VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

POSOUZENÍ KORESPONDENCE ZÁJMOVÝCH BODŮ V OBRAZE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Jan Křehlík
AUTHOR

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

POSOUZENÍ KORESPONDENCE ZÁJMOVÝCH BODŮ V OBRAZE

SIMILARITY MEASURE OF POINTS OF INTEREST IN IMAGE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Jan Křehlík
AUTHOR

VEDOUCÍ PRÁCE Ing. Adam Herout, Ph.D.
SUPERVISOR

BRNO 2008

Abstrakt

Tento dokument se zabývá experimentálním ověřením možnosti použití trénovacích algoritmů AdaBoost a WaldBoost pro vytvoření klasifikační funkce, která by dokázala ve druhém snímku nalézt bod, který koresponduje s bodem v prvním snímku v sekvenci snímků. Práce také popisuje nalezení význačných bodů v obraze, jejichž detekce patří k prvním z kroků hledání korespondence. Dále je popsáno vytvoření deskriptorů nalezených bodů zájmu. Takovéto nalezené korespondující body ve dvojici snímků mohou například sloužit jako předstupeň pro vytvoření 3D modelu nasnímané scény.

Klíčová slova

význačné body, počítačové vidění, předzpracování obrazu, Harrisův detektor, hranový detektor, Sobelův operátor, Laplaceův operátor, korespondence snímků, deskriptory, AdaBoost, WaldBoost, ROC, klasifikace

Abstract

This document deals with experimental verifying to use machine learning algorithms AdaBoost or WaldBoost to make classifier, that is able to find point in the second picture that matches original point in the first picture.
This work also depicts finding points of interest in image as a first step of finding correspondence. Next there are described some descriptors of points of interest. Corresponding points could be useful for 3D modeling of shooted scene.

Keywords

Points of interest, computer vision, image preprocessing, Harris detector, edge detector, Sobel operator, Laplace operator, image correspondence, descriptors, AdaBoost, WaldBoost, ROC, classification

Citace

Křehlík Jan: Posouzení korespondence zájmových bodů v obraze. Brno, 2008, diplomová práce, FIT VUT v Brně.

Posouzení korespondence zájmových bodů v obraze

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Adama Herouta, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jméno Příjmení
Datum

Poděkování

Rád bych tímto poděkoval panu Ing. Adamu Heroutovi, Ph.D., za poskytnuté rady a odborné vedení celé práce, jakožto Ing. Romanu Juránkovi a Ing. Michalu Hradišovi za poskytnuté konzultace k aplikaci pro strojové učení.

Obsah

1 Úvod

Díky stále rostoucí výkonnosti výpočetní techniky byla řada náročných úkolů převedena z „papíru“ a teoretických úvah do praktického nasazení za pomocí výpočetní techniky. Jednou z oblastí zájmu se stalo také zpracování obrazu a s ním související snaha „naučit počítač rozumět“ obrazovým informacím. Toto odvětví zpracování obrazu se nazývá počítačové vidění. Existuje stále mnoho otevřených úkolů, kde je možné zlepšovat výkonnost a rychlost řešení problémů hledáním nových postupů, jak daný úkol řešit. Touto oblastí je např. zpracování obrazových informací ze sekvencí snímků v reálném čase. V některých úkolech, např. sledování pohybujících se objektů, bývá často důležité nalezení význačných bodů v obraze a následné posouzení korespondence nalezených bodů ve dvojici po sobě následujících snímků.

Tato práce se zabývá jednak popisem existujících metod nalezení význačných bodů a také snahou experimentálně zjistit, zda je možné posoudit korespondenci nalezených bodů pomocí klasifikační funkce získané trénovacím algoritmem AdaBoost, což je stěžejní částí a hlavním cílem této diplomové práce.

Druhá kapitola se zabývá obecně popisem významných znaků v obraze a jejich detekce. Následují kapitoly, které podrobněji rozebírají typy jednotlivých významných znaků a způsoby jejich detekce. Popis je především zaměřen na detekci rohů a hran, které se obvykle používají coby významné znaky v obraze. Třetí kapitola se stručně zabývá deskriptory, které popisují významné znaky v obrazech a ve čtvrté kapitole jsou popsány současné způsoby a postupy hledání korespondencí mezi dvěma snímky.
Pátá kapitola popisuje rozpoznávání vzorů v obraze pomocí klasifikátorů, natrénovaných pomocí algoritmu AdaBoost a WaldBoost, jež jsou také následně popsány. Součástí práce je také knihovna pro generování obrazových dat pro tyto trénovací algoritmy, jejíž vlastnosti a možnosti použití jsou popsány v téže kapitole.
Závěrečná šestá kapitola seznamuje čtenáře s navrhnutými klasifikátory pro řešení problému hledání korespondence bodů ve dvojici snímků. Současně jsou také prezentovány výsledky experimentů provedených při trénování klasifikátorů.

2 Typy významných znaků v obraze a jejich detekce

Detektory významných bodů v obraze hrají důležitou roli při předzpracování obsahu obrázku a vyhledání vhodných struktur pro další zpracování. Důležitou vlastností všech takových detektorů je stabilita nalezených struktur. To znamená, aby byly dobře lokalizovatelné a aby je bylo možno opakovaně nalézt i po působení geometrických a fotometrických změn. Pojem významný bod obecně označuje místo v obraze, které má:

- jasnou a matematicky dobře podloženou definici
- jasně definovanou pozici v obrazovém prostoru
- lokální struktura v obraze kolem významného bodu je bohatá na informace vhodné pro pozdější zpracování vizuálním systémem
- je stabilní z hlediska působení lokálních a globálních deformací v obrazové doméně, tak aby byl bod opět nalezen s vysokým stupněm opakovatelnosti

V běžném prostředí, které nás obklopuje, můžeme každý bod v prostoru ohodnotit jeho skutečnou světelnou intenzitou a barvou. Pro jednoduchost se omezíme na dvourozměrný prostor. Pod pojmem dvourozměrný spojitý obraz si můžeme představit například běžnou fotografii. Takovýto obraz lze matematicky vyjádřit spojitou funkcí dvou proměnných, kde funkční hodnota v každém bodě je rovna např. jeho jasu a barvě či hodnotám RGB složek. Funkční hodnotu obrazové funkce si můžeme ještě zjednodušit, aby obsahovala pouze informaci o jasu. Takto vznikne fotografie v odstínech šedi. Toto zjednodušení je často žádoucí, neboť pro detekci významných bodů většinou postačí jasová informace.

