasu, vzorku textúry, rýchlosné pole a pod. Môže to byť jedna hodnota alebo vektor hodnôt (vtedy sa jedna o problém klasifikácie v n-rozmernom priestore).

5.1.2 Úroveň segmentácie

- kompletívna – ak segmentované objekty priamo zodpovedajú hľadaným objektom reálneho sveta. Zvyčajne sa pri tom využíva spracovanie na vyššej úrovni spojené so znalosťou vlastností objektov.
- čiastočná – segmentovane časti obrazu nie sú priamo hľadanými objektami, ale ich časťami (treba ich ďalej spracovávať).

5.1.3 Rozdelenie metód

- globálne (napr. prahovanie)
- určovanie hraníc (detekcia hrán)
- určovanie oblastí (napr. metódy založené na narastaní oblastí)

5.2 Prahovanie

Prahovanie je najjednoduchší spôsob segmentácie. Vychádza z predpokladu, že veľa objektov a im zodpovedajúcich oblastí obrazu je charakterizovaných konštantnou odrazivosťou, či pohltivosťou svetla na svojom povrchu. Ak obraz obsahuje dostatočne kontrastné objekty vzhľadom na pozadie, je možné použiť určitú jasovú úroveň tzv. **prah** k ich vzájomnému oddeleniu. Prah môže byť buď globálny (rovnaký pre celý obraz), alebo lokálny (závislý od pozície na obraze).

Prahovanie je najpoužívanejšou a z pohľadu výpočtovej nenáročnosti aj najrýchlejšou segmentačnou metódou. Možno ju definovať ako priradenie

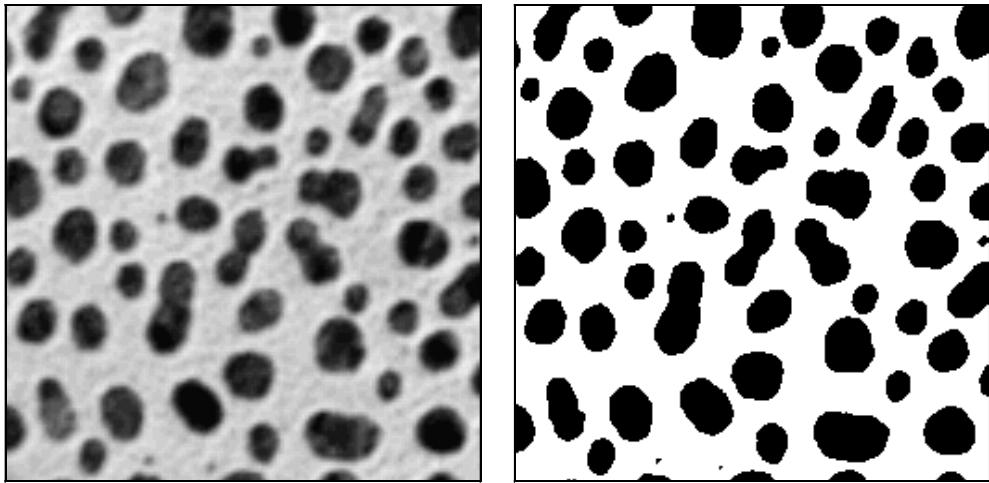
$$f(i, j) = d, \text{ ak } t_{d-1} \leq f(i, j) < t_d; d = 1, 2, \dots, D \quad (54)$$

kde t_d je d -tý prah a $f(i, j)$ je segmentovaná hodnota v bode obrazu (i, j) . Vo všeobecnosti môže mať hodnota prahu zložitejšiu závislosť na parametroch obrazu, napr.

$$t_d = t_d\{(i, j), o(i, j), f(i, j)\} \quad (55)$$

Takýto prah sa nazýva lokálny **dynamický prah**. Ak prah t_d nie je závislý na súradniciach bodu obrazu, nie je dynamický. Ak nie je závislý na okolí bodu $o(i, j)$, potom prah nie je lokálny. Najčastejšie používaným prípadom je **globálny prah**, kedy prah t_d závisí len na intenzite jasu obrazu.

Na vyjadrenie zastúpenia jednotlivých jasových úrovní v obraze sa často používa **histogram**.



Obrázok 23. Pôvodný šedotónový obrázok a jeho binárny obraz získaný prahovaním globálnym prahom

Globálny prahový výber je vhodný pre aplikácie, v ktorých sa objekty diametrálne líšia svojimi jasovými charakteristikami a jeho špeciálnym prípadom je, keď $d = 1$. V takomto prípade vzniká tzv. binárna reprezentácia obrazu (binárny obraz - viď Obrázok 23b). Odlišnosť objektu od pozadia v prípade ich rozdielnych jasových charakteristík sa prejavuje napr. v tom, že takýto obraz má bimodálny histogram a prah t možno určiť ako hodnotu jasovej úrovne u , v ktorej má histogram lokálne minimum. Pre **binárny obraz** potom platí

$$g(i, j) = \begin{cases} 1 & \text{pre } f(i, j) \geq u \\ 0 & \text{pre } f(i, j) < u \end{cases} \quad (56)$$

pričom $g(i, j) = 1$ pre elementy patriace po segmentácii objektu a $g(i, j) = 0$ pre elementy pozadia (alebo môže to byť aj naopak).

Len málokedy je možné úspešne prahovať s tým istým prahom na celej ploche obrazu. Je to spôsobené zmenami jasu objektu a pozadia zavinenými napr. nerovnomernosťou osvetlenia či nerovnakými vlastnosťami snímacieho zariadenia na ploche obrazu. Vtedy je možné použiť už spomínany lokálny, prípadne lokálny dynamický prah. Inou modifikáciou je **prahovanie s viacerými prahmi**, keď výsledkom už nie je binárny obraz, ale obraz s obmedzeným počtom jasových úrovní. Tento prístup je možné aplikovať aj v prípade, keď samotné objekty majú navzájom rozdielny jas. Potom platí

$$g(i, j) = \begin{cases} 1 & \text{pre } f(i, j) \in U_1 \\ 2 & \text{pre } f(i, j) \in U_2 \\ \vdots & \\ S & \text{pre } f(i, j) \in U_S \\ 0 & \text{inak} \end{cases} \quad (57)$$

kde U_s , pre $s = 1, 2, \dots, S$ sú podmnožiny jasových úrovní. Toto je možné v zjednodušenom prípade považovať za klasifikáciu objektov do tried. Nevýhodou segmentačných metód založených na prahovaní je, že nie sú vhodné pre zašumené alebo rozmazané obrazy.

Poloprahovanie je modifikáciou predošlého prípadu, keď je odstránené pozadie, ale objekty ostanú v nezmenenej podobe.

$$g(i, j) = \begin{cases} f(i, j) & \text{pre } f(i, j) \in D \\ 0 & \text{inak} \end{cases} \quad (58)$$

5.2.1.1 OpenCV

double threshold(InputArray src, OutputArray dst, double thresh, double maxval, int type). Funkcia aplikuje prah *thresh* na vstupný obraz *src*. Výsledkom je prahovaný obraz *dst* pričom spôsob prahovania je určený parametrom *type* (binárne, inverzne binárne, poloprahovanie a pod.).

Funkcia void adaptiveThreshold(...) nepotrebuje zadať globálny prah, nakoľko si odvodí sama lokálny prah na základe vlastnosti obrazu v danom mieste. Využíva na to buď priemer pixelov, alebo Gaussovské rozdelenie hodnôt jasu na danom lokálnom mieste.

5.2.2 Metódy určovania prahu

5.2.2.1 Percentuálne prahovanie.

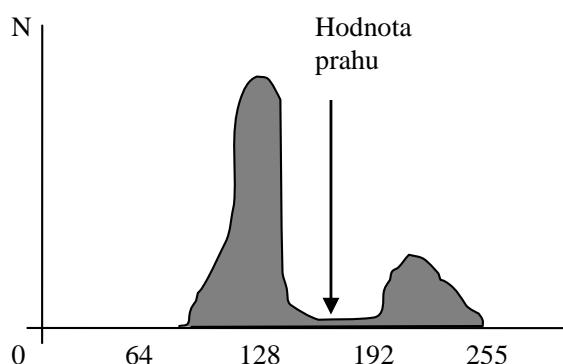
Ak vopred poznáme (máme predikovanú) určitú vlastnosť snímaného obrazu. Vieme napr. že písma pokrývajú 1/p plochy strany. Túto vlastnosť predpokladáme aj u práve spracovanej strany a preto nastavíme prah tak, aby práve 1/p počtu pixelov bolo tmavých.

5.2.2.2 Interaktívny výber prahu

Prah nastaví užívateľ manuálne, pričom pozoruje mieru zhody jednotlivých segmentovaných objektov binárneho obrazu s očakávaným výsledkom.

5.2.2.3 Analýza bimodálneho histogramu.

Histogram vyjadruje zastúpenie jednotlivých jasových úrovní v obraze. **Bimodálny** histogram (Obrázok 24) reprezentuje obraz, na ktorom sú približne rovnako šedé objekty (s jasom cca 160) umiestnené na pozadí, ktoré má približne konštantný jas (zhruba 128). Ak je objektov málo, bude mať pík zodpovedajúci objektom malú amplitúdu a bude ľahšie detektovateľný.



Obrázok 24. Bimodálny histogram a určenie prahu

- nájdeme dve lokálne maximá vzdialené o určitý počet jasových úrovní

- nájdeme minimum medzi týmito maximami, čo je zhodné s hľadaným prahom
- overí sa vierohodnosť získaného prahu (čím plochejší histogram, tým menej dôveryhodný).

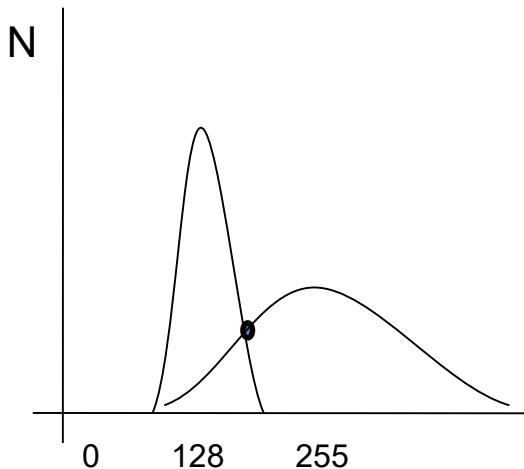
Multimodálny histogram môže byť jednak dôsledkom viacerých typov objektov s rôznym jasom, ale môže byť tiež dôsledkom šumu a nerovnomerného osvetlenia (obraz bez objektov, ktorého jedna polovica je čierna a druhá biela má rovnaký histogram, ako obraz s bielym pozadím a množstvom čiernych objektov). Preto použitie histogramu bez znalosti obrazu negarantuje správnu segmentáciu.

5.2.2.4 OpenCV

GetMinMaxHistValue nájde pozíciu a hodnotu extrémov v danom histograme. To umožní nájsť prah bimodálneho histogramu vhodného na prahovanie.

5.2.3 Optimálne prahovanie.

Miesto histogramu použijeme dve Gaussové krivky vyjadrujúce hustotu pravdepodobnosti, že objekt bude mať určitý jas (Obrázok 25). Hodnota prahu sa potom určí ako hodnota udávajúca minimálnu chybu oboch distribúcií, teda hodnota v priesecníku oboch kriviek. Takáto hodnota síce teoreticky zaručuje minimálnu chybu klasifikácie (optimálna segmentácia), avšak iba za predpokladu, že Gaussova krivka skutočne vyjadruje pravdepodobnosť rozloženia jednotlivých jasových úrovní. Bohužiaľ, v praxi je táto podmienka málokedy splnená a preto aj tzv. optimálne prahovanie nemusí byť ideálne.



Obrázok 25. Bimodálny histogram podľa rozloženia pravdepodobnosti (Gaussové krivky)

5.2.4 Iteratívny algoritmus optimálneho prahovania

Predpokladáme, že na obrázku je okrem relatívne homogénneho pozadia aj niekoľko typov objektov, každý z nich je charakterizovaný určitým jasom. Jasy jednotlivých pixelov v rámci jedného typu objektu sa od tejto priemernej hodnoty odlišujú v zmysle Gaussového rozdelenia. Algoritmus sa pre každý typ objektov snaží nájsť optimálny tvar príslušnej Gaussovej krivky (modelu) tak, aby súčet štvorcov vzdialenosí medzi skutočnými hodnotami histogramu a modelovou krivkou bol minimálny.

$$h_{model}(g) = \frac{\sum_{i=1}^n a_i e^{-(g-\mu_i)^2}}{2\sigma_i^2} \quad (59)$$

kde g predstavuje hodnoty jasu množiny G objektov, a_i , σ_i a μ_i predstavujú parametre Gaussovej distribúcie. Optimálne parametre Gaussovej distribúcie dostaneme minimalizáciou fitovacej funkcie

$$F = \sum_{g \in G} (h_{model}(g) - h_{region}(g))^2 \quad (60)$$

pre dve triedy (objekty a pozadie) hľadáme iba 1 hodnotu prahu nasledovným algoritmom:

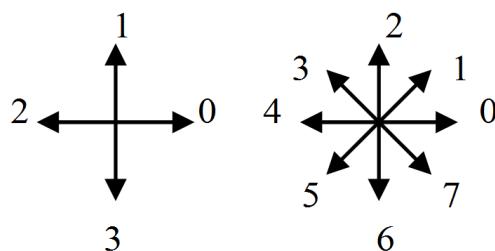
1. Keďže nevieme nič o objektoch, považujeme za pozadie hodnotu jasu bodov v okolí rohov obrazu, ostatné pixely sú považované za objekty.
2. Určíme priemernú hodnotu jasu pre pixely pozadia a pre pixely objektov.
3. Novú hodnotu prahu udáva aritmetický priemer medzi priemerným jasom bodov pozadia a priemerným jasom bodov objektov.
4. Opakujeme 2) a 3) dovtedy, pokiaľ sa nová hodnota prahu líši od predošlej.

5.3 Segmentácia detekciou hrán

Pripomeňme si opäť rozdiel medzi hranou a hranicou (kontúrou). Kým *hrana* je iba vlastnosť pixelu vyjadrujúca jeho gradient a poprípade smer, *hranica* (kontúra) je výsledkom segmentácie a predstavuje obrys segmentovaného regiónu. Teda pixely patriace do hranice musia spĺňať viaceré požiadavky než iba vysoký gradient. Dá sa očakávať, že čím viac segmentačná metóda rešpektuje globálne požiadavky a apíornu informáciu o segmentovanom objekte, tým bude výsledok segmentácie lepší.

5.3.1 Sledovanie vnútornej hranice objektov.

Tento postup používame, keď máme objekty reprezentované v tvare binárneho obrazu. Využíva sa pritom postup, ktorého výsledkom je vnútorná hranica objektov o šírke 1 pixel. Táto forma môže byť považovaná za polygón, z ktorého je možné v ďalšom kroku vylúčiť redundantné body. Nasledovný postup (prípadne príbeh „*Stratený samopal. (Sledovanie vnútornej hranice)*“ popisujú rovnaký algoritmus. Obrázok 26 znázorňuje kódovanie smerov.



Obrázok 26. Kódovanie smerov

Postup

1. Prechádza obraz po riadkoch, až kým nenájde bod patriaci novej oblasti. Označíme bod ako P0, predstavuje prvý bod hranice novej oblasti. Vo zvláštnej premennej SMER uchovávame kód poslednej hodnoty smeru (pre 4-susedstvo smer=3, pre 8-susedstvo smer=7)
2. Prehľadaj okolie aktuálneho bodu 3x3, pričom sa začne v smere:

(smer+3) mod 4 pre 4-susedstvo

(smer+7) mod 8 (8-susedstvo párný smer)

(smer+6) mod 8 (8-susedstvo nepárný smer)

3. Prvý takto nájdený hraničný element sa stáva novým bodom hranice, určí sa nová hodnota smeru. Ak P_i je aktuálny bod hranice, P_1 je druhý bod hranice a platí $P_n = P_1$ a zároveň $P_{n-1} = P_0$, potom ukončí hľadanie hranice, inak opakuj krok 2)
4. Opakuj kroky 1) až 3) pre celý obraz

Algoritmus teda testuje nasledujúci bod v smere pohybu a jeho susedov v smere kolmom na smer pohybu. Prvý z testovaných bodov, ktorý je hraničný, sa stáva novým vyšetrovaným bodom. Postup sa končí po návrate do východiskového bodu, teda keď algoritmus nájde dvojicu bodov, ktorú už testoval. Algoritmus funguje pre objekty o veľkosti väčšej než 1 pixel a s určitými úpravami aj pre objekty hrúbky 1 pixel.

5.3.1.1 OpenCV

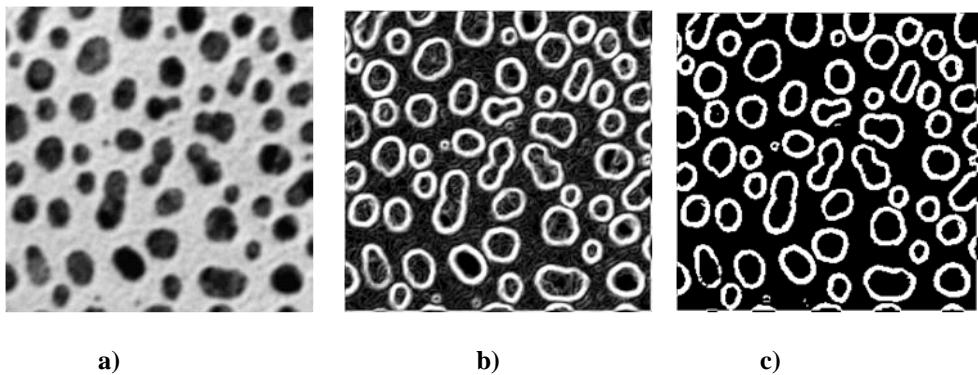
Knižnica poskytuje veľmi účinnú a často využívanú metódy nájdenia kontúr objektov.

`void findContours(InputOutputArray image, OutputArrayOfArrays contours, int mode, int method, Point offset=Point())`

kde `image` je binárny obraz a `contours` je výsledné pole (`vector`) polygónov, pričom každý polygón je pole bodov (`vector<vector<Point>>`). Funkcia má rôzne módy a využíva rôzne metódy (viď dokumentácia OpenCV). Modifikovaná verzia tejto funkcie zaznamenáva aj hierarchické rekurzívne usporiadanie kontúr ktoré vznikne tak, že binárny objekt obsahuje dieru v ktorej sú ďalšie objekty, ktoré môžu tiež obsahovať diery atď.

5.3.2 Prahovanie gradientného obrazu

Táto metóda je najjednoduchšou segmentačnou metódou založenou na detekcii hrán. Vychádza z predpokladu, že hranice reálnych objektov sa prejavia na obrázku ako nespojitost (jasy, farby, textúry). Aplikáciou hranového detektora vieme túto nespojitosť zvýrazniť v podobe gradientného obrazu (Obrázok 27b). Bohužiaľ, gradientný obraz zvýrazní okrem samotných kontúr aj množstvo neželaných kontúr či už vnútri objektov, alebo v priestore medzi nimi. Keďže intenzita týchto falošných kontúr je slabšia, ako intenzita reálnych obrysov objektov, môžeme aplikovať **prahovanie**, ktoré potlačí všetky nespojitosťi, ktorých amplitúda je menšia než nastavený prah (výsledok zobrazuje Obrázok 27c). Výsledok súčasťne neobsahuje falošné hrany a pripomína kontúry objektov, avšak problémom je, že **hrúbka** takýchto hrán (resp. prstencov) je väčšia než 1 pixel. Nasleduje teda redukcia do tvaru obrysovej kontúry o šírke 1 pixel (polygón).



Obrázok 27. a) pôvodný obrázok na ktorom potrebujeme nájsť kontúry objektov.
b) gradientný obraz ako výsledok aplikácie hranového detektora,
c) výsledok prahovania aplikovaného na gradientný obraz.

5.3.3 Cannyho detektor

Filozofia tohto detektora berie do úvahy nielen hodnotu (gradient) hrany na danom mieste, ale aj spoľahlivosť jeho susedov. Tento algoritmus popisuje nasledovný postup (prípadne v zjednodušenej podobe príbeh „Zdravotná prehliadka (Cannyho detektor)“).

Pozostáva z niekoľkých krokov:

1. Eliminácia šumu Gaussovým filtrom
2. Určenie veľkosti a smeru gradientu (Sobelovým operátorom)
3. Eliminácia bodov, ktoré nie sú lokálnym maximom. Znamená to, že nájdeme pixely, ktoré majú v lokálnom okolí maximálny gradient, čiže jeho susedia v smere gradientu (vypočítanom v predošлом kroku) majú menší gradient. Tieto pixely považujeme za miesta ktorými hrana určite prechádza.
4. Prahovanie ktoré používa 2 prahy T_1 a T_2 pri ktorých sa uplatňuje **hysterézia**. Teda ak je gradient pixelu prevyšuje horný prah, pixel je hranový, ak je nižší ako dolný prah, nie je hranový. Pixely ležiace v intervale (T_1 , T_2) sú označené ako hranové iba ak majú za suseda pixel označený predtým ako hranový.

Tento princíp je ďalej zdokonaľovaný použitím rôznych **relaxačných techník**, ktoré kombinujú intenzitu hrany a konfiguráciu pixelov využívajúc pritom fakt, že niektoré kombinácie sú pravdepodobnejšie ako iné a teda treba ich preferovať. Napríklad ak pixel so slabým gradientom leží medzi dvoma pixelmi so silným gradientom, je vysoko pravdepodobné, že má tvoriť hranicu na rozdiel od pixelu so silným gradientom, ktorý však leží medzi dvoma „slabými“ susedmi.

5.3.3.1 OpenCV

void Canny(InputArray image, OutputArray edges, double threshold1, double threshold2, ...). Táto funkcia nájde na vstupnom šedotónovom obrazu hrany použitím dvoch prahov s hysteréiou. Výstupom je gradientný šedotónový obraz rovnakého typu ako vstupný, avšak so zvýraznenými hranami.

5.4 Hľadanie hrán pomocou prehľadávania grafu

Využitie **apriórnej informácie** je jednou z najúčinnejších spôsobov ako zvýšiť efektívnosť a presnosť segmentácie. Ak poznáme napr. dva body, o ktorých vieme, že ležia na hrane objektu, bude hľadanie

zvyšných bodov hranice ležiacich medzi nimi menej zložitou úlohou než hľadanie hrany bez tejto informácie.

Graf je všeobecne známa štruktúra pozostávajúca z množiny vrcholov n_i a množiny hrán medzi týmito vrcholmi (n_i, n_j). Tieto hrany majú priradenú určitú hodnotu (cenu). Proces prehľadávania grafu pozostáva z hľadania najlepšej (najlacnejšej) cesty medzi štartovacím a konečným vrcholom grafu.

Cena cesty sa zvyčajne počíta ako súčet cien jednotlivých hrán, ktorými cesta z daného štartovacieho x_A po daný koncový bod x_B prechádza (čiže medzi elementmi x_A a x_B obrazu). Za tým účelom zvykne byť vytvorená hodnotiaca funkcia $f(x_i)$, ktorá pre každý uzol grafu vie vyhodnotiť cenu cesty od počiatočného bodu x_A po bod x_i (a tiež od x_i po koncový bod x_B). Ukážeme si niekoľko prípadov hodnotiacej funkcie $f(x_i)$, pričom pre jednoduchosť uvažuje iba cestu medzi bodom x_A po bod x_i :

Dĺžka cesty $dist(x_A, x_i)$ je najjednoduchším kritériom ceny. Za optimálnu cestu z x_A do x_i bude považovaná najkratšia cesta je (prechádzajúca najmenším počtom uzlov grafu).

Veľkosť hrán tvoriačich hranicu $M - s(x_i)$, kde M je maximum zo všetkých hodnôt veľkosti hrán. Zabudovaním tohto kritéria do hodnotiacej funkcie zohľadníme intenzitu gradientu jasu, čo môže viesť k dôsledku, že najkratšia cesta nemusí byť vždy optimálnou.

Zakrivenie hranice $dif(\phi(x_i), \phi(x_{i+k}))$, pričom dif je vhodná funkcia rozdielov smerov po sebe nasledujúcich hrán. Prudké zakrivenie je hodnotiacou funkciou penalizované, čo dovoľuje vyhnúť sa prudkým zmenám smeru hrany.

Vzdialenosť od predpokladanej hranice $dist(x_i, H)$. Keď poznáme predpokladanú trasu hranice, môžme týmto preferovať trasu, ktorá sa od nej príliš nevzdiali.

Vzdialenosť od koncového bodu $dist(x_i, x_B)$. Pokiaľ vieme, že hranica by mala pokračovať pomerne priamočiaro ku koncovému bodu, posilníme preferenciu bodov, ktoré sa nachádzajú v blízkosti koncového bodu.

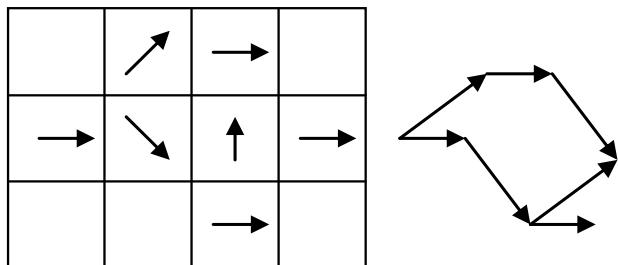
Najmenšia maximálna cena spojnice je inšpirovaná poznatkom, že sila reťaze je daná silou najslabšieho článku. Cestou z x_A po x_i sa teda nespočítavajú jednotlivé príspevky ceny, ale registruje sa iba najdrahšia spojnica. Optimálna cesta potom vedie trasou, ktorá sa vyhne slabým článkom.

Prehľadávanie s obmedzením ceny. Pokiaľ vopred poznáme cenu cesty, ktorá sa nesmie prekročiť, môžme vylúčiť vopred cesty cez „drahé“ spojnice a tým značne zefektívniť prehľadávanie.

Nasledovný algoritmus predstavuje jeden z konkrétnych spôsobov segmentácie pomocou detekcie hrán používajúceho prehľadávanie grafu:

1. Pomocou gradientného operátora (detektora hrán) získame obraz veľkostí hrán $s(x)$ a obraz smerov hrán $\phi(x)$. Každý element obrazu smerov budeme považovať za uzol grafu ohodnotený veľkosťou $s(x)$.
2. Uzly n_i a n_j sú spojené spojnicou, keď smery hrán $\phi(x_i)$ a $\phi(x_j)$ súhlasia s lokálnym smerom generovanej hranice a keď uzly n_i a n_j zodpovedajú susedným obrazovým elementom x_i a x_j v zmysle 8-susedstva. Inak povedané, aby sme mohli spojiť uzly n_i (zodpovedajúci obrazovému elementu x_i) a n_j (zodpovedajúci obrazovému elementu x_j), musí byť x_j jedným z troch možných 8-susedov x_i v smere z intervalu $\langle \phi(x_i) - \pi/4, \phi(x_i) + \pi/4 \rangle$ a zároveň musí platiť $s(x_i) > T$, $s(x_j) > T$, kde T je vhodný prah veľkosti hrany.

Podmienkou je teda, aby rozdiel smerov hrán $\phi(X_i)$ a $\phi(X_j)$ bol menší než 90 stupňov. Okrem toho je možné do zahrnúť do pravidiel pre spájanie uzlov aj iné požiadavky, podľa riešeného typu segmentácie (vid' Obrázok 28).



Obrázok 28. Vytváranie grafu z obrazu hrán
a) obraz smerov hrán b) vytvorený orientovaný graf

5.4.1 Prehľadávanie grafu.

Existuje veľa spôsobov, ako v grafe s ohodnotenými hranami hľadať najlepšiu cestu zo štartovacieho po konečný bod. Lišia sa spôsobom prehľadávania (do šírky, do hĺbky), stratégou (heuristicky, kombinatoricky) a pod. Nasledovný algoritmus (Live wire) sme vybrali za reprezentanta celej skupiny algoritmov pracujúcich na podobnom princípe.

5.5 Hľadanie hraníc Houghovou transformáciou

Pomocou Houghovej transformácie dokážeme nájsť na obraze objekty, ktorých tvar je možné popísť analytickým výrazom (priamka, kruh, elipsa a pod.). Pritom je metóda invariantná na otočenie, zmenu mierky a niektorých iných parametrov. Navyše je metóda značne necitlivá na šum a deformácie objektov.

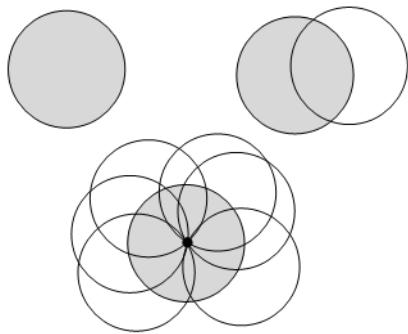
5.5.1 Detekcia kruhových objektov

Pre hľadanie kruhových objektov platí rovnica

$$(X_1 - a)^2 + (X_2 - b)^2 = r^2 \quad (61)$$

Pokiaľ rovnicu zjednodušíme tak, že budeme považovať r za konštantu, bude úloha hľadania pozície kružník známej veľkosti veľmi jednoduchá, keďže v priestore parametrov bude transformovaná na súradnice bodov.

Vo všeobecnom prípade (keď r nie je konštanta), je transformácia zložitejšia. Pre každý obrazový element X s nadprahovou hodnotou veľkosti hrany s v priestore parametrov hľadajú všetky kombinácie pri ktorých je bod (a,b) vo vzdialosti r od X . Teda musíme nájsť všetky trojice (a,b,r) splňajúce rovnicu kružnice.



Obrázok 29. Detekcia kruhových objektov pomocou Houghovej transformácie

Algoritmus v praxi funguje tak, že najprv sa detektorom hrán nájdú všetky hranové body, ktoré sú potenciálnymi obrysmi všetkých objektov (teda aj hľadaných kruhových). Z každého takého bodu opíšeme kružnicu o danom polomere. Pokiaľ sa veľa kružník pretnie v jednom bode, je zrejmé, že predstavujú stred kruhového objektu o rovnakom polomere (viď Obrázok 29).

5.5.1.1 OpenCV

`void HoughCircles(InputArray image, OutputArray circles, int method, ...)`

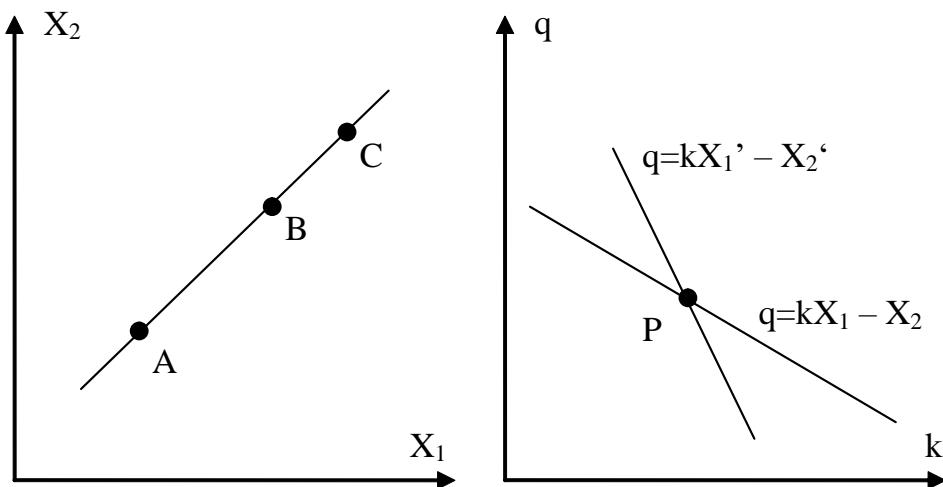
kde na vstupný šedotónový obraz sa aplikuje horeuvedený postup a výsledkom je pole circles, kde každý prvok poľa je reprezentovaný trojicou hodnôt (X, Y, R) . Pritom XY je pozícia stredu kruhového útvaru na obrazu a R je jeho polomer. Funkcia obsahuje viacero parametrov vymedzujúcich napr. povolený rozsah polomerov, vzdialenosť a pod. (viď dokumentácia).

5.5.2 Detekcia priamok a úsečiek

Transformáciu priamky do priestoru parametrov znázorňuje Obrázok 30. Priamka je daná dvoma bodmi $A=(X_1, X_2)$ a $B=(X_1', X_2')$. To znamená, že všetky priamky, ktoré prechádzajú bodom A musia vyhovovať rovnici priamky $X_2 = k X_1 + q$.

Pretože body X_1, X_2 sú pre bod A konštanty, môžeme rovnicu transformovať do priestoru parametrov (k, q) . Tam sú všetky priamky, ktoré v obrazovom priestore prechádzajú bodom A reprezentované priamkou $q = kX_1 - X_2$.

Podobne sa dajú všetky príznaky, ktoré prechádzajú bodom B , reprezentovať v priestore parametrov priamkou $q = kX_1' - X_2'$. Jediným spoločným bodom oboch priamok v priestore parametrov je bod P zodpovedajúci v obrazovom priestore priamke AB . Pre bod C , ktorý by sme na tejto priamke zvolili, tiež platí že mu v priestore parametrov zodpovedá priamka, ktorá by tiež prechádzala bodom P . Teda priamka z obrazového priestoru sa transformuje do bodu v priestore parametrov. Pokiaľ priamka nie je dokonalá, transformuje sa do zhluku bodov. V tom prípade vyberiem ľažisko zhluku ako bod reprezentujúci nedokonalú priamku.



Obrázok 30. Princíp Houghovej transformácie pre priamky
vľavo- obrazový priestor, vpravo - priestor parametrov

5.5.2.1 OpenCV

```
void HoughLines(InputArray image, OutputArray lines, ... )
```

Funkcia nájde pole priamok *lines* na vstupnom binárnom obraze *image*. Každá priamka je reprezentovaná dvojicou hodnôt (*k,q*) predstavujúcich jej smernicu a posunutie. Funkcia obsahuje viacero ďalších parametrov určujúcich vlastnosti algoritmu.

5.6 Segmentácia založená na spájaní a delení oblastí

Na rozdiel od metód založených na detekcii hrán, metódy založené na vlastnostiach oblastí vychádzajú z klasickej definície, ktorá definuje segmentáciu ako rozdelenie obrazu na spojité homogénne podoblasti, ktoré sú vzájomne disjunktné, pričom zjednotením podoblastí je celý obraz:

$$R = \bigcup_{i=1}^S R_i, \quad R_i \cap R_j = 0 \quad \text{pre } i \neq j \quad (62)$$

Kritérium homogeneity môže mať rôznu podobu. Najjednoduchším kritériom homogeneity je stredná hodnota jasu oblasti, ale môže to byť napr. veličina hodnotiaca textúru alebo vektor hodnôt.

$H(R_i) = \text{TRUE}; \quad i=1, 2, \dots, S,$

$H(R_i \cup R_j) = \text{FALSE}; \quad (i \neq j) \wedge (R_j \text{ je susedná s } R_i)$

kde *S* je počet oblastí a $H(R_i)$ je dvojhodnotové hodnotenie homogeneity R_i

Výsledná oblasť teda musí byť homogénna a súčasne aj maximálna, čo znamená pridaním ďalšej podoblasti by sa porušilo kritérium homogeneity (Obrázok 31).

Rozdelenie metód založených na vlastnostiach oblastí:

1. Narastanie oblastí
2. Spájanie oblastí
3. Štiepenie oblastí

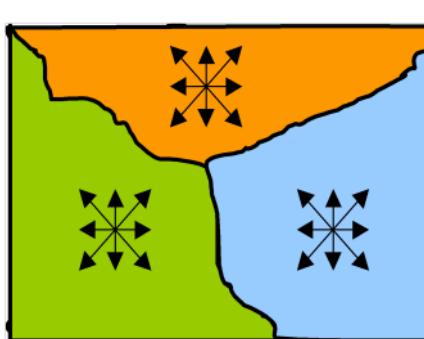
4. Spájanie a štiepenie oblastí
5. Seeded Region Growing
6. Interaktívne metódy
7. Porovnávanie so vzorom

Vo všeobecnosti sú metódy založené na vlastnostiach oblastí oveľa odolnejšie voči šumu než tie, ktoré sú založené na detekcii hrán, ale sú aj výpočtovo náročnejšie. Rôzne metódy segmentácie dávajú rôzne výsledky na tom istom obrazu.

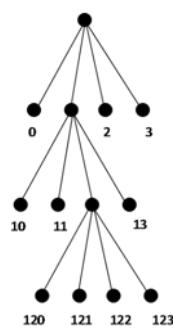
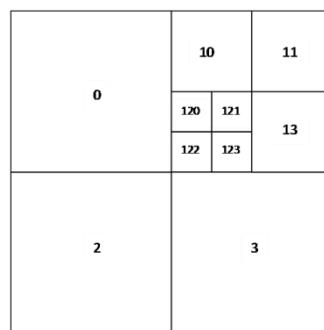
5.6.1 Spájanie oblastí

Základný algoritmus je možné definovať nasledovne

- Definuje sa počiatočné rozdelenie obrazu do veľkého množstva malých oblastí
- definuje sa kritérium spájania dvoch susedných oblastí
- Spájajú sa susedné oblasti, ktoré vyhovujú definovanému kritériu. Algoritmus končí, ak sa už nedajú spojiť žiadne dve susedné oblasti bez porušenia kritéria homogeneity



Obrázok 31. Segmentácia narastaním oblastí.



Obrázok 32. Kvadrantový strom. Nehomogénny kvadrant sa delí na 4 menšie kvadranty a kóduje sa iba konečný homogénny kvadrant spôsobom obvyklým u stromových štruktúr.

Počiatočné rozdelenie obrazu zvyčajne nepredstavujú samotné pixely (kvôli ich citlivosti na šum), ale o niečo väčšie zoskupenia pixelov (2×2 , 4×4 , 8×8). Dve susedné oblasti sa spoja, pokiaľ ich spojením nedôjde k prípadu, že by bolo porušené kritérium homogeneity. Spojením vzniknutá nová oblasť spĺňa kritérium homogeneity a prepočítá sa aj hodnota priradená novému regiónu.

Povedzme, že kritériom homogeneity slúži na to aby jas regiónu (aritmetický priemer jasu jednotlivých pixelov) bol v určitom konštantnom rozsahu. Toto kritérium znemožní pripojenie regiónov mimo tohto rozsahu. Po pripojení regiónu sa nanovo prepočítá hodnota jasu už aj s príspevkom nového regiónu. Celý proces končí, ak už nedokážeme pripojiť žiadny región bez porušenia kritéria homogeneity.

Algoritmus je závislý na poradí, to znamená, že ak segmentácia začne spracovaním bodov v ľavom dolnom rohu obrazu, dostaneme iné rozdelenie, ako keď začneme v opačnom rohu. Existuje veľa modifikácií algoritmu založených na rôznych kritériach homogeneity (jas, histogram rozloženia jasu a iné štatistické vlastnosti). Ak nastane zhoda vektorov parametrov, oblasti sa spoja.

5.6.2 Štiepenie oblastí

Je to opačný postup voči spájaniu oblastí. Počiatočné rozdelenie predstavuje celý obraz, ktorý samozrejme nevyhovuje kritériu homogeneity, preto je ďalej delený (štiepený) na menšie časti a to až do chvíle, keď všetky časti spĺňajú kritérium homogeneity.

Postup je duálny k už spomínanému procesu spájania oblastí, avšak výsledok segmentácie bude v oboch prípadoch odlišný (aj napriek použitiu toho istého kritéria homogeneity). Pri štiepení sa niektoré časti môžu javiť ako homogénne a nie sú ďalej delené, pričom pri postupe zdola nahor bolo ich spájanie odmietnuté. Typickým príkladom obrazu, pri ktorom bude proces štiepenia neúčinný je šachovnica, kde kritériom homogeneity je stredná hodnota jasu v oblasti. Už v prvom kroku nebude obraz rozdelený na podoblasti, pretože podoblasti budú mať rovnakú strednú hodnotu jasu ako nerozdelený obraz.

Naproti tomu pri postupe zdola nahor (spájanie oblastí) bude výsledkom prvého a jediného kroku tiež zastavenie algoritmu, avšak jednotlivé elementy (polička šachovnice) zostanú nespojené so susedmi, čo zodpovedá realite.

5.6.2.1 OpenCV

`int floodFill(InputOutputArray image, Point seedPoint, Scalar newVal, ...)`

Vypĺní súvislú oblasť hodnotou newVal pričom vypĺňanie začína na mieste dané seedPoint. Súvislá oblasť je definovaná ďalšími parametrami napr. ako binárna maska alebo rozsah hodnôt patriacich do oblasti.

5.6.3 Kvadrantový strom.

Pokiaľ štvorica pixelov tvoriacich pixel nadradenej hladiny je homogénna, stáva sa ich reprezentantom v hierarchickom strome. **Kvadrantové stromy** sú špeciálnym prípadom T-pyramídy, kde sa nekódujú všetky úrovne stromu, ale keď je určitý kvadrant homogénny, označí sa daný uzol stromu ako terminálny. Takže ak by sme mali napr. štvorcový obraz 256x256 a prvý kvadrant by bol homogénny, potom na zakódovanie celej podliehajúcej oblasti 128x128 bodov stačí zakódovať jedený pixel na hladine bezprostredne pod vrcholom (Obrázok 32). Bolo publikovaných množstvo algoritmov, ktoré spracovávajú obraz priamo v tvare kvadrantového stromu.

5.6.4 Štiepenie, spájanie a ich kombinácia

Kombinuje oba duálne procesy za účelom odstránenia problémov spomínaných v predošej časti. Využíva sa pritom pyramidálna reprezentácia obrazov.

Ked' je na niektornej úrovni pyramídy (okrem najnižšej) nehomogénny obraz, je rozštiepená na 4 podobrazy na nižšej úrovni pyramídy. Podobne, ak sa na niektornej hladine pyramídy nachádzajú 4 homogénne podoblasti, ktorých spojením vznikne homogénna oblasť, tak takáto oblasť vznikne na vyššej hladine pyramídy. Takto vznikne segmentačný **kvadrantový strom**, v ktorom každý uzol predstavuje homogénnu oblasť.

Podľa stratégie spájania a štiepenia existuje viacero modifikácií takéhoto kombinovaného postupu „split and merge“, „Split and link“. Jedným z príkladov môže byť aj nasledovný algoritmus.

5.6.5 Algoritmus štiepenia a spájania

Definujte počiatočné rozdelenie obrazu, kritérium homogeneity H a pyramidálnu dátovú štruktúru

Ak platí pre niektorú oblasť R_i -tej úrovne pyramídy, že je nehomogénna ($H(R) = FALSE$), rozdeľ oblasť R na 4 samostatné podoblasti. Ak platí pre niektoré 4 oblasti $R1, R2, R3, R4$ ktoré sa dajú spojiť do 1 otcovskej oblasti že

$$H(R1 \cup R2 \cup R3 \cup R4) = \text{TRUE}, \quad (63)$$

spojte ich do 1 oblasti. Ak sa nedá žiadna oblasť ani spojiť ani rozdeľiť, pokračuj ďalším krokom.

Ak existujú susedné oblasti R_i a R_j (nepatriace jednemu otcovi, alebo pochádzajúce z rôznych hladín pyramídy), pre ktoré platí

$$H(Ri \cup Rj) = \text{TRUE}, \quad (64)$$

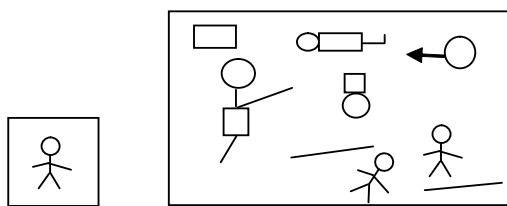
spojte ich do jedinej oblasti

Ak treba, odstráňte oblasti malých rozmerov tak, že ich spojíte s najpodobnejšou susednou oblastou.

5.7 Segmentácia porovnávaním so vzorom

Porovnávanie so vzorom (anglicky „Matching“) je pomerne častá úloha v oblasti spracovania obrazov. Cieľom je nájsť na obraze miesto, kde sa vyskytuje daný vzor, pričom vzor má tiež tvar obrazu (napr. kreslená figúrka na Obrázok 33 vľavo je vzor, ktorý sa na obraze vpravo vyskytuje 2x, raz v priamej a raz v pootočenej polohe). Táto úloha sa v modifikovanej podobe vyskytuje aj pri stereo videní, analýze pohybu a pod.

V tradičnom slova zmysle je porovnávanie so vzorom vlastne hľadaním vzájomnej **korelácie**, čiže miesta súhlasu spracovávaného obrazu f a hľadaného vzoru h umiestneného v polohe (u, v) obrazu f , pričom V je množina všetkých pixelov vzoru. Je možné si to vysvetliť na príklade, kde máme vzor nakreslený na priesvitnej fólii, ktorú postupne posúvame po obraze až kým nenájdeme miesto maximálnej zhody (absolútna zhoda je praxi zriedkavá).



Obrázok 33. Segmentácia porovnávaním so vzorom

5.7.1 Testovanie súhlasu vzoru a obrazu

Hľadanie presnej kópie vzoru na obraze je triviálna úloha, pokiaľ nie je obraz voči vzoru porušený šumom, geometrickou deformáciou, otočením a pod. Mieru zhodu určujú vzorce **vzájomnej korelácie**. Porovnávajú sa jednotlivé pixelu vzoru a zodpovedajúceho výrezu obrazu a vypočítá sa maximálna odchýlka, súčet odchýliek alebo súčet štvorcov odchýliek.

$$C_1(u, v) = \left(\max |f(i+u, j+v) - h(i, j)| \right)^{-1} \quad (65)$$

$$C_1(u, v) = \left(\sum_{(i,j) \in V} |f(i+u, j+v) - h(i, j)| \right)^{-1} \quad (66)$$

$$C_1(u, v) = \left(\sum_{(i,j) \in V} [f(i+u, j+v) - h(i, j)]^2 \right)^{-1} \quad (67)$$

Ako je známe z teórie, je možné konvolúciu dvoch obrazov nahradieť súčinom Fourierovských obrazov a následne uskutočniť spätnú transformáciu výsledku. Dané riešenie je možné s výhodou aplikovať vtedy, keď máme k dispozícii rýchle algoritmy FFT a podporou príslušného hardvéru. Pri tom je treba riešiť niektoré čiastkové problémy, ktoré sa pri práci s bežnými obrazmi nevyskytujú ako napr. že vzor aj obraz musia byť rovnako veľké, čo v praxi znamená, že treba vzor doplniť nulovými riadkami a stĺpcami na veľkosť obrazu.

5.7.2 Stratégia porovnávania

Pretože samotný algoritmus hľadania vzájomnej korelácie testuje iba posunutie vzoru na obraze, negarantuje to správny výsledok v prípade čiastočne pootočeného alebo geometricky deformovaného objektu. Preto testovanie na rotáciu, resp. zmenu mierky znamenajú ďalšie stupene voľnosti pri prehľadávaní. Každý ďalší stupeň voľnosti však exponenciálne zvyšuje výpočtovú náročnosť.

Čiastočným riešením je rozdeliť vzor na menšie časti a tie jednotliво korelovať s obrazom. Nájdene pozície s najväčšou pravdepodobnosťou nebudú presne v takom istom vzájomnom pomere ako v originálnom vzore. Drobne posnutia je možné eliminovať tzv. **pružným spojením** (ako keby jednotlivé časti boli spojené pružinami). Výsledok korelačnej analýzy potom predstavuje nájdenie takého spojenia, aby súčet pnutia v jednotlivých pružinách bol minimálny.

V mnohých prípadoch je možné algoritmus výrazne urýchliť použitím **pyramidálnej reprezentácie** obrazu aj vzoru. Aj hrubé rozlíšenie obrazu a vzoru (vyššie hladiny pyramídy) zvyčajne stačia na vtipovanie miesta, ktoré má vysokú pravdepodobnosť zhody. Potom stačí testovať zhodu obrazu a vzoru pri plnom rozlíšení, avšak iba na predtým vtipovaných perspektívnych miestach. Pokiaľ sa zistí, že na danom mieste (napriek perspektívnosti) vzor nekoreluje, je možné sa vrátiť opäť na vyššiu hladinu pyramídy a opakovať pokus s ďalšími perspektívnymi miestami.

Ďalším spôsobom, ako zredukovať množstvo výpočtov je nedokončiť testovanie zhody v prípade, keď už prvých niekoľko pixelov prinášajú príliš veľkú chybu, aby bolo možné považovať pozíciu za perspektívnu. V tom prípade je možné rovno pokračovať ďalšou pozíciou. V tomto prípade je výhodné začať porovnávanie v inom poradí, než po riadkoch a stĺpcach. Niektoré miesta majú totiž vyššiu pravdepodobnosť rozdielu než iné časti.

5.7.2.1 *OpenCV*

void matchTemplate(InputArray image, InputArray templ, OutputArray result, int method). Táto funkcia posúva obraz templ po obraze image a porovnáva mieru zhody. Výstupný obraz result má najväčšie hodnoty pixelov v miestach najväčšej zhody.

6 Pokročilé metódy segmentácie

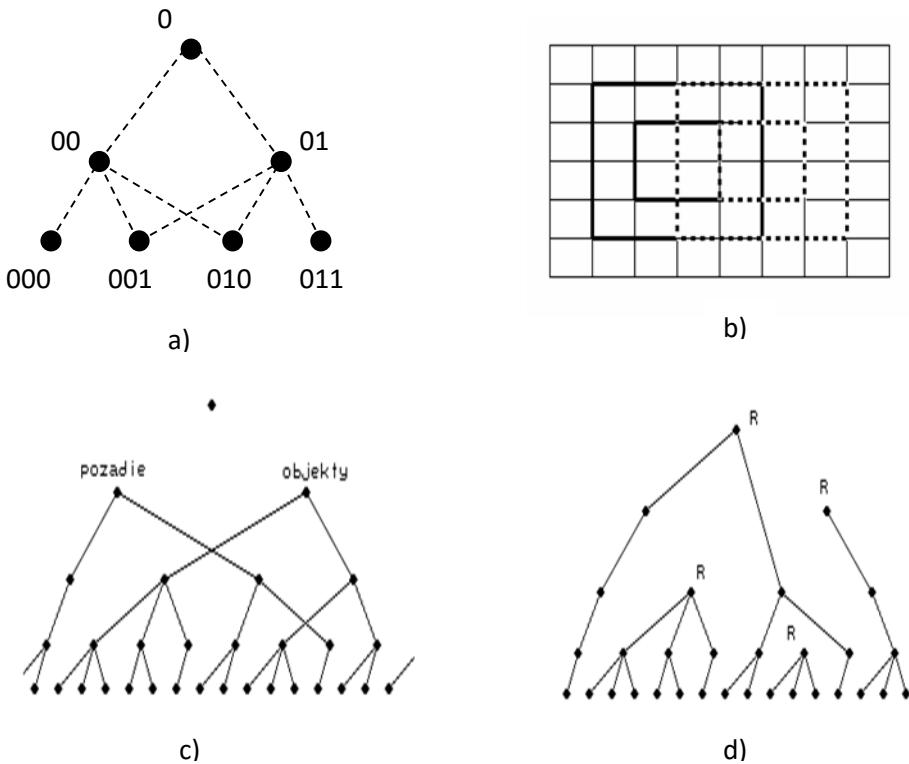
Dôležitosť segmentácie ako významnej etapy v procese číslicového spracovania obrazu sa premietlo aj do množstva nových prác publikovaných na túto tému. Aj keď často sú to iba variácie na tému klasických metód spomínaných v predošej kapitole, predsa je možné u nich pozorovať niektoré spoločné (moderné) črty a trendy. Sú to napr.:

- Využívanie a minimalizácia tzv. „Energetickej funkcie“ popisujúcej vplyv vonkajších faktorov (obraz) aj vnútorných faktorov (model objektu) na výsledok segmentácie.
- Využívanie zložitejších údajových štruktúr (ako napr. orientované grafy) na reprezentáciu obrazov a vzťahov jednotlivých pixelov a regiónov. Výhodou je existencia prepracovaného matematického formalizmu a existujúca softwarová podpora urýchľujúca vývoj aplikácií.
- Využitie štatistických a stochastických príncipov (modely na báze Markovových reťazcov, Bayesovských pravidiel pravdepodobnosti a pod.) je kľúčom k ich vyššej účinnosti.
- Algoritmy sa konštruujujú tak, aby boli aplikovateľné aj na viacozmerné obrazy (aspoň 3D)

6.1 Pyramid linking

Prekrývateľná pyramída 4×4 sa vyznačuje hlavne tým, že bloky 4×4 elementov podlieajúcich sa na vytváraní susedných bodov vyššej hladiny sa na 50% prekryvajú v horizontálnom i vertikálnom smere (Obrázok 34 znázorňuje prekryvanie iba v horizontálnom smere). Každý bod o súradničach (k, i, j) je (resp. môže byť) tvorený blokom 4×4 možných synov na hladine $k+1$, ktorých súradnice sú z intervalu $<2i-1, 2i+2>$ resp. $<2j-1, 2j+2>$. Zároveň každý bod (k, i, j) je potenciálnym "tvorcom" niektorého zo 4 bodov nadradenej hladiny, ktoré označíme termínom "možný otec" a ktoré majú súradnice $[k-1, (i \pm 1)/2, (j \pm 1)/2]$, kde znamienko " \pm " môže nadobudnúť hodnotu buď "+", alebo "-", takže využitím všetkých kombinácií to predstavuje 4 výrazy na výpočet súradníc možných otcov. Keďže delíme v celočíselnej aritmetike, výsledok je delenie modulo 2.

Pyramidálna reprezentácia sa už svojou filozofiou, dovoľujúcou lokálnou operáciou obsiahnuť ľubovoľné okno obrazu, zaraďuje medzi reprezentácie mimoriadne vhodné na segmentáciu. Keďže aplikácia lokálnych operátorov dovoľuje zistiť homogénnosť sledovanej oblasti v určitom okne (či už z hľadiska rozdelenia jasu, textúry, korelačných hodnôt a pod.) a keďže súčasne voľba veľkosti okna je závislá od predpokladanej veľkosti segmentovanej oblasti, vedie tento problém prirodzene k iteratívному postupu. Podstatné je, aby segmentácia bola zložka iteratívneho postupu, ktorý bude konvergovať. Klasický algoritmus segmentácie na báze prekrývajúcej sa pyramídy 4×4 sa nazýva *Pyramid linking*. Je založený na postupe zdola nahor (ako to znázorňuje príbeh „Demokratizácia armády (Pyramid linking)“). Každý bod pyramídy 4×4 má možnosť prilinkovať sa k jednému zo svojich štyroch potenciálnych otcov, čo znamená, že v každej iterácii je k danému bodu prilinkovaných 0 až 16 synov vidť zjednodušený 1-rozmerný prípad linkovania (Obrázok 34).



Obrázok 34. Princíp algoritmu "Pyramid linking".

- a) Príklad linkovanie v 1-rozmernom prípade. Syn (001) sa môže prilinkovať buď k vlastnému otcovi (00), alebo k nevlastnému (01). Takú istú možnosť má aj (010), vlastný syn otca (01).
- b) 2-rozmerná interpretácia tohto postupu vedie k prekryvu oblastí v pyramíde,
- c) Iterácia linkovanie/spríemernenie vedúca ku klasifikácii pixelov do 2 tried (objekt/pozadie),
- d) Klasifikácia s možnosťou zakorenenia viedie k vytvoreniu viacerých tried podľa podobnosti jasu.

Predpokladajme ďalej, že daný obraz obsahuje objekty maximálne troch typov, ktoré sa odlišujú navzájom a od pozadia iba stupňom jasu. Proces segmentácie na základe rozdelenia stupňov jasu je možné potom vyjadriť nasledovným algoritmom:

1. Inicializuj hodnoty všetkých bodov, t.j. zostav počiatočnú pyramídu 4×4 tak, že hodnota aritmetického priemeru jednotlivých blokov 4×4 je priradená jednotlivým bodom vyššej hladiny.
2. Prilinkuj každý bod k jednému zo 4 otcov a to k tomu, ktorého hodnota je najbližšie k hodnote daného bodu.
3. Prepočítaj nové hodnoty bodov ako aritmetický priemer tých synov, ktorí sú k nemu prilinkovaní.
4. Pre nové hodnoty bodov uskutoční nové linkovanie postupom zhodným s krokom b).
5. Opakuj kroky c) a d) podľa potreby (zvyčajne stačí menej ako 10 iterácií).

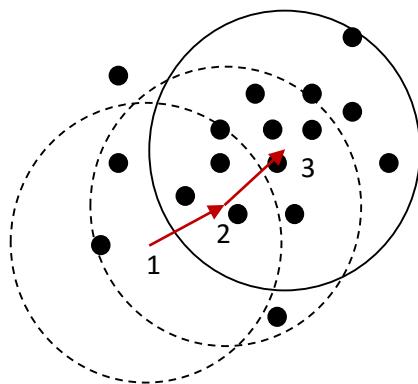
Tento proces je v podstate špeciálnym prípadom klasifikačného algoritmu ISODATA u ktorého je zaručená konvergencia. Výsledok segmentácie zobrazuje Obrázok 34c, ktorý kvôli zjednodušeniu zobrazuje iba jednorozmerný prípad. V časti) je výsledok segmentačného algoritmu, ktorý vždy rozklasifikuje pixely do dvoch (v 2D prípade do štyroch) skupín, zodpovedajúcich dvom (štvorm)

pixelom na druhej hladine od vrcholu pyramídy. Naproti tomu obrázok d) zobrazuje alternatívu algoritmu s možnosťou zakorenenia. To znamená, ak by linkovaním na ľubovoľného otca došlo k porušeniu kritéria homogeneity, stáva sa daný pixel „koreňom“ a reprezentantom podliehajúcej podoblasti.

6.1.1.1 OpenCV

funkcia `cvPyrSegmentation(src, dst...)` vybuduje pyramídu zo vstupného obrazu a realizuje na nej algoritmus pyramid linking. Jednotlivé parametre definujú štartovaciu hladinu a prahové hodnoty pre sily jednotlivých liniek. Táto funkcia už v novších verziách OpenCV nie je podporovaná.

6.2 Algoritmus MeanShift



Obrázok 35. Algoritmus Mean Shift. Šípky znázorňujú posun kruhu z zo štartovacej pozície do optimálnej pozície predstavujúcej ďažisko bodov vnútri kruhu.

Algoritmus je iba modifikáciou notoricky známeho postupu aplikovaného v rôznych iných algoritnoch a znázorňuje ho Obrázok 35, resp. v zjednodušenej interpretácii aj príbeh „Akcia na záchrana stromčekov (Mean shift)“ V jednoduchom prípade sa môžeme na tmavé guličky na obrázku dívať ako na tmavé objekty na svetlom pozadí. V tom prípade je výsledkom algoritmu optimalizovaná pozícia kružnice ktorá obsahuje najväčší počet guličiek.

Oveľa väčší význam ale má algoritmus MeanShift vtedy, keď sa na horeuvedený obrázok dívame ako na príznakový priestor a na guličky ako na hodnoty príznakov. To môžme využiť napr. pri analýze pohybu, ako to bude uvedené neskôr. Mapujeme príznakový priestor predstavovaný kvantitatívnymi parametrami obrazu

1. Inicializácia – kruh umiestnime hocikde do príznakového priestoru
2. Spočítame ďažisko pomocou bodov vnútri kruhu
3. Presunieme stred kruhu do vypočítaného ďažiska

Opakujeme 2) a 3) kým dochádza ku zmene

6.2.1.1 OpenCV

`int meanShift(InputArray probImage, Rect& window, TermCriteria criteria)`. V tejto funkcii predstavuje probImage obraz, ktorého pixely reprezentujú pravdepodobnosti s akou daný pixel patrí do hľadanej

oblasti. Tento pravdepodobnosný obraz vieme získať napr. z histogramu vopred známej hľadanej oblasti s následnou spätnou projekciou (viď dokumentácie funkcie calcBackProject).

6.3 Modely aktívnych kontúr (had, balón)

Pri segmentácii sa často stretávame s dvojetapovým prístupom ku hľadaniu hrán. V prvej fáze sa buď interaktívne alebo nejakým vyšším procesom určí približná poloha hrany objektu, ktorá sa v druhej etape optimalizuje. Vhodným modelom pre druhú etapu je použitie aktívnej kontúry, niekedy volanej aj „had“ (snake). Výsledná poloha hada resp. hadice v príbehu „Hadica v zákope (Modely aktívnych kontúr)“ je kompromisom medzi minimálnou potenciálnou energiou a mechanickými vlastnosťami tela, ktoré preferuje vystretý tvar. Aktívna kontúra je schopná je najst optimálne iba vtedy, keď sa nachádza v blízkosti polohy s minimálnou energiou. V prípade hľadania hrán budeme za energeticky minimálnu polohu považovať miesto s maximálnym gradientom (hrana).

Had je v podstate spline, ktorý minimalizuje energiu, pričom energia závisí na jeho tvare a umiestnení v obrazu. Lokálne minimá tejto energie zodpovedajú hľadaným vlastnostiam obrazu. Funkcia energie, ktorá sa minimalizuje je vážená kombinácia interných a externých síl. Vnútorné sily vyplývajú z tvaru hada a zohľadňujú napr. jeho mechanické vlastnosti, kým externé sily vyplývajú z obrazu.

Had je definovaný parametricky ako $v(s) = [x(s), y(s)]$, kde $x(s)$ a $y(s)$ sú súradnice pozdĺž hranice a s je z $[0,1]$. Minimalizovaná energia sa dá napísat ako

$$E_h^* = \int_0^1 E_h(v(s)) ds = \int_0^1 \{[E_{\text{int}}(v(s))] + [E_{\text{img}}(v(s))] + [E_{\text{con}}(v(s))] \} ds \quad (68)$$

pričom E_{int} predstavuje **vnútornú „internú“ energiu** (elasticitu), E_{img} predstavuje **vonkajšiu energiu obrazu**, ktorá núti aktívnu kontúru zaujať polohu s minimálnou energiou – v danom prípade je to hrana. E_{con} je zasa **vonkajšia „apriórna“ energia**, ktorá môže zohľadňovať vopred dané vedomosti (napr. poloha bodov, ktorími by mala kontúra „povinne“ prechádzať).

6.3.1 Interná energia

môže byť vyjadrená rovnicou

$$E_{\text{int}} = \alpha(s) \left| \frac{dv}{ds} \right|^2 + \beta(s) \left| \frac{d^2v}{ds^2} \right|^2 \quad (69)$$

kde $\alpha(s)$ a $\beta(s)$ vyjadrujú elasticitu resp. tuhost' kontúry

6.3.2 Vonkajšia energia obrazu

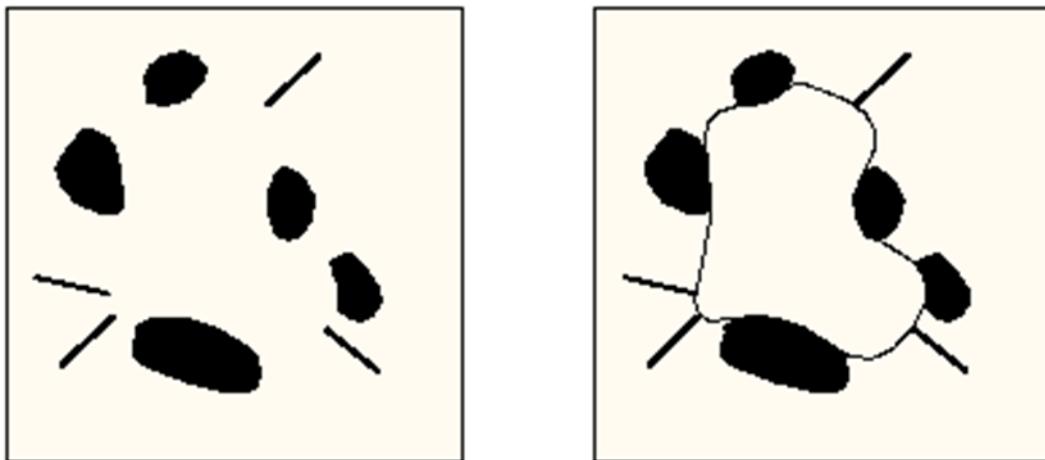
$$E_{\text{image}} = w_{\text{line}} E_{\text{line}} + w_{\text{edge}} E_{\text{edge}} + w_{\text{term}} E_{\text{term}} \quad (70)$$

kde E_{line} predstavuje energiu **lineárnych úsekov** o konštantnom jase, E_{edge} energiu vyplývajúcu z gradientu (**hrana**) a E_{term} energiu významných bodov akými sú napr. rohy alebo **koncové body** u ktorých je vyššia pravdepodobnosť, že by nimi mala aktívna kontúra prechádzať.

Z uvedeného vzorca vyplýva, že energia obrazu môže byť daná nielen gradientom (ako bolo väčšinou doteraz prízvukované) ale vplývajú na ňu aj rovné úseky s konštantným jasom alebo existencia rohových, koncových a terminálnych bodov na ktorých aktívna kontúra má tendenciu sa „zachytiť“.

6.3.3 Vonkajšia „apriórna“ energia

Značíme ju E_{con} a je do rovnice zavedená kvôli tomu, aby „vyššia moc“ mala možnosť ovplyvniť segmentáciu vo svoj prospech. Pod „vyššou mocou“ môžeme rozumieť napríklad človeka, ktorý zo svojej skúsenosti vie, kde určite má segmentovaná hrana prechádzať pozitívne ovplyvní prácu algoritmu tým, že posilní pozíciu takýchto bodov. Môže to byť aj nezávislý algoritmus pracujúci na vyšej úrovni, ktorý zistí, že poloha, ktorú zaujala aktívna kontúra je súčasťou minimálnej, ale nie optimálnej hľadiska celkovej stratégie. Vhodnou voľbou E_{con} môže potom algoritmus „vychýliť“ kontúru z energetického minima a usmerní ho na miesto, kde by mal nájsť iné lokálne minimum,



Obrázok 36. Kompozícia rôznych objektov na obraze vplyv vonkajšej energie obrazu na tvar aktívnej kontúry)

Existuje veľmi veľké množstvo rôznych modelov aktívnych kontúr, ktoré riešia špecifické problémy, napríklad ako prekonať pri optimalizácii drobné lokálne minimá, ako vhodne zabudovať do modelov iné fyzikálne vlastnosti, ako zachovať topológiu objektov a pod.

Balóny sú modely aktívnych kontúr podobné hadom s tým rozdielom, že sú úmyselne iniciované do stavu s minimálnou plochou, ktorá sa postupne zväčšuje (nafukuje).

Obrázok 36 zobrazuje „nafukovanie“ aktívnej kontúry. Po naradení na prekážku (objekty na ľavom obrázku) musí kontúra prekonávať ďaleko väčší odpor až zastane. Výhodou balónov je skutočnosť, že ľahšie prekonávajú lokálne minimá (diery, malé údolia).

Predstavme si aktívnu kontúru na úplne bielom obrazu (t.j. takom, ktorý neobsahuje žiadne prvky ktorá by mohli predstavovať vonkajšiu energiu). Kontúra sa riadi iba internou energiou, má snahu sa vyrovnávať, čo za daných okolností vedie k jej úplnému zmršteniu. Keby boli na obrázku nejaké hrany, kontúra by sa o nich zachytila a ďalej by sa nezmŕšťovala.

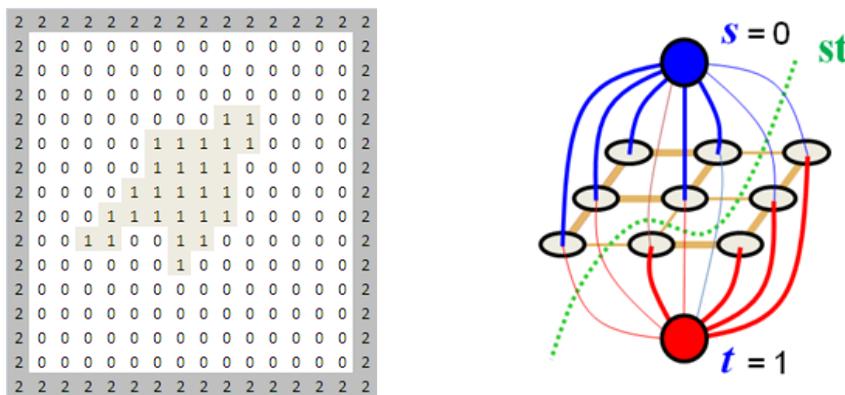
6.4 Metóda „Preseknutia grafu“ (Graph cut)

Podobne ako u aktívnych kontúr aj tu je segmentácia založená na minimalizácii energetickej funkcie E_T ktorá pozostáva z dvoch častí E_R (regional term, resp. data term) a E_B (boundary term).

$$E = E_R + E_B \quad (71)$$

E_R vychádza z tradičného predpokladu, že každý segmentovaný región by mal byť čo najhomogénejší. Využíva sa pritom model Markovovských náhodných polí (Markov Random Field - MRF), ktorý sa aplikuje na úlohu optimálneho značkovania. To znamená, že ak priradíme každému pixelu vnútri segmentovaného regiónu rovnakú značku, táto značka by štatisticky mala vyskúvať čo najväčšiemu počtu pixelov. Najjednoduchším príkladom takejto štatistiky je aritmetický priemer, čiže E_R bude rovná sume odchýliek jasov jednotlivých pixelov regionu od priemeru. Zložitejšie príklady štatistiky sú napr. založené na histograme pixelov regiónu alebo tzv GMM (Gaussian Mixture Model).

E_B predstavuje tú časť energetickej funkcie, ktorá je zodpovedná za vlastnosti hranice. V najjednoduchšom prípade môžeme uvažovať intenzitu gradientu v horizontálnom aj vertikálnom smere, čiže pixel má vysokú energiu, keď má vo svojom susedstve pixely s výrazne odlišou intenzitou (hranica).