K historicky první zmínce významných bodů v obraze dochází v souvislosti s detektory rohů. Nicméně v praxi většina rohových detektorů nedetekuje přímo rohy, ale spíše jsou citlivé na lokální oblasti v obrázku, které mají vysoký stupeň variability ve všech směrech. Body zájmu v obraze jsou také historicky spjaty s oblastmi zájmu v obraze, které značí výskyt nějakých objektů, často detekovaných pomocí blob - detektorů. Termíny detekce blobů a detekce rohů se často překrývají. Dnešní hlavní aplikací bodů zájmu je signalizovat body / oblasti v obrazové doméně, které by mohli být dobrými kandidáty na rozpoznávání objektů či párování obrázků, rozpoznávání, analýzu textur. Pro tyto účely byla navržena spousta detektorů, které jsou v praxi velmi užitečné. Některé z těchto detektorů budou popsány v následujících kapitolách

2.1 Rohy a jejich detekce

Jedním z prvních významných bodů v obraze je roh. Roh může být definována jako průsečík dvou hran. Stejně tak může být roh definován jako bod, v jehož blízkém okolí jsou dva dominantní a rozdílné směry hran.

Naneštěstí jsou v literatuře pojmy „roh“, „významný bod“ a „znak“ (feature) často zaměňovány, což situaci poněkud zamlžuje. Posuzování kvality rohových detektorů je často založeno na schopnosti detekovat stejné body v obrázcích stejné scény, které jsou podobné, ale ne identické, např. jsou posunuté, rotované nebo jinak transformované.

Nejpoužívanější metodou k detekci rohů je algoritmus vyvinutý Harrisem a Stephensem, který je výsledkem zdokonalení Moravcova detektoru.

2.1.1 Moravcův detektor

Jeho metoda je založená na hledání regionů, které jsou lokálním maximem ve vypočtených směrových změnách intenzity analyzovaného obrázku. Pro jejich určení jsou použita okénka o stejných velikostech, která reprezentují určitou oblast v obrázku. Jedno je vždy umístěno nad právě zkoumaným bodem, označme ho O, a dalších osm posunuto ve směrech $(u, v) \in \{(1,0), (1,1), (0,1), (-1,1), (-1,0), (1,-1), (0,-1), (1,-1)\}$. Mezi O a každým z osmi posunutých okének je spočten součet čtverců rozdílu podle vzorce:

$$E(u,v) = \sum_{(x,y) \in O} \left(I(x,y) - I(x+u, y+v)\right)^2 \tag{1}$$

kde $I(x,y)$ představuje hodnoty pixelů obrázku. Z těchto osmi součtů $E(u,v)$ je pro tvorbu mapy „rohovatosti“ použita pouze minimální hodnota a významné body leží v lokálních maximech s hodnotou větší než je zvolená prahová hodnota.

Hlavní výhodou tohoto detektoru je ovšem pouze ona jednoduchost a tím i výpočetní nenáročnost, jinak trpí řadou neduhů. Velice rád reaguje na hrany a také je náchylný na šum.

2.1.2 Harrisův detektor

Vymysleli ho roku 1988 Chris Harris a Mike Stephens, když se zabývali systémem pro počítačové vidění zpracovávajícím sekvenci snímků z jedné pohybující se kamery. Potřebovali tedy novou dostatečně stabilní metodu, která by z obrázku získala bohatší informace potřebné pro rozpoznávání ploch a objektů. Zaměřili se na sledování jak rohů mezi snímky, tak také hran.

Harrisův operátor vychází z Moravcova operátoru, ale liší se od něj v používání tak zvané lokální autokorelační funkce.

Moravcův operátor používá obdélníkové okénko, což způsobuje nízkou odolnost vůči šumu a závislost na natočení obrázku. To je odstraněno použitím kruhového okénka O s Gaussovým vyvážením vnitřních hodnot. Gaussovo vyvážení způsobí, že hodnotám vzdálenějších bodů bude přikládán menší význam a omezí se tak vliv šumu.

Další negativní vlastností Moravcova operátoru jsou anizotropní odezvy. K odstranění tohoto nedostatku je potřeba funkce, která je schopna měřit změnu intenzity v okolí bodu ve všech směrech. Použije se tato funkce ve zjednodušeném zápisu:

$$V_{u,v}(x,y) = \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \tag{2}$$

kde M je:

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix} \tag{3}$$

Matice M obsahuje všechny diferenční operátory a její vlastní čísla určují hlavní křivost plochy v okolí bodu (x, y). Toho se dá využít při rozhodování, zda daný bod je významný a jestli se jedná o hranu či roh. Vlastní čísla matice M označíme jako λ_1 a λ_2 a mohou nabývat těchto hodnot:

- $\lambda_1, \lambda_2 \approx 0$ – okolí bodu je relativně ploché a nejedná se o významný bod
- $\lambda_1 \gg \lambda_2$ nebo $\lambda_2 \gg \lambda_1$– jedná se o hranu
- λ_1, λ_2 jsou obě velké – na okolí bodu je významná křivost ve více směrech a jedná se tedy o roh nebo izolovaný bod

2.1.3 SUSAN

S novým přístupem k detekci rohů přišli v roce 1997 Smith a Brady. Smallest Univalue Segment Assimilating Nucleus (SUSAN) představoval zcela odlišný přístup k této problematice v porovnání s dosavadními postupy. Tento detektor nepoužívá žádné derivace (nepočítá se ani zakřivení ani se nehledají hrany). Způsob této detekce řadíme do skupiny alternativních detektorů rohů. Tato metoda předpokládá, že uvnitř malé kruhové oblasti budou mít pixely daného objektu téměř stejný jas. Algoritmus zjistí počet pixelů s podobnou hodnotou jasu vzhledem k hodnotě středového pixelu masky (jádro masky). Takové pixely se pak nazývají Univalue Segment Assimilating Nucleus (USAN). Rohové body se detekují tak, že se tato maska aplikuje na každý bod v obraze a poté se hledají lokální minima.