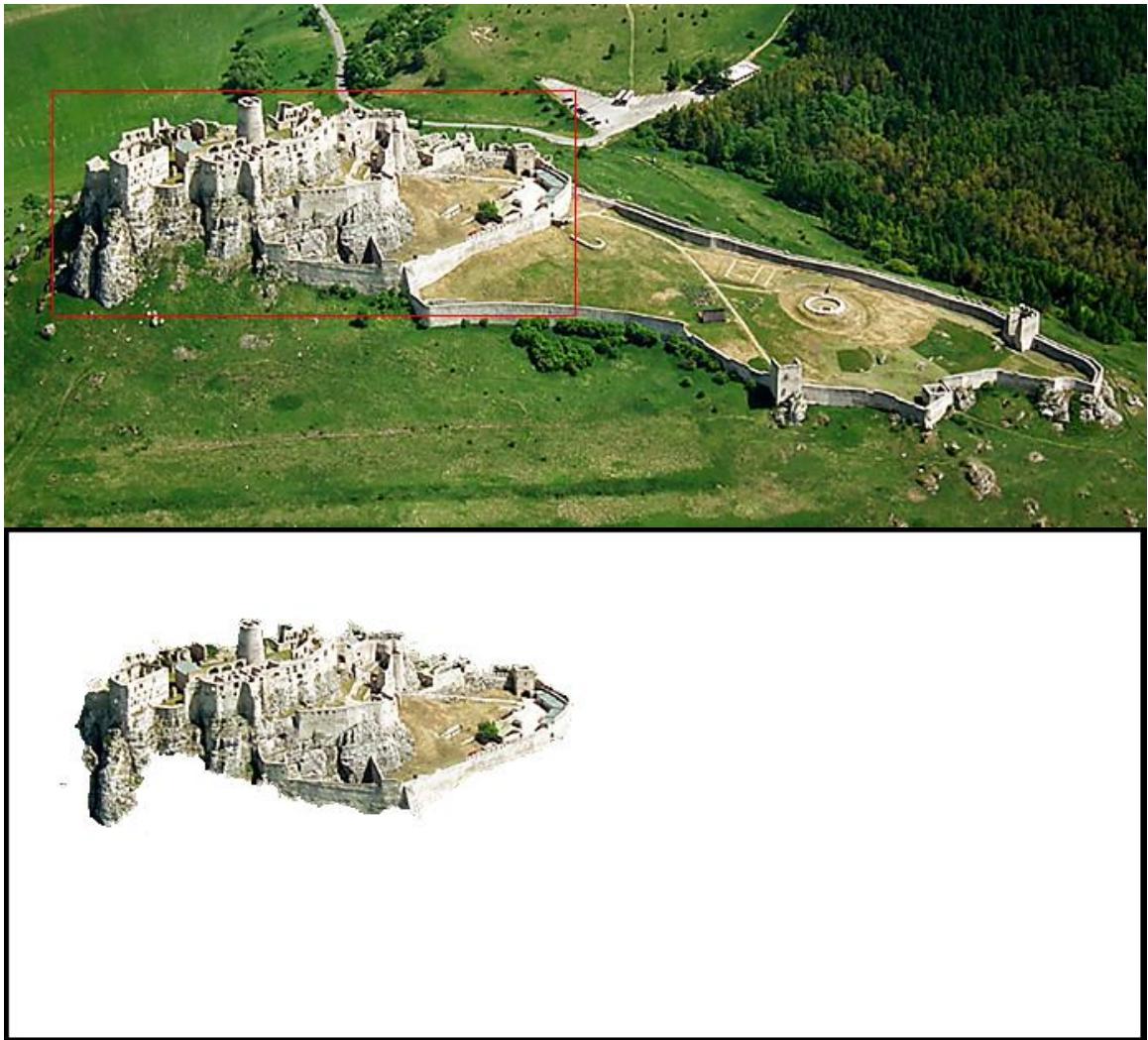
Obrázok 37. Vpravo: inicializácia algoritmu značkami: 1 = objekt, 2= pozadie, 0=neurčitý stav (určí sa segmentáciou či bude 1 alebo 2). Vľavo: Minimálny rez (prasklina) vznikne na mieste najslabších väzieb.

Inicializáciu algoritmu (apriornu informáciu) poskytne užívateľ (alebo iný algoritmus) tak, že označí napr. značkou 1 pixely určite patriace objektom a značkou 2 pixely určite patriace pozadiu, pixely ležiace medzi nimi nechá neoznačené (značka 0). Úlohou algoritmu je označiť neoznačené pixely buď ako pozadie, alebo ako objekt.

Energia E_R (region) je vyjadrená v grafe väzbou medzi pixelom a dvoma špeciálnymi uzlami grafu S a T reprezentujúce objekt resp. pozadie. Keď bol pixel iniciovaný ako objekt, bude mať maximálne silnú väzbu na uzol S a nulovú na T, v prípade pixelov pozadia je to naopak. Neoznačené pixely majú väzby na oba uzly a ich sila je úmerná podobnosti k uzlu (jas alebo iná štatistika).

Energia E_B (boundary) je vyjadrená silou väzieb medzi pixelmi - čím väčšia podobnosť, tým silnejšia väzba.

Úlohou algoritmu Graph-cut je uskutočniť rez oddelujúci S a T. Je logické, že systém väzieb "pukne" tam, kde sú najslabšie tak, ako je to naznačené na obrázku. Na túto úlohu použije algoritmy "Min-cut" (resp. k nemu duálny "Max flow"), ktoré sú známe z teórie grafov. Graph-cut má množstvo modifikácií, jednou z najčastejšie používaných je GrabCut implementovaný napr. v knižnici OpenCV. Prednosťou tejto metódy je možnosť manuálnej inicializácie



Obrázok 38. Segmentácia pomocou metódy Grab-cut.

Horný obrázok znázorňuje vstupný obrázok (Spišský hrad) a inicializáciu metódy (červený obdĺžnik). Spodný obrázok zobrazuje výsledok segmentácie.

6.4.1.1 OpenCV

void grabCut(InputArray img, InputOutputArray mask, Rect rect, ...) je funkcia, ktorá realizuje rôzne modifikácie rovnomenného algoritmu. Umožňuje iniciovať niektoré pixely pomocou obrazu mask. Alternatívou je inicializácia iba pomocou obdĺžnika rect predpokladajúc, že pixely mimo jeho vnútra sú pozadie. Výstupom je obraz mask, ktorého jednotlivé pixely predstavujú značky oblasti - výsledok segmentácie.

7 Príznaky a rozpoznávanie

7.1 Základy rozpoznávania

V predchádzajúcich častiach bola popísaná segmentácia ako etapa vedúca k rozdeleniu obrazu do vzájomne disjunktných oblastí. Podstatným krokom je porozumenie, do ktorej triedy zaradiť určitý región tak, aby koreloval s objektami vonkajšieho sveta. Rozpoznanie vyžaduje exaktný **popis oblastí** tak, aby tento mohol byť predložený klasifikátoru, ktorý ich zatrieď do určitých vzájomne disjunktných podmnožín – **tried**. **Segmentácia** teda väčšinou bezprostredne predchádza etape **rozpoznávania**, pripadne medzi nimi môže byť ďalšia činnosť (**medzietapa**) vedúca k extrahovaniu informácie o objektoch – napr. charakteristických príznakov. Niekoľko rôznych metód rozpoznania sú užívane v rôznych aplikáciach.

Základom úspešného rozpoznania je dobrá **reprezentácia znalostí** v celej svojej podstate, tak ako je základom umelej inteligencie. Rozpoznanie je **klasifikačná úloha** (do ktorej triedy patrí predmet). Existujú dve základné formy opisu objektu, podľa nich delíme aj metódy rozpoznania na dve základné skupiny:

- **Príznakové** (štatistiké). Príznaky popisujúce objekt majú zvyčajne charakter vektora, ktoré sú vstupnými dátami príznakových (štatistikých) algoritmov.
- **Syntaktické** (štrukturálny, symbolicky, lingvisticky) - pomocou primitív (spravidla symbolického charakteru). Tieto metódy pracujú s pojмami ako formálne gramatiky, produkčné pravidlá, sémantická sieť fuzzy logika a pod. Aj keď syntaktické rozpoznanie predstavuje významnú skupinu metód, predstavuje pomerne samostatnú oblasť a preto sa v ďalšom týmto metódam venovať nebudeme.

7.1.1 Identifikácia oblastí o reprezentácia objektov

Je to jedna zo základných úloh nevyhnutných pre popis regiónov. Vychádza z klasickej definície segmentácie ako rozdelenia obrazu na vzájomne disjuktné regióny. Identifikáciu regiónov vykonáva ju tzv. algoritmus farbenia (coloring) alebo značkovania (labeling). Jeho úlohou je priradiť každému spojitému regiónu jedinečné prirodzené číslo. Po skončení algoritmu reprezentuje najvyššie číslo súčasne aj počet objektov. Klasický dvojprechodový algoritmus značkovania funguje nasledovne:

7.1.1.1 Algoritmus značkovania.

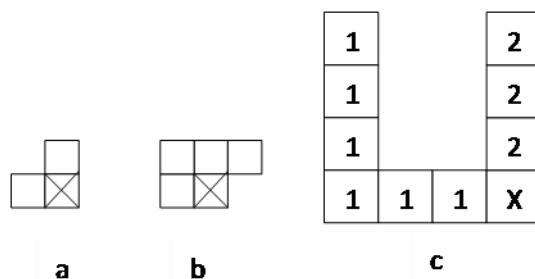
1. Prvý prechod: Prechádza systematicky riadok po riadku celý obraz a každému nenulovému pixelu $R(i,j)$ priradí hodnotu v , ktorá sa určuje nasledovne:
 2. Ak všetci susedia vyšetrovaného bodu sú pixely pozadia (s hodnotou 0), priradí v novú, doteraz nepoužitú hodnotu značky (napr. inkrementáciou počítadla)
 3. Ak existuje iba jedený sused nepatriaci pozadiu, ktorý už bol označkovaný, priradí bodu $R(i,j)$ hodnotu susedovej značky.
 4. Ak existuje viac susedov nepatriacich pozadiu, priradí bodu $R(i,j)$ značku ktoréhokoľvek z nich. Ak vznikne konflikt značiek (to znamená, že susedia majú odlišné značky), eviduj oboch susedov v tabuľke a pokračuj ďalej.

5. Pri druhom prechode algoritmus využíva informáciu konfliktov značiek v tabuľke a preznačkuje oblasti tak, že napr. priradí obom účastníkom konfliktu nižšiu značku. Takto sa preznačkujú všetky pixely majúce vyradenú značku.

Tento algoritmus sa zvyčajne zobrazuje graficky pre objekt v tvare písmena „U“, kde pozadie je reprezentované hodnotou 0 a objekty nenulovou hodnotou. Predpokladajme najtradičnejší spôsob prechodu obrázkom – po riadkoch, pričom zjednodušene by sa jeho činnosť dala vyjadriť nasledovnými dvoma krokmi:

Po prvom prechode v prvom riadku sa priradí prednej časti písmena hodnota značky „1“, naproti tomu zadná strana bude mať hodnotu „2“, keďže algoritmus netuší, že obe nožičky sa v ďalšom spoja(v zmysle Obrázok 39c). Spodok tiež nadobudne hodnotu „1“ s výnimkou posledného bodu algoritmu (vpravo dole), ktorý je konfliktný, keďže susedí aj so značkou „1“, aj so značkou „2“ .

V druhom kroku sa tento konflikt vyrieši tak, že sa všetkým pixelom so značkou „2“ vnúti značka „1“.



Obrázok 39. a), b) Masky značkovania pre 4-susedstvo a 8-susedstvo. c) konflikt značiek v bode X.

Priradenie značiek jednotlivým pixelom vnútri segmentovaných objektov by sme mohli v niektorých prípadoch považovať za výsledok. Sofistikované algoritmy by však nemohli efektívne fungovať na pixelovej reprezentácii obrazu kvôli jej obrovskej náročnosti na pamäť. Preto sa na reprezentáciu objektov používajú iné spôsoby zvyčajne založené na registrácii obrysových bodov či už ako postupnosti súradníc v tvare polygónu alebo postupnosti smerov hranice v prípade Freemanovho reťazového kódu (viď podkapitolu o dátových typoch). Miesto lineárnych úsekov kód sa môžu použiť zakrivené (B-splajny), Fourierovské transformácie obrys, hierarchické kvadrantové stromy a pod. Najpoužívanejšou formou reprezentácie objektov je vyjadrenie pomocou postupnosti bodov (polygon).

7.1.1.2 OpenCV

`int floodFill(InputOutputArray image, Point seedPoint, Scalar newVal, ...)`

Vyplní súvislú oblasť hodnotou newVal pričom vyplňanie začína na mieste dané seedPoint. Súvislá oblasť je definovaná ďalšími parametrami napr. ako binárna maska alebo rozsah hodnôt patriacich do oblasti.

funkcia void findContours(InputOutputArray image, OutputArrayOfArrays contours, int mode, int method, Point offset=Point() nájde priamo izolované objekty binárneho obrazu a jej výstupom je pole polygonov.

7.2 Skalárne deskriptory.

Príznakom resp. deskriptorom rozumieme výsledok merania, ktorý kvantifikuje nejakú vlastnosť objektu. Zvyčajne príznaky majú číslicový charakter a spájajú sa do vektora príznakov.

7.2.1 Jednoduché skalárne deskriptory oblastí.

Na základe segmentovaného objektu reprezentovaného hranicou resp. pixelmi vnútri oblasti vieme vypočítať niektoré skalárne parametre, ktoré objekt charakterizujú. Najčastejšie sa používajú nasledujúce parametre:

Veľkosť resp. plocha (Area) – je daná počtom pixelov vnútri oblasti. Pokiaľ je oblasť reprezentovaná obrysou krivkou (kontúrou), stačí jednoducho spočítať pixely ležiace vnútri hraničnej kontúry. V prípade, keď je hranica určená polygónom, hodnota veľkosti sa dá spočítať aj priamo so súradnicami polygónu

$$A = \frac{1}{2} \left| \sum_{k=0}^{n-1} (x_k y_{k+1} - x_{k+1} y_k) \right| \quad (72)$$

kde polygón je určený n bodmi (x_k, y_k) , pričom prvý a posledný bod sú totožné $(x_0, y_0) = (x_n, y_n)$.

Obvod (Perimeter) – vyplýva priamo z dĺžky hranice objektu a spočíta sa ako súčet Euklidovských vzdialenosťí jednotlivých obrysových bodov pozdĺž kontúry.

Okrúhosť (Roundness) je definovaná ako: $(4 \times \pi \times \text{plocha}) / \text{obvod}^2$. Hodnota je z intervalu $<0,1>$. Hodnota 1.0 predstavuje kruh, v praxi sa ale môže stať, že kvôli zaukrúhľovacím chybám dostaneme aj hodnotu vyššiu než 1.0.

Feretov priemer (Feret diameter) – priemer kruhu majúceho rovnakú plochu ako je plocha sledovaného objektu.

Dĺžka hlavnej osi / Major axis length – dĺžka najdlhšej úsečky, ktorú vieme nakresliť medzi ľubovoľnou dvojicou obrysových bodov.

Uhol hlavnej osi / Major axis angle – uhol medzi horizontálou a hlavnou osou objektu.

Kompaktnosť / Compactness – je definovaná ako $\text{obvod}^2 / \text{plocha}$. Pre kruhové objekty má hodnotu rovnú 1.0.

Ťažisko / Centroid – centrálny bod uzavretého objektu bez ohľadu na hodnoty jasu vo vnútri.

Gray centroid (brightness-weighted center of mass) – ťažisko, ktoré berie do úvahy aj jas vnútri objektu (pixely s jasom 255 sú najťažšie, pixely s jasom 0 najľahšie)

Integrovaná optická hustota / Integrated density – je výsledkom násobenia priemernej hodnoty jasu objektu a počtu pixelov.

Minimálna hodnota jasu / Minimal gray level – je minimálna hodnota jasu vnútri objektu

Priemerná hodnota jasu / Mean gray level – je priemerná hodnota jasu vnútri objektu.

Mediánová hodnota jasu / Median gray level a **Maximálna hodnota jasu** / Maximal gray level – sú analogicky zrejmé z názvu.

Eulerove číslo – je dané:

$$E = S - N \quad (73)$$

kde S je počet súvislých častí objektu a N je počet dier. Táto hodnota sa nemení s geometrickými transformáciami objektu.

Pomer strán (Aspect ratio) – je parameter udávajúci pomer najväčšieho a najmenšieho

$$A_R = \frac{d_{\min}}{d_{\max}} \quad (74)$$

priemeru v dvoch na seba kolmých smeroch. Používa sa napr. aj ako parameter určujúci rozmery displeja.

Ďalšie parametre, ako projekcia, výška, šírka, výstrednosť, pozdĺžnosť, pravouhlosť, smer, nekompaktnosť sa dajú do určitej miery predstaviť na základe ich názvu, presné definície sú uvedené v literatúre.

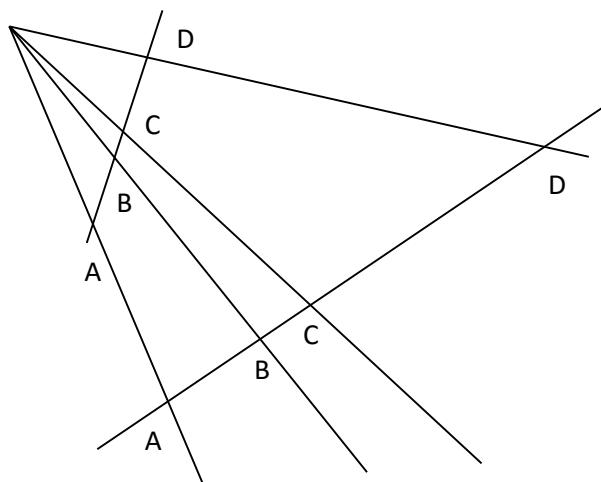
7.2.1.1 OpenCV

Funkcia `findContours` nájde objekty binárneho obrazu v tvare polygónov. Pre nich sú k dispozícii funkcie na výpočet (`contourArea`), na approximáciu (`fitLine`, `fitEllipse`, `approxPolyDP`, ...). K dispozícii sú funkcie na výpočet momentov (`moments`, `HuMoments`), z ktorých sa dajú ľahko odvodiť ďalšie deskriptory ako napr. tăžisko a pod. Užitočná je tiež funkcia `matchShapes` ktorá určí mieru podobnosti dvoch polygonov.

7.2.2 Tvarové invarianty.

Sú to také deskriptory tvaru objektu, ktoré sa nemenia pri určitej triede transformácií. Napr. sploštený objekt sa javí inak pri pohľade zhora a pri pohľade zboku. Z toho vyplýva, že napr. plocha priemetu nie je invariantná voči projekčnej transformácii.

Obrázok 40 reprezentuje jednoduchý deskriptor invariantný voči projekčnej transformácii.



Obrázok 40. Príklad tvarovo invariantného deskriptora

Krížový pomer / predstavuje koeficient, ktorý sa pri projekcii úsečky nemení

$$I = \frac{(A - C)(B - D)}{(A - D)(B - C)} \quad (75)$$

O ďalších invariantných deskriptoroch sa budeme venovať v kapitole Analyza pohybu. (skontroluj)

7.2.3 Vektory príznakov resp. jednoduché viacozmerné skalárne deskriptory oblastí.

Podľa obsahu vektory príznakov vieme rozdeliť do niekoľkých skupín:

- Vektor obsahujúci **obraz** – digitálny obraz je reprezentovaný ako matica čísel a jej riadkovým rozvojom do vektora vieme využiť obraz ako deskriptor. Takýmto obrazom často býva výrez z farebného alebo gradientného obrazu, ktorý obsahuje objekt záujmu.
- Vektor obsahujúci **histogram** – histogram v sebe udržiava rozdelenie nejakej charakteristickej vlastnosti objektu alebo celého obrazu a preto je vhodný ako deskriptor na rozpoznávanie objektov. Najčastejšie používané lokálne či globálne histogramy sú: jasové, farebné (histogram textúry), vzdialenosné histogramy, SIFT deskriptor, ...
- Vektor obsahujúci **charakteristické čtry** – kombinácia jednoduchých skalárnych deskriptorov alebo tvarových invariantov.
- **Kombinácia** predošlých vektorov.

Vektory obsahujúce obraz alebo histogram sú najčastejšie kombinované s prístupom **Sliding window**. Tento prístup prechádza všetky možné výrezy obrazu (pre každú veľkosť výrezu prejde celý obraz) a tie posielajú klasifikátoru. V prípade obrazového deskriptora výrez obrazu pred poslaním klasifikátoru sa ešte preškáluje na veľkosť obrazu, ktorá bola použitá v trénovacej vzorke.

7.3 Momenty

Všeobecná definícia momentu v diskrétnom priestore je:

$$m_{pq} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} i^p j^q f(i, j) \quad (76)$$

kde $f(i, j)$ je vo všeobecnosti definovaná ako hustota pravdepodobnosti dvojrozmernej náhodnej premennej, v prípade šedotónového obrazu si ju môžeme predstaviť ako hodnotu jasu.

Na základe momentov vieme vypočítať viaceré charakteristiky oblasti, najznámejšie sú súradnice ťažiska oblasti, $x_c = m_{10}/m_{00}$, $y_c = m_{01}/m_{00}$. V prípade binárneho obrazu je m_{00} rovné ploche objektu.

Centrálny moment je potom definovaný pomocou súradníc ťažiska:

$$m_{pq} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} (i - x_c)^p (j - y_c)^q f(i, j) \quad (77)$$

Jeho význačnou vlastnosťou je, že je **invariantný voči posuvu**, keďže súradnice objektu sú teraz vzťahované voči vlastnému ťažisku, ktoré sa posúva spolu s objektom.

Podobným, aj keď trochu zložitejším spôsobom je možné odvodiť momenty invariantné voči zmene škály aj voči rotácii, čo je možné nájsť v príslušnej literatúre.

Okrem invariantnosti voči posunutiu, zmene škály a rotácií boli odvodené aj pomocou momentové deskriptory ktoré sú invariantné voči všeobecnej afinnej transformácii (viď kapitola Predspracovanie).

Táto zahrnuje posun, rotáciu aj zmenu škály a je základom projekčnej geometrie (projekcia 3D objektu na 2D).

7.3.1.1 OpenCV

Moments moments(InputArray array, bool binaryImage=false)

void HuMoments(const Moments& m, OutputArray hu)

Štruktúra Moments obsahuje jednotlivé momenty. HuMoments na základe nich vypočíta zložitejšie tvarovo invariantné deskriptory (viď dokumentácia).

7.4 Príznakové (štatistické) metódy rozpoznávania.

Tieto metódy úzko súvisia s pojmom **klasifikácia** a ich základom sú štatistické postupy.

Príbeh „Basketbalista alebo žokej (rozpoznávanie).“ demonštruje klasifikáciu športovcov do dvoch skupín (žokeji, basketbalisti), ktorá môže byť uskutočnená v príznakovom priestore (výška, prípadne výška+váha), čo je možné ľahko zobraziť graficky, keďže každá os predstavuje jeden parameter. Na základe toho vzniknú dve **triedy** - zhľuky bodov A a B. Predmety jednej triedy tvoria zhľuk so svojim centrom, ťažiskom. Pomocou priamky (alebo viacerých priamok) vieme od seba oddeliť body patriace do rozličných tried. Pokiaľ ich vieme oddeliť tak, že každá časť rozdelenej plochy (priestoru) obsahuje iba prvky jednej triedy, potom sa jedná o úlohu so **separovateľnými** triedami. Zovšeobecnením spomínaného prípadu pre n-rozmerný priestor je klasifikácia do tried na základe vektora parametrov obsahujúceho n čísel charakterizujúcich každý objekt. V takomto n-rozmernom priestore (ktorý ale nevieme graficky znázorniť) je klasifikácia uskutočnená pomocou deliacej hyperplochy. Pokiaľ takéto hyperplochy sú roviny, hovoríme o **lineárnej separovateľnosti**. Nástroj, ktorý na základe príznakov zaradí objekty do tried sa volá **klasifikátor**, funkcie oddeľujúce jednotlivé triedy sú **diskriminačné funkcie**. Pokiaľ tieto diskriminačné funkcie vieme znázorniť ako priamky, hovoríme o **lineárnom klasifikátore**. Existuje mnoho typov klasifikátorov, ich štúdium je predmetom samostatných vedných disciplín.

Typy klasifikátorov

1. *deterministický* – presná syntéza funkcií
2. *optimálne, stochastický* – Bayesov klasifikátor
3. *heuristický* – neurónové siete, k-najbližších susedov, logistická regresia, Support Vector Machines (SVM), rozhodovacie stromy (Decision trees), ... - sú predmetom samostatných vedných disciplín a nebudeme sa s nimi ďalej zaoberať

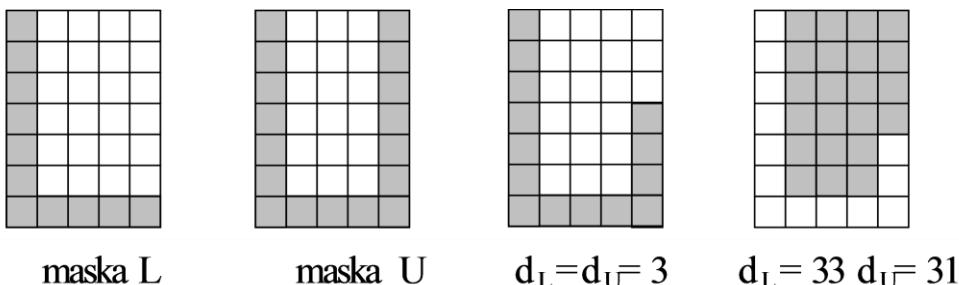
Klasifikátory sú aj základom vednej disciplíny zvanej strojové učenie, pri ktorej sa na vstup klasifikátora sa privádzajú objekty z *trénovacej množiny*, potom sa overujú na *testovacej množine*.

Rozoznávame dva hlavné prístupy k učeniu:

1. *s učiteľom* - učiteľ rozhoduje, či klasifikátor zaradil nový objekt do správnej triedy: ak áno - nastavenie klasifikátora sa „upevní“, ak nie - urobia sa korektúry nastavenia, vhodné najmä vtedy, keď sú vopred známe triedy,
2. *bez učiteľa* je vhodné najmä keď nepoznáme vopred ani počet tried ani ich masky (klasifikátor „vytvára“ triedy automaticky).

7.4.1 Deterministický klasifikátor

Je založený na diskriminačných funkciach – určitá vzorka je vždy zaradená do určitej triedy. Diskriminačná funkcia d_i (jedna pre každú triedu). Pre „správnu“ triedu má extrémnu hodnotu. Obrázok 41 ukazuje jednoduchý príklad, kde diskriminačná funkcia je počet odlišností od masky (v prípade „L“ je jej hodnota 3 v prípade „U“ tiež 3); kódová vzdialenosť medzi „L“ a „U“ je 6.



Obrázok 41. Deterministický klasifikátor a príklad výpočtu kódovej vzdialenosť.

7.4.2 Stochastický klasifikátor

Je založený na princípe pravdepodobnosti, kde musíme vždy počítať s tým, že klasifikátor niektorý objekt zaradí do nesprávnej triedy. Pokiaľ máme neseparabilné triedy, každé naše rozhodnutie o zaradení do určitej triedy nesie v sebe riziko určitej chyby – my sa len snažíme, aby pravdepodobnosť tejto chyby bola čo najmenšia.

Kedže trieda ω je náhodná premenná, nevieme jednoznačne rozhodnúť o zaradení, vieme len určiť pravdepodobnosť $P(\omega)$, že do nej nejaký (neznámy) objekt patrí. Teda pre R tried $\omega_1, \omega_2, \dots, \omega_R$ s pravdepodobnosťami $P(\omega_1), P(\omega_2), \dots, P(\omega_R)$ platí:

$$\sum_{r=1}^R P(\omega_r) = 1 \quad (78)$$

7.4.3 Bayesovské pravidlo

V praxi sa však často vyskytuje prípad, keď okrem apriorných pravdepodobností, máme k dispozícii aj aposteriorne (podmienené) pravdepodobnosti $p(\mathbf{x}/\omega_r)$, čiže pravdepodobnosť objavenia sa príznakového vektoru \mathbf{x} za predpokladu, že je známa trieda ω_r . Potom podmienená pravdepodobnosť, že vektor príznakov \mathbf{x} patrí do triedy ω_r je daná vzťahom:

$$Pr(\omega_r | X) = \frac{Pr(X | \omega_r) Pr(\omega_r)}{Pr(X)} \quad (79)$$

kde

$$Pr(X) = Pr(Pr(X | \omega_r) Pr(\omega_r) + Pr(Pr(X | \omega'_r) Pr(\omega'_r)) \quad (80)$$

Príbeh „Vzácná choroba (Bayesovský klasifikátor)“ zasa ukazuje, k akým dôsledkom môže viest, keď sa pri interpretácii meraní neberie do úvahy apriorná pravdepodobnosť.

7.4.4 K-najbližších susedov

Medzi najjednoduchšie klasifikátory patrí metóda k-najbližších susedov, ktorý si pamatá celú trénovaciu vzorku. Jeho princípom je klasifikácia objektu na základe jeho okolia v príznakovom

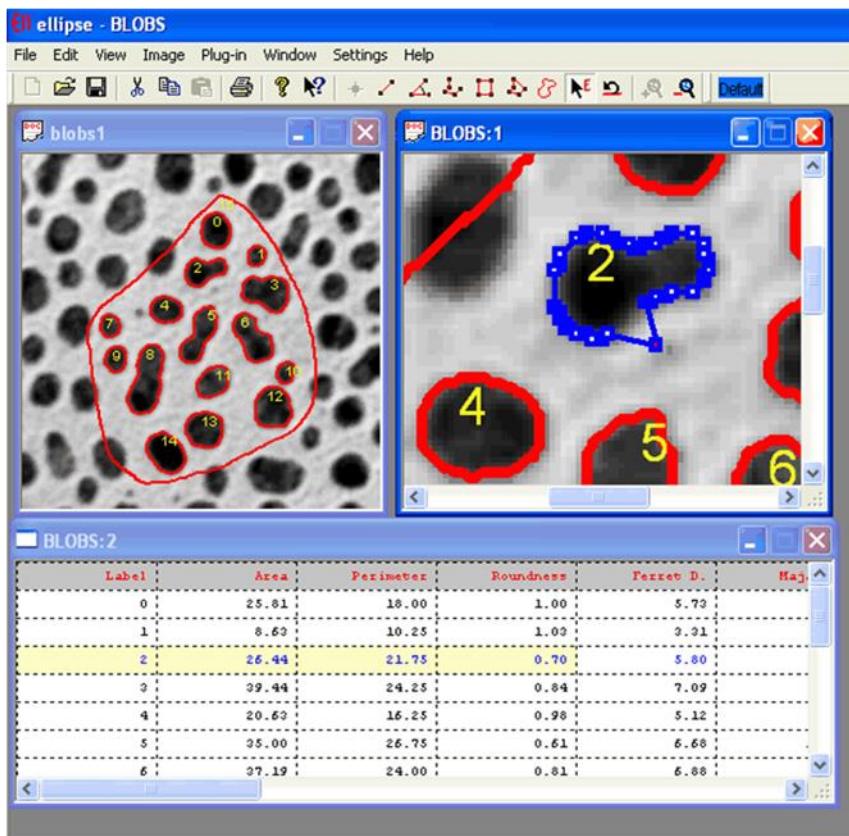
priestore. Metóda hľadá k-najbližších deskriptorov (využitím niektoréj vzdialenosnej metriky) a objektu priradí triedu podľa majoritného zastúpenia tried najbližších susedov. Vzhľadom na jej jednoduchosť metóda funguje veľmi spoľahlivo, no jej daňou je pomalá klasifikácia pri veľej trénovacej vzorke.

7.4.5 Klasifikátory s učením bez učiteľa

Toto učenie je úzko späté s problematikou odhadu hustoty v štatistike, kde klasifikátor v učiacej fáze vytvára triedy resp. zhluky deskriptorov s podobnými vlastnosťami na základe špecifickej znalosti. Príkladom znalosti je počet zhlukov v príznakovom priestore (algoritmus K-Means) alebo minimálna vzdialenosť dvoch tried (algoritmy ako zhlukovanie pomocou Mean Shiftu alebo DBSCAN). Tieto triedy sú najčastejšie charakterizované svojím ťažiskom, z čoho vyplýva, že klasifikácia je založená na najmenšej vzdialnosti deskriptora od ťažísk zhlukov.

7.4.6 Klasifikácia mikroskopických buniek

Je častou úlohou riešenou pri analýze mikroskopických obrazov [12]. Predpokladajme, že pre segmentované obrys sú vypočítané ich príznaky tak, že každý riadok tabuľky predstavuje vektor príznakov zodpovedajúceho objektu.



Obrázok 42. Vykreslenie hraníc segmentovaných objektov a výpočet príznakov.

Uvažujme príklad jednoduchej deterministickej klasifikácie do troch tried:

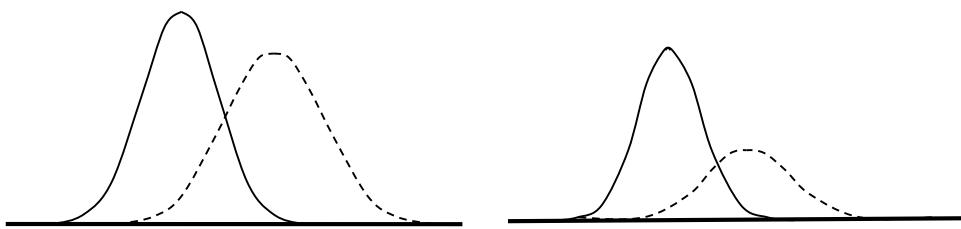
1. guľaté s polomerom 20-30 um,
2. guľaté s polomerom 10-20 um,
3. ostatné.

Za guľaté považujeme tie objekty, ktorých parameter okrúhlosti (Roundness) je väčší než 0,5. Pri kalibrovanej mierke vzdialenosí nám stačí usporiadať objekty podľa okrúhlosti a potom v rámci okrúhlych objektov podľa veľkosti.

Uvažujme teraz zložitejší prístup na základe Bayesovského pravidla, pri ktorých chceme klasifikovať bunky do dvoch tried (normálne, abnormálne). Čo o bunkách vieme?

1. Na základe dlhodobých štatistik vieme (apriórna informácia), že 90% všetkých buniek je normálnych a zvyšných 10% je abnormálnych.
2. Meraním sme získali kvantitatívne údaje (polomer, plocha, ...) jednotlivých buniek.
3. Zistili sme štatistickú distribúciu určitého príznaku (napr. polomer) s ohľadom na jednotlivé triedy.

Ked' tieto informácie dáme dohromady, vieme bunky klasifikovať tak, aby sme znížili pravdepodobnosť chybnej klasifikácie na minimum.



Obrázok 43. Gaussovské rozdelenie nameraných hodnôt pre jednotlivé triedy. Vľavo: neberie sa do úvahy apriorna pravdepodobnosť (početnosť). Vpravo: po škálovaní príslušnou apriornou pravdepodobnosťou.

Je zrejmé, že keď škálujeme Gaussovskú krivku apriórnu informáciou, znižuje sa výška menej početnej triedy a to posúva prah klasifikácie v prospech početnejšie zastúpenej triedy.

Bayesovské pravidlo nie je v rozpore s tzv. "zdravým sedliackym rozumom", iba so zjednodušenými úvahami. Cieľom je maximálna pravdepodobnosť správnej klasifikácie a tá musí brať ohľad aj na spoľahlivosť meraní na základe dlhodobých (apriórnych) znalostí.

8 Tretí rozmer v obrazoch

8.1 Základy 3D videnia

Základným princípom 3D videnia je **projekcia** lúčov vyžarovaných alebo odrazených od trojrozmerných objektov okolitého sveta na 2D plochu senzora (sietnica oka, CCD kamera atď.). Pritom dochádza ku strate informácie v dôsledku nasledovných príčin:

1. do jedného bodu v 2D rovine sa projektuje viacero bodov z 3D priestoru,
2. vzťah medzi vlastnosťami dopadajúceho lúča a vlastnosťami objektu je zložitý a nie je možné ho vyjadriť bez znalostí fyzikálnej podstatu objektu a zdroja svetla apod.,
3. objekty resp. časti objektu sa môžu prekryvať, čo zabráni aby informácia o zakrytej časti dostala ku senzoru,
4. neoddeliteľnou súčasťou signálu je šum, ktorý znehodnocuje informáciu.

Našťastie existujú spôsoby ako tieto problémy eliminovať, aby sme dosiahli cieľ a to **porozumiť** objektom v 3D. Ide o kombináciu apriórnych vedomostí o objekte a informácie získanej z viacerých obrazov snímaných pri pohybe, viacerými kamerami, pod určitým uhlom, pri určitom zaostrení a pod.

Na získanie informácie o priestorových súradničiach sa využívajú rôzne metodiky ako napr.:

- uplatnením znalosti o objektoch (1 pohľad),
- kombináciou dvoch pohľadov (stereo videnie),
- na základe sériových rezov kolmých na os Z,
- získanie informácie o hĺbke hĺbkomerom (aktívny, pasívny),
- špeciálne usporiadaným zdrojom svetla a snímača (Kinect),
- shape from focus,
- kombináciou obrazov získaných projekciou na detektory pod rôznymi uhlami (napr. tomografia).

Niekedy sa stretнемe aj s pojmom 2,5D ktorý vyjadruje, že sa nejedná o plnohodnotnú 3D informáciu (včítane vnútorných voxelov), ale iba o kombináciu 2D obrazu a hĺbkovej mapy. Čiže pre každý pixel ktorý kamera vidí poznáme aj jeho z-ovú súradnicu (vzdialenosť od kamery).

8.2 Základy projekčnej geometrie, model kamery

Základnou literatúrou v oblasti projekčnej geometrie s aplikáciu na počítačové videnie je (Hartley & Zisserman, 2004), kde sú podrobne vysvetlené základné pojmy a princípy. Vzhľadom na obmedzený rozsah tejto kapitoly ale musíme akceptovať určité zjednodušenia. Prejdeme na formu zápisu v maticovom vyjadrení, ktorá nám umožní zapisovať jednoducho súradnice v n-rozmernom priestore.

Pri vysvetľovaní projekcie vychádzajme z modelu dierkovej kamery spomínanej v úvodných kapitolách a pomocou neho odvodeného základného vzorca.

Aj v reálnej CCD kamere považujeme snímací čip za dvojrozmernú rovinu, ale súradnice (x, y) obvykle rátame od ľavého horného rohu. Snažíme sa umiestniť stred čipu oproti dierke kolmo na optickú os. Vzhľadom na spôsob výroby (hlavne lacných web kamier) však nedokážeme ani jednu z týchto podmienok garantovať, preto zakomponujeme posunutie voči optickej osi (c_x, c_y) priamo do vzorca.

Ďalšie rozšírenie modelu predstavuje zavedenie dvoch ohniskových vzdialenosťí. Vychádzame z toho, že v reálnej kamere nevystačíme s množstvom svetla prechádzajúcim cez dierku a musíme ho koncentrovať pomocou šošovky na políčko čipu. Pretože v reálnej kamere nemusia byť rozmery čipu (počet pixelov) v horizontálnom a vertikálnom smere rovnaké, tak budeme uvažovať s dvoma

hodnotami f_x a f_y . Ak poznáme reálny rozmer 1 pixelu, snažíme sa vyjadriť hodnotu ohniskovej vzdialenosťi f v násobkoch rozmeru pixelu, nie v absolútnych jednotkách (napr. mm). Potom pôvodný vzorec nadobudne tvar:

$$x = f_x \left(\frac{X}{Z} \right) + c_x; \quad y = f_y \left(\frac{Y}{Z} \right) + c_y \quad (81)$$

8.2.1 Vnútorná matica (intrinsic matrix)

Prepísaním predchádzajúceho vzťahu do maticového tvaru dostaneme:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (82)$$

pričom boli použité tzv. homogénne súradnice s použitím škály w .

Homogénne súradnice sú často používané v projekčnej geometrii, keďže značne zjednodušujú pochopenie a hlavne počítanie v 3D. Na rozdiel od tradičnej Euklidovskej geometrie uvažujeme, že všetky body priestoru, ktoré sa premietajú do toho istého bodu sú aj v priestore identické. Tento fakt vyjadríme pridaním hodnoty škály w k vektoru súradníc obrazu. Pretože z rovnice vyplýva, že $w=Z$, môžeme ľahko deliť všetky hodnoty matice.

Matica 3×3 v rovnici je vnútorná matica (intrinsic) a určuje vzťah medzi súradnicami v rovine projekcie a súradnicami v 3D priestore, pričom súčasne koriguje posunutie optickej osi kamery. Matica môže obsahovať aj ďalšie korekčné členy napr. na elimináciu vady šošoviek (*radiálneho a tangenciálneho skresenia*), ktorých výsledkom je súdkovité geometrické skreslenie alebo tzv. efekt "rybieho oka" známy fotografom. Toto nelineárne skreslenie sa dá modelovať pomocou pomerne zložitých matematických vzťahov popísaných v príslušnej literatúre. Táto matica sa volá **vnútorná matica (intrinsic matrix)** preto, lebo obsahuje iba členy, ktoré sú spojené s "vnútom" kamery a nemenia sa napr. zmenou polohy kamery. Vnútornú maticu môžeme získať kalibráciou kamery tak, čo je popísané v nasledujúcich sekciách. Okrem toho existuje tzv. "vonkajšia matica" (extrinsic matrix) vyjadrujúca polohu (pózu) kamery v priestore (je popísané v nasledujúcich sekciách).

8.2.2 Typy projekčných transformácií

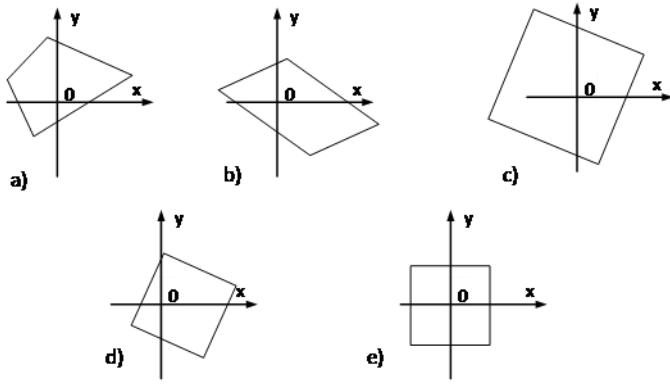
Na pochopenie kalibrácie kamery je potrebné poznať mechanizmus, ako sa určitá rovinná plocha umiestnená v 3D priestore (napr. šachovnica) zobrazuje na inú rovinnú plochu (CCD čip kamery). Vieme, že keď pozorujeme kruhovú obruč v 3D priestore pod určitým uhlom, javí sa nám ako elipsa alebo dokonca ako úsečka. Teda 2D objekt (kruh) sa cez 3D priestor transformuje na iný 2D objekt (elipsa).

Homografia je projektívna transformácia z P^d do P^d prostredníctvom R^{d+1} , ktorú môžeme zapísat:

$$u' \cong Hu \quad (83)$$

kde H je matica rozmeru $(d+1) \times (d+1)$.

Spôsob, akým sa to deje, určuje typ projekcie. Najpoužívanejšie projekčné transformácie znázorňuje Obrázok 44.



Obrázok 44. Rôzne typy projekčných transformácií:
a) všeobecná, b) affinná, c) podobnosť, d) metrická, e) identita.

Projekčná transformácia predstavuje najzložitejší prípad, kde ostávajú invariantné iba najzákladnejšie projekčné parametre – napr. krížový pomer.

Affinná transformácia. Okrem základných projekčných parametrov z prípadu a) ostávajú navyše invariantné (zachované) aj:

- paralelizmus hrán, ktoré boli paralelné,
- pomer dĺžok v paralelných hranách,
- pomer plôch,
- lineárna kombinácia vektorov ľažísk.

Podobnosť. Okrem predošlých parametrov ostávajú invariantné aj:

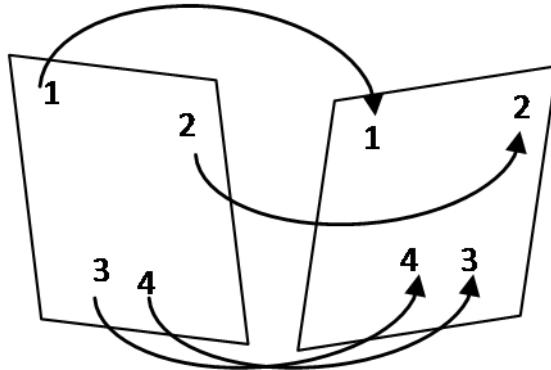
- uhly a pomer dĺžok.

Metrická (Euklidovská) transformácia. Okrem všetkých doteraz spomínaných parametrov ostáva invariantná aj:

- plocha a dĺžky strán.

Identita – predstavuje najjednoduchší prípad, keď sa transformuje obraz do identického obrazu. Všetky parametre sú invariantné.

Všetky spomínané transformácie zachovávajú topológiu v rastúcej miere od a) po e). Topológia ale môže byť narušená napr. šumom, ktorý naruší vzťah korešpondencie bodov. Obrázok 45 ukazuje prípad transformácie neprípustnej z hľadiska zachovania topológie.



Obrázok 45. Porušenie topológie a vzájomnej korešpondencie bodov pri projekčnej transformácii.

8.3 Kalibrácia kamery

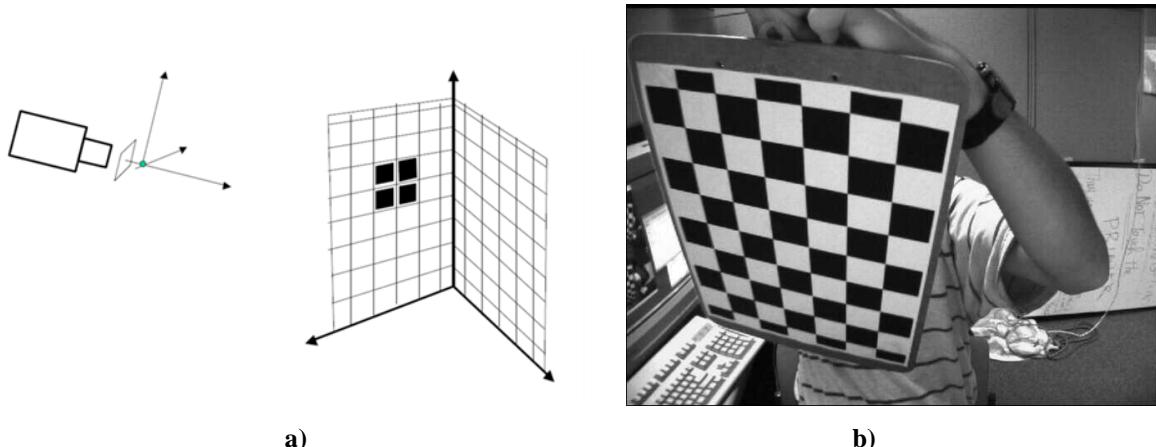
Kalibrácia je nevyhnutným stupňom pri väčšine úloh 3D rekonštrukcie. V literatúre je popísaných viacerých prístupov, v podstate ale najpoužívanejšie spôsoby demonštruje Obrázok 46.

Využívame podobnú filozofiu ako korekcia geometrického skreslenia v 2D, kde sme získali deformovaný obraz mriežky a snažili sa nájsť takú kombináciu parametrov transformačnej funkcie, ktorá čo najviac priblíži výsledný obraz ideálnemu stavu (pravidelná mriežka).

V prípade kalibrácie kamery v 3D aj snímaná kalibračná mriežka musí byť trojrozmerná (Obrázok 46a). Ale pre kvalitnú kalibráciu potrebujeme viaceré bodov z tej istej časti obrazu. Preto je výhodnejšie použiť spôsob zobrazený na Obrázok 46b), kde sa kamerou zosníma viaceré obrazy šachovnice, pričom sa snažíme, aby kamera bola v rôznych častiach obrazu a mala rôzne uhly pootočenia. O šachovnici vieme, že políčka sú rovnako vzdialé od seba a poznáme aj ich počet. Poznáme teda 3D súradnice rohových bodov v súradnom systéme šachovnice (v rovine XY je to rovnomerná mriežka s jednotkovou vzdialenosťou t.j. Z=0). Nájdenie súradníc na snímanom obrazu (v rovine čipu kamery) nám veľmi uľahčia existujúce pomocné funkcie, ktoré automaticky nájdú hodnoty súradníc v pixeloch. Projekčná matica nám vyjadruje, ako sa transformujú rohy šachovnice z 3-rozmerného priestoru do dvojrozmerného (projekčnej roviny). Používame pritom homogénne súradnice, ktorých princíp sme si už vysvetlili.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} & h_{03} \\ h_{10} & h_{11} & h_{12} & h_{13} \\ h_{20} & h_{21} & h_{22} & h_{23} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (84)$$

Všetky body sa premietnu do obrazovej roviny iba na základe jednej matice. Kedže poznáme presné súradnice priesecníkov, vieme ako sa ľubovoľný bod premietne z priestoru na snímač. Ale nevýhodou je, že posunutím kamery sa kalibrácia znehodnotí.



Obrázok 46. Kalibrácia kamery: a) pomocou fixnej vzorky (dvoch na seba kolmých mriežok v priestore),
b) pomocou série obrazov šachovnice v rôznych pozíciách a s rôznym natočením. Zdroj: Demo príklady knižnice OpenCV

8.3.1 Kalibrácia kamery s dekompozíciou.

Ide o rozloženie projekčnej matice na viac častí s rozličnými vlastnosťami a to:

- vnútornú maticu obsahujúcu hodnoty kamery f_x , f_y , c_x , c_y ,
- nelineárne skreslenia (rybie oko a pod.) v tvare poľa koeficientov - pomocou nich vieme deformovaný obraz šachovnice „vyrovnať“,
- vonkajšiu maticu udávajúcu pootočenie a posun medzi rovinami šachovnice a obrazovou rovinou (rovina čipu kamery).

Takýto model už netrpí spomínaným nedostatkom, že akékoľvek posunutie kamery poruší kalibráciu, keďže vnútorná matica kamery je pri rovnakej ohniskovej vzdialnosti konštantná. Rozšírením rovnice (82) o jednotkovú maticu 3×4 môžeme kalibráciu vyjadriť nasledovne:

$$s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (85)$$

Tento zápis používa homogénne súradnice aj v 3D priestore pridaním štvrtého prvku vektora súradníc v 3D priestore s hodnotou 1 (ako už vieme, pridanie ďalšieho prvku vektora nemení hodnotu ostatných prvkov v homogénnych súradničiach). Hodnota s predstavuje škálu ktorou sa delia jednotlivé súradnice.

8.3.2 Vonkajšia matica (extrinsic matrix)

Kým prvá matica vo výraze (85) je už známa vnútorná matica zohľadňujúca vlastnosti kamery, druhá predstavuje projekčnú maticu v najjednoduchšom tvaru, ktorý platí iba pre projekciu kolmú na os s centrom totožným so stredom kamery. V praxi nevieme zaručiť spomínané podmienky, preto projekčná matica bude upravená do tvaru, aby zohľadňovala fakt, že šachovnica je posunutá a pootočená voči osi. Pretože pootočenie a posunutie v priestore predstavujú „vonkajšiu“ vlastnosť kamery, nazýva sa vonkajšia (extrinsic) matica.

$$s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_1 & r_2 & r_3 & t_1 \\ r_4 & r_5 & r_6 & t_2 \\ r_7 & r_8 & r_9 & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (86)$$

Ľavá časť vonkajšej matice (podmatica 3×3 s hodnotami r_1, \dots, r_9) vyjadruje rotáciu šachovnice voči kamere a vektor s hodnotami t_1, \dots, t_3 jej posun v smeroch osí x,y,z. Podmatica pre rotáciu okolo jednotlivých osí je násobkom troch matíc vyjadrujúcich posuny okolo jednotlivých osí (každú rotáciu si vieme predstaviť ako postupnosť rotácií okolo jednotlivých osí).

$$R = R_x R_y R_z \quad (87)$$

kde

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \varphi_x & \sin \varphi_x \\ 0 & -\sin \varphi_x & \cos \varphi_x \end{bmatrix} \quad (88)$$

$$R_y = \begin{bmatrix} \cos \varphi_y & 0 & -\sin \varphi_y \\ 0 & 1 & 0 \\ \sin \varphi_y & 0 & \cos \varphi_y \end{bmatrix} \quad (89)$$

$$R_z = \begin{bmatrix} \cos \varphi_z & \sin \varphi_z & 0 \\ -\sin \varphi_z & \cos \varphi_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (90)$$

Posledný výraz predstavujúci pootočenie okolo osi z je identický so vzťahom pre rotáciu v kapitole o afinnej transformácii.

Rovnica (86) vyjadruje projekciu 3D bodov na 2D obrazovú rovinu. My však máme v priestore šachovnicu, na ktorú sa môžeme tiež dívať ako na 2D plochu. Teda proces kalibrácie je vlastne homografia, čiže projekcia množiny 2D bodov (súradnice rohov šachovnice) do inej množiny 2D bodov (na čipe). Tento dej pozostáva z dvoch častí a to:

- samotná projekcia pomocou rotácie, translácie a zmeny škály (zodpovedná tej vonkajšej matice a škálový faktor s),
- korekcia premietnutého obrazu v obrazovej rovine (vnútorná matice kamery).

Pre užívateľa je výhodou, že vstupom do kalibračného programu virtuálna mriežka predstavujúca šachovnicu v súradnom systéme šachovnice (body XY mriežky sú vo vzdialosti 1, Z=0). Stačí nám iba zistiť ako sa tieto body zobrazia (snímaný obraz), nájsť ich súradnice bodov a poskytnúť tieto hodnoty kalibračnej funkcie. Podmienkou je, že takýchto obrazov musíme nasnímať viacero, každý v inej polohe šachovnice.

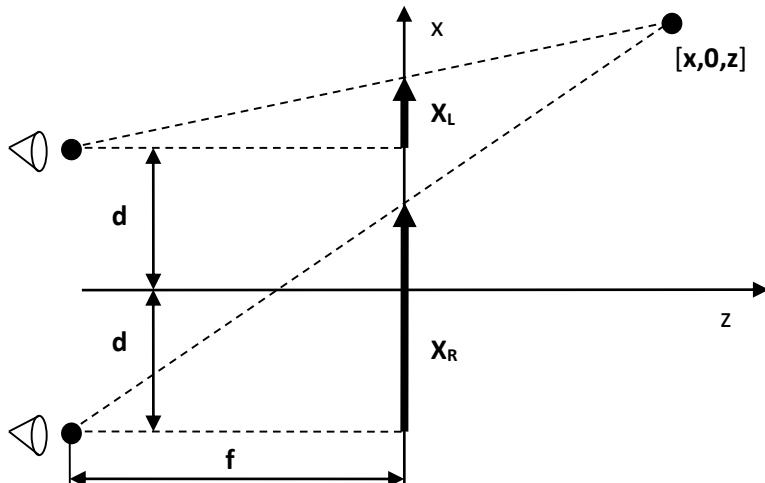
8.3.2.1 OpenCV

`bool findChessboardCorners(InputArray image, Size patternSize, OutputArray corners, ...).` Táto funkcia nájde na obraze `image` postupnosť bodov `corners` reprezentujúcich súradnice rohových bodov šachovnice o rozmeroch `patternSize`.

`double calibrateCamera(InputArrayOfArrays objectPoints, InputArrayOfArrays imagePoints, Size imageSize, InputOutputArray cameraMatrix, InputOutputArray distCoeffs, OutputArrayOfArrays rvecs, OutputArrayOfArrays tvecs, ...).` Táto funkcia realizuje samotnú kalibráciu pričom vstupom sú vzájomne si zodpovedajúce množiny 3D bodov `objectPoints` a 2D bodov `imagePoints`. Výstupom je vnútorná matica `cameraMatrix`, pole koeficientov nelineárnych skreslení `distCoeffs` a vonkajšia matica so zložkami `rvecs` (rotácia), a `tvecs` (posunutie).

8.4 Snímanie dvoma kamerami - stereo videnie

Snímanie pomocou dvoch kamier sledujúcich ten istý objekt pod ľubovoľným uhlom je pomerne zložitý problém. My sa tu obmedzíme iba na prípad konfigurácie tzv. smerovaných kamier (rectified configuration), pričom kamery majú paralelné optické osi (Obrázok 47). Pomocou nich odhadneme **hĺbku** v obraze. Táto metóda je založená na triangulácii, ktorú používali už moreplavci v dávnej minulosti na meranie vzdialenosť. Príbeh „Ako ďaleko je cel? (triangulácia a stereo videnie)“ zasa ukazuje iné netradičné využitie.



Obrázok 47. $2d$ = vzdialenosť kamier, f = ohnisková vzdialenosť, X_L, X_R = zobrazované body.

Z podobnosti trojuholníkov dostaneme:

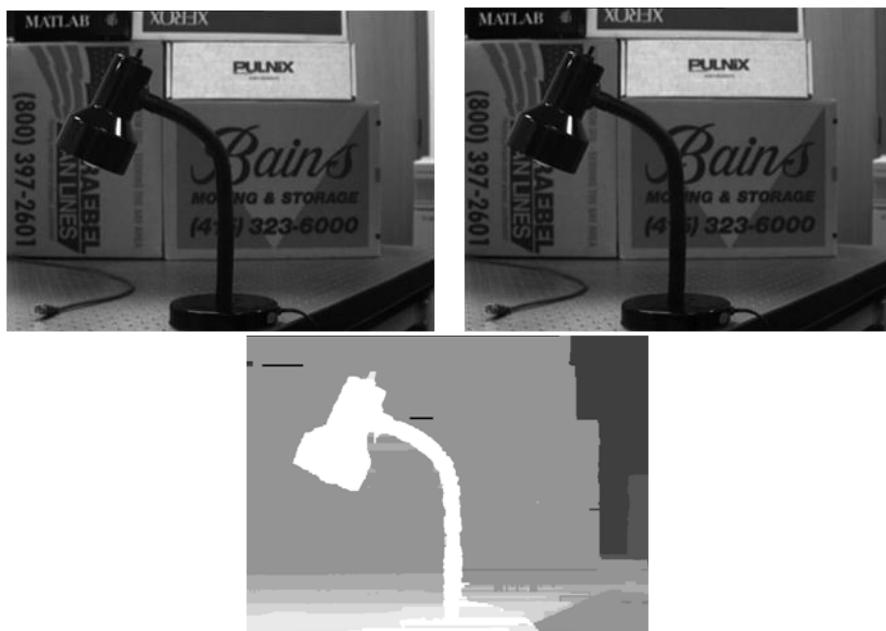
$$\frac{X_L}{f} = \frac{x - d}{z}; \quad \frac{X_R}{f} = \frac{x + d}{z} \quad (91)$$

odstránením x dostaneme:

$$z = \frac{-2df}{X_L - X_R} \quad (92)$$

Tento vzorec vyjadruje, že pri známych parametroch f a d kamery a z rozdielu ako vidí ten istý bod pravá a ľavá kamera vieme určiť vzdialenosť bodu od obrazovej roviny resp. od kamery. Ľudovo povedané, ak striedavo zatvárame pravé a ľavé oko a pozorujeme napr. vztýčený vlastný ukazovák v tesnej blízkosti očí, bude sa javiť rozdiel pozícii oveľa väčší ako keď je pozorovaný z väčšej vzdialenosťi. Tento rozdiel sa volá **disparita**. Takže každý pixel tej istej scény pozorovanej dvoma kamerami má určitú hodnotu disparity, pixely patriace blízkym objektom majú disparitu vysokú, vzdialenejšie naopak malú. Stereo obraz teda môžeme vyjadriť bud' dvojicou obrazov (z pravej a ľavej kamery), alebo iba jedným obrazom plus disparitnou mapou priradujúcou každému pixelu hodnotu predstavujúcu rozdiel vo vnímaní pozície videnej druhou kamerou.

Aj napriek jednoduchosti vzorca na výpočet hĺbky, nie je stereo videnie jednoduchý problém. Hľadanie korešpondujúcich bodov je zložité a vyžaduje zvyčajne aj špeciálne algoritmy založené na informácii o hranách. Jedným z najpoužívanejších algoritmov v tejto oblasti je RANSAC, ktorého podrobnejší popis je možné nájsť v literatúre.



Obrázok 48. Vytvorenie disparitnej mapy (dole) na základe dvoch stereo obrazov (hore).

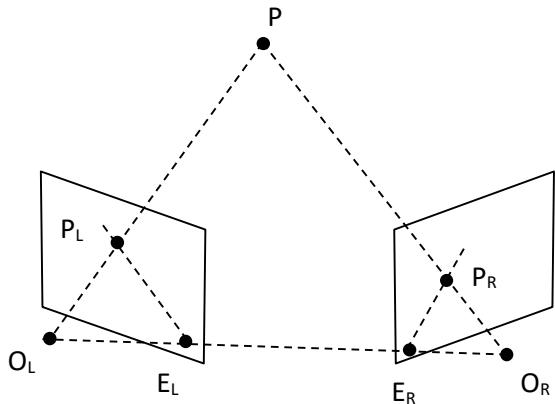
8.4.1.1 OpenCV

void StereoBM::operator() (InputArray left, InputArray right, OutputArray disparity, int disptype=CV_16S). z dvojice obrazov left a right spočíta disparitný obraz disparity. Predpokladom použitia funkcie je že obe kamery sú rektifikované. Pokiaľ tomu tak nie je, je možné použiť na tento účel funkciu void stereoRectify(...), ktorá ale vyžaduje množstvo parametrov získaných kalibráciou oboch kamier.

8.5 Epipolárna geometria a fundamentálna matica

Obrázok 47 znázorňoval prípad stereo videnia, keď obe kamery sú rektifikované, teda majú rovnobežnú optickú os a sú umiestnené v rovnakej výške (os y). Potom disparita sa počíta iba z rozdielu súradník v smere osi x. Tento model zodpovedá ľudskému stereo videniu, keďže obe oči sú v rovnakej výške a dívajú sa obe rovnakým smerom.

Ale v technickej praxi je možné ľubovoľné umiestnenie kamier sledujúcich tú istú scénu, tak ako to znázorňuje Obrázok 49. Príbeh „Vzdušný cieľ (Epipolárna geometria)“ vychádza z rovnako aranžovanej scény.

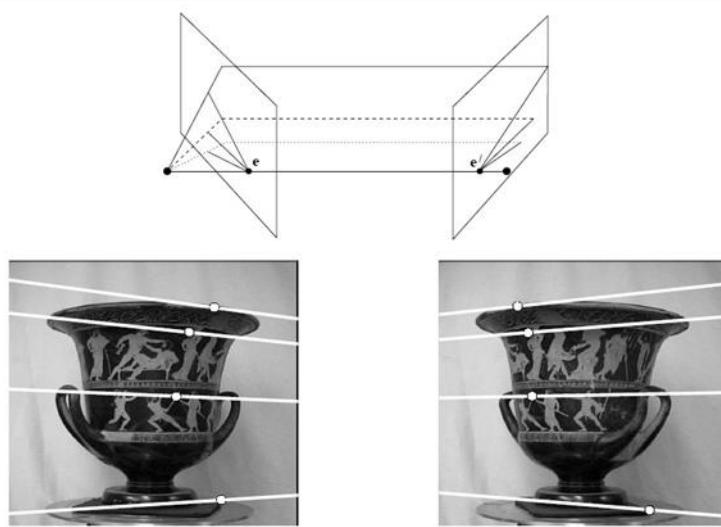


Obrázok 49. Epipolárna geometria. Body $E_L P_L$ a $E_R P_R$ udávajú epipolárne priamky, rovina daná bodmi $O_L O_R P$ je epipolárna rovina.

Epipolárna geometria

Umožní zjednodušiť komplikované vzťahy medzi dvoma kamerami v nerektifikovanej polohe. Bod P sa premietne na obrazovú rovinu pravej kamery do pozície P_R . Do tohto bodu sa ale okrem bodu P premietne aj iný bod ležiaci na priamke $O_R P$. Pravá kamera teda sama nedokáže zistiť, ako ďaleko je bod P . Túto službu jej môže poskytnúť ľavá kamera, napríklad tým že z bočného pohľadu vidí celú úsečku $O_R P$ premietnutú do podoby úsečky $P_L E_L$. Táto výnimočná úsečka je **epipolárna priamka**, podobne ako priamka $P_R E_R$ na obrazovej rovine pravej kamery. Celá rovina daná bodmi $O_L O_R P$ sa volá **epipolárna rovina**.

Výhodou epipolárnej geometrie je, že redukuje hľadanie bodov vzájomnej korešpondencie z dvojrozmerného (obrazová rovina) na jednorozmerný (epipolárna priamka). Priamka $O_L O_R$ spájajúca stredy oboch kamier je výnimočná v tom, že pretne obe obrazové roviny v bodoch zvaných **epipolárny bod**. Cez tento bod potom prechádzajú všetky epipolárne priamky zodpovedajúce ľubovoľnému bodu z druhej obrazovej roviny. Na nasledovnom obrázku sú znázornené vzájomne si zodpovedajúce body i epipolárne priamky na ktorých ležia.



Obrázok 50. Využitie epipolárnej geometrie. Na oboch obrazoch sa nájdú vzájomne si zodpovedajúce body. Každému bodu ľavého obrazu zodpovedá príslušná priamka na pravom obraze a naopak. Rovnica priamky sa z hodnoty bodu vypočíta pomocou fundamentálnej matice.

Fundamentálna matica

V predošej sekcií bolo povedané, že každému bodu z jednej obrazovej roviny zodpovedá epipolárna priamka z druhej obrazovej roviny, ktorá nutne prechádza epipolárnym bodom. Tento jednoznačný vzťah určuje fundamentálna matica:

$$\begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} = F \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (93)$$

pričom boli opäť použité homogénne súradnice, čiže pre súradnice v euklidovskom priestore (x,y) môžeme zanedbať tretiu súradnicu rovnú hodnote 1 a hľadaná epipolárna priamka je definovaná rovnicou:

$$L_1 x + L_2 y + L_3 = 0 \quad (94)$$

Potom bodu P_L (z ľavého obrazu) zodpovedá bod P_R ktorý leží na takto vypočítanej priamke na pravom obraze (a naopak). Pritom platí:

$$P_L F P_R = 0 \quad (95)$$

Toto epipolárne obmedzenie (epipolar constraint) dovoľuje vypočítať fundamentálnu maticu, pokiaľ poznáme minimálne dostatočný počet vzájomne si zodpovedajúcich bodov. Na výpočet môžeme použiť viaceré známe algoritmy:

1. základný sedem-bodový algoritmus,
2. základný osem-bodový algoritmus,
3. algoritmus RANSAC,
4. „least mean square“ algoritmus.

Výpočet je jednoduchý a použitím vhodných funkcií (napr. v knižnici OpenCV) aj rýchly. Najväčším problémom sa preto stáva spoľahlivo identifikovať zodpovedajúce si dvojice bodov. Automatické detektory týchto bodov často zlyhávajú a preto sa nezriedka využíva interaktívny mód zadávania (napr. kliknutie alebo potvrdenie myšou).

8.5.1.1 OpenCV

`Mat findFundamentalMat(InputArray points1, InputArray points2, int method=FM_RANSAC, ...)`

Funkcia na základe dvoch množín vzájomne si zodpovedajúcich bodov (points1, points2) nájde fundamentálnu maticu.

`void computeCorrespondEpilines(InputArray points, int whichImage, InputArray F, OutputArray lines).`

Pre bod (alebo množinu bodov) points na obrázku whichImage nájde pomocou fundamentálnej matice F priamku (alebo množinu zodpovedajúcich priamok) lines na druhom obrázku.

8.6 Snímanie objektov v 3D

Problém ako získať informáciu o hĺbke sa dá riešiť viacerými spôsobmi, stereo videnie spomínané v predošej časti je iba jedným z nich. Medzi najpoužívanejšie metódy patria:

1. na základe sériových rezov kolmých na os Z,
2. kombináciou dvoch pohľadov (stereo videnie),
3. získanie informácie o hĺbke híbkomerom (aktívnym, pasívnym),
4. špeciálne usporiadaným zdrojom svetla a snímača,
5. kombináciou obrazov získaných projekciou na detektory pod rôznymi uhlami (napr. počítačová tomografia).

My sa zameriame na niektoré najčastejšie používané resp. tie, ktoré sa momentálne javajú ako najperspektívnejšie.

8.6.1 Híbkomery

Delia sa na **aktívne** a **pasívne** podľa toho, či híbkomer vysiela nejakú energiu, alebo ju len príma.

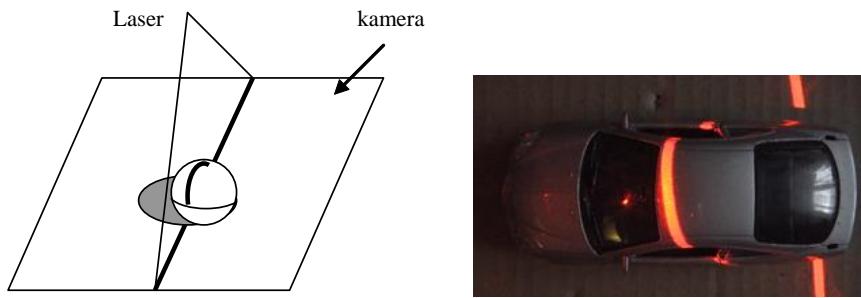
Príkladom **pasívneho** híbkomera je taký, ktorý zaostruje na povrch objektívom s malou ohniskovou vzdialenosťou. Ak by sme vnímali sledovaný povrch ako prírodný topologický terén, tak zmenou ohniskovej vzdialenosť zaostri najprv na bližšie objekty (vrcholky hôr), ostatné ostanú nezaostrené. Postupne preostri a zaznamená celý povrch vzorky pričom jednotlivé hladiny si môžeme predstaviť ako vrstevnice mapy terénu. Analogicky funguje konfokálny mikroskop, ktorý má schopnosť zaostrovať aj pod povrch vzorky.

Aktívne híbkomery vysielajú energiu v nejakej fyzikálnej forme (akustické vlny, svetlo určitej vlnovej dĺžky a pod.) a zaznamená sa čas, za ktorý sa energia vyšle, odrazí od povrchu a vráti sa naspäť. Na takomto princípe sú konštruované Time-of-Flight kamery, ktoré sú pomerne drahé.

8.6.2 Špeciálne usporiadanie zdroja svetla a snímača

Tieto snímače pracujú na základe premietania laserového lúča resp. štruktúrovaného svetla (Obrázok 51).

Technika Moire prúžkov – 3D scéna sa osvetlí rovnobežnými prúžkami. Tam, kde sú prúžky bližšie k sebe, je väčší sklon povrchu (analógia s vrstevnicami na mape).