Označme oblast pod kruhovou maskou jako M, potom tuto oblast reprezentují pixely $\vec{m} \in M$. Jádro masky je v $\vec{m_0}$. Každý pixel porovnáme s jádrem požitím srovnávací funkce

$$c(\vec{m}) = e^{\frac{\left(I(\vec{m}) - I(\vec{m}_0)\right)^6}{t}}, \tag{4}$$

kde t vymezuje šířku. Mocnost exponentu byla určena empiricky. Oblast USAN je dána vztahem

$$n(M) = \sum_{\vec{m} \in M} c(\vec{m}). \tag{5}$$

Odezva USAN operátoru je dána vztahem

$$R(M) = \begin{cases} g - n(M) & \text{pokud } n(M) < g \\ 0 & \text{jinak} \end{cases},$$

kde g je geometrický práh. Jinými slovy, USAN operátor má kladné hodnocení pouze, pokud je oblast dostatečně malá. Nejmenší USAN lze nalézt potlačením nemaximálních hodnot a tímto dostáváme celý SUSAN operátor. Hodnota t říká, jak podobné musí být hodnoty bodů vzhledem ke středu masky, aby mohly patřit do segmentu bodů s podobnou intenzitou. Hodnota g určuje minimální velikost segmentu bodů s podobnou intenzitou. Pokud je g veliké, pak se operátor stává detektorem hran. Při detekci rohů se používají další dva kroky. Nejdříve se nalezne těžiště oblasti USAN. U skutečného rohu nalezneme těžiště daleko od středu masky. Další krok spočívá v tom, že se všechny body na spojnici středu masky s těžištěm USAN oblasti musí nalézat uvnitř této USAN oblasti. Detektor umožňuje hledat stejně dobře hrany jako rohy, je poměrně odolný vůči šumu a díky této vlastnosti není potřeba obraz nejprve vyhladit filtrem. Odolnost vůči šumu zvýšíme nastavením větší tolerance hodnoty t. Detektor je rychlý na výpočet, ale jeho hlavní nevýhodou je poměrně malá stabilita.

2.2 Hrany a jejich detekce

Hrana je místo v obraze, kde dochází k ostrým změnám hodnot jasu. Ostré změny jasu nám dávají dobrou informaci o vlastnostech a probíhajících událostech pozorovaného světa. Tyto změny nám indikují nesouvislosti v hloubce obrazu, změny v orientaci povrchu a vlastnostech materiálu a v neposlední řadě také informují o různorodosti osvětlení scény.

Hrany můžeme rozdělit podle směru pohledu na:

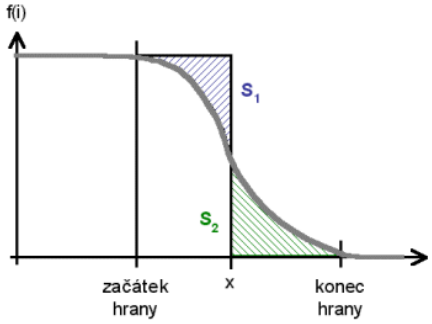
- **směrově závislé** – se směrem pohledu se mění a typicky odráží geometrické vlastnosti scény, překrývající se objekty, atd.
- **směrově nezávislé** – tyto hrany typicky vyjadřují vlastnosti pozorovaných objektů jako je tvar povrchu objektů

V přírodě se obvykle nevyskytují hrany skokové, které by byly svou charakteristikou ideální pro detekci, ale často jsou hrany ovlivněny několika jevy:

- ohniskové rozmazání způsobené konečnou hloubkou ostrosti
- polostínem - oblast, ze které je zdroj světla částečně vidět a částečně jej překrývá předmět, jež stín vrhá
- lokálními odlesky a odrazy světla od nerovných povrchů

Pro hledání hran v obraze se obvykle používá některá z následujících metod:

- **Komparace** – signál je komparován s komparační úrovní (pevnou nebo adaptivně nastavovanou). Místa hran se stanoví jako místa průchodu signálu touto komparační úrovní. Pro zvýšení rozlišení tohoto algoritmu se používá interpolace mezi vzorky signálu.
- **Metody využívající vlastnosti 1. a 2. derivace** – tyto metody určují polohu hrany jako polohu inflexního bodu v grafu jasového profilu. Metody využívají toho, že 1. derivace (resp. její diskrétní obdoba: diference) nabývá v inflexním bodě maxima, resp. že graf 2. derivace (diference) signálu prochází v místě inflexního bodu nulou.
- **Fotometrická interpolace** – poloha hrany se určí jako poloha *x* pro kterou platí, že plochy S_1 a S_2 jsou shodně velké.



Obrázek 2.1 - Princip fotometrické interpolace [14]

Pokud chceme hranu v diskrétním obraze detekovat a zobrazit, nemůže být nekonečně úzká a musí jí odpovídat určité pixely v obraze. Přitom předpokládáme, že tyto pixely by měly být právě v takových místech, kde je lidské vnímání obrazu předpokládá.

Princip hledání hran je většinou odvozen od principu hledání hran ve spojitém obraze. Pixely s velkou hodnotou gradientu vnímáme jako hranu. Většina metod je založena na aproximaci 1. nebo 2. derivace obrazové funkce a v diskrétním obraze tak gradient pouze odhadujeme.

V diskrétním prostoru aproximujeme parciální derivaci *diferencí* a 1. diference může vypadat např. takto:

$$\Delta_x g(x,y) = g(x,y) - g(x-n,y), \tag{6}$$

$$\Delta_y g(x,y) = g(x,y) - g(x,y-n), \tag{7}$$

kde n je malé celé číslo, obvykle rovno jedné. Diferenci lze vyjádřit i symetricky

$$\Delta_x g(x,y) = g(x+n,y) - g(x-n,y), \tag{8}$$

$$\Delta_y g(x,y) = g(x,y+n) - g(x,y-n), \tag{9}$$

ale většinou se nepoužívá, neboť zanedbává vliv samotného pixelu (x,y).

Gradientní operátory udávající strmost funkce můžeme rozdělit do tří kategorií:

1. Operátory aproximující derivace pomocí diferencí. Některé operátory aproximují první derivaci a využívají několik masek, které odpovídají příslušné orientaci. Z nich se vybere ta, která nejlépe lokálně aproximuje obrazovou funkci. Tímto výběrem je zároveň určen i směr gradientu (orientace). Jiné, např. Laplacián aproximují druhou derivaci, jsou invariantní vůči rotaci a mohou být počítány konvolucí s jedinou maskou.
2. Operátory založené na hledání v místech, kde druhá derivace obrazové funkce prochází nulou. Příkladem je Marrův-Hildrethové operátor a Cannyho hranový detektor.
3. Operátory snažící se lokálně aproximovat obrazovou funkci poměrně jednoduchým parametrickým modelem, např. polynomem dvou proměnných.