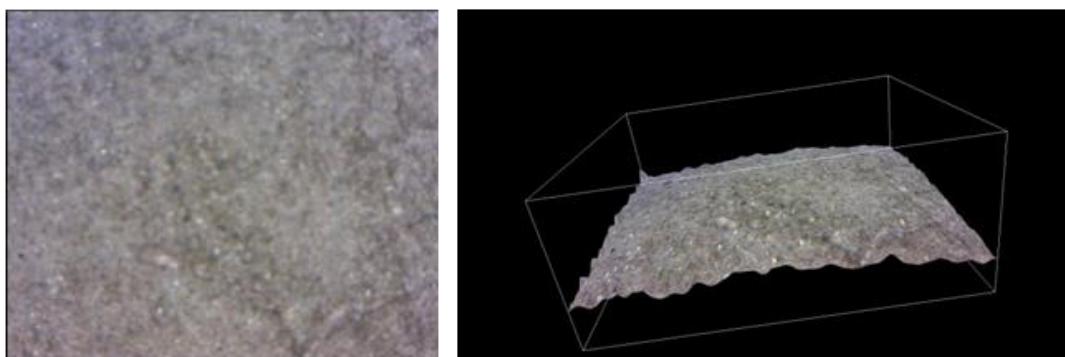


Obrázok 51. Hĺbkomer využívajúci trianguláciu za pomocí prúžku svetla.
Oblasti neviditeľné pre kameru (šráfované) a neosvetliteľné laserom (spodná časť gule).
Vpravo – snímanie 3D tvaru karoséria auta.

Kamery snímajúce v reálnom čase farebný RGB obraz a súčasne aj hĺkovú mapu sa v poslednej dobe výrazne presadili nielen ako ovládače počítačových hier telom (Kinect), ale aj v priemyselných a zdravotníckych aplikáciách.

8.7 Získanie informácie o hĺbke na základe ostrosti obrazu.

Tento spôsob sa s obľubou pri používa mikroskopii (ale aj vo fotografii), kde je možné pri kvalitných objektívoch selektívne zaostriť na určitú vzdialenosť a zvyšok obrazu je rozostrený. Pokiaľ je na danom mieste obraz ostrý, priradí sa mu hĺbka, na ktorú je mikroskop zaostrený. Potom mikroskop zmení zaostrenie o jeden krok a proces sa opakuje, až pokým nie je priradená informácia o hĺbke všetkým pixelom (Obrázok 52). To, či je obraz na určitom mieste zaostrený, je možné určiť pomocou rôznych lokálnych operátorov známych pri detektoroch hrán.



Obrázok 52. Vytvorenie 3D obrazu zo série obrazov pomocou zaostrovania. Vľavo: jeden z obrazov pod mikroskopom. Vpravo: rekonštruovaný 3D obraz. Zdroj Erna Demjén, výsledok experimentu „Shape from Focus“.

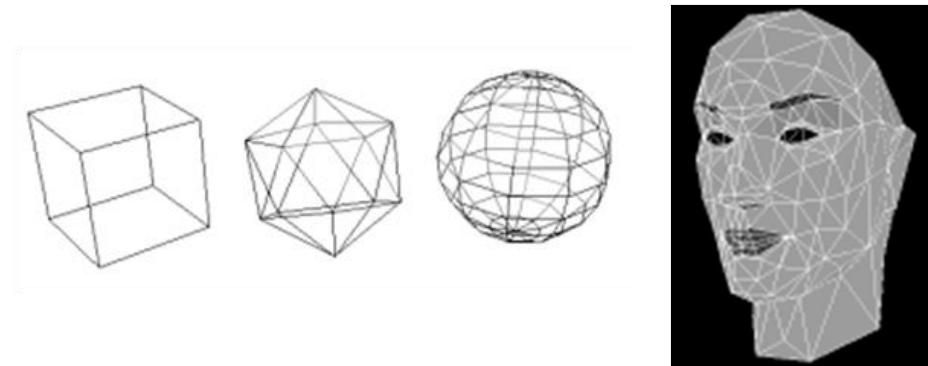
8.8 Videnie na základe modelu, typy modelov

Oblast modelov je základom počítačovej grafiky, keďže dovoľuje úsporne vizualizovať a manipulovať s grafickými objektmi (napr. postavy v počítačových hrách). Na vytvorenie funkčného modelu potrebujeme zvyčajne sériu 2D obrazov reálneho objektu a navyše vedomosti o správaní sa objektu v rôznych podmienkach. Aj v počítačovom videní, ktoré je s počítačovou grafikou úzko prepojené, využívame modely veľmi často. Napríklad keď máme všeobecný model ľudskej tváre (viď Obrázok 53). Tento model vieme použiť na fitovanie 2D obrazu reálnej ľudskej tváre. Predstavme si, že človek na obraze sa nedíva priamo do kamery, ale má pootočenú hlavu a my potrebujeme vedieť polohu

hlavy. Uhol pootočenia a posun voči optickej osi sa volá **póza** a je daná transformačnou maticou podobne ako určenie vzájomnej pozície dvoch kamier. Najprv nájdeme na modeli charakteristické body (oči, koreň nosa, špička nosa) a tieto body premietneme do 2D roviny pri určitej póze modelu. Potom hľadáme takú pózu, aby odchýlka medzi projektovanými bodmi modelu a zodpovedajúcimi bodmi na obraze reálnej tváre bola čo najmenšia.

8.8.1 Drôtový a povrchový model

Najstaršia a najjednoduchšia metóda popisu telesa, ktoré je reprezentované iba pomocou hrán a vrcholov je drôtový model. Jej názov vyplýva z podobnosti so školskými modelmi vyrobenými z drôtu. Povrchový model je taký, ktorý zobrazuje povrch telesa pomocou elementárnych plôšok – napríklad trojuholníkov (Obrázok 53).



Obrázok 53. Jednoduché drôtové modely geometrických telies a plôškový model ľudskej hlavy.

8.8.2 Objemové modely

Využívajú malé elementy objemu (voxely). Analógia so stavebnicou LEGO.

Novšie elementárne objemy sú tzv. *superkvadratiky* založené na modifikovanom elipsoide:

$$\left(\left(\frac{x}{a_1} \right)^{\frac{2}{\varepsilon_1}} + \left(\frac{y}{a_2} \right)^{\frac{2}{\varepsilon_1}} \right)^{\frac{\varepsilon_2}{\varepsilon_1}} + \left(\frac{z}{a_3} \right)^{\frac{2}{\varepsilon_1}} = 1 \quad (96)$$

pričom parametre a_1 , a_2 , a_3 definujú veľkosť superkvadratiky v smeroch x , y , z súradného systému. Parameter ε_1 udáva hranatosť superkvadratiky v pozdĺžnom smere a ε_2 vo zvislom smere.

Existuje veľké množstvo rozličných objemových modelov (zovšeobecnené valce, GEONY, atď.), ich popis je možné nájsť hlavne v literatúre venovanej počítačovej grafike.

8.8.2.1 OpenCV

`bool solvePnP(InputArray objectPoints, InputArray imagePoints, InputArray cameraMatrix, InputArray distCoeffs, OutputArray rvec, OutputArray tvec, ...)`

Táto funkcia pracuje podobne ako funkcia na kalibráciu kamery pomocou šachovnice. Na vstupe potrebuje 3D súradnice modelu (objectPoints) v jeho vlastnom súradnom systéme a zodpovedajúce 2D body na obraze (imagePoints). Okrem toho potrebuje predtým získané kalibračné údaje

kamery(cameraMatrix, distCoeffs). Funkcia potom nájde také pootočenie (rvec) a posunutie (tvec) modelu, pri ktorom je najlepšia zhoda projektovaných objectPoint a zadaných imagePoints.

9 Matematická morfológia (pre binárne obrazy)

9.1 Základy matematickej morfológie

Matematická morfológia predstavuje relatívne samostatnú etapu spracovania obrazu. Táto „samostatnosť“ je spôsobená hlavne odlišným spôsobom matematického aparátu používaného pri výpočtoch. Aj keď nadväzuje na tradičné metódy spracovania obrazu (hlavne na tie založené na konvolúcii), používa pojmy ako „množina bodov“, „konektivita“, „tvar“ a pod. Väčšinou kladú dôraz na topológiu objektov, zjednodušujú obraz, a preto sú výpočtovo veľmi efektívne, čo je príčinou ich obľúbenosti. Výhodou binárnej matematickej morfológie je, že mnohé operácie sa dajú efektívne realizovať operáciou bitového posuvu v registroch sériových počítačov.

Tradične sa sústredí hlavne na binárne obrazy, ale existuje aj množstvo aplikácií pre šedotónové obrazy a to hlavne:

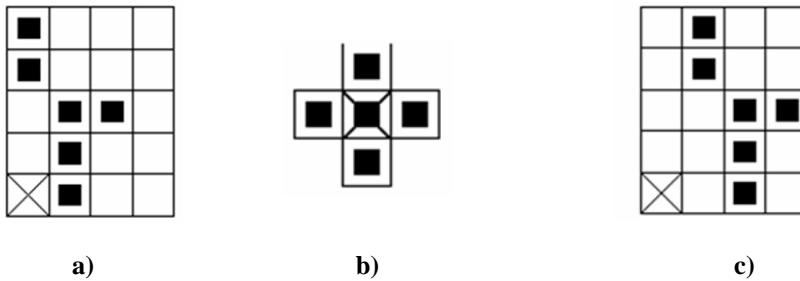
- predspracovanie (odstránenie šumu, zjednodušenie tvaru),
- zdôraznenie štruktúry (kostra, stenčovanie, zosilňovanie, konvexný obal, značkovanie objektov),
- popis tvarových vlastností číselnými príznakmi (plocha obvod, ...).

Matematická morfológia je založená na pojme bodovej množiny.

Bodová množina X – reprezentuje pixely objektov (hodnota 1). Pixely pozadia (hodnota 0) – tvoria množinu X^c (doplnok X).

Počiatok $(0,0)$ je označený krížikom. Prvky bodovej množiny sú súradnice tmavých pixelov tvoriacich objekt. Napr. pre bodovú množinu na

Obrázok 54 sú $X = \{(1,0), (1,1), (1,2), (2,2), (0,3), (0,4)\}$.



Obrázok 54. Základné pojmy: a) bodová množina b) štruktúrny element c) bodová množina posunutá o vektor $\{1,0\}$.

Štruktúrny element B je bodová množina (zvyčajne menšia) obsahujúca reprezentatívny bod označený krížikom. Pomocou nich sa uskutočňujú morfologické transformácie.

Morfologická transformácia je daná reláciou medzi množinami X a B. Z uvedeného je zjavná analógia medzi operáciou konvolúcie a morfologickou transformáciou, kde bodová množina zodpovedá obrazu a štruktúrny element konvolučnému jadru.

Ku každej morfologickej transformácii $\phi(X)$ existuje duálna transformácia $\phi'(X)$, taká že:

$$\phi(X) = (\phi'(X^c))^c \quad (97)$$

9.1.1 Posunutie bodovej množiny (translácia)

Posunutie bodovej množiny X o vektor h sa označuje X_h a je definované

$$X_h = \{p \in E^2 : p = x + h \text{ pre } x \in X\} \quad (98)$$

Transláciu bodovej množiny zobrazuje Obrázok 54c)

9.2 Dilatácia

Dilatácia dvoch bodových množín je definovaná pomocou vektorového súčtu. V matematike sa tejto operácii hovorí aj Minkowského súčet a z nej bol prevzatý aj symbol ktorý sa na označenie dilatácie používa.

$$X \oplus B = \{p \in E^2 : p = x + b, x \in X \text{ and } b \in B\} \quad (99)$$

Tento zápis znamená: x patrí do X a **zároveň** b patrí do B .

Obrázok 55a) je príkladom dilatácie pre:

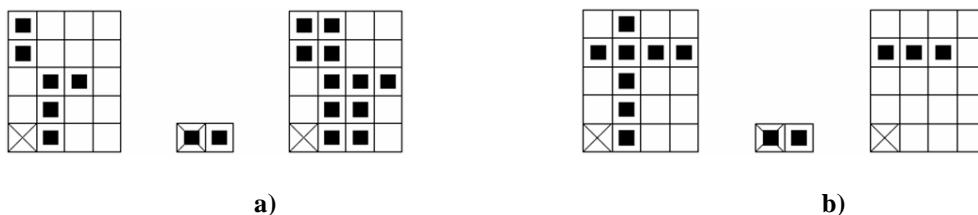
$$X = \{(1,0), (1,1), (1,2), (2,2), (0,3), (0,4)\}$$

$$B = \{(0,0), (1,0)\}$$

$$X \oplus B = \{(1,0), (1,1), (1,2), (2,2), (0,3), (0,4), (2,0), (2,1), (2,2), (3,2), (1,3), (1,4)\}$$

Dilatáciu môžeme vyjadriť aj ako zjednotenie posunutých bodových množín:

$$X \oplus B = \bigcup_{b \in B} X_b \quad (100)$$



Obrázok 55. Príklady morfológickej operácie: a) dilatácia b) erózia.

9.2.1 Použitie dilatácie

Zvyčajne sa používa izotropický štruktúrny element 3×3 , ktorý spôsobí expanziu objektov vo všetkých smeroch. Táto transformácia zmení všetky body pozadia susediace s objektmi na body objektov. Pridá objektom jednu vrstvu na úkor pozadia. Vyplní diery o veľkosti 1 pixelu. Na Obrázok 56 je vidieť, ako sa v dilatovanom obraze vyplňia tenké medzery (šráfované časti v pôvodnom obraze). Na druhej strane to spôsobí aj zväčšenie 1-pixelových bodov predstavujúcich šum (body na oblohe).

9.3 Erózia

Erózia skladá dve bodové množiny s využitím rozdielu vektorov. Podobne ako pri dilatácii aj tu je táto operácia totožná s Minkovského rozdielom včítane symbolu na jej označenie

$$X \ominus B = \{p \in E^2 : p = x - b \in X, \forall b \in B\} \quad (101)$$

Obrázok 55b) zobrazuje nasledovné množiny:

$$X = \{(1,0), (1,1), (1,2), (0,3), (1,3), (2,3), (3,3), (1,4)\}$$

$$B = \{(0,0), (1,0)\}$$

$$X \ominus B = \{(0,3), (1,3), (2,3)\}$$

Pre každý bod d obrazu sa overuje, či pre všetky možné $d+b$ leží výsledok v X . Ak áno, zapíše sa v reprezentatívnom bode do výsledku 1, inak 0.

Inak povedané: Posúvajme ľubovoľne B po X . Ak B je posunutý o vektor d a zároveň celý obsiahnutý v obraze X , potom bod zodpovedajúci reprezentatívному bodu B patrí do erózie. Podobne ako dilatácia, je aj erózia vyjadriteľná pomocou množín posunutých o vektory $-b \in B$:

$$X \ominus B = \bigcap_{b \in B} X_{-b} \quad (102)$$

9.3.1 Použitie erózie

Izotropná erózia zmení všetky body objektov susediac s pozadím na body pozadia. Ubertie objektom jednu vrstvu na úkor pozadia.

- Odstráni objekty o veľkosti (hrúbke) 1 pixel ako napr. šum. To vidíme na Obrázok 56 kde sú odstránené body predstavujúce šum (bodky na oblohe).
- Oddelí dotýkajúce sa objekty.
- Odčítaním erodovaného objektu od pôvodného získame jeho obrys.

9.4 Kombinovanie dilatácie a erózie

Erózia a Dilatácia sú **duálne**, no nie sú to vzájomne inverzné funkcie (ich inverznosť je garantovaná iba za určitých predpokladov vyplývajúcich z vlastnosti Minkowského množín). Preto je možná ich vzájomná kombinácia, ktorej výsledkom je zjednodušený obraz s menším počtom detailov, pričom sa približne zachová pôvodná hrúbka čiar (Obrázok 56).

9.4.1 Otvorenie (Opening)

Je to erózia nasledovaná dilatáciou. Ak sa obraz X nezmení po otvorení štruktúrnym elementom B , hovoríme že X je otvorený vzhľadom k B . Používa sa na oddelenie objektov dotýkajúcich sa úzkym pásiom.

$$X \circ B = (X \ominus B) \oplus B \quad (103)$$

9.4.2 Uzavretie (Closing)

Je to dilatácia nasledovaná eróziou. Spojí oddelené objekty, ktoré sú blízko seba, zaplní malé diery a vyhľadí obrys tým, že vyplní úzke zálivy.

$$X \bullet B = (X \oplus B) \ominus B \quad (104)$$

9.4.3 Vlastnosti otvorenia a uzavretia

Idempotentnosť otvorenia a uzavretia znamená, že ich opakované použitie nemení predchádzajúci výsledok. Preto má zmysel hovoriť o otvorennej resp. uzavretej množine vzhľadom na B :

$$X \circ B = (X \circ B) \circ B \quad X \bullet B = (X \bullet B) \bullet B \quad (105)$$

Podobne ako dilatácia a erozia, tvoria aj otvorenia a uzavretie **duálne operácie**.

9.4.3.1 OpenCV

`void dilate(InputArray src, OutputArray dst, InputArray kernel, ...)`

`void erode(InputArray src, OutputArray dst, InputArray kernel, ...)`

Tieto funkcie vykonajú operáciu dilatácie resp. erózie na bodovej množine src, použitím štruktúrneho elementu kernel. Výstupom je obraz dst.

`void morphologyEx(InputArray src, OutputArray dst, int op, InputArray kernel, ...)` má rovnaké použitie ako predošlé funkcie, pomocou kódu operácie op však vieme definovať aj operácie otvorenia a uzavretia.



Obrázok 56. Morfologické operácie nad skicou Dómu sv. Alžbety v Košiciach:

- originál,
- po dilatácii,
- po erózii,
- po otvorení,
- po uzavretí.

9.4.4 Transformácia „Hit-or-miss“

Transformácia hľadá lokálnu konfiguráciu skupiny pixelov (roh, určitý tvar hranice), pričom jej veľkosť je daná veľkosťou štruktúrneho elementu. Je to určitý ekvivalent porovnávania so vzorom (**template matching**). Využívame pritom zložený štruktúrny element $B = (B_1, B_2)$.

$$X \otimes B = \{x : B_1 \subset X, B_2 \subset X^c\} \quad (106)$$

Hľadá sa teda presná konfigurácia objektových pixelov a zároveň presná konfigurácia pixelov pozadia. Štruktúrny element pozostáva z pixelov dvoch typov B_1 a B_2 . Musia byť splnené dve podmienky súčasne – časť B_1 zloženého štruktúrneho elementu musí byť obsiahnutá v X a časť B_2 v doplnku X^c .

9.5 Šedotónová dilatácia a erózia

Vychádzajme z často využívanej topologického modelu, pri ktorom jas predstavuje výšku terénu. Keďže objekty „vyčnievajú“ nad pozadím, hrany objektu potom pripomínajú strmé útesy. Operácia erózie znamená nahradenie hodnoty pixelu najmenšou hodnotou z jeho susedstva (to je len zovšeobecnenie predošej definície platnej pre binárne obrazy, kde boli iba dve hodnoty - 0 a 255). V praxi sa to prejaví tak, že z ostrého útesu ubudne smerom do vnútra objektu a malé vyčnievajúce výbežky sa zmenšia. Časti objektu s homogénym jasom to nepostihne. Opačný prípad nastane v prípade šedotónovej dilatácie, pri ktorej sa vyplňia malé priehlbiny a objekt sa zväčší, keďže útes sa posunie smerom von od objektu.

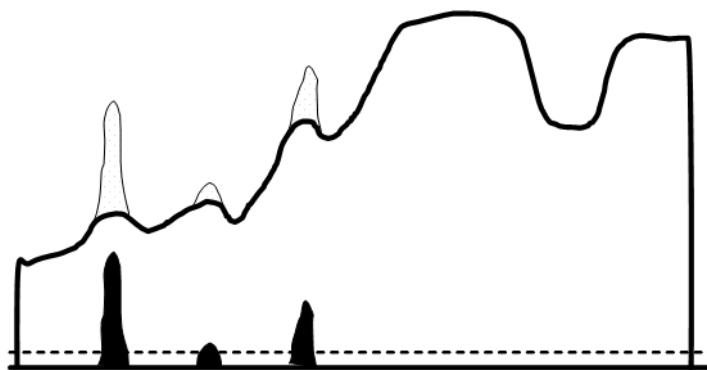
Táto predstava umožňuje jednoduchú detekciu hrán na základe „Baucherovho gradientu“:

$$grad(x) = (X \oplus B) - (X \ominus B) \quad (107)$$

Pomocou tejto morfologickej funkcie sa vypočíta rozdiel medzi dilatáciou a eróziou vstupného obrazu. Je to určitá analógia klasickej detekcie hrán (derivácia), pri ktorej vznikne gradientný obraz.

9.6 Top hat transformácia (vrch klobúka)

Oddeľuje objekty výrazne sa líšiace jasom na pomaly sa meniacom pozadí (napr. v dôsledku nerovnomerného osvetlenia). Použije sa operácia Otvorenie. Objekty, ktoré sa nevojdú do veľkosti štruktúrneho elementu (diery v klobúku) sa odstránia. Potom sa odčíta filtrovaný obraz od pôvodného a výsledok sa prahuje (Obrázok 57).



Obrázok 57. Top hat transformácia

9.6.1.1 OpenCV

Funkcie dilate, erode a morphologyEx spomínané v predošej časti sú sice najčastejšie aplikované na binárne obrazy (t.j. šedotónové iba s dvoma hodnotami 0 a 255), avšak je možné ich aplikovať na šedotónové obrazy s viacerými hodnotami a dokonca aj na farebné, pri ktorých sa každý farebný kanál spracováva zvlášť.

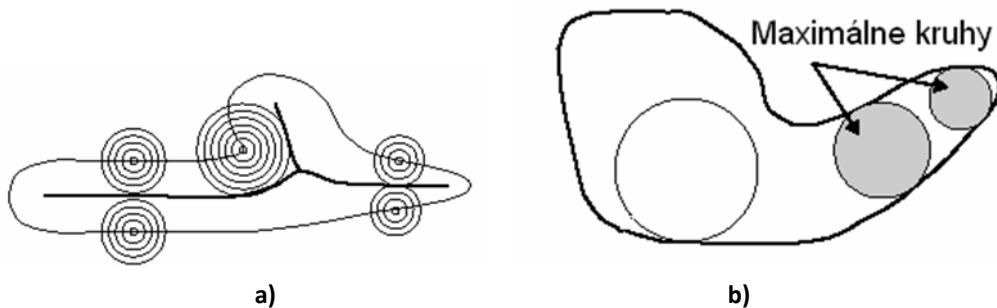
Funkcia morphologyEx má možnosť aj voľby operácie typu TopHat. Operácia typu Baucherov morfologickej gradient odčíta od výsledky jeho erodovanú verziu. Funkcia tiež dovoľuje definovať

rôzne štruktúrne elementy (roh, kríž a pod.) a metódou "Hit-or-miss" nájsť na obraze oblasti analogických vlastností.

9.7 Skeleton, topologické vlastnosti

9.7.1 Morfologická kostra (skelet, skeleton)

Často potrebujeme získať štruktúru, ktorá zjednodušene reprezentuje objekt, pričom rešpektuje jeho topológiu. Takto reprezentáciou je skelet. Bez ohľadu na presnú definíciu si môžeme predstaviť skelet ako výsledok pôsobenia tzv. „Algoritmu horiacej trávy“. Predstavme si zapálenú trávu, ktorá súčasne začne horieť po celom obvode objektu. Skelet je miesto, kde sa horiace ohne stretnú (viď Obrázok 58a).

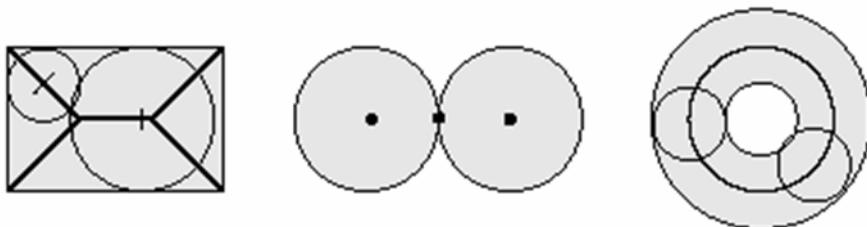


Obrázok 58. a) Skelet definovaný pomocou algoritmu horiacej trávy. b) Skelet pomocou maximálnych vpísaných kruhov.

Skelet je možné definovať pomocou rozdielov morfologických otvorení Lantuéjoulovou formulou:

$$S_B(X) = \bigcup_n [(X \ominus nB) - (X \ominus nB) \circ B] \quad (108)$$

kde nB je operácia použitá n -krát so štruktúrnym elementom B .

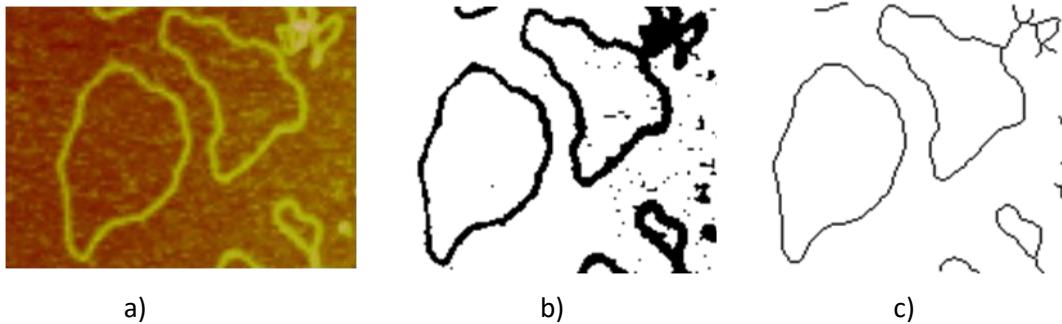


Obrázok 59. Príklady skeletov ďalších objektov.

Skelet $S(Y)$ množiny Y je možné si predstaviť aj zjednotením stredov x kružníc $D(x)$, ktoré sú obsiahnuté v Y a dotýkajú sa hranice množiny Y aspoň v 2 bodoch. Takto definícia nezaručuje homotopiu vždy (napríklad 2 dotýkajúce sa kruhy na Obrázok 59b). Podobne to nezaručí ani opakovaná erozia a dilatácia. Preto sa v praxi používajú iné tzv. homotopické metódy.

Existuje skupina morfologických transformácií, ktorým sa hovorí homeotopické.

Transformácia je **homotopická** ak nemení spojitosť objektov a dier vyjadrených homotopickým stromom. Predstavujeme si túto vlastnosť tak, že na gumovú podložku nakreslíme objekty s dierami a potom ju za okraje naťahujeme. To spôsobí zmenu tvaru objektov i dier, ale ich počet a vzájomný pomer sa nemení. Koreň stromu reprezentuje pozadie obrazu. Prvá úroveň stromu zodpovedá objektom, druhá dieram v týchto objektoch, tretia objektom v týchto dierach atď.



Obrázok 60. a) Obraz štruktúry DN z AFM mikroskopu. b) Binárny obraz získaný prahovaním. c) Skeleton DNA štruktúr.

9.8 Vzdialenosťná funkcia (Distance transform)

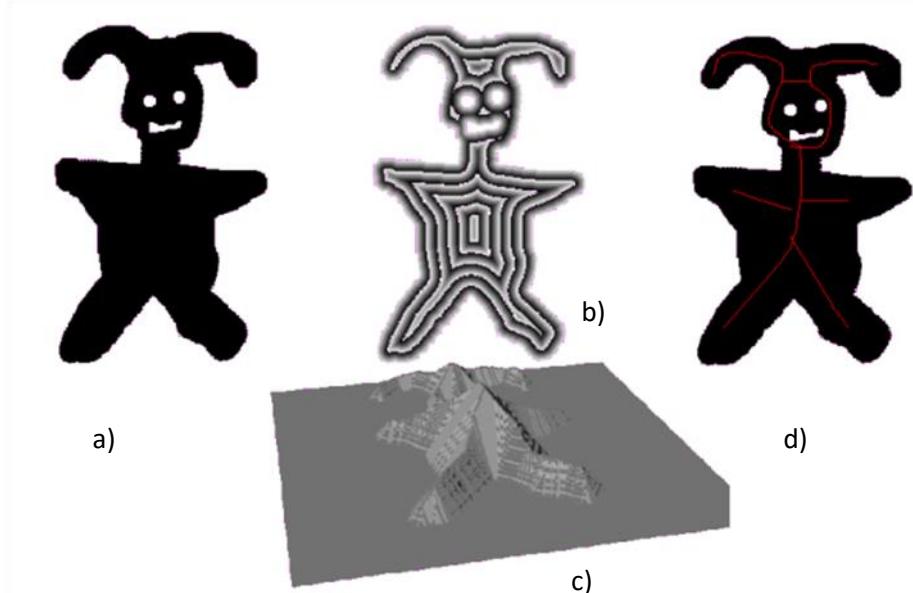
Aplikujme na binárny obraz operáciu erózie dovtedy, pokým zmiznú všetky objekty, zároveň priraďujeme každému pixelu z množiny X veľkosť prvej erózie množiny, ktorá už neobsahuje pixel p . Hodnotou pixela je teda najkratšia vzdialenosť od okraja a doplnkom množiny X^c , teda:

$$\forall p \in X, \quad dist_X(p) = \min\{n \in N, p \notin (X \ominus nB)\} \quad (109)$$

Obrázok 61c) zobrazuje výsledok takejto operácie a je z neho zrejmý aj spôsob ako zo vzdialenosnej transformácie dostaneme skeleton.

9.8.1.1 OpenCV

`void distanceTransform(InputArray src, OutputArray dst, int distanceType, int maskSize)`. Táto funkcia realizuje vzdialenosťnú transformáciu vstupného binárneho obrazu `src`. Je pritom možné použiť rôzne typy výpočtu vzdialenosťi `distanceType` a veľkosti masky `maskSize`.



Obrázok 61. a) Binárny obraz, b) opakovaná erózia, c) topologická reprezentácia vzdialenosnej transformácie, kde výška predstavuje vzdialosť od najbližšieho okraja, d) skeleton ako množina lokálnych maxim vzdialenosnej transformácie.

Zdroj obrázku: David Coeurjolly, Laboratoire LIRIS, Université Claude Bernard Lyon 1

Morfologická segmentácia (watershed)

Podobný postup ako pri výpočte skeletoru vieme využiť aj pri segmentácii založenej na predstave topológie a analógie s vodnou hladinou. Kvapka vody, ktorá dopadne na terén sa posúva smerom k najhlbšiemu miestu – lokálnemu minimu. Lokálne minimum predstavuje rezervoár vody (bazén), ktorý sa takto postupne plní. Po jeho naplnení voda začne pretekáť do susedného bazéna a miesto pretekania tvorí prirodzenú hranicu medzi lokálnymi minimami.

Je zrejmé, že výsledkom tohto postupu je veľké množstvo segmentovaných objektov, z ktorých potom musíme vylučovať tie falošné resp. menej dôležité. Preto je veľmi dôležitým vstupom algoritmu tzv. Marker čiže obraz obsahujúci značky na miestach, o ktorých určite vieme, že patria objektu. Tento obraz značiek môžeme zadať aj interaktívne (kliknutím myšou), väčšinou je ale výsledkom nejakého algoritmu predspracovania.

9.8.1.2 OpenCV

`void watershed(InputArray image, InputOutputArray markers)`

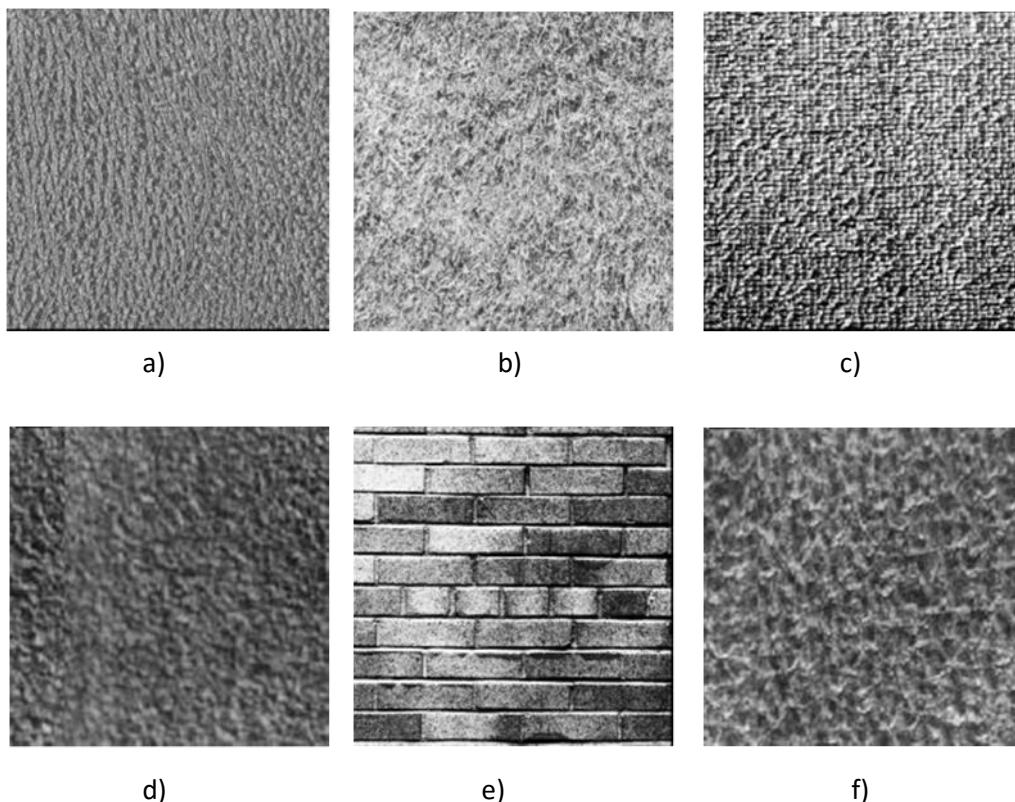
Táto funkcia očakáva na vstupe obraz markers pozostávajúci zo značiek 1, 2, ... atď. ktoré označujú pixely určite patriace objektom. Ostatné pixely obrazu markers sú nulové. Po ukončení segmentácie budú nulovým pixelom priradené značky objektov ku ktorým patria.

10Textúry a ich vlastnosti

10.1 Základné vlastnosti textúr

10.1.1 Definícia textúr

Textúra je (často intuitívne) označenie pre triedu nefigurálnych obrazov, ktoré charakterizujú spravidla povrch objektu, a ktoré môžeme nájsť prakticky u všetkých obrazov (viď Obrázok 63). Prvky, z ktorých sa textúra skladá sa nazývajú **textúrne primitíva** (textúrne elementy, angl. texel). Nie je jednoduché určiť primitíva – niekedy majú veľkosť iba o málo väčšiu než je rozmer pixelu a niekedy sa periodicitu zoskupení prejavuje na viacerých úrovniach rozlíšenia. Tak napríklad u pletenej textílie je primitívom jednak samotné vlákno prechádzajúce z jednej strany textílie na druhú, ale pri nižšom rozlíšení sa môže objaviť pravidelný vzor (ornament) úmyselne vytváraný pri pletení. Z toho vyplýva, že popis textúry je **škálovo závislý**.

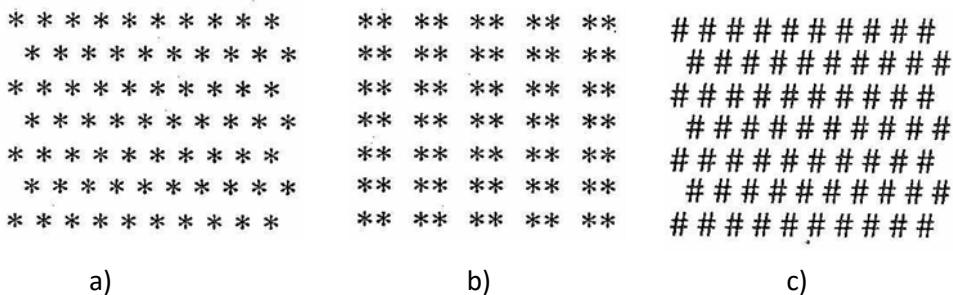


Obrázok 62. Príklady rôznych textúr:

a) tel'acia koža, b) tráva, c) ráfiová rohož, d) korok, e) tehlová stena, f) prasačia koža.

10.1.2 Textúrne vlastnosti

Pod **tónom** rozumieme tónové vlastnosti primitív z hľadiska celkového štruktúrneho usporiadania primitív v celom obraze. Pojmom **štruktúra** chápeme plošné usporiadanie primitív s ohľadom na ich tónové vlastnosti.



Obrázok 63. a), b) – rovnaké primitíva v rôznom usporiadaní,
a), c) – rôzne primitíva v rovnakom usporiadani.

10.1.3 Rozdelenie textúr

- **Jemná textúra** – rozmiestnenie primitív obrazu je náhodné a tónové rozdiely medzi susednými malými primitívami sú veľké.
- **Hrubá textúra** – rozmiestnenie primitív má viac určitosti a primitíva obsahujú viac buniek.
- **Slabé textúry** – majú malé plošné interakcie medzi primitívami.
- **Silné textúry** – ich plošné interakcie sú pomerne pravidelné.
- **Konštantná textúra oblasti obrazu** – súbor miestnych vlastností je v tejto oblasti konštantný, pomaly sa meniaci alebo približne periodický.

10.2 Metódy pre popis textúr

Poznáme dve hlavné skupiny metód:

- **Príznakové:** využívané pre textúry, ktorých veľkosti primitív sú porovnateľné s veľkosťami obrazových elementov. Tieto metódy budú popisované vo zvyšnej časti tejto kapitoly.
- **Syntaktické:** využívajú na vyhodnocovanie zvyčajne pravidlá formálnej gramatiky, vyhovujú textúram, v ktorých môžeme primitíva popísat väčším počtom vlastností ako sú len tónové vlastnosti. V praxi sa používajú menej často a preto ich nebudeme v ďalšom spomínať.

10.2.1 Charakteristika

Každej rozpoznávanej textúre je jednoznačne priradený bod vo viacrozmerskom priestore príznakov. Úloha spočíva vo vytvorení rozhodovacieho pravidla, priradujúceho textúru k triede.

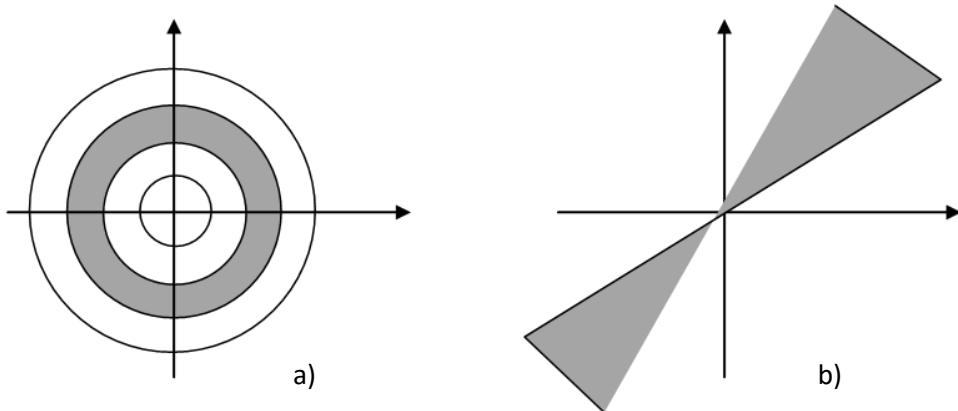
10.2.2 Metódy založené na určovaní plošných frekvencií

Vychádzajú zo skutočnosti, že charakter textúry je v priamom vzťahu s plošnou veľkosťou textúrnych primitív - pre hrubšie textúry sú príznačné väčšie primitíva, pre jemnejšie textúry menšie primitíva.

Autokorelačná funkcia textúry je najjednoduchším prípadom, kde textúrnemu primitívu zodpovedá 1 pixel a tónu jeho jas. Priestorovú organizáciu textúry určuje korelačný koeficient. Pri relatívne veľkých primitívach sa hodnota autokorelačnej funkcie s rastúcou vzdialenosťou, pri malých primitívach sa zmenšuje. V prípade kruhovo symetrickej funkcie je autokorelačná funkcia nezávislá na smere – stáva sa funkciou 1 premennej.

$$C_{ff}(p, q) = \frac{MN}{(M-p)(N-q)} \frac{\sum_{i=1}^{M-p} \sum_{j=1}^{N-q} f(i, j)f(i+p)(j+q)}{\sum_{i=1}^M \sum_{j=1}^N f^2(i, j)} \quad (110)$$

Fourierová transformácia je prirodzeným nástrojom na posudzovanie spektrálnych vlastností obrazov a ich častí - teda aj texelov.



**Obrázok 64. a) Všesmerový filter vo Fourierovskej oblasti,
b) smerový filter vo Fourierovskej oblasti.**

Na obrázku a) je klasický Fourierovský všesmerový filter. V okolí počiatku súradného systému sú nízke frekvencie. Čím ďalej od stredu sa vzdialujeme, tým vyššie frekvencie súradnice reprezentujú. Tmavo vyfarbené medzikružie teda predstavuje pásmový filter. Keď takýto filter „priložíme“ na Fourierovský obraz skúmaného obrazu a nastavíme všetky pixely „pod“ medzikružím na nulu. Tým dostaneme po spätej transformácii obraz zbavený frekvencií v určitom pásme. Keď takúto Fourierovskú masku využijeme nie na filtrovanie, ale ako nástroj na vyšetrenie textúr, budeme vedieť zistiť, ktoré frekvencie v obraze dominujú, a teda aká je dominantná frekvencia opakovania primitív. Použitím masky na obrázku b) vieme zasa určiť dominantné smerové charakteristiky. Pokiaľ sa pod vyfarbenú časť filtra dostane po Fourierovskej transformácii dostane väčšina svetlých bodov znamená to, že v smere orientácie vyfarbenej časti je výrazná periodicitá – jej frekvenciu určuje vzdialenosť od stredu.

10.2.3 Kookurenčná matica (coocurrence matrix)

Spôsob výpočtu znázorňuje nasledovný popis:

$$\begin{aligned}
 P_{0^\circ,d}(a,b) &= |\{(k,l), (m,n) \in D : |k-m|=0, |l-n|=d : f(k,l)=a, f(m,n)=b\}| \\
 P_{90^\circ,d}(a,b) &= |\{(k,l), (m,n) \in D : |k-m|=d, |l-n|=0 : f(k,l)=a, f(m,n)=b\}| \\
 P_{45^\circ,d}(a,b) &= |\{(k,l), (m,n) \in D : |k-m|=d, |l-n|=-d \vee |k-m|=-d, |l-n|=d : f(k,l)=a, f(m,n)=b\}| \\
 P_{135^\circ,d}(a,b) &= |\{(k,l), (m,n) \in D : |k-m|=d, |l-n|=d \vee |k-m|=-d, |l-n|=-d : f(k,l)=a, f(m,n)=b\}|
 \end{aligned}$$

$$P_{135^\circ,d}(a,b) = |\{(k,l), (m,n) \in D : |k-m|=d, |l-n|=d \vee |k-m|=-d, |l-n|=-d : f(k,l)=a, f(m,n)=b\}|$$

kde $d=1$ (vzdialenosť), horizontálny smer P_0 , $D = (M \times N) \times (M \times N)$

$a, b = 0..255$ (počet úrovní jasu), $k, m, l, n = 1..M, 1..N$

Príklad na výpočet kookurenčnej matice:

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

a)

4	2	1	0
2	4	0	0
1	0	6	1
0	0	1	2

b)

2	1	3	0
1	2	1	0
3	1	0	2
0	0	2	0

c)

Obrázok 65. a) Matica predstavujúca pôvodný obraz. b) Kookurenčná matica $P_{0^\circ,1}$.
c) Kookurenčná matica $P_{135^\circ,1}$.

Napríklad $P_{0^\circ,1}(0,0) = 4$. Znamená to, že prvok matice o súradničach (0,0) reprezentuje početnosť objavenia sa dvojice s hodnotou 0, pričom vzdialenosť medzi nimi je 1 (bezprostrední horizontálni susedia). Analogicky napr. $P_{0^\circ,1}(3,2) = 1$, čiže dvojica horizontálnych susedov (3,2) sa v matici obrazu objaví iba raz (samozrejme kvôli symetrii aj $P_{0^\circ,1}(2,3) = 1$). Medzi dobré vlastnosti tejto metódy patrí, že priamo popisuje vzájomné priestorové rozloženie primitív, a že je odolná voči monotónnym zmenám úrovne jasu. Nevýhodou je značná náročnosť na pamäť a tiež že nepopisuje celkový tvar primitív, a preto sa nehodí na väčšie primitíva zložitejšieho tvaru.

10.2.4 Kritéria odvodené z kookurenčnej matice

10.2.4.1 Energia:

(čím homogénnejší obraz, tým väčšia hodnota)

$$\sum_{a,b} P_{\phi,d}^2(a,b) \quad (111)$$

10.2.4.2 Entropia

$$\sum_{a,b} P_{\phi,d}(a,b) \log_2 P_{\phi,d}(a,b) \quad (112)$$

10.2.4.3 Maximálna pravdepodobnosť

$$\max_{a,b} (P_{\phi,d}(a,b)) \quad (113)$$

10.2.4.4 Kontrast

Miera lokálnych zmien v obraze – prudké zmeny jasu. Zvyčajne $k=2, \lambda=1$

$$\sum_{a,b} |a-b|^k P_{\phi,d}^\lambda(a,b) \quad (114)$$

10.2.4.5 Korelácia

Miera linearity obrazu v danom smere. Ak obraz obsahuje veľa lineárnych objektov v danom smere, korelácia je vysoká.

$$\frac{\sum_{a,b} [(a,b) P_{\phi,d}(a,b)] - \mu_x \mu_y}{\sigma_x \sigma_y} \quad (115)$$

10.2.5 Početnosť hrán v textúre

Porovnávame početnosť hrán textúry. Hrany môžu byť detekované jednak ako „mikro-hrany“, ale aj ako „makro-hrany“. Detektor hrán (napr. Robertsov alebo iné) musí v sebe zohľadňovať aj vzdialenosť pixelov.

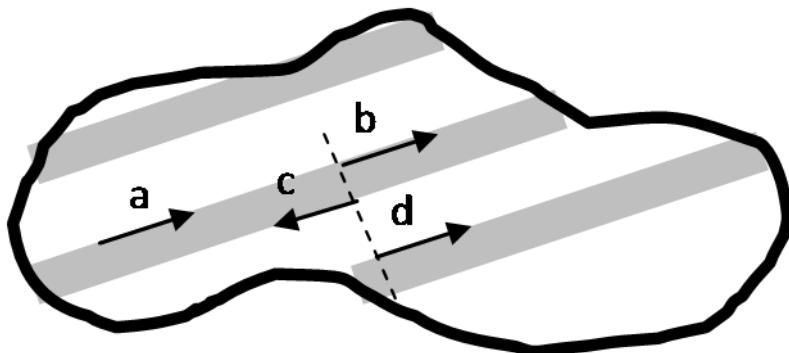
Príkladom takého prístupu je funkcia $g(d)$, ktorá popisuje textúru zohľadňujúc pritom vzdialenosť d :

$$g(d) = |f(i, j) - f(i + d, j)| + |f(i, j) - f(i - d, j)| + \\ |f(i, j) - f(i, j + d)| + |f(i, j) - f(i, j - d)| \quad (116)$$

Táto funkcia je podobná autokorelačnej funkcií, avšak má minimum tam, kde má autokorelačná funkcia maximum a naopak.

Na základe hustoty a iných vlastností hrán môžeme charakterizovať niektoré ďalšie parametre textúry ako napr.:

- Hrubosť (coarseness) – textúra je tým jemnejšia, čím je vyšší počet hrán vo vyšetrovanej časti obrazu.
- Kontrast. Vysoko kontrastné obrazy sú charakterizované vysokou hodnotou hrany (gradientu).
- Náhodnosť – dá sa vyjadriť ako entropia histogramu gradientného obrazu.
- Smerovosť – pokial väčšina hrán má podobný smer – vieme to posúdiť na základe histogramu smerov hrán.
- Linearita – ak sa podarí detegovať dvojice hranových bodov ležiacich na tej istej priamke v konštantnej vzdialnosti od seba (prípady a), b) na Obrázok 66).
- Periodicita – ak nachádzame dvojice hranových bodov nachádzajúcich sa v konštantnej vzdialosti od seba, avšak v smere kolmom na smer hrán. (prípady b, d.).
- Veľkosť texelu je určená dvojicou hranových vektorov opačného smeru ležiace na kolmici voči smeru hrany v konštantnej vzdialosti (vektory b, c).



Obrázok 66. Definovanie pojmov linearity, periodicity a veľkosti.

10.2.6 Dĺžka primitív.

Pomerne častým prípadom je, že primitíva majú podlhovastý tvar, teda viacero pixelov v určitom smere má podobný jas a a tvorí lineárnu štruktúru určitej dĺžky r . Keď máme obraz rozmerov $M \times N$,

pričom pixely môžu nadobudnúť L hodnôt jasu. Keď N_r je maximálna povolená dĺžka primitíva, K je počet primitív:

$$K = \sum_{a=1}^L \sum_{r=1}^{Nr} B(a, r) \quad (117)$$

potom vieme zdôrazniť krátke primitíva:

$$\frac{1}{K} \sum_{a=1}^L \sum_{r=1}^{Nr} \frac{B(a, r)}{r^2} \quad (118)$$

zdôraznenie dlhých primitív:

$$\frac{1}{K} \sum_{a=1}^L \sum_{r=1}^{Nr} B(a, r) r^2 \quad (119)$$

rovnomernosť jasu:

$$\frac{1}{K} \sum_{a=1}^L \left[\sum_{r=1}^{Nr} B(a, r) \right]^2 \quad (120)$$

rovnomernosť dĺžky primitív:

$$\frac{1}{K} \sum_{r=1}^{Nr} \left[\sum_{a=1}^L B(a, r) \right]^2 \quad (121)$$

10.2.7 Iné metódy popisu textúry.

V odbornej literatúre bolo publikovaných množstvo iných metód popisu textúry využívajúcich napr. fraktály, hierarchické dátové štruktúry (napr. pyramídy a wavelety), matematickú morfológiu, autoregresný model a pod. Ich podrobnejší popis by však vyžadoval ďaleko väčší priestor prekračujúci daný rozsah predmetu.

10.2.7.1 OpenCV

Knižnica sice nemá priame nástroje na hodnotenie textúr, poskytuje však viaceré funkcie z oblasti detekcie príznakov, štatistiky obrazu a strojového učenia, ktoré sú k tomu potrebné.

void calcBackProject(const Mat* images, int nimages, const int* channels, InputArray hist, OutputArray backProject, ...). Táto funkcia má na vstupe histogram určitej časti obrazu (napr. texelu). Algoritmus skúma jednotlivé časti obrazu images a porovná ich histogram so vstupným histogramom hist. To predstavuje mieru pravdepodobnosti ktorú priradí príslušnému pixelu výstupného obrazu backProject.

11 Analýza pohybu

Rozvojom videotechniky na jednej strane a výpočtovej kapacity na strane druhej akceleruje vývoj v oboch oblastiach. Existuje viacero klasických nosných problémov ako napr.:

- Detekcia samotného pohybu – statická kamera sleduje určitú scénu a snaží sa zistiť, či na nej nedošlo k nežiaducemu pohybu. Pritom systém nesmie reagovať na náhodné a prirodzené pohyby prostredia. Táto problematika je najjednoduchšia a najstaršia, keďže našla v praxi rozsiahle uplatnenie pri strážení objektov pomocou kamery.
- Detekcia a lokalizácia pohybujúceho sa predmetu. Oproti predošej kategórii má algoritmus zvyčajne za úlohu nielen detegovať pohyb, ale aj určiť jeho trajektóriu a tiež predpovedať smer ďalšieho pohybu. Uplatňuje sa hlavne pri analýze pohybu mrakov a predpovede počasia, sledovanie premávky vozidiel v mestách, riadenia autonómneho vozidla, sledovanie pohybujúceho sa objektu pomocou satelitu a pod. Medzi najzložitejšie úlohy tejto kategórie patria tie, pri ktorých sa môže pohybovať objekt aj kamera súčasne.
- Určenie vlastností 3D objektu zo série snímok pohybujúceho sa objektu. Tu sa využíva skutočnosť, že pri pohybe kamery je možné získať sériu 2D obrazov a na základe nich rekonštruovať 3D objekty. Keďže každý snímok bol získaný v inom čase, reprezentuje iný pohľad na pohyblivý 3D objekt.

Okrem spomínaných oblastí neustále pribúdajú nové, takže akákoľvek snaha o ich vymenovanie by bola kontraproduktívna.

11.1 Diferenčná metóda

Je najjednoduchšou metódou, ktorá iba registruje, že došlo k nejakej zmene medzi dvoma po sebe nasledujúcimi snímkami. Z rozdielu oboch obrazov vytvoríme binárny obraz, kde výrazne odlišné body budú mať hodnotu 1 (z praktických dôvodov sa väčšinou miesto hodnoty 1 používa 255).

$$\begin{aligned} d(i, j) &= 0 \quad \text{ak } |f_1(i, j) - f_2(i, j)| \leq \varepsilon \\ d(i, j) &= 1 \quad \text{inak} \end{aligned} \tag{122}$$

Medzi dvoma po sebe nasledujúcimi snímkami f_1 a f_2 môže nastať niekoľko situácií:

1. $f_1(i, j)$ je pixel pohyblivého objektu, $f_2(i, j)$ je pixel statického pozadia (resp. naopak),
2. $f_1(i, j)$ je pixel pohyblivého objektu, $f_2(i, j)$ je pixel iného pohyblivého objektu,
3. $f_1(i, j)$ je pixel pohyblivého objektu, $f_2(i, j)$ je pixel inej časti toho istého pohyblivého objektu.

Diferenčná metóda v princípe nevie odhaliť smer pohybu. Tento nedostatok čiastočne odstraňuje nasledovná metóda založená na kumulatívnom princípe.

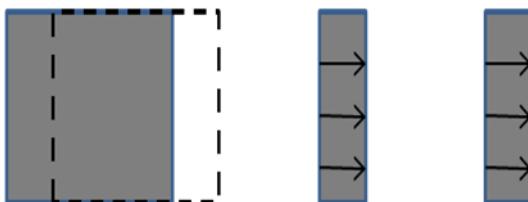
Kumulatívny diferenčný obraz zahrňuje viacero snímok a vyjadruje rozdiely oproti prvému (referenčnému obrazu).

$$d_{cum}(i, j) = \sum_{i=1}^n a_k |f_1(i, j) - f_k(i, j)| \tag{123}$$

kde a_k predstavuje váhu (dôležitosť) konkrétnej snímky (neskoršie snímky majú zvyčajne väčšiu váhu z hľadiska odhadu budúcej trajektórie).

Diferenčná metóda je najjednoduchšia a dobre pochopiteľná, no nemá veľký praktický význam kvôli viacerým obmedzeniam. Jedným z problémov je zrejmý z Obrázok 67, kde sú detektovateľné iba časti v smere pohybu a vôbec nevidno časti paralelné so smerom pohybu.

Na vylepšenie nedostatkov sa používajú rôzne metódy – napr. miesto porovnávania pixelov porovnávame tzv. **superpixely** – body zahrnujúce v sebe vlastnosti susedov.



Obrázok 67. Diferenčná metóda a jej nedostatky.

Metóda pohybujúcich sa hrán (Moving edges) kombinuje obraz $D(i,j)$ získaný diferenčnou metódou a obraz $S(i,j)$, ktorý obsahuje hrany po sebe nasledujúcich obrazov. Hrany sa získajú ľubovoľným hranových detektorom a do obrazu $S(i,j)$ sa premietnú pomocou operácie AND, ktorý takto obsahuje hranice oboch vyšetrovaných obrazov. Potom výsledok vyzerá nasledovne:

$$d_{med}(i,j) = S(i,j)D(i,j) \quad (124)$$

11.1.1 Modelovanie pozadia

Spomínané algoritmy založené na odčítaní obrazu od referenčného obrazu pozadia (pričom referenčná snímka môže byť aj predchádzajúca) budú fungovať iba v najjednoduchších prípadoch. V praxi však pozadie nie je konštantné a taktiež vykazuje určité zmeny. Preto musíme pre každý pixel pozadia tolerovať určitú úroveň zmien, čiže vytvoriť **model pozadia**. Najjednoduchším modelom je aritmetický priemer (alebo medián) hodnôt v jednotlivých snímkach. Tento model je nutné pravidelne aktualizovať (napr. miesto obyčajného priemeru použiť kĺzavý aritmetický priemer z posledných N snímkov).

Zložitejšie modely pozadia si vedia poradiť aj s rýchlejšími periodickými zmenami pozadia ako je šum, pohyb listov v dôsledku vetra, zmeny tieňa a pod. Napr. miesto jedného priemeru použijeme dva reprezentujúce horný a dolný jasový limit, v ktorom ešte považujeme pixel za pozadie.

Gaussovské modely využívajú okrem aritmetického priemeru aj rozptyl jasov. Priemer a smerodajná odchýlka definujú Gaussovskú krivku a my vieme spočítať pravdepodobnosť, s akou aktuálna hodnota jasu pixelu patrí do pozadia resp. či reprezentuje štatisticky významnú zmenu (objekt). Samozrejme, aj tu musí algoritmus zabezpečiť pravidelnú aktualizáciu modelu na základe vlastnosti dát.

11.1.1.1 OpenCV

trieda `BackgroundSubtractorMOG` predstavuje Gaussovský model pozadia (Mixture of Gaussian).
Vytvoríme objekt tejto triedy - napr. `MOG` a pri každej novej snímke `image` voláme `MOG(InputArray image, OutputArray fgmask, double learningRate=0)`. Volanie tejto funkcie model aktualizuje a vráti obraz `fgmask` predstavujúci objekty.

11.2 Detekcia bodov vzájomnej korešpondencie.

Málokedy má význam porovnávať všetky pixely dvoch obrazov, tak ako to je popísané v predchádzajúcich častiach. Keďže v drvinej väčšine praktických aplikácií sledujeme pohyb objektov s danou konšteláciou pixelov, je výhodnejšie nájsť a porovnávať na oboch obrazoch iba výrazné a dobre detekovateľné črty (pixely). Tieto body by nemali byť výsledkom náhodných procesov (šum v homogénnych častiach obrazu). Obrázok 67 demonštruje, že ani hrany nie sú dobrými kandidátmi, keďže nevieme identifikovať pohyb v smere hrany. Oveľa lepšími kandidátmi sú rohy (corners), ktoré vieme nájsť rôznymi spôsobmi.

Harrisov detektor

Nech vektor $[u, v]$ predstavuje bod z okolia určitého bodu p . Harrisov detektor hľadá priemernú (smerovo závislú) zmenu intenzity jasu v určitom okne (okoli) bodu p .

$$R = \sum (I(x+u, y+v) - I(x, y))^2 \quad (125)$$

Použitím rozvoja do Taylorovho radu a prepísaním do tvaru matice dostaneme

$$H(p) = \begin{pmatrix} \sum \left(\frac{\delta I}{\delta x} \right)^2 & \sum \frac{\delta I}{\delta x} \frac{\delta I}{\delta y} \\ \sum \frac{\delta I}{\delta x} \frac{\delta I}{\delta y} & \sum \left(\frac{\delta I}{\delta y} \right)^2 \end{pmatrix} \quad (126)$$

Táto matica je kovariačnou maticou, ktorá popisuje rýchlosť smerovej zmeny intenzity. Harrisov detektor výrazne reaguje na body, v ktorých sa stretáva horizontálna a vertikálna hrana, čiže na rohové body.

Uvedený detektor je základom množstva modifikácií ktoré možno nájsť v literatúre (napr. subpixelová detekcia rohov, odolnosť voči rotácii, aplikácia rôznych váh a pod.). Keďže Harrisov detektor produkuje veľké množstvo rohových bodov, ktoré sú často nahostené na jednom mieste obrazu, viacero modifikácií rieši tento problém zadáním určitej minimálnej vzdialenosť medzi rohmi.

11.2.1.1 OpenCV

`void cornerHarris(InputArray src, OutputArray dst, int blockSize, ...)`

Funkcia nájde na vstupnom obraze `src` rohové body a umiestní ich do výstupného obrazu `dst`. `blockSize` udáva veľkosť prehľadávacieho okna.

`void goodFeaturesToTrack(InputArray image, OutputArray corners, int maxCorners, double qualityLevel, double minDistance, ...)` táto funkcia používa Harrisov detektor avšak umožňuje zadáť požadovanú kvalitu a minimálnu vzdialenosť rohov.

`void cornerSubPix(InputArray image, InputOutputArray corners, ...)` vstupom je obraz `image` a celočíselné súradnice rohov nájdené klasickými detektormi. Výsledkom sú rohy `corners` so subpixelovou presnosťou.

11.2.2 Škálovo invariantné detektory príznakov (SURF a SIFT)

Pri porovnávaní obrazov v čase musíme brať do úvahy zmeny sledovaných objektov v dôsledku rôznych faktorov. Jedným z najdôležitejších je zmena škály pri snímaní (videozáZNAM približujúceho sa auta pri ktorom sa zväčšuje vzdialenosť medzi reflektormi).

V časti venovanej konvolučným filtrom boli uvedené príklady (napr. Gaussovský filter), pri ktorom jeho koeficienty aproximovali Gaussovou krivku s určitou šírkou (danou hodnotou σ). Podobne aj koeficienty detektorov hrán (napr. Laplacian) predstavujú šírku filtra danú hodnotou σ . Považujme teda σ za premennú a aplikujme detektor rohov s meniacou sa hodnotou σ na obraz. Pri určitej hodnote je hodnota odozvy detektora najvyššia. Keby sme zmenili škálu celého obrazu (napr. na polovicu), optimálna odozva by bola pri polovičnej hodnote pôvodnej optimálnej σ .

Škálovo invariantný detektor rohov $H(x,y, \sigma)$ teda musí prehľadávať obraz nielen pre rôzne hodnoty polohy (x,y) , ale aj škály (σ).

Na uvedenom princípe pracujú aj detektory príznakov **SURF** (Speeded Up Robust Features) a **SIFT** (Scale Invariant Feature Transform). Výsledkom ich práce je poloha príznaku (x,y) aj škála t.j. veľkosť okna okolo (x,y) , v ktorej by mal príznak platiť, aj keď sa objekt priblíži. Dobre vybraný príznak mal by okrem toho zaručovať aj stabilitu pri premenlivom osvetlení, pri malých deformáciách v dôsledku zmeny perspektívy a mal by sa dať ľahko porovnávať s inými príznakmi jednoduchou metrikou (napr. Euklidovská vzdialenosť).

11.2.2.1 OpenCV

Oblast' robustných detektorov príznakov je v OpenCV rozsiahla a stále vylepšovaná. Bola vytvorená univerzálna trieda FeatureDetector ktorá nájde príznaky (polohu, škálu, orientáciu, kvalitu atď). Pri vytváraní detektora môžme zvoliť jeho typ (napr. FAST, STAR, SIFT, SURF, ORB, MSER, GFTT, HARRIS, ...).

11.3 Optický tok

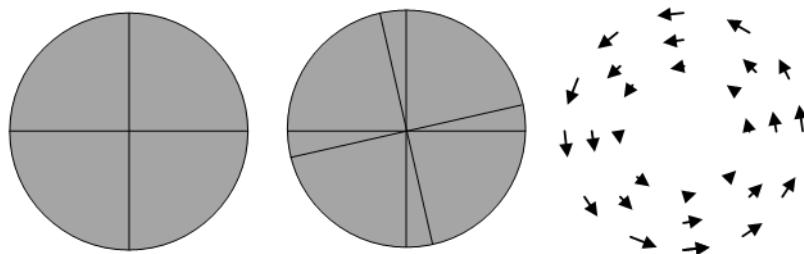
Optický tok je forma analýzy pohybu, kde sa predpokladá taký malý časový interval medzi jednotlivými snímkami, že nedôjde s žiadnej významnej zmene smeru alebo rýchlosťi pohybu. Podľa toho, či využívame všetky pixely porovnávaných obrazov alebo iba vybrané príznaky (napr. rohy), poznáme hustý a riedky optický tok.

Analýza pohybu využíva niekoľko zjednodušujúcich predpokladov:

1. Maximálna rýchlosť je obmedzená na hodnotu C_{max} . Pohybujúci sa objekt snímaný v časových intervaloch dt , teda bude vždy vnútri kruhu ohraničujúceho maximálnu vzdialenosť, kam sa mohol dostať pri svojom pohybe od poslednej snímky.
2. Malá akcelerácia (kladná alebo záporná) pohybujúceho sa objektu.
3. Spoločný pohyb všetkých bodov objektu.
4. Vzájomná korešpondencia – určitému bodu pevného telesa zodpovedá konkrétny bod na nasledovnej snímke a naopak. Výnimku tvoria body, ktoré sú zakryté v dôsledku rotácie alebo prekrytie časťou objektu.

Vyjadruje zmeny obrazu v dôsledku pohybu a to v krátkom časovom intervale dt . Je založený na dvoch predpokladoch:

1. Jas každého pixelu pohybujúceho sa objektu sa s časom nemení.
2. Blízke okolie pohybujúceho sa pixelu sa pohybuje podobným spôsobom ako samotný pixel (tzv. velocity smoothness constraint).



Obrázok 68. Optický tok

Predpokladajme, že máme „dynamický obraz“, čiže spojité obrazové funkcie $I(x,y)$, kde x,y sú súradnice a t je čas. Podľa jedného z uvedených predpokladov sa nemení intenzita jasu v čase medzi dvoma snímkami. Potom hľadáme také posunutie (u,v) pre ktoré platí

$$I_t(x, y) = I_{t+1}(x + u, y + v) \quad (127)$$

Uvedený výraz môžeme rozložiť do Taylorovho radu:

$$I_{t+1}(x + u, y + v) \approx I_t(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} \quad (128)$$

dosadením predposledného vzťahu do posledného (čiže aplikovaním predpokladu o konštantnosti jasu) dostaneme:

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v = -\frac{\partial I}{\partial t} \quad (129)$$

čo je základná rovnica pre optický tok.

Algoritmus vypočíta optický tok tak, že pre všetky body (i,j) na základe hodnôt z predošej snímky vypočíta hodnoty nasledovnej snímky. Tento výpočet je iba zjednodušený hrubým odhadom, ktorý predpokladá vektor rýchlosť v smere maximálneho gradientu. V literatúre je popísaných viacero rozšírených interpretácií tohto pravidla, ktoré vedie k presnejším výpočtom. Jedným zo spôsobov je zavedenie podmienky „hladkého toku“, to znamená, aby sa vektor rýchlosť medzi jednotlivými snímkami menil iba pomaly. Na rovnici pre optický tok, ako aj na ďalších obmedzujúcich podmienkach vyplývajúcich z definície, je založených viacero algoritmov (napr. Lucas-Kanade).

Kedže algoritmy používajú na prehľadávanie okno konštantnej veľkosti, v praxi môže sa stať, že príliš malé okno nezaregistruje ten istý príznak v dvoch po sebe nasledujúcich snímkach. Z tohto dôvodu sa niekedy používa pyramidálna verzia algoritmov, ktorá hľadá príznaky na určitej hladine pyramídy a v prípade potreby prechádza na presnejšiu alebo hrubšiu verziu obrazu na inej hladine.

11.3.1.1 OpenCV

`void calcOpticalFlowPyrLK(InputArray prevImg, InputArray nextImg, InputArray prevPts, InputOutputArray nextPts, ...)` je pyramidálna verzia algoritmu Lucas-Kanade. Na základe dvojice

obrazov prevImg a nextImg ako aj množiny príznakových bodov prevPts spočíta novú množinu príznakov nextPts.

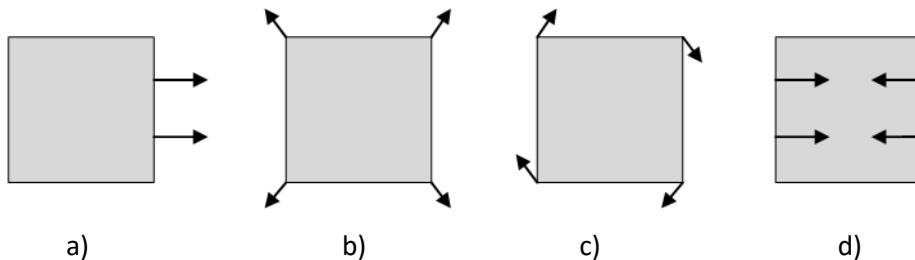
Existuje aj funkcia calcOpticalFlowFarneback - implementácia podobného algoritmu.

Na analýzu pohybu sa časti používa aj algoritmus int meanShift(InputArray probImage, Rect& window, TermCriteria criteria) resp. jeho modifikácia CamShift, ktoré boli už spomínané v súvislosti so segmentáciou. Ide o to, že obraz transformujeme na mapu vyjadrujúcu pravdepodobnosť, že pixely vyhovujú určitému vopred danému modelu (napr. histogram objektu ktorého pohyb sledujeme). Algoritmus najde polohu okna obsahujúceho pixely s vysokou pravdepodobnosťou. Kedže poznáme polohu takéhoto okna na predošej snímke, vieme nájsť trajektóriu okna v čase.

11.3.2 Použitie optického toku pri analýze pohybu

Obrázok 69 znázorňuje pohyb telesa voči pozorovateľovi, ktorý môžeme vyjadriť ako kombináciu nasledovných 4 akcií:

1. Posunutie v konštantnej vzdialnosti od operátora.
2. Posunutie do hĺbky voči pozorovateľovi. Pri tomto pohybe existuje jeden zdroj optického toku – tzv. Focus of Extension (FOE).
3. Rotácia v konštantnej vzdialnosti voči pozorovateľovi, pričom os rotácie je totožná s osou pohľadu.
4. Rotácia okolo osi ležiacej v rovine kolmej na os pohľadu pozorovateľa.