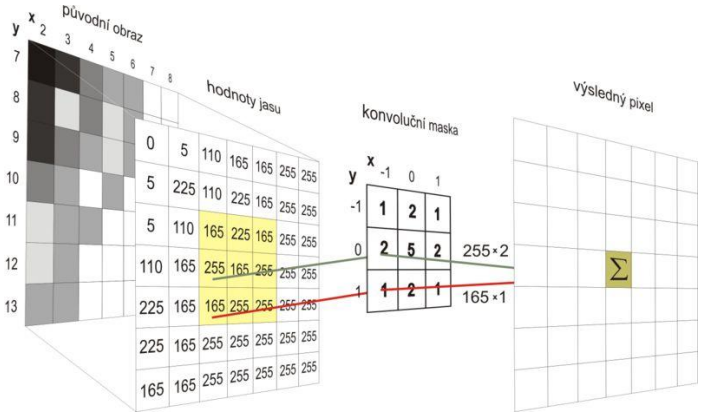
2.2.1 Lokální filtrace

Při zpracování obrazu se často využívá tzv. *lokální filtrace*. Ta spočívá v tom, že k výpočtu nové hodnoty pixelu se použije jeho malé okolí. Obraz se systematicky, např. po řádcích prochází. Kolem každého pixelu je zkoumáno jeho malé okolí, např. tvaru čtverce se stranou o lichém počtu pixelů.

Metody lokální filtrace se dělí na *lineární* a *nelineární*. Lineární operace počítají hodnotu ve výstupním obraze jako lineární kombinaci hodnot vstupního obrazu v malém okolí středového pixelu. Pro detekci hran nebo i pro odstraňování šumu se používají tzv. *konvoluční lokální* operace (filtry). V diskrétním prostředí se používá diskrétní konvoluce, pro diskrétní obraz jde o její dvourozměrnou podobu

$$I'_{i,j} = I_{i,j} \star h_{i,j} = \sum_{x=-k}^{k} \sum_{y=-k}^{k} I_{i-x,j-y} h_{x,y}\,, \tag{10}$$

kde $I'_{i,j}$ je nová hodnota pixelu na pozici (x,y), $I_{i,j}$ je původní hodnota pixelu a $h_{x,y}$ je konvoluční jádro (maska). Hodnoty konvolučního jádra určují způsob výpočtu nového pixelu v obraze. Konvoluční jádro můžeme popsat jako tabulku o rozměrech $\langle -k,k\rangle \times \langle -k,k\rangle$. Princip disktrétní konvoluce popisuje obrázek 2.2.



Obrázek 2.2 - Princip diskrétní konvoluce [15]

2.2.2 Odhad první derivace

Operátory této kategorie aproximují první derivaci. Gradient je většinou odhadován v okolí 3×3 pro osm směrů. Vybrána právě jedna maska, které odpovídá největší modul gradientu. Je přirozeně možné vytvářet větší masky s podrobnějším směrovým rozlišením, které jsou méně citlivé na šum, ovšem takovýto „komfort“ draze zaplatíme kvadraticky narůstajícím počtem kroků, které jsou potřeba k výpočtu konvoluce.

2.2.2.1 Sobelův operátor

Sobelův operátor je velmi podobný asi nejjednodušímu Robertsovu operárotu. Na rozdíl od něj je založen na konvoluci. Sobelův operátor je vždy složen z dvojice komplementárních konvolučních masek označených jako h a $\bar{h}$. Komplementární maska se získá z původní masky rotací o 90° kolem středu. Protože se v druhém kroku vypočítává druhá mocnina, je nepodstatné, zda otáčíme vlevo či vpravo. Absolutní velikost gradientu $|G|$ je pak získán aplikací obou komplementárních jader následujícím vztahem:

$$|G| = \sqrt{{G_x}^2 + {G_y}^2}. \tag{11}$$

Možné konvoluční masky mají tvar:

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{12}$$

Sobelův operátor je směrově orientovaný. Pro detekci vodorovných a svislých hran je vhodné použít uvedené dvě masky. Existují i další masky vhodné např. pro detekci šikmých hran. Důležité je získání

směru gradientu. Protože se Sobleův operátor skládá ze dvou masek, z nichž každá je citlivá na jiný směr hran, je získání směru gradientu triviální:

$$\theta = \tan^{-1}\left(\frac{|G_y|}{|G_x|}\right), \tag{13}$$

z vlastností funkce arctan, neboli $\tan^{-1}$ je zřejmé, že musíme ošetřit stav, kdy $|G_x| = 0$, pak $\theta = \frac{\pi}{2}$, pokud je $|G_x| = |G_y| = 0$, potom je $\theta = 0$.

2.2.3 Odhad druhé derivace

2.2.3.1 Laplaceův operátor

K detekci hrany můžeme využít maxima první derivace obrazové funkce nebo nulové hodnoty druhé derivace. Detekce hrany ve strmějším průchodu druhé derivace nulou je obvykle spolehlivější. Při lokálním vyšetřování změn jasu (bez potřeby detekce směru) je možné s výhodou užít všesměrový lineární Laplaceův operátor neboli Laplacián, který vychází z druhých parciálních derivací a je definován vztahem:

$$\nabla^2 g(x,y) = \frac{\partial^2 g(x,y)}{\partial x^2} + \frac{\partial^2 g(x,y)}{\partial y^2} \tag{14}$$

Operátor je invariatní vzhledem k otáčení a udává pouze velikost gradientu hrany, přicházíme tedy o směr hrany. V digitálním obraze je Laplacián aproximován pomocí disktrétní konvoluce, ta mívá většinou jednu ze dvou následujících často používaných masek:

$$h_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, h_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -8 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{15}$$

První z masek se používá k aproximaci 2. derivace pro čtyřokolí, druhá pro osmiokolí. Existují ještě další varianty masek, používané pro zvýraznění středu:

$$h_3 = \begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix}, \; h_4 = \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix} \tag{16}$$

které dávají větší váhu středovému pixelu. V tomto případě již ovšem operátor ztrácí invariantnost vůči otáčení. Hlavní nevýhodou Laplaciánu je velká citlivost na šum. Další nevýhodou jsou dvojité odezvy na hrany odpovídající tenkým liniím v obraze.

2.2.3.2 LoG (Laplacián Gaussiánu)

Operátory aproximující derivaci diferencemi, jsou lokálně závislé s velkou mírou citlivosti na šum, protože musí mít dostatečně malou masku, aby odpovídaly analyzovaným detailům.
Dobrým kompromisním řešením je konvoluce obrazové funkce s vyhlazujícím filtrem, jehož koeficienty v konvoluční masce odpovídají 2D gaussovskému rozdělení podle vztahu

$$G(x,y) = e^{\frac{x^2+y^2}{2\sigma^2}} \tag{17}$$

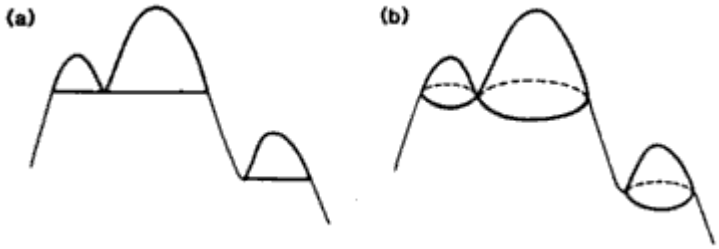
kde x, y jsou souřadnice bodu v obraze a σ je střední odchylka definující velikost okolí bodu. Největší váhu v masce mají pixely blíže středu. Pixely vzdálené více než 3σ mají jen zanedbatelný vliv. Obrazová funkce se pak po aplikaci gaussovského filtru rozostří.
Druhou derivaci obrazové funkce je možné odhadnout všesměrovým Laplaciánem. Postup je někdy nazýván jako LoG operátor (Laplacian of Gaussian), který je definován jako

$$LoG\big(f(x,y)\big) = \nabla^2\big(G(x,y) * f(x,y)\big) \tag{18}$$

LoG nedetekuje ostré rohy a má tendenci k vyhlazování ostrých hran a spojování do uzavřených křivek.

2.3 Bloby a scale-space teorie

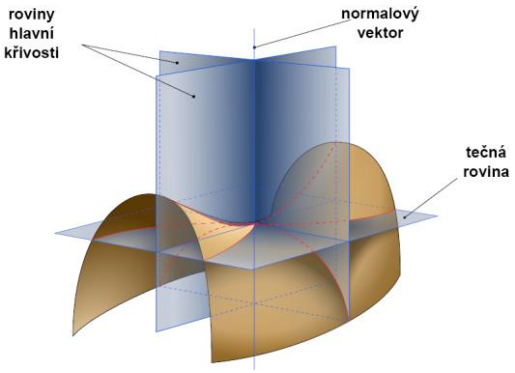
Intuitivně můžeme blob definovat jako spojitou oblast, která je významně světlá či významně tmavá než její nejbližší okolí. Blob je oblast v obraze asociovaná (nejméně) s jedním lokálním extrémem. Definice blobu je jasnější z obrázku 2.3.



Obrázek 2.3 - Ilustrace definice šedotónového blobu (a) 1D signál a (b) 2D signál. V 1D signálu je blob určen párem obsahujícím jedno lokální maximum a jedno lokální minimum. V 2D je to maximum a sedlo. [16]

S bloby souvisí též pojem *hřeben a údolí*. V topografii je hřeben definován jako oddělovač oblastí, v nichž teče voda různými směry. Nicméně přesná matematická formulace této vlastnosti vedla k mnoha nejasnostem.

V počítačovém vidění byl světlý (tmavý) hřeben definován Haralickem (1983) jako body, pro které se předpokládá v rovině hlavní křivosti maximum (minimum) v hlavním směru křivosti a Crowleyem a Parkerem (1984), kteří hledali směrová maxima v pásmových filtrech.



Obrázek 2.4 – roviny křivosti [17]

Byla formulována spousta dalších způsobů popisu hřebenů z jasové složky obrázků. Přirozeným způsobem, jak definovat koncept hřebenů pomocí jasu v obrázku, je definováním světlých (tmavých) hřebenů jako spojené množiny bodů, pro které intenzita předpokládá lokální maximum (minimum) ve směru hlavní křivosti.

Výše uvedené pojmy jsou spjaty s tzv. scale-space teorií, která bude následně popsána.

Scale-space teorie

Scale-space teorie je základ pro prvotní obrazové operace, které byly vyvinuty komunitou pro počítačové vidění, aby se mohlo pracovat s vlastnostmi obrazových dat v prostoru měřítek. Hlavním argumentem tohoto přístupu je to, že pokud neexistuje žádná apriorní informace o tom, jaká jsou vhodná měřítka pro daná vstupní data, pak jedinou racionální cestou pro počítačové vidění je reprezentovat vstupní data v několika měřítkách. To znamená, že původní signál by měl být začleněn do jednoparametrových odvozených signálů, ve kterých jsou struktury v jemném měřítku postupně potlačovány.

Jak by měla být tato myšlenka realizována v praxi? Rozhodujícím požadavkem je, že struktury v hrubých měřítkách by měly představovat zjednodušení korespondujících struktur v jemných měřítkách. Neměly by být náhodnými fenomény, které by byly vytvořeny metodou pro potlačení struktur v jemném měřítku. Tato myšlenka byla formalizována různými autory různými způsoby. Stojí však za povšimnutí, že tito autoři došli k podobným závěrům, i když každý začínal jinak. Koenderink a Lindeberg ukázali, že různé důvodné předpoklady pro to stát se jádrem konvoluce v prostoru měřítek, má pouze Gaussova funkce.

3 Deskriptory

Jakmile již máme významné body nalezeny, potřebujeme lokální fotometrické deskriptory k popisu těchto významných bodů. Deskriptor je struktura popisující vlastnosti bodu, které daný bod charakterizují a umožní jeho opětovné vyhledání. Hrají tedy hlavní roli při porovnávání obrázků. Tyto deskriptory musí být význačné a robustní vůči různým transformacím obrázku. Je tedy důležité, aby bylo okolí bodu popisováno ve formě odolné proti geometrickým a světelným změnám obrazové informace, neboť nese hlavní zodpovědnost za správné nalezení shod mezi obrázky. Dále musíme brát zřetel na to, aby deskriptory nebyly příliš rozměrné. Běžné deskriptory obsahují vektor jasů okolních pixelů, histogram jasů, histogram gradientů nebo jiný invariantní popis. Způsob zakódování ovlivňuje dvě věci: invarianci vůči transformacím a diskriminativitu popisu. Práce Mikolajczyka a Schmida ukázaly, že SIFT deskriptory v mnohém předčí ostatní deskriptory. SIFT a SURF metody využívají deskriptory založené na orientacích gradientů získaných z oblasti obklopující daný bod. SIFT popis je robustně vypočtený histogram orientací v okolí bodu resp. na oblasti. Okolí je rozděleno na 4×4 čtverce a v každém čtverci se orientace rozdělí do osmi přihrádek (*bins*). Výsledkem je 128-dimenzionální vektor pro každou oblast. Prvky vektoru se označují jako biny a deskriptor, který má 128 prvků, se tedy označuje jako 128 binový.