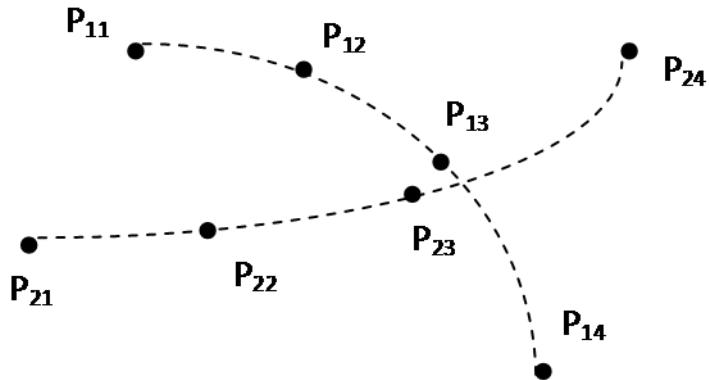


Obrázok 69. Možné spôsoby pohybu telesa voči pozorovateľovi.

11.4 Trajektória objektov a Kalmanov filter

11.4.1 Trajektória objektu

Doteraz spomínané postupy sú vhodné pre jeden alebo niekoľko málo pohybujúcich sa objektov. V prípade viacerých, nezávisle sa pohybujúcich objektov musíme sledovať individuálnu trajektóriu každého z nich.



Obrázok 70. Výpočet trajektórií viacerých objektov pohybujúcich sa nezávisle.

Takáto trajektória je však idealizovanou verziou skutočnej, nakoľko na pohyb vplývajú viaceré náhodné faktory. Preto takáto trajektória (aproximovaná nejakou funkciou) slúži ako odhad skutočného pohybu a musí byť doplnená procesom hodnotenia skutočnosti voči odhadu. **Funkcia koherentnosti trajektórie** (path coherence function) udáva do akej miery je trajektória koherentná, teda bez náhlych zmien smeru a rýchlosťi pohybu. Odchýlku od tejto trajektórie potom udáva tzv. **deviačná funkcia**.

Prekrytie objektov. Pri určitom počte nezávisle pohybujúcich sa objektov je takmer isté, že dôjde k ich vzájomnému prekrytiu (occlusion). Preto sa musí počítať aj s tým, že niektoré body trajektórie nebude možné verifikovať, nakoľko budú v určitom momente prekryté iným objektom. V takomto prípade budú skutočné body trajektórie nahradené tzv. **fantómovými bodmi**, ktoré dokážu zastúpiť na určitú dobu tie skutočné.

Odlišný prístup k odhadu trajektórie sa dá použiť aj tam, kde je pohyb dôsledkom určitého dobre definovaného procesu (napr. pohyb ľavej srdcovej komory). Vtedy je možné umiestniť na obrázok rovnomernú mriežku bodov (markerov), ktoré sa pohybujú spolu s objektmi, na ktorých ležia. Pri tomto pohybe dochádza k deformácii pravidelnosti mriežky, pričom posun každého bodu mriežky udáva trajektóriu v zodpovedajúcej časti obrazu.

11.4.2 Kalmanov filter

Predstavuje vhodný a často používaný model pre analýzu pohybu založený na dvojici prediktor – korektor, teda že odhad nasledujúceho stavu systému je nasledovaný pozorovaním (meraním) skutočnosti a odchýlka odhadu od skutočnosti je vzápäť použitá na korekciu odhadu pre ďalšie kroky. Príbeh „Odhad pozície (Kalmanov filter)“ to dokumentuje na príklade leteckej (prípadne námornej) navigácie, ktorá bola historicky motiváciou jeho vzniku.

Predpokladá, že systém je lineárny a pozorovania systému je možné tiež vyjadriť lineárnymi funkiami a naviac odhad i merania sú začlenené bielym šumom, ktorý má Gaussovú distribúciu okolo nulovej hodnoty. Z vlastností Gaussovskej distribúcie vyplýva, že ak máme dve merania charakterizované hodnotami priemeru a smerodajnej odchýlky (x_1, σ_1) a (x_2, σ_2), potom môžeme tieto dve merania kombinovať a výsledok je charakterizovaný hodnotou priemeru a smerodajnej odchýlky určenej vzťahmi:

$$\bar{x}_{12} = \left(\frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \right) x_1 + \left(\frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \right) x_2 \quad (130)$$

$$\sigma_{12}^2 = \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2} \quad (131)$$

Uvedené vlastnosti využijeme v úlohách predpovedania trajektórie tak, že (x_1, σ_1) predstavuje pravdepodobnosť polohy na predchádzajúcej snímke a (x_2, σ_2) na snímke novej. Potom hodnoty na ľavej strane predchádzajúcich dvoch rovníc predstavujú priemer a smerodajnú odchýlku nového **odhadu (predikcie)** budúcej pozície na základe predošej a súčasnej snímky. Ak tieto hodnoty vyjadríme v trochu upravenej podobe, kde strieška nad symbolom vyjadruje fakt že sa jedná o odhad:

$$\hat{x}_2 = \hat{x}_1 + \frac{\hat{\sigma}_1^2}{\hat{\sigma}_1^2 + \sigma_2^2} (x_2 - \hat{x}_1) \quad (132)$$

$$\hat{\sigma}_2^2 = \left(1 - \frac{\hat{\sigma}_1^2}{\hat{\sigma}_1^2 + \sigma_2^2} \right) \hat{\sigma}_1^2 \quad (133)$$

zavedieme kvôli zjednodušeniu hodnotu K vyjadrujúcu príspevok nového merania.

$$K = \frac{\hat{\sigma}_1^2}{\hat{\sigma}_1^2 + \sigma_2^2} \quad (134)$$

Dosadením do predošlých rovníc:

$$\hat{x}_2 = \hat{x}_1 + K(x_2 - \hat{x}_1) \quad (135)$$

$$\hat{\sigma}_2^2 = (1 - K) \hat{\sigma}_1^2 \quad (136)$$

Tieto rovnice vyjadrujú spôsob, ako upravíme model na základe hodnôt získaných z posledného merania. Ak má nové meranie veľký rozptyl (t.j. je nespoľahlivé), hodnota K bude malá a nové meranie na výsledný odhad bude mať na minimálny vplyv (čiže nový odhad bude takmer totožný s predošlým). Naopak, vysoká hodnota K zapríčiní výrazný vplyv nového merania na model (odhad).

Uvedené rovnice sú základom veľkého množstva aplikácií v rôznych oblastiach. V oblasti počítačového videnia často hodnota x predstavuje stavový vektor pozostávajúci zo 4 hodnôt $[x, y, v_x, v_y]^T$ (poloha a rýchlosť v horizontálnom i vertikálnom smere). Okrem toho obsahuje vektory umožňujúce externú korekciu stavového vektoru a tiež zložku predstavujúcu šum s Gaussovým rozložením okolo nulovej hodnoty. Konkrétnie vzťahy pre výpočet trajektórie pohybujúceho sa objektu vyžadovali by zavedenie ďalších vzťahov, ktoré sú dostatočne vysvetlené v príslušnej špecializovanej literatúre.

11.4.2.1 OpenCV

Na predikciu trajektórie pomocou Kalmanovho filtra bola vytvorená trieda Kalman. Funkcie predict(const Mat& control=Mat()) a correct(const Mat& measurement) vykonávajú operácie predikcie a korekcie.

12Literatúra

- [1] Sonka, M., Hlavac, V., & Boyle, R. (2008). Image processing, analysis, and machine vision. Toronto: Thomson.
- [2] Hlaváč, V., & Šonka, M. (1992). Počítačové vidění. Praha: Grada.
- [3] Szeliski, R. (2011). Computer Vision Algorithms and Applications. Springer Verlag.
- [4] Szeliski, R. (2011). Computer Vision Algorithms and Applications. Dostupné na Internete: <http://szeliski.org/Book/>
- [5] Svoboda, T., Kybic, J., & Hlaváč, V. (2007). Image Processing, Analysis, and Machine Vision: A MATLAB Companion. Thomson Learning
- [6] Bradski, G., & Kaehler, A. (2008). Learning OpenCV. Beijing ; Sebastopol, CA: O'Reilly.
- [7] Laganière, R. (2011). OpenCV 2 Computer Vision Application Programming Cookbook. Packt Publishing: Birmingham - Mumbai.
- [8] Russ, J. (2002). The image processing handbook (4th edition). Boca Raton: CRC Press.
- [9] OpenCV documentation. Dostupné na Internete: <http://docs.opencv.org/>
- [10] Hartley, R., & Zisserman, A. (2004). Multiple View Geometry in Computer Vision. Cambridge University Press.
- [11] Strharsky, I. (2006). Fotoalbum. Kosice.
- [12] Q. Wu, F. Merchant a K. Castleman, Microscope image processing, Amsterdam , Boston: Elsevier/Academic Press, 2008.

13 Návody na cvičenia s využitím knižnice OpenCV

Na cvičeniaci si precvičíme praktické využitie metód popísaných v predchádzajúcich kapitolách použitím knižnice OpenCV verzie 3.0.0 v programovacom jazyku C++.

13.1. Cvičenie 1. Príprava prostredia, priamy prístup k obrazu

Na tomto cvičení su ukážeme ako pripraviť prostredie pre vývoj aplikácií v OpenCV, ukážeme si priamy prístup k obrazovým dátam a základnú manipuláciu s nimi. Dokumentáciu ku knižnici nájdeme na stránke <http://docs.opencv.org/>.

13.1.1 Príprava prostredia (Windows)

Pre Windows je odporúčané používať už skompliovanú knižnicu OpenCV, ktorú si viete stiahúť zo stránky: <http://opencv.org/downloads.html> linkou OpenCV for Windows. Za ňou sa skýva samorozbaľovací archív, ktorý štandardne sa rozbaluje do zložky c:\opencv. Ako významná pomôcka sa často premenná prostredia `OPENCV_DIR`, ktorá uchováva cestu ku priečinku knižnice OpenCV. Najčastejšie používané vývojové prostredie pre Windows je Microsoft Visual Studio.

13.1.2 Nový projekt (Visual Studio)

Nastavenie nového projektu Visual Studio pre vývoj aplikácií s využitím OpenCV je kúsok zdĺhavejší proces popísaný v týchto krokoch (za predpokladu, že `OPENCV_DIR` je premenná prostredia, ktorá uchováva cestu k priečinku knižnice OpenCV):

1. FILE -> New -> Project...
2. Win32 -> Win32 Console Application
3. Next> -> zaškrtnúť Empty project a ostatné odškrtnúť
4. Project -> Properties (hore je výber, či nastavujete vlastnosti Debug alebo Release konfiguácie)
5. C/C++ -> Additional Include Directories:
 \$ (OPENCV_DIR) \build\include
6. Linker -> Additional Library Directories:
 \$ (OPENCV_DIR) \build\x86\vc11\lib
7. Linker -> Input -> Additional Dependencies:
v prípade Debug konfigurácie:
 opencv_world300d.lib
v prípade Release konfigurácie:
 opencv_world300.lib
 Debugging -> Environment:
 PATH=\$ (OPENCV_DIR) \build\x86\vc11\bin

13.1.3 Príprava prostredia (Ubuntu/Mint)

Pre Linuxové distribúcie sa odporúča inštalovať knižnice pomocou balíčkovacieho systému z repozitára. V čase písania týchto materiálov knižnica OpenCV v Ubuntu repozitári sa nachádzala vo verzii 2.4.x, čomu sme sa chceli vyhnúť. V Linixe odporúčame používať vývojové prostredie QT Creator. Budeme postupovať nasledujúcimi krokm:

1. pridajme si nový zdroj balíčkov ppa k repozitáru:
 sudo add-apt-repository ppa:amarburg/opencv3
2. obnovíme si zoznam balíčkov:
 sudo apt-get update
3. nainštalujeme knižnicu OpenCV 3.0:
 sudo apt-get install libopencv3-dev

4. (odporúčané) nainštalujme si vývojové prostredie QT Creator:
sudo apt-get install qtcreator

13.1.4 Nový projekt (QT Creator)

Nastavenie projektu prevývoj aplikácií s využitím knižnice OpenCV prebieha v týchto krokoch:

1. New Project -> Non-Qt Project -> Plain C++ Application -> ...
2. nájdeme v projekte **NázovProjektu.pro** a pridáme nasledujúce riadky:

V prípade Windowsu:

```
INCLUDEPATH += $${OPENCV_DIR}/build/include  
LIBS += -L$${OPENCV_DIR}/build/x86/vc11/lib libopencv_world300d
```

V prípade Linuxu:

```
CONFIG += link_pkgconfig  
PKGCONFIG += opencv
```

13.1.5 Kompilácia pomocou GNU Make

Kompilovať vieme pomocou kompilátora g++ nasledovne:

```
g++ source.cpp -o output `pkg-config opencv --cflags --libs`
```

13.1.6 Začíname

Na začiatok si pripojíme knižnicu OpenCV k hlavičke:

```
#include <opencv2/core/core.hpp> //jadro OpenCV  
#include <opencv2/highgui/highgui.hpp> //GUI
```

Metódy a triedy OpenCV sa nachádzajú v namespace cv:

```
using namespace cv;
```

Načítanie obrázku zo súboru:

```
Mat img = imread("linux_logo.jpg");
```

Zobrazenie obrázku na formulári:

```
namedWindow("cvtest"); //vytvorenie formulára  
imshow("cvtest", img); //zobrazenie obrázku  
waitKey(0); //čakanie na stlačenie klávesu
```

Zmena farby jedného pixelu (x = 50; y = 100 na modrú farbu RGB: 0, 0, 255):

```
img.at<Vec3b>(100, 50) = Vec3b(255, 0, 0);
```

Prechod obrázkom priamym prístupom k pamäti a pridanie farby (RGB: 30, 50, 70):

```
for(int y=0; y.rows; y++) {  
    Vec3b* row = img.ptr<Vec3b>(y);  
    for(int x=0; x.cols; x++) {  
        Vec3b color = row[x];  
        row[x] = Vec3b(color[0]+70, color[1]+50, color[2]+30);  
    }  
}
```

Uloženie výsledného obrázku na disk:

```
imwrite("negativ.jpg", img);
```

13.1.7 Úlohy

1. Načítajte, uložte obrázok zo/do súboru.
2. Zobrazte načítaný obrázok.
3. Zmeňte niekoľko pixelov obrázku.
4. Zmeňte jas obrázku.
5. Vytvorte negatív obrazku.

13.2 Cvičenie 2. Kreslenie a farebné priestory

Na tomto cvičení si vyskúšame získať obraz z videa, vytváranie obrázkov v pamäti, vyskúšame základné kresliace prvky a použijeme prechod medzi farebnými priestormi.

13.2.1 Získavanie obrazu z videa

O získavanie obrazu z videa sa stará trieda `VideoCapture`. Príklad:

```
VideoCapture cap(fileName);
if(!cap.isOpened()) return -1;

namedWindow("video");
while(true) {
    Mat frame;
    cap >> frame;
    if(!frame.data) break;
    imshow("video", frame);
    if(waitKey(40) >= 0) break;
}
```

kde `frame` je zísakný obraz, `waitKey(40)` je čakanie 40 ms a znamená, že prehrávame video v rýchlosťi 25 FPS.

13.2.2 Vytváranie obrázkov

Obrázok v OpenCV je reprezentovaný pomocou dátovej štruktúri matica `Mat`. Maticu štandardne vytvárame pomocou volania konštruktora:

```
Mat M(Size(320, 240), CV_8UC3, Scalar(0,0,255));
```

kde `Size(320, 240)` sú rozmery obrázku, `CV_8UC3` je typ obrázku a `Scalar(0,0,255)` je výplň obrázku. Typ obrázku je v tvare `CV_8U` alebo `CV_8UC3`, kde `C` a nasledujúce číslo hovorí o počte kanálov a može byť v rozsahu 1-4. Číslo 8 reprezentuje počet bitov uchovávajúcich číselnú hodnotu a to môže byť 8/16/32/64. Písmeno `U` znamená unsigned alebo cele číslo, čo može byť nahradené písmenom `F` čiže floating alebo desatinné číslo.

13.2.3 Základné kresliace prvky

Kresliace prvky, ktoré si ukážeme bude priamka, obdĺžník, kružnica a mnohouholník.

```
line(obraz, Point(x1, y1), Point(x2, y2), Scalar(b,g,r), hrubkaCiary);
circle(obraz, Point(x, y), polomer, Scalar(b,g,r), hrubkaCiary);
rectangle(obraz, Point(x1, y1), Point(x2, y2), Scalar(b, g, r),
hrubkaCiary);
```

Z predchádzajúceho kódu je zjavné ako použiť dané metódy. Ak zadáte záporné číslo `hrubkaCiary` v prípade kružnice alebo obdĺžníka, tak útvar bude vyplnený. Mnohouholník sa kreslí podobne, ale kvôli častému použitiu C++ dátovej štruktúry `vector` možeme nakresliť vyplnený mnohouholník takto:

```

void drawPolygon(Mat& image, vector<Point>& polygon, Scalar& color) {
    const Point* ppt[1] = {&polygon[0]};
    int poiCount[] = {polygon.size()};
    fillPoly(image, ppt, poiCount, 1, color);
}

```

13.2.4 Prechod medzi farebnými priestormi

O prechod medzi nimi sa stará metóda `cvtColor`:

```
cvtColor(vstupnyObraz, vystupnyObraz, kodPrechodu);
```

Najčastejšie použité kódy prechodu sú `CV_BGR2GRAY` na prechod do šedotónového obrazu, `CV_BGR2HSV` a `CV_BGR2HLS` na prechod do farebných priestorov HSV a HSL. Pre opačný prechod kód vyzerá `CV_HSV2BGR` a `CV_HLS2BGR`.

13.2.5 Úlohy

1. Načítajte video zo súboru po snímkach.
2. Vyskúšajte si základné kresliace funkcie.
3. Konvertujte obrázok do iných farebných priestorov (šedotónový obraz, HSL...).
4. Upravte farby obrazu zmenou jas, kontrastu, gamma korekcie.
5. Upravne farby obrazu zmenou sýtosti farieb, svetelnosti.
6. Spočítajte histogram obrázku a nakreslite ho.
7. Upravte farby obrázku pomocou rozťahnutia histogramu.

13.3 Cvičenie 3. Filtrovanie periodického šumu

Na cvičení si ukážeme práca s kanálmi obrazu, ako orámovanie a orezať obraz, robiť jeho maskované duplikáty, čo je základ pre filtrovanie periodického šumu.

13.3.1 Práca s kanálmi obrazu

Ak chceme pristupovať k jednotlivým farebným kanálom samostatne, tak máme nato metódu `split`:

```

Mat parts[3];
split(image, parts); //ak image má BGR zložky
//parts[0] = matica modrého kanálu
//parts[1] = matica zelenáho kanálu
//parts[2] = matica červeného kanálu

```

Rovnakým spôsobom dokážeme poskladať jednotlivé kanály do jedného obrázku (matice) pomocou metódy `merge`:

```

Mat parts[] = {bluePart, greenPart, redPart};
Mat output;
merge(parts, 3, output);

```

13.3.2 Orámovanie a orezanie obrazu

Nie vždy potrebujeme pracovať s celým obrázom, niekedy nám stačí iba malá časť. Pomocou konštruktora triedy `Mat` vieme extrahovať časť existujúceho obrázku (je to "in place", čiže jeho úpravou upravujeme pôvodný obrázok).

```
Mat vyrez = Mat(obraz, Rect(x, y, sirka, vyska));
```

Opačný prípad nastáva, ak potrebujeme mať z nejakého dôvodu vačšie rozmery obrazu (napr. ako pri rýchlej fourierovej transformácii) a nato použijeme metódu `copyMakeBorder`:

```
copyMakeBorder(obraz, vystup, hore, dole, vľavo, vpravo, BORDER_CONSTANT,  
Scalar::all(0));
```

kde hore, dole, vľavo a vpravo hovoria, koľko pixelov sa má pridať do danej strany a Scalar::all(0) hovorí, aké farby majú byť pridané pixely.

13.3.3 Maskované kopírovanie

Na kopírovanie časti obrazu máme metódu copyTo, ktorá patrí triede Mat:

```
obraz.copyTo(vystup, maska);
```

Maska obsahuje je jednokanálová matica, ktorá obsahuje 8-bitové celá čísla. Hodnota 255 znamená, že pixel z obrazu sa bude kopírovať a pri hodnote 0 sa nebude. V prípade, ak chcete kopírovať celú matiku, tak masku zadávať netreba.

13.3.4 Fourierová analýza

Pre spočítanie fourierovského pektra použijeme nasledujúcu metódu:

```
Mat computeFourierTransform(Mat& image) {  
    Mat paddedImage;  
    int newWidth = getOptimalDFTSize(image.cols);  
    int newHeight = getOptimalDFTSize(image.rows);  
    copyMakeBorder(image, paddedImage, 0, newHeight-image.rows, 0,  
                  newWidth-image.cols, BORDER_CONSTANT, Scalar::all(0));  
    Mat parts[] = {Mat_<float>(paddedImage),  
                  Mat::zeros(paddedImage.size(), CV_32F)};  
    Mat output;  
    merge(parts, 2, output);  
    dft(output, output, DFT_COMPLEX_OUTPUT);  
    return output;  
}
```

Vo výsledku fourierovej transformácie sú kvadranty šikmo vymenené, čiže nasledujúca metóda koriguje túto výmenu:

```
void shiftQuadrants(Mat& image) {  
    int cx = image.cols/2;  
    int cy = image.rows/2;  
  
    Mat q0(image, Rect(0, 0, cx, cy));  
    Mat q1(image, Rect(cx, 0, cx, cy));  
    Mat q2(image, Rect(0, cy, cx, cy));  
    Mat q3(image, Rect(cx, cy, cx, cy));  
  
    Mat tmp;  
    q3.copyTo(tmp);  
    q0.copyTo(q3);  
    tmp.copyTo(q0);  
  
    q2.copyTo(tmp);  
    q1.copyTo(q2);  
    tmp.copyTo(q1);  
}
```

Samozrejme výstup je zložený z reálnej a imaginárnej zložky, čo k našej analýze je postačujúca veľkosť komplexných čísel, načo použijeme nasledujúcu metódu:

```
void vizualize(Mat& complex){  
    Mat parts[2];  
    split(complex, parts);
```

```

    Mat distances;
    magnitude(parts[0], parts[1], distances);

    distances += Scalar::all(1);
    log(distances, distances);
    normalize(distances, distances, 1, 0, NORM_INF);

    shiftQuadrants(distances);
    imshow("fourier", distances);
}

```

Operáciu násobenia spektier má OpenCV implementovanú pod názvom mulSpectrums:

```
mulSpectrums(fourieroveSpektrum, komplexnaMaska, vystup, DFT_ROWS);
```

A napokon posledná užitočná metóda inverzná fourierová transformácia:

```

Mat obraz;
idft(fourieroveSpektrum, obraz);

```

13.3.5 Úlohy

1. Extrahujte jednotlivé kanále farebného obrazu.
2. Pridajte rám k obrázu.
3. Vytvorte výrez obrázu.
4. Pomocou masky skopírujte časť obrázu do druhého.
5. Spočítajte fourierovské spektrum obrazu.
6. Odfiltrujte periodický šum pomocou spodno prieplustného filtra na fourierovskom spektre.

13.4 Cvičenie 4. Generovanie šumu, konvolúcia a hranové detektory

Na tomto cvičení sa pozrieme na jednotlivé typy šumov a ako ich odstrániť konvolúciou. Zároveň sa budeme venovať hranovým detektorom.

13.4.1 Generovanie šumu

OpenCV má už predpripravené metódy na generovanie bieleho a gaussového šumu:

```

randu(obraz, min_hodnota, max_hodnota); //biely šum
randn(obraz, priemerna_hodnota, rozptyl); //gaussov šum

```

ktoré môžme pridať do obrazu ako aditívny alebo multiplikatívny šum:

```

obraz = obraz + šum; //aditívny šum
obraz = obraz * šum; //multiplikatívny šum

```

Manuálne generovanie šumu typu pepper & salt:

```

//sum pepper & salt
//pepper
for(int i=0; i<15000; i++){
    int x = rand() % obraz.cols;
    int y = rand() % obraz.rows;
    obraz.ptr<Vec3b>(y)[x] = Vec3b(0,0,0);
}
//salt
for(int i=0; i<15000; i++){
    int x = rand() % obraz.cols;
    int y = rand() % obraz.rows;
    obraz.ptr<Vec3b>(y)[x] = Vec3b(255,255,255);
}

```

13.4.2 Konvolúcia

Všeobecná operácia konvolúcie je implementovaná metódou `filter2D`, ktorá vyžaduje na vstup povačine floaty (kvôli násobeniu s maskou):

```
filter2D(Mat<float>(obraz), vystup, -1, maska);
```

kde parameter `-1` je typ výstupnej matice a znamená, že je zhodný so vstupom. Vytváranie konvolučnej masky po jednotlivých číslach môže byť zdlhavé, tak trieda `Mat` ma preťažený operátor `<<`, kde môžeme napísť jednotlivé hodnoty po riadkoch za sebou:

```
Mat maska = (Mat<float>(3,3) << 0, -1, 0, -1, 5, -1, 0, 1, 2);
maska *= 1.f/2;
```

Častou používané konvolučné operácie so známymi maskami implementované ďalšími metódami:

```
blur(obraz, vystup, Size(i, i)); //s priemerňovanie
GaussianBlur(obraz, vystup, Size(i, i), 0, 0); //gaussovské vyhľadenie
medianBlur(obraz, vystup, i); //medaánový filter
Laplacian(obraz, vystup, CV_8U); //laplaceov hranový detektor
Sobel(obraz, vystup, CV_32F, 1, 0); //sobelov hranový detektor
```

kde `i` znamená veľkosť masky, čísla 0 u gaussa znamenajú rozptyl v smere x a y a pri sobelovi je to smer hľadanej hrany. Mediánový filter tu bol zaradený kvôli spojitosťi so šumom a vyhľadenia šumu.

13.4.3 Úlohy

1. Generujte šum do obrázkov (aditívny, peper & salt).
2. Použite konvolúciu na odfiltrovanie predchádzajúcich šumov.
3. Použite konvolúciu na detekciu hrán a uvažujte nad hľadaním telies na obraze.

13.5 Cvičenie 5. Segmentácia a hľadanie hraníc

Toto cvičenie budeme venovať segmentácii obrazu. Najjednoduchšia metóda segmentácie je prahovanie, ako oddelenie popredia od pozadia. Používa sa na šedotónové obrazy a vyzerá nasledovne:

```
threshold(obraz, vystup, 200, 255, CV_THRESH_BINARY_INV);
```

kde 200 je vopred určený prah a 255 je hodnota, ktorá sa zapíše do výsledku, ak je spĺňená podmienka určená ďalším parametrom. Pre automatické určenie prahu môže použiť Otsuho metódu, ktorú pridávame v parametri type:

```
CV_THRESH_BINARY_INV | CV_THRESH_OTSU
```

Výsledok prahovania dokážeme získať aj pomocou preťažených operátorov `<`, `>` (funguje aj pre viackanálové obrazy):

```
Mat vystup = obraz < 230;
```

13.5.1 Cannyho hranový detektor

Cannyho hranový detektor používa 2 prahy na detekciu hrany v gradientnom obraze. Pixely gradientného obrazu, ktoré sú menšie ako prvý prah určite hranou nebudú. Tie, ktoré sú medzi nimi sú pravdepodobná hrana a tie, ktoré sú väčšie ako druhý prah, je určitá hrana. To je iba laické vysvetlenie Cannynho detektora, ktorý vieme takto používať:

```
Canny(obraz, vystup, dolnyPrah, hornýPrah);
```

13.5.2 Hľadanie hraníc

V knižnici OpenCV je implementovaný upravený algoritmus prehľadávania do šírky na hľadanie hraníc objektov. Príklad použitia:

```
vector<vector<Point>> contours;
findContours(input.clone(), contours, CV_RETR_LIST,
CV_CHAIN_APPROX_SIMPLE);
```

Toto je najčastejšie použitie tejto metódy. Jej ďalšie variácie a vysvetlenie parametrov nájdete v dokumentácii.

13.5.3 Úlohy

1. Pomocou prahovania oddelte objekty záujmu od pozadia.
2. Použitím Cannyho hranoveho detektora určte hrany na obrázku.
3. S využitím predchádzajúcich metód nájdite hranice objektov obrazu a zvýraznite ich.
4. Využitím approximácie zjednodušte hranice telies.

13.6 Cvičenie 6. Houghové transformácie, Graph cut segmentácia

Na cvičení sa naučíme používať metódy Houghovej transformácie a segmentáciou pomocou rezu v grafe.

13.6.1 Houghová transformácia pre kružnicu

Táto transformácia už bola vysvetlovaná, až na niekoľko implementačných detailov. Príklad použitia:

```
vector<Vec3f> kruznice;
HoughCircles(obraz, kruznice, CV_HOUGH_GRADIENT, 1, 20, 200, 100, 10, 100);
for(unsigned i=0; i<circles.size(); i++) {
    circle(circlesMat, Point(kruznice[i][0], kruznice[i][1]),
           kruznice[i][2], Scalar(0,0,255), 3);
}
```

Táto metóda používa veľa parametrov, ktoré treba vysvetliť:

- 1 = prevrátená hodnota násobku veľkosti akumulátora voči vstupnému obrázku
- 20 = minimálna vzdialenosť medzi dvoma nájdenými kružnicami
- 200 = horná hranica prahu pre Cannyho detektor (dolná je určená polovicou hornej)
- 100 = prah priesečníkov v akumulátore na detekciu kružnice
- 10 = najmenší polomer hľadanej kružnice
- 100 = najväčší polomer hľadanej kružnice

13.6.2 Houghová transformácia pre úsečky

Výklad bol podaný skôr, čiže tu je príklad použitia:

```
vector<Vec4i> ciary;
HoughLinesP(obrazHran, ciary, 100, CV_PI/360, 10, 4, 5);
for(unsigned i=0; i<lines.size(); i++) {
    line(linesMat, Point(ciary[i][0], ciary[i][1]),
          Point(ciary[i][2], ciary[i][3]), Scalar(0,0,255), 3);
}
```

Táto metóda používa tiež viac parametrov, ktoré treba vysvetliť:

- 100 = rozlíšenie akumulátora pre parameter dĺžky
- CV_PI/360 rozlíšenie akumulátora pre parameter uhla

- 10 = prah priesečníkov v akumulátore na detekciu úsečiek
- 4 = minimálna dĺžka úsečky
- 5 = maximálna možná medzera medzi bodmi, aby tvorili jednu úsečku

13.6.3 Grab cut

Metóda grabCut delí obraz na popredie a pozadie. Jej detailne vysvetlenie parametrov nájdete v dokumentácii a v tutoriáli zahrnutého v dokumentácii. Prvé použitie metódy je inicializáciou oblasti, kde sa nachádza objekt záujmu. Algoritmus si v pozadí vytvára foreground a background matice, ktoré určujú popredie a pozadie a ich úpravou možeme ovplyvniť výsledok ďalšej iterácie. Pre jednoduchosť použijeme len jednu iteráciu, vytiahneme z nej masku objektu a oddelíme objekt od pozadia:

```
grabCut(obraz, maska, hraniceVyrezu, background, foreground, 1,
GC_INIT_WITH_RECT);
compare(maska, GC_PR_FGD, maskaObjektu, CMP_EQ);
obraz.copyTo(vystup, maskaObjektu);
```

13.6.4 Myšacie udalosti

V knižnici OpenCV je možné nastaviť na každé okno zvlášť reakcie na myšacie udalosti. Využívajú sa nato globálne metódy. Príklad použitia:

```
void onMouse(int event, int x, int y, int flag, void* params) {
    if(flag == CV_EVENT_FLAG_LBUTTON) {
        }
}
```

kde priradenie metódy riešiacu myšacie udalosti okna nastavujeme nasledovne:

```
setMouseCallback(nazovOkna, onMouse, vlastnyDoplňujuciParameter);
```

13.6.5 Úlohy

1. Pomocou Houghovej transformácie nájdite kruhové objekty na obraze.
2. Pomocou Houghovej transformácie nájdite priamky na obraze.
3. Pomocou nájdených priamok môžete nájsť bod v nekonečne.
4. Pomocou myšacích udalostí označte na obraze objekt záujmu a pomocou metódy rezov v grafe oddelte objekt záujmu od pozadia.

13.7 Cvičenie 7. Matematická morfológia

Cvičenie ukazuje implementáciu základných morfologických operácií a ukazuje ich rôzne kombinácie. Ich základom týchto binárnych operácií je štruktúrny element, ktorý vieme získať jednoducho:

```
Mat element = getStructuringElement(MORPH_CROSS, size(5,5));
```

Namiesto parametra MORPH_CROSS vieme použiť aj MORPH_RECT alebo MORPH_ELLIPSE. Základné operácie sú implementované takto (morphologické operácie sa dajú robiť „in place“):

```
erode(obraz, obraz, element);
dilate(obraz, obraz, element);
```

Ich kombinácie sú implementované všeobecnejšou metódou morphologyEx:

```
morphologyEx(obraz, obraz, MORPH_OPEN, element);
```

kde namiesto MORPH_OPEN môžeme použiť MORPH_CLOSE, MORPH_GRADIENT, MORPH_TOPHAT, MORPH_BLACKHAT alebo MORPH_HITMISS.

13.7.1 Úlohy

1. Použite dilatáciu a eróziu na čiernobielé obrazy a pozorujte, čo sa deje s obrazom.
2. Použitím dilatácie a erózie implementujte operácie otvorenia, uzavrenia, detekcie vonkajšieho/vnútorného okraja.
3. Pomocou otvorenia a uzavretia filtrovajte šum peper & salt.
4. Pomocou Lantuéjoul-vej formuly implementujte morfologickú kostru.

13.8 Cvičenie 8. Template matching

Na cvičení sa oboznámime ako hľadať vzor na obrazoch, kde vzor môže mať zmenenú škálu, može byť pootáčaný a aj mierne zdeformovaný.

13.8.1 Lokalizácia charakteristických bodov

Ako najčastejšie charakteristické body sú sice rohy, ale v niektorých prípadoch sa nám zíjde poznať také charakteristické body, ktoré sú invariantné voči zmene škály, rotácie a podobne. V OpenCV sú implementované ako Feature Detectors. Využitím polymorfizmu sa ľahko nahradza typ detektora charakteristických bodov. Príklad použitia BRISK detektora:

```
Ptr<BRISK> detector = BRISK::create();
vector<KeyPoint> keyPoints;
detector->detect(obraz, keyPoints);
drawKeypoints(obraz, keyPoints, vystup); //vizualizácia
```

Každý charakteristický bod (vieme ich vizualizovať pomocou metódy drawKeypoints) má vlastné príznaky, ktoré získame nasledovne:

```
Mat descriptors;
detector->compute(obraz, keyPoints, descriptors);
```

13.8.2 Spárovanie charakteristických bodov

Najjednoduchší prístup(aj keď najzložitejší) je vyskúšať všetky možné kombinácie bodov, či majú podobné deskriptory. V OpenCV sú nato Matcher-í a tento najjednoduchší sa používa nasledovne:

```
BFM matcher;
vector<DMatch> matches;
matcher.match(descriptors1, descriptors2, matches);
Mat vystup; //vizualizácia spárovania
drawMatches(obraz1, keyPoints1, obraz2, keyPoints2, matches, vystup);
```

13.8.3 Homografia a transformácie

Pomocou spárovaných bodov vieme nájsť zobrazenie, ktoré transformuje jeden obraz do druhého. Hľadanie zobrazenia s využitím metódy RANSAC vyzerá nasledovne:

```
Mat maticaZobrazenia = findHomography(zoznamBodov1, zoznamBodov2,
                                         CV_RANSAC);
```

Ak už máme maticu zobrazenia, tak zobrazenie zoznamu bodov vykonáme nasledovne:

```
perspectiveTransform(zoznamBodov, vystup, maticaZobrazenia);
```

Alebo ak chceme zobrazíť celý obraz pomocou transformačnej matice:

```
warpPerspective(obraz, vystup, maticaZobrazenia, velkosťVystupu);
```

13.8.4 Úlohy

1. Nájdite charakteristické body na obrázku pomocou dostupných metód (SURF, SIFT, BRISK, ...).
2. Nájdite zhody charakteristických bodov a ich deskriptorov medzi dvoma obrázkami a nakreslite ich.
3. Nájdite umiestnenie vzorového obrázku na druhom obrázku (kde môže byť vzorový obrázok pod ľubovoľným natočením) pomocou homografie (Template matching).
4. Pomocou predchádzajúcej metódy zlepťe 2 obrázky do panorámy.