4 Algoritmy hledání korespondencí bodů zájmu

V této části budou uvedeny některé metody hledání korespondencí bodů zájmu, které využívají lokálního chování obrazové funkce. Pod pojmem lokálního chování se myslí vlastnosti obrazové funkce v nějakém okolí bodu zájmu. Naleznou se tedy body zájmu a charakterizují se pomocí jejich okolí. Tyto charakteristiky se potom porovnávají. Přesný postup je následovný:

1. detekce bodů zájmu ve vstupních obrazech
2. popis lokálního chování obrazové funkce v okolí bodů zájmu
3. porovnání popisů a generování dvojic se stupněm příslušnosti (stupeň příslušnosti je míra, do jaké lze očekávat výskyt korespondence mezi dvěma body zájmu).

4.1 Korelace

Jednou z nejběžnějších metod na určení podobnosti dvou okolí je korelační metoda. Dívá se na funkční hodnoty obrazové funkce I v okolí bodu zájmu jako na n výskytů náhodné proměnné. Těchto n výskytů se pro každý bod zájmu uspořádá do vektoru a míra podobnosti mezi dvěma vektory se určí výpočtem korelačního koeficientu neboli kosinem úhlu mezi nimi.
Metoda používá čtvercové okolí bodu zájmu $p_i = (x_0, y_0)$. Vektor v_i popisující bod p_i získáme z funkčních hodnot obrazové funkce I na okolí B_i, tak že uspořádáme hodnoty $I(B_i)$ do vektoru.

Stupeň příslušnosti η dvou bodů p_i a p_j potom získáme výpočtem korelačního koeficientu mezi vektory v_i a v_j. Čím bude korelační koeficient bližší jedné, tím jsou si obě okolí více podobná. Výpočet koeficientu je vyjádřen rovnicí:

$$\eta = C(v_i, v_j) = \frac{(v_i - \bar{v}_i)(v_j - \bar{v}_j)}{\|(v_i - \bar{v}_i)\|\|(v_j - \bar{v}_j)\|}, \tag{19}$$

kde v_i a v_j jsou vektory popisující daná okolí bodů p_i a p_j a $\bar{v}_i$, $\bar{v}_j$ jsou střední hodnoty těchto vektorů.

Tato metoda umožňuje snímání obrazů pouze translací ortografické kamery, protože požaduje pro svoji funkci stejná okolí korespondujících bodů zájmu. Použít lze i perspektivní kameru, ale body scény, se kterými pracujeme, musí všechny být přibližně ve stejné vzdálenosti od kamery, jinak dochází k tomu, že korespondující okolí bodů mají různá měřítka. Dále je nutné, aby detektor bodů zájmu detekoval stále stejné body scény. Na tom spočívá invariance vůči translaci všech metod. Další druhy pohybu například rotace nelze použít, protože okolí korespondujících bodů se při nich mění. Lze je použít pouze v případě, kdy se poloha kamery změní velmi málo tak, že se okolí bodů zájmu změní jen nepatrně.

4.2 Neparametrické invarianty

Neparametrické invarianty používají k popisu okolí bodu zájmu uspořádání funkčních hodnot obrazové funkce v tomto okolí. To je rozdíl od korelační metody, která využívá vlastní funkční hodnoty. Metoda je založena na výpočtu tzv. neparametrických korelací. Aby se dosáhlo rotační invariance, používá se kruhové okolí nebo okolí tvořené mezikružím. Popis okolí se potom získá pomocí uspořádání hodnot jasu na kružnicích kolem bodu zájmu, které patří do zvoleného okolí. Protože metoda používá kruhové okolí, je vhodné pracovat v polárních souřadnicích. Neparametrické invarianty jsou velmi citlivé na šum.

Tyto invarianty se mohou použít jak k translaci, tak i k rotaci kamery. Nelze ovšem použít tuto metodu pro libovolnou změnu polohy kamery, protože používá kruhové okolí v každém obraze, což není invariantní vůči afinní transformaci. Výhodou této metody však je, že je invariantní vůči jakékoliv monotónní jasové transformaci.
Pro praktické použítí těchto invariantů je nutné si uvědomit, že pokud vstupní obrazy obsahují šum, tak musí být typ, rozptyl a střední hodnota šumu stejná ve všech vstupních obrazech, protože jinak se budou střední hodnoty neparametrických invariantů pro korespondující okolí bodů zájmu lišit mezi jednotlivými obrazy.

4.3 Diferenciální invarianty

Jestliže se na obrazovou funkci díváme jako na reálnou spojitou funkci dvou proměnných a pokud je tato funkce totálně diferencovatelná do řádu N, pak lze s pomocí Taylorovy řady funkci aproximovat v jistém okolí bodu $p_i = (x_0, y_0)$ polynomem řádu N. K popisu lokálního chování funkce I v bodě p lze tedy použít derivaci funkce I v bodě p. Tyto derivace samozřejmě nejsou invariantní vůči rotaci, ale rotační invarianty z nich lze sestavit pomocí tensorového počtu, jako polynomy obsahující tyto derivace. Pro představu toho, co vlastně tyto invarianty obsahují, je možno uvést, že zahrnují například velikost gradientu, která je rotačně invariantní.
Podobně jako neparametrické invarianty jsou i diferenciální invarianty použitelné pro translaci a rotaci kamery. Ze stejných důvodů jako předchozí metoda nelze tuto metodu použít pro libovolnou změnu polohy kamery. Její výhodou je ovšem snadné a přirozené rozšíření na práci s obrazy s více měřítky.

Všechny popisované metody kromě korelační předpokládají, že obrazová funkce je spojitá reálná funkce $I : \mathcal{R} \times \mathcal{R} \rightarrow \mathcal{R}$. Tento předpoklad, ale sebou nese jisté problémy. Skutečný obraz vznikne z této funkce I vzorkováním. Proto, čím větší frekvence se ve frekvenčním spektru funkce I vyskytují, tím větší chyby to způsobuje. Typickým příkladem může být ztráta rotační invariance metody. Dalším příkladem jsou chyby, které vznikají díky diskretizaci, jsou náhrady derivací diferencemi a integrálů sumami. Důsledky těchto jevů jsou dobře patrné v experimentech studujících rotační invarianci metod [6].
Dalším problematickým předpokladem, který se používá při hledání korespondencí v obrazech, které jsou nasnímány z různých míst, je předpoklad rovinnosti okolí bodu zájmu. Všechny metody se dívají na okolí bodu zájmu jako rovinu pokrytou texturou. Změny, ke kterým dochází při změně polohy, se snaží obsáhnout pomocí různých lineárních transformací. Předpokládají tedy, že obraz při změně polohy vzniká takto:

$$I'(x,y) = I\big(g(x,y)\big), \tag{20}$$

kde g je lineární zobrazení (transformace) z roviny do roviny.

Tento předpoklad platí ovšem pouze při malé změně polohy kamery, nebo pokud zobrazuje okolí bodu zájmu rovinnou část scény (příkladem může být plakát na stěně). Pokud je tato změna větší, dochází díky trojrozměrnosti scény k zákrytům. To způsobí to, že se v obraze I' objeví informace, která v I obsažena není a naopak.

5 Rozpoznávání vzorů v obraze

Oblast rozpoznávání vzorů má za úkol získat informace ze signálu na základě nějaké znalosti nebo statistických informací. Klasifikované signály bývají obyčejně měření reprezentovaná v multidimensionálním prostoru vektorem příznaků (*feature vector*). Obecný klasifikační systém se skládá ze tří částí – získání dat, extrakce příznaků a klasifikace. Výsledek klasifikace záleží tedy především na kvalitě naměřených dat a příznaků z nich získaných. V případě počítačového vidění naměřená data představuje obraz (např. fotografie, obraz z kamery) a příznaky jsou funkce počítané nad tímto obrazem. Klasifikace je použitelná obecně v prakticky libovolné úloze. Klasifikační metody lze nalézt například v dolování dat nebo zpracování řeči, kde se takto rozpoznávají fonémy a slova nebo se určuje jazyk, kterým je řeč mluvena.

V počítačovém vidění má klasifikace za úkol zařadit obraz do jedné z definovaných tříd a tak identifikovat, zda se v obraze vyskytuje hledaný objekt či nikoliv. V binárních úlohách se tedy obvykle vyskytují dvě třídy – *hledaný objekt* a *pozadí*. Některé úlohy ale vyžadují větší počet klasifikačních tříd (např. OCR – každý znak má svoji třídu). Třída hledaných objektů není nijak omezena a hledat lze téměř cokoliv, např. obličeje, určitý geometrický tvar, poznávací značka auta, znak, atd.
Existuje mnoho metod klasifikace. V praktických aplikacích je při výběru klasifikační metody nutné zohlednit kritéria jako množství dostupných trénovacích dat, dostupné výpočetní prostředky pro trénování a běh aplikace, atd. A podle nich zvolit použitou metodu, protože každá má jiné nároky a mají i různě kvalitní výstup.

5.1 Klasifikátory

Klasifikace jako metoda rozpoznávání vzorů je použitelná v mnoha komerčních i průmyslových aplikacích. V mnoha případech se jedná o systémy, kde je klasifikace pouze jako jedna součást a celek využívá řadu dalších metod. Mnoho webkamer nebo digitálních fotoaparátů má v sobě jednoduchý detektor obličeje, který používají pro automatické zaostřování, nastavení expozice (AF/AE) a odstranění červených očí. Další využití rozpoznávání vzorů lze nalézt v OCR programech, které musí na digitalizovaném dokumentu rozpoznat jednotlivá písmena (a řezy písma). OCR software se dnes běžně dodává k většině scannerů. Z průmyslových aplikací jde například o kontrolu dopravy, kde se klasifikace používá například při detekci a rozpoznávání poznávacích značek aut. Další využití lze nalézt v průmyslové kontrole kvality výrobků.

5.1.1 Trénování klasifikátoru

Obyčejně se trénování klasifikátoru provádí učením s učitelem (*supervised learning*). Jde o metodu zjišťování klasifikační funkce ze známých dat, u kterých víme, do které třídy patří (*ground truth*). Klasifikátor se tak podle dat, která během učení *viděl*, naučí předpovědět třídu klasifikace pro neznámá data – tzv. generalizovat. Je zřejmé, že při učení nelze v datech popsat všechny možné případy vstupu (až na některé velmi speciální případy), které má klasifikátor umět zařadit a výsledek tedy bude mít vždy určitou chybu. Při trénování se s použitím co nejlepších trénovacích dat snažíme, aby tato chyba byla co nejmenší. Platí, že se snižující se chybou roste komplexnost klasifikátoru a tedy i čas potřebný pro jeho trénování a vyhodnocení. Jednodušší klasifikátory mají na druhou stranu chybu vyšší.

Při trénování se používají vzorky dat. Existují dvě sady vzorků – trénovací a testovací. Na trénovací sadě se učí klasifikační funkce a na testovací sadě se ověřuje její chyba. Obecně lze říci, že trénovací data obsahují obrazy objektů a jejich zařazení do třídy. Například v úloze detekce obličejů jsou dvě třídy objektů – *obličej(possitives)* a *pozadí(negatives).* Třída *obličej* obsahuje obrázky různých obličejů za různých podmínek. Ve třídě *pozadí* je vše, co nemá klasifikátor považovat za detekovaný objekt. Čím lepší sadu vzorků reprezentující podmínky při použití klasifikátoru máme k dispozici, tím lepší klasifikátor lze natrénovat. Trénovací sada se někdy rozděluje na dvě části – vlastní trénovací sada a validační sada. Druhá jmenovaná se používá pro detekci přetrénování klasifikátoru.

Před spuštěním trénování jsou ze vstupních dat extrahovány příznaky a vytvořen příznakový vektor – ten se v průběhu nemění. Samotné trénování je iterační proces, který lze popsat následujícím kroky:

1. *Klasifikace vzorků*
   Příznakový vektor se použije ke klasifikaci vzorků aktuální verzí klasifikátoru. Takto se zjistí jeho chyba.
2. *Úprava klasifikátoru podle chyby*
   Trénovací algoritmus se pokusí opravit klasifikační funkci podle chyby klasifikace z předchozího kroku.

Tyto kroky se opakují, dokud není splněno kritérium zastavení. Nejjednodušší kritérium je počet trénovacích kroků. Trénování se zastaví po dosažení stanoveného počtu iterací. Dalším kritériem může být velikost chyby. Trénuje se, dokud je chyba vyšší než stanovená hranice. Jiná možnost je, že pokles chyby mezi po sobě následujícími kroky musí být vyšší než určitá hodnota. Trénuje se tedy do té doby, dokud klesá chyba na celé trénovací sadě. Metoda, která dosahuje nejlepších výsledků, je použití validační sady vzorků. Pokud chyba na trénovací i validační sadě klesá, lze trénovat dál. Ve chvíli, kdy začne chyba na validační sadě stoupat, znamená to, že klasifikátor začíná být přetrénovaný a je třeba trénování zastavit.