13.9 Cvičenie 9. Optický tok

Na cvičení si ukážeme sledovanie charakteristických bodov na obrazu. Charakteristické body, ktoré sa dobre sledujú sú rohy. Z toho vyplýva, že treba nám take nájsť:

```
goodFeaturesToTrack(obraz, zoznamRohov, 500, 0.01, 10);
cornerSubPix(obraz, zoznamRohov, Size(10, 10), Size(-1, -1),
            kriteriumKonvergencie);
```

kde 500 je maximálny počet rohov, 0.01 je minimálna kvalita rohu a 10 je minimálna vzdialenosť medzi dvoma rohami. Metóda cornerSubPix slúži na spresnenie odhadu pozície rohu podľa okolia. Metóda počítania optického toku Lucas-Kanade je implementovaná v OpenCV a používa sa nasledovne:

```
calcOpticalFlowPyrLK(predchadzajucaSnimka, aktualnaSnimka,
                      zoznamRohovPredchadzajucejSnimky, noveUmiestnenieRohov,
                      zoznamStavovRohov, zoznamChybRohov);
```

Interpretácia zoznamov stavov a chyb a ďalšie parameter sú vysvetlené v dokumentácii.

13.9.1 Úlohy

1. Nájdite charakteristické body, ktoré sú vhodné na sledovanie.
2. Pomocou optického toku využitím metódy Lucas-Kanade nájdite charakteristické body na ďalšej snímke videa.

13.10 Cvičenie 10. RANSAC

Na cvičení si ukážeme ako funguje RANSAC(Random Sample Consensus). RANSAC je iteratívna metóda na odhad parametrov matematického modelu. Vyzerá nasledovne:

Vstup: zoznam dát, ϵ (veľkosť okolia), k (počet iterácií)

Algoritmus: opakuj k -krát kroky 1-4:

1. vyberieme minimálnu náhodnu vzorku, z ktorej sa dá vytvoriť model (v prípade úsečky: sú to 2 body)
2. vypočítame parametre modelu (v prípade úsečky: z dvoch bodov vypočítame smerový vektor $u=(\Delta x, \Delta y)$, z neho normálový vektor $v=(-\Delta y, \Delta x)$ a pomocou jedného bodu z nich dopočítame parametre modelu priamky: $ax + by + c = 0$, kde (a, b) sú súradnice normálového vektora)
3. spočítame počet inlinerov tj. koľko dát sa nachadza v ϵ -okolí (v prípade úsečky: inliner je bod, ktorý ma kolmú vzdialenosť od priamky menšiu ako ϵ) vzdialenosť bodu od priamky: $d(ax + by + c = 0, (x_0, y_0)) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$.

4. ak má nový model viac inlinerov ako doteraz najlepší, tak nový model sa stáva doteraz najlepším

13.10.1 Úlohy

1. Pomocou metódy RANSAC nájdite najdlhšiu úsečku na obrázku.
2. Navrhnite metódu, ktorá pomocou RANSAC-u nájde aj ostatné priamky na obrázku.
3. Odvodte kroky RANSAC-u pre detekciu kružníc, rovín...

13.11 Cvičenie 11. Tracking

Na cvičení si vysvetlíme, prečo je tracking dôležitý a ako ho implementovať.

13.11.1 Úlohy

1. Implementujte diferenčnú metódu, ktorá detektuje pohyb na obraze videa.
2. Pomocou diferenčnej metódy nájdite pohybujúce sa objekty na obraze videa.
3. Priradte jednotlivým telesám svoje id-čka, ktoré sa počas svojej púte obrazom nebudú meniť, využite problém spárenia.
4. Pomocou princípu duchov vylepšite pridávanie id-čiek.

14 Cyklus príbehov

Cieľom tejto časti je čo najprístupnejšou formou (aj za cenu značného zjednodušenia) demonštrovať činnosť algoritmov spomínaných v hlavnej časti. Čitateľom hlavnej cieľovej skupiny (študenti prírodovedných a technických oborov VŠ) môže poslužiť ako mnemotechnická pomôcka a relaxačná vsuvka, tým ostatným môže poslužiť ako prvý kontakt s problematikou, pred prípadným hlbším štúdiom. Prostredie vojenského útvaru počas základnej vojenskej služby jedného z autorov má okrem nostalgiej stránky aj stránku racionálnu a tou sú určité analógie medzi prácou algoritmov a armády (vykonávanie presne špecifikovaných príkazov, paralelizmus, princípy hierarchie a pod.).

14.1 Fotokomora (model dierkovej kamery)

„Budíceák. Náastup na rozvíčku“ - rozliehal sa rev „dozorčího“ po opustenej chodbe kasárni tesne pred svitaním. Vojak Zvara si užíval posledné sladké sekundy na hornom poschodí železnej posteľe, ktorá bola uplynulé tri mesiace jeho jediným súkromným útočiskom počas dvojročnej základnej vojenskej služby. Dnes sa to však malo zmeniť a zoznam jeho nemovitostí sa mal zdvojnásobiť. Mazák Fero, pochádzajúci zo susednej dediny, odchádzal do civilu a vybral si ho ako svojho nástupcu vo funkcií správca fotokomory. Dal mu však podmienku - vyhotoviť fotografiu preberaného priestoru, ktorá bude zároveň veliteľovi slúžiť ako dôkaz funkčnosti fotokomory.

Ked' po vyčerpávajúcim celodennom kolotoči zamestnaní vstúpil do fotokomory, bolo mu jasné, že slúžila na všetko možné len nie na vyvolávanie fotografií. Prázdne fľaše od alkoholu, ohorky cigariet a podlažia pokrytá vrstvou predmetov, z ktorých by sa dala vyčítať história vojenského útvaru ako z kroniky. Najhoršie bolo, že starý fotoaparát (ktorý mal pracovník kontrašpionáže z historických dôvodov v evidencii rovnako ako zbrane) mal rozbitý objektív a bol teda nefunkčný. Odmietať prebrať poškodený inventár a ohroziť tak odchod kamaráta do civilu bol v zmysle útvarových tradícií neodpustiteľný hriech a tvrdo sa trestal neformálnymi, ale veľmi účinnými prostriedkami.

Vojak Zvara teda musel nájsť improvizované riešenie, ako vyrobiť fotografiu bez fotoaparátu. Medzi predmetmi na zemi našiel papierovú krabičku, urobil do nej dierku špendlíkom a umiestnil oproti dierke držiak filmového políčka (tak ako znázorňuje Obrázok 2). Po viacerých pokusoch s dĺžkou osvitu sa mu podarilo vyvolať fotografiu ako dôkaz funkčnosti, ktorý garantoval mazákovi Ferovi bezproblémové odovzdanie inventáru a odchod do civilu.

Pointa: Dierkový model kamery je založený na reálnej fyzickej zostave schopnej fotografovať. Aj v dnešnej dobe dostupných digitálnych fotoaparátov existujú nadšenci využívajúci tento princíp.

14.2 Vrtule (vzorkovanie)

Major Zápotocký rád sledoval spravodajstvo z rôznych nepokojných častí sveta, v ktorých obyčajne nejaký urastený a vetrom ošľahaný generál v poľnej uniforme pred televíznymi kamerami zamieril svoj ďalekokohľad smerom k nepriateľskej línií a pritom druhou rukou vo vzduchu naznačoval strategiu, na čo vedľa stojaci dôstojníci uznanivo pokyvovali hlavou. Ked' dostal od veliteľa útvaru oznam, že má sprevádzať televízny štáb relácie „Plná poľná dobrých správ“, od vzrušenia takmer nevedel zaspať. Pretože bol veliteľom letky dopravných turbovrtuľových lietadiel, hned' ráno zvolal mužstvo s inštrukciami čo robiť aby reportáž pred lietadlom s točiacimi sa vrtuľami bola čo najpresvedčivejšia.

V nedeľu sedela celá rozvetvená rodina pred televízormi a čakala na avizovanú reportáž. Na majorove zdesenie ale jeho dlho trénovaná fráza „ako vidíte, vrtule sa už točia, sme pripravení vyraziť pred“

nasledovaná gestom ukazujúcim na vrtuľu, ktorá sa ale na obrazovke trochy mykala ale netočila. Vinníka našiel rýchlo - bol to vojak Zvara, ktorý vtedy asistoval pri predletovej príprave. „Ja Vás roztrhnem - jasne som povedal že vrtule treba poriadne roztočiť“. Vojakovi ale prídel adrenalínu náhle osvetil mysel' a spomenul si na vzorkovací teorém, ktorý preberal v rámci jedného z dvoch semestrov neukončenej vysokej školy. „Práve preto že sa vrtule točili rovnakou frekvenciou ako bola snímacia frekvencia kamery sa zdalo že sa netočia“.

Pointa: Frekvencia vzorkovania musí byť aspoň 2x väčšia ako je maximálna frekvencia, ktorú chceme zachytiť, inak dochádza ku stroboskopickému efektu.

14.3 Košeľa. (Aliasing)

Veliteľ letky vydával dcéru. Mala si brať miestneho funkcionára s povesťou človeka, ktorý vie vybaviť takmer všetko. Keď sa objavili prvé digitálne fotoaparáty, podarilo sa mu získať dva – jeden lacný model do útvarovej fotokomory ako náhradu rozbitého Flexaretu a druhý (drahší model) pre seba. Pri organizácii svadby zabudli dohodnúť profesionálneho fotografa, a tak sa stalo, že sa tam objavili dvaja náhradníci - vojak Zvara s lacným modelom a ženichov známy - zbohatlík s digitálnym fotoaparátom s vysokým rozlíšením.

„Zabudni na uniformu - vezmeš si oblek a tú jemne prúžkovanú košeľu“ - zavelila veliteľova manželka. Na svadobnej hostine postupne saká ostali prevesené stoličkách a ich majitelia tancovali, pili a fotili sa ostošest. Na druhý deň sa pozerali fotky a všetci ostali prekvapení, že drahý fotoaparát dáva horšie obrázky ako lacný (na košeli s prúžkami boli Moire škvry podobne ako ukazuje Obrázok 13). „Mal si nasadiť dolnopriepustný filter“ - poznamenal mentorsky Zvara. „Môj lacný foťák trochu rozostruje takže filter nepotrebujem.“

Pointa: Keď chceme obmedziť falošné kontúry (aliasing), musíme buď snímať z bližšej vzdialenosťi, alebo použiť hustejšie vzorkovanie alebo jednoducho odfiltrovať vysoké frekvencie pred snímaním. Túto úlohu môže plniť aj menej kvalitný objektív, ktorý rozostruje takže obraz bude sice menej ostrý, ale bez rušivých Moire škvŕn.

14.4 Čudné terče. (Ekvalizácia histogramu)

Na strelnici bolo rušno. Dnešok bol výnimcočný tým, že sa streľby zúčastňovalo aj elitné družstvo zo susedného pluku pripravujúce sa na celoarmádnu súťaž. Keďže si zabudli priniest špeciálne terče s jemným delením, museli použiť tie, ktoré sa používali na bežný výcvik. Rozdiel bol viditeľný ihned, väčšina terčov mala zásahy výhradne v stredovom krahu.

„Ako to vyhodnotím?“ - pýtal sa vedúci streľby a uprel zrak na vojaka Zvaru, ktorý si medzitým získal povesť technického Guru.

„No, nakresli do čierneho stredového krahu ešte jeden menší biely a ten označ číslom 10, pôvodný obrys číslom 9 atď. Keďže si jednu vrstvu v strede pridal, musíš jednu ubrať a to tak, že spojíš dve medzikružia na okraji do jedného.“

Tak sa stalo, že výstrely, ktoré boli predtým v desiatke sa dali jemnejším delením odlišiť na 9 resp. 10 a dali sa zapísať do predpisanej tabuľky a vyhodnotiť úspešnosť.

Pointa: Keď je veľa údajov akumulovaných jednej triede, zjednime jej delenie na úkor tried s malým početným zastúpením.

14.5 Výhľad z basy. (Geometrické transformácie)

Vojakovi Zvarovi sa stalo to, čo by sa mu v civile ťažko mohlo stať - díval sa von cez zamrežované okno útvarovej väznice. Dlho očakávanú vychádzku totiž strávil u piateľky, ktorá bývala tesne za územím posádky. Pri návrate ho chytila hliadka a dostal tri dni ostré za opustenie miesta posádky.

Zrazu jeho pozornosť upútal veľký predmet neurčitého tvaru zabalený v plachte, ktorý neznámi vojaci vyložili z dopravného lietadla, položili vedľa hangáru priamo pred okno väznice a odišli. Zrazu videl súvislosti – plastové nádoby so zvyškami vodky nedávno objavila kontrola v hangári. Pretože nebolo jasné, kto a ako ich tam dopravil, nič netušiacemu kolegovi strážiacemu hangár hrozil vojenský prokurátor.

Teraz bolo jasné, že letecké návštevy družobného letiska za našimi východnými hranicami nie sú iba o výmene skúseností. To by chcelo fotku ako dôkaz, pomyslel si Zvara. S výrobou dierkovej kamery už skúsenosti mal, krabičku od zápaliek aj neexponovaný film mal náhodou vo vrecku. Postavil provizórnu dierkovú kameru tak aby snímala predmet a v popredí mrežu na okne väznice. Stihol to ukončiť tesne pred tým ako sa dvere hangáru otvorili a iná skupina predmet vzala dovnútra.

Po návrate z väznice film vyvolal avšak fotografia bola veľmi skreslená keďže film sa v kamere trochu skrútil. Naštastie mreža v popredí bola viditeľná a poslúžila ako etalon pri geometrickej korekcii obrazu (Obrázok 19). Fotka potom poslúžila ako klúčový dôkazový materiál v procese, ktorý sa ďahal ešte niekoľko rokov, ale vojak strážiaci hangár bol hned' na začiatku zbavený viny.

Pointa: pomocou referenčnej mriežky vieme vyrovnať geometricky deformovaný obraz.

14.6 Konvolučné družstvo (lokálne operátory)

Na konci pristávacej dráhy bolo jazero so stabilnou výškou hladiny a tak niekto na veliteľstve navrhol postaviť na jazere plošinu tak, aby sa náklad dal presunúť z člna priamo do dopravného lietadla pomocou obyčajného vozíka. Plošina stála na deviatich nosníkoch v konfigurácii 3x3 s rozostupom 1 meter. Keďže ju už iniciatívne z veliteľstva dopravili, bolo treba nájsť v jazere miesto blízko brehu, s rovným dnom a hĺbkou presne 1,2 metra.

Touto úlohou bol poverený vojak Zvara za pomoci družstva ženistov, ktorí na letisku zabezpečovali stavebné a terénne práce. Aj napriek dôležitosti ich činnosti boli ženisti na letisku v rebríčku pomyselnej hierarchie niekde na konci a nikto ich nevolal inak ako „Lopaty“. Zvara dostal k dispozícii skupinu deviatich ženistov s rôzne dlhémi záznamami v registri trestov. Každý z nich bol vybavený meracou tyčou, najspoločalivejší z nich dostal aj kalkulačku a papier.

Zoradil ich do trojstupov do tvaru 3x3 tak, aby zodpovedali rozmiestneniu nosníkov, nahnal ich do vody a pokúsil sa o zrozumiteľnú interpretáciu problému:

„Každý z vás predstavuje jeden nosník. Na povel postúpite všetci o meter smerom pozdĺž brehu, zmeriate hĺuku a budete ju postupne hlásiť vojakovi v strede. Ten hodnoty (včítane svojej) spočítava na kalkulačke, vydeli deviatimi a výsledok mi oznámi. Zistí tiež či je dno rovné, teda či rozdiely všetkých deviatich meraní sú malé. To budete opakovať až kým nepoviem dosť.“

Po úspešnom a sofistikovanom prieskume dna už členov družstva nikdy nevolali „Lopaty“, ale „Konvolučné družstvo“.

Pointa: Prepočítaná hodnota bodu v strede konvolučnej masky s jednotkovými prvkami je aritmetickým priemerom susedov. To je v podstate dvojrozmerný ekvivalent kľzáváho aritmetického priemeru. Rozdiel susedov oproti centrálnemu bodu konvolučnej masky predstavuje gradientný obraz, ktorý je v prípade konštantného jasu nulový.

14.7 Vojak Median (Mediánový filter)

„Uff, to bude problém.“ poznamenal neisto veliteľ výcviku brancov dívajúc sa z okna na jednotku nastúpenú podľa veľkosti. V ruke držal obežník zo štábu s príkazom vybrať z každého družstva jedného fyzicky priemerného vojaka (slovo priemerný bolo dva krát podčiarknuté). Vybraní branci sa mali zúčastniť armádneho štatistického prieskumu. Na jednom konci družstva sa týčil dvojmetrový obézny plzeňák Honza, ktorému výstrojný náčelník nevedel nájsť vhodný opasok, takže mu ho poskladal z dvoch. Na druhom konci radu sa snažil udržať na nohách astmatický Ďuri, ktorý považoval vojenskú službu za vec osobnej prestíže a pred odvodmi intenzívne jedol aby dosiahol limit 50 kg potrebných na odvedenie.

„Obaja majú časté zdravotné problémy, keď jeden vypadne na ošetrovňu, úplne nám to zmení priemer.“ - rozmyšľal nahlas.

„A čo tak median?“ spýtal sa slobodník Zvara, ktorý bol privolaný ako poradca?

„A to je ktorý, myslíte že ich všetkých poznám po mene?“

„Ten v strede“

Pointa: Mediánový filter reprezentuje vzorku hodnotou, ktorú má prvok v strede jej triedeného zoznamu. Tým je odolný voči vplyvu extrémnych hodnôt.

14.8 Stratený samopal. (Sledovanie vnútornej hranice)

Čatárovi Fidrmucovi práve skončila vcelku príjemná služba keď dohliadal na skupinu väzňov z útvarovej väznice pracujúcich na úprave sadu za kasárňami. Stmievalo sa a on si v duchu premietal detaile plánu vytúženej cesty do civilu, ktorá sa mala stať skutočnosťou už o necelý mesiac. „Ako bolo?“ pýtal sa službukanajúci kolega na bráne.

„Ušlo to, teraz ešte odovzdať sapík a mám veget...“ Chcel dokončiť vetu ale nemohol, lebo ho obliaľ pot a podlomili sa mu kolená. Uvedomil si že samopal nechal niekde na mieste kde strážil väzňov. Otočil sa na opätku a ozlomky bežal späť ako mu to len podlamujúce sa kolená dovolili. Tradovalo sa, že za stratu samopalu s ostrými nábojmi boli dva roky vo vojenskej väznici v Sabinove automaticky.

Ked' sa dopotácal na miesto činu, bola už tma ako v rohu, naštastie cestou stihol nájsť nejakú palicu, ktorou si oťukával cestu pred sebou ako slepec. Spomenul si, že sedel na jednej z viacerých terénnych vyvýšení pripravených na sadenie kvetov a samopal oprel niekde tam.

„Hlavne nepanikáriť a postupovať systematicky.“ opakoval si v duchu.

Ked' narazil na prvú z vyvýšení, vydal sa po jej okraji tak, že túkol palicou pred seba a potom o 45 stupňov vľavo i vpravo. Tým zistil, ktorým smerom vyvýšenina pokračuje a posunul sa o krok v danom

smere. Aby sa netočil v kruhu, nechal na mieste kde začal svoju čiapku. Keď na ňu narazil zobrajal ju a presunul sa na druhú vyvýšeninu kde celý proces opakoval. Keď potom počul pri ľuknutí kovový zvuk bol to najkrajší zvuk jeho života.

Pointa: Sledovaním miesta najväčšieho gradientu v danom smere nájdeme uzavretú kontúru oblasti. Po návrate do východiskovej pozície pokračujeme ďalšou kontúrou.

14.9 Zdravotná prehliadka (Cannyho detektor)

Piloti boli na letisku elitou a dávali to patrične najavo okatým ignorovaním vojenského vystupovania. Ich sebavedomie bolo prirodzené a keď sa niekto na vychádzke pokúšal oklamať skúsené mestne krásavice a predstierať že je pilot, nikdy s klamstvom neuspel. Existovala však jedna autorita, pred ktorou stáli piloti v servilnom predklone a to bol vojenský lekár na pravidelných zdravotných prehliadkach. Od jeho rozhodnutia záviselo, či bude dotyčný ďalej lietať alebo sa stane „personálom“. Výsledky všetkých vyšetrení sa bodovali a tí, ktorí nedosiahli limit skončili.

Doktor Kany presadil novú metódu bodovania. Najprv vybral skupinu pilotov s najlepšími výsledkami z jednotlivých útvarom, ktorým automaticky predĺžil licenciu. U všetkých ostatných stanobil spodný prah, pod ktorým už dotyčný nemal šancu a horný prah, ktorý automaticky znamenal že môže lietať ďalej. Tí ktorí sa ocitli medzi oboma prahmi mali ešte jednu šancu a síce že sa za nich zaručil pilot, ktorý už prešiel testom a pritom bol z jeho lokálneho okolia (útvaru), a teda ho poznal.

Pointa: Keď je priemerne spoľahlivý prvok v tesnej väzbe s prvkom vysoko spoľahlivým, zvyšuje to jeho šance na zaradenie do triedy spoľahlivých .

14.10 Demokratizácia armády (Pyramid linking)

Prišla doba zmien, slovo demokratizácia sa skloňovalo vo všetkých súvislostiach a neobišli ani armádu. Generálny štáb chcel ukázať, že mu tieto princípy nie sú cudzie a tak vydal brožúrku s nasledovnými inštrukciami.

- 1) Každý vojak si môže vybrať, pod ktorého veliteľa družstva v rámci jeho čaty chce patriť. Ten sa potom stretne so svojimi priaznivcami z rôznych družstiev a upraví plán na základe ich požiadaviek.
- 2) Každý vojak zváží či mu po úpravách veliteľov plán ešte stále vyhovuje alebo radšej prestúpi do družstva iného veliteľa.
- 3) Body 1) a 2) sa opakujú dovtedy, kým nie sú všetci vojaci spokojní so svojimi veliteľmi družstiev.
- 4) Taký istý proces prebehne aj na vyšších úrovniach tak, že velitelia družstiev si vyberajú veliteľov čiat, tí potom veliteľov roty atď. až po najvyššie velenie armády.

Akcia sa stretla s mimoriadnym ohlasom, už v prvej fáze sa väčšina vojakov priklonila k veliteľom družstiev, ktorí deklarovali potrebu zrušenia základnej vojenskej služby. Generálny štáb akciu vyhodnotil ako úspešný experiment a urýchlene ju ukončil.

Pointa: Opakovaný výber najpodobnejšieho nadradeného prvku a následná aktualizácia jeho hodnoty konverguje k vytvoreniu homogénnych zhlukov reprezentovaných vrcholom subpyramídy.

14.11 Akcia na záchrana stromčekov (Mean shift)

Veliteľ útvaru pochádzal zo starej dôstojníckej rodiny a to znamenalo že mali k dispozícii vždy dostatok lacnej pracovnej sily. Kým starý otec mal ešte oficiálne prideleného „pucfleka“, jeho potomkovia si to už museli zariadiť inak. Sadenie okrasných drevín v záhrade domu veliteľovej dcéry bola oficiálne „brigáda na skrášlovanie prostredia v okolí vojenského útvaru“. S blížiacimi sa mrazmi vyvstala potreba ochrany zasadencích drevín. Tak sa stalo, že jedného rána stála pred domom veliteľovej dcéry skupina vojakov a niesli veľký kruhový plastový bazén, ktorý na letisku mal slúžiť na vypúšťanie ropných produktov. Jeho úloha bola prikryť čo najviac čo najvyšších drevín.

Slobodník Zvara, ktorý skupinu viedol na to išiel intuitívne. Prikázal položiť bazén na zem spočítal prikryté rastliny pričom tie vysoké rátal viackrát (podľa výšky). V duchu si predstavil, kde by asi bolo ťažisko na kruhovej doske rovnakých rozmerov ako bazén keby rastliny neboli v zemi ale položené na doske. Potom dal presunúť bazén do ťažiska a postup opakoval dovtedy kým sa stará a nová hodnota ťažiska líšili.

Pointa: Opakovane spočítavame ťažisko bodov vnútri daného kruhu a presunieme stred kruhu do ťažiska. Tento algoritmus konverguje k lokálne optimálnej polohe. Tento princíp je možné využiť aj na optimalizáciu v n-rozmersnom príznakovom priestore.

14.12 Hadica v zákope (Modely aktívnych kontúr)

Aj keď u letectva boli cvičenia v teréne zriedkavé, občas si nejaký veliteľ spomenul, že „aj oni sú vojaci“. Tentoraz bolo cieľom vybudovať poľné veliteľské stanovisko s prívodom vody na chladenie agregátov. Voda sa mala priviesť gumenou hadicou zakopanou v zemi. Čata leteckých mechanikov, elektrikárov a radistov s tým nemala skúsenosti a podľa toho ten kanál aj vyzeral - bol krivoňáký a smerom do hĺbky sa zužoval. Na dno krivolakého kanálu uložili gumovú hadicu na prívod vody vcelku bez problémov. Tie nastali keď začali hadicu tlakovať. Keď zvýšili tlak, hadica sa stala menej pružnou a tlačila sa z kanála smerom nahor. Rozkaz znel jasne - zakopať čo najhlbšie ale tak, aby v hadici mohol byť čo najvyšší tlak. Po viacerých experimentoch nakoniec našli optimálny tlak, pri ktorom ešte hadica kopírovala dno zákopu s akceptovateľnou presnosťou.

Pointa: Vnútorná energia hadice je daná tlakom - čím je vyšší, tým je menej ohybná. Vonkajšia energie je daná prostredím - hĺbka zákopu(derivácia gradientu). Poloha hadice v zákope je kompromisom medzi našou snahou vtlačiť ju čo najhlbšie ale pritom rešpektovať je pružnosť.

14.13 Basketbalista alebo žokej (rozpoznávanie).

Vojenský útvar bol tento rok organizátorom pravidelnej celoarmádnej súťaže vo vybraných športoch. Aby sa predišlo dezorganizácii, bola registrácia a ubytovanie športovcov z jednotlivých vojenských útvarov organizovaná podľa harmonogramu. Službokonajúci dozor na bráne dostal inštrukcie, že počas ich služby budú prichádzať basketbalisti a účastníci jazdeckých pretekov – žokeji a podľa toho ich treba poslať na miesto podrobnej registrácie.

Poddôstojník slúžiaci na bráne najprv podrobne kontroloval doklady a posielal prichádzajúcich na správne miesto. Po krátkom čase ale zistil že je to zbytočná námaha keďže bolo na prvý pohľad jasné, že vysoký účastník je basketbalista a nízky a ľahký bude asi žokej. Omyl bol menej pravdepodobný ako pri kontrole množstva dokumentov v časovej tiesni.

Pointa: Pri klasifikácii delíme objekty do tried podľa vektora príznakov. Pri najjednoduchšej klasifikácii do dvoch tried na základe jediného príznaku (výška) bude plniť funkciu diskriminačnej funkcie jediná konštantná hodnota.

14.14 Vzácna choroba (Bayesovský klasifikátor)

Poručík Bajus sa vrátil z dvojročného pobytu v spriatelej africkej krajine kde na letisku zaškoľoval miestnych vojenských technikov. Prvé kroky každého navrátilca viedli poviňne do vojenskej nemocnice kde lekári zistovali, či si okrem spomienok a slušného dolárového konta nepriniesol aj nejakú tropickú chorobu. Pretože takéto pobity boli vysoko lukratívne aj pre štát, ten nešetril na vybavení a poskytol vojenskej nemocnici diagnostické zariadenia s vysokou spoľahlivosťou 99%. Výsledok vyšetrenia bol pre poručíka Bajusa šokom. Týmto spoľahlivým zariadením mu diagnostikovali veľmi vzácnu tropickú chorobu (1 prípad na 1 milión obyvateľov), ktorá sa zatiaľ vždy skončila smrťou a nejestvoval na ňu liek. Výsledky opakovaných testov mali byť až o týždeň.

Vojenský psychológ, ktorý mal za úlohu dostať poručíka z ľažkej depresie mal však naštastie aj znalosti z oblasti štatistiky.

„Podľa Bayesovského pravidla apriórna pravdepodobnosť jedna ku milión aj pri 99% spoľahlivosti testov znamená riziko iba jedna ku desaťtisíc že dotyčný tú chorobu má. Je na to vzorec! (79)“. Táto správa okamžite prebrala pacienta z depresie. Opakované výsledky testov potvrdili že je v poriadku.

Pointa: V prípade podmienenej pravdepodobnosti je veľmi malá apriórna pravdepodobnosť javu príčinou nízkej celkovej pravdepodobnosti.

14.15 Ako ďaleko je ciel? (triangulácia a stereo videnie)

Vojaci sa presúvali na štábne cvičenie špeciálnym vlakom, ktorý zastal na slepej koľaji v poli mimo stanice. Prepravovaní vojaci si všimli v diaľke stojacu výletnú reštauráciu, ku ktorej viedla poľná cesta.

„Budeme tu stať presne hodinu, bodlo by pivo. Keď mi presne povieš ako je to ďaleko aby som sa stihol vrátiť, tak tam skočím a donesem aj tebe“.

„Viem to spočítať!“ - odpovedal Zvara, ktorý si zo školy pamätał metódu triangulácie. Vošiel do prvého vagóna, postavil sa meter od okna a označil fixkou na sklo bod kolmý na smer pohľadu a tiež bod kde sa mu premietala budova reštaurácie. Potom pravítkom zmeral vzdialenosť týchto bodov. To isté urobil vo vagóne na konci vlaku, ktorého dĺžku poznal. Spočítať vzdialenosť od objektu podľa Obrázok 47 a príslušného vzorca už bolo jednoduché. Pivo získaný týmto sofistikovaným spôsobom potom chutilo dvojnásobne.

Pointa: Keď poznáme posunutie (disparitu) korešpondujúcich bodov na stereo obrazoch, vieme určiť vzdialenosť objektu od kamery.

14.16 Vzdušný ciel (Epipolárna geometria)

Súčasťou cvičenia mal byť aj nácvik zostreľovania vzdušného výsadku kde bolo treba z guľometu zasiahnuť gumovými projektílmi cvičný terč vypustený na padáku z lietadla. Z technických dôvodov sa ale cvičenie oneskorilo a začalo sa stmievat. Na nočnú streľbu chýbalo vybavenie (svetlomet), jediným zdrojom slabého svetla boli vojenské baterky. Slobodník Zvara neostal nič dlžný svojej povesti a bleskovo vymyslel plán.

„Musíš ísť bližšie k miestu predpokladaného dopadu a keď uvidíš ciel, stále naňho sviet baterkou. Ja sice ciel nebudem vidieť, ale budem z boku vidieť lúč baterky. Keď pokropím dráhu lúča dávkou z guľometu, nemôžem ho minúť.“ (Obrázok 49)

Pointa: Objekt, ktorý sa javí jednému pozorovateľovi ako bod, leží z bočného pohľadu druhého pozorovateľa na epipolárnej priamke. To znamená, že keď bočný pozorovateľ hľadá polohu bodu, nemusí prehľadávať celý 2D priestor ale stačí keď bude sledovať body na epipolárnej priamke.

14.17 Odhad pozície (Kalmanov filter)

Jednou z nacvičovaných letových situácií bolo aj lietanie bez vizuálneho kontaktu keď sa pilotovi zatiahla na prednom okne roleta tak že bol nútený sa plne spoliehať iba na prístroje. Starý skúsený pilot nalietal na dopravnom lietadle mnoho hodín a spoliehal sa na svoju intuíciu. Keď k nemu do dopravného lietadla nastúpili na svoj prvý let mladí študenti leteckej akadémie, neváhal svoje skúsenosti náležite dať najavo:

„Poletíme rýchlosťou 640km/hod takže s ohľadom na protivietor budeme za hodinu nad Brnom“. Po hodine skontrolovali polohu podľa prístrojov (avšak pomerne nespoľahlivých, keďže GPS vtedy nebolo) a prístroj ukazoval že nad Brnom budú až za 10 minút. Starý pilot sa spýtal:

„Tak komu veríte – intuícii skúseného pilota alebo prístroju s obmedzenou presnosťou“

Jeden zo skúšaných mladých pilotov aplikoval znalosti z prednášky o Kalmanovom filtro a odpovedal:

„Obom – spoľahlivosť odhadu s rastúcou preletenou vzdialenosťou klesá, spoľahlivosť prístroja je malá ale konštantná. Pravda bude niekde medzi“

Pointa: Pri Kalmanovom filtro sa opakovane kombinuje predikcia nasledovaná korekciou odhadu. Spoľahlivosť odhadu i merania je daná Gaussovým rozdelením. Výsledný odhad je ich kombináciou podľa vzorcov (130) a (131) takže keď je napr. vysoká presnosť merania (úzka Gaussova krivka), bude výsledný odhad bližšie k nameraným hodnotám.

Otzky:

- *Typy vzdialenosťí*
- *Cesta, oblasť, objekty, pozadie*
- *Typy šumov a vzorce*
- *Konvolúcia vzorec*
- *Vlastnosti Fourierovej trasformácie, konvolučný teorém*
- *Jasová korekcia a vzorec*
- *Ekvalizácia histogramu*
- *Transformácia suradníc, obecná, bilineárna, affína, základné typy*
- *Zniženie šumu spriemernením*
- *Detekcia hrán*
- *Laplacov operátor*
- *Definícia segmentácie*
- *Prahovanie, poloprahovanie, určenie prahu*
- *Optimálne prahovanie*
- *Houghová transformácia pre kružnice a priamky vzorce*
- *Template matching, vzorce na testovanie súhlasu*
- *Pyramid linking a mean shift*
- *Snake energia, z čoho pozostáva*
- *Algoritmus značkovania*
- *Tvarové invarianty*
- *Momenty vzorec*
- *Model dierkovej kamery, vnútorná matica*
- *Kalibrácia kamery s dekompozíciou*
- *Vonkajšia matica vzorce netreba*
- *Stereo videnie, triangulacia*
- *Epipolárna geometria a fundamentálna matica*
- *Matematická morfológia dilatácia, erozia, oprening, closing*
- *Distance transform*
- *Autokorenačná funkcia textury*
- *Kookurenčná matica príklady, kritéria odvodené z nej - vzorce, dĺžku primitív netreba*
- *Harrisov detektor*
- *Optický tok vzorce a jeho použitie*
- *Kalmanov filter - bez vzorcov*