5.2 AdaBoost

Základní ideou „boostovacích“ algoritmů je iterativně kombinovat relativně jednoduchá predikční pravidla (slabý klasifikátor) k vytvoření velmi přesného rozhodovacího pravidla (silný klasifikátor). Ve většině „boostovacích“ algoritmů dochází k lineární kombinaci slabých klasifikátorů.

Boosting je původně pojem z teorie strojového učení a v analýze dat se tak obvykle označuje algoritmus AdaBoost. Algoritmus AdaBoost (Adaptive Boosting) byl představen v roce 1995. Jeho hlavním přínosem je schopnost exponenciálně snižovat chybu výsledného klasifikátoru na libovolně nízkou úroveň (s danou množinou vzorků a slabých klasifikátorů). Formální důkaz tohoto faktu lze nalézt v článku [8]. AdaBoost dokáže produkovat v relativně krátkém trénovacím čase klasifikátory s velmi malou chybou i za použití jen velmi jednoduchých slabých klasifikátorů. Takové klasifikátory pak mají výhodu, že jsou při použití v reálných podmínkách velice přesné a jejich vyhodnocení lze provést ve velmi krátkém čase. Ukazuje se, že jsou vhodné pro úlohy jako například detekce obličeje ve videosekvenci v reálném čase [9, 10].

Základní varianta algoritmu [6, 7] je určena ke kombinování několika klasifikátorů do jednoho tak, aby výsledná klasifikační funkce byla přesnější než všechny použité klasifikátory. Výsledkem je silný klasifikátor (strong classifier), který se skládá z několika slabých klasifikátorů (weak classifier). Slabé klasifikátory mohou mít libovolnou složitost, ale v mnoha případech se volí jen jednoduché funkce. Algoritmus každému přiřadí váhu na základě jeho chyby.

5.2.1 Algoritmus

Základním úkolem algoritmu je vybrat z velkého množství slabých klasifikátorů malou podmnožinu tak, aby tyto klasifikátory co nejlépe rozdělovaly dané vzorky do svých tříd. V článku od autorů Freund a Schapire [8] je popsána základní varianta algoritmu pro klasifikaci do dvou tříd. Článek také obsahuje dvě rozšíření algoritmu pro klasifikaci do více tříd.

Vstupem algoritmu je sada trénovacích dat x a jejich ohodnocení y (třída, do které náleží) – $(x_1, y_1) \dots (x_m, y_m)$, kde každé $x \in X$ je instance hledaného vzoru a $y \in Y$ je jeho ohodnocení. V našem případě platí, že $Y = \{-1. +1\}$. Hodnotou 1 jsou označeny hledané objekty, -1 mají vzorky protipříkladů. Hlavní myšlenkou algoritmu je, že uchovává distribuci trénovacích vzorků neboli jejich váhu. Díky této distribuci se algoritmus přizpůsobuje těžko klasifikovatelným vzorkům v trénovacích datech a postupně opravuje svou funkci tak, aby i tyto vzorky dokázal správně zařadit. Váha vzorku

i v trénovacím kroku t je $D_t(i)$. Na začátku jsou váhy všech vzorků nastaveny stejně. V každém dalším kroku je váha chybně klasifikovaných vzorků zvýšena a u správně klasifikovaných snížena. To dovoluje algoritmu opravovat rozhodnutí u chybně klasifikovatelných vzorků.

Úkolem algoritmu je v každém kroku nalézt právě jeden slabý klasifikátor z dané množiny, který pro distribuci $D_t(i)$ nejlépe klasifikuje vzorky v trénovací sadě. Slabý klasifikátor je funkce $h_t: X \rightarrow \{-1, +1\}$, která každému vzorku z množiny X přiřadí jeho ohodnocení. Vhodnost klasifikátoru h_t se zjišťuje pomocí jeho chyby ε_j . Jedná se o součet vah špatně klasifikovaných vzorků.

$$\varepsilon_j = \sum_{i=1}^{m} D_t(i) \left[y_i \neq h_j(x_i)\right] \tag{21}$$

Pro každý slabý klasifikátor h_t je vypočítán parametr α_t , který znamená důležitost klasifikátoru – čím nižší je jeho chyba, tím vyšší je α_t .

$$\alpha_t = \frac{1}{2} \log\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right) \tag{22}$$

Aktualizace distribuce pro další iteraci algoritmu (krok 3 v následujícím pseudokódu) má za úkol zvýšit důležitost u chybně klasifikovaných vzorků a snížit u správně klasifikovaných. Tímto se algoritmus zaměřuje na obtížné vzorky v trénovací sadě.

Celý algoritmus lze pak zapsat pseudokódem.

Vstup algoritmu: $(x_1, y_1) \dots (x_m, y_m), x \in X, y \in \{-1. +1\}$

Inicializace: $D_1(i) = \frac{1}{m},\ i = 1, \dots, m$

Pro $t = 1, \dots, T$

1. Určení slabých klasifikátorů. Nalezení optimálních parametrů (pokud nějaké má) každého slabého klasifikátoru tak, aby měl co nejmenší chybu ε_j na aktuální distribuci D_t

$$h_t = min\, \varepsilon_j = \sum_{i=1}^{m} D_t(i) \left[y_i \neq h_j(x_i)\right]$$

2. Pokud chyba klasifikátoru je $\varepsilon_t \leq 0{,}5$ tak stop
3. Výpočet váhy klasifikátoru $\alpha_t = \frac{1}{2} \log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$
4. Aktualizace vah vzorků v trénovací množině

   $D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$, kde Z_t je normalizační faktor, zvolený tak, aby funkce $D_{t+1}(i)$ zůstala pravděpodobnostním rozložením

Algoritmus 1: AdaBoost

Výsledkem je lineární klasifikátor $H(x)$, který je lineární kombinací tzv. slabých klasifikátorů.

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t \, h_t(x)\right) \tag{23}$$

Podmínka v kroku 2 zajišťuje to, aby byl nalezený slabý klasifikátor lepší, než klasifikátor vracející náhodnou třídu. To je potřeba k zajištění konvergence algoritmu.

Výběr příznaku podle minima vážené chyby společně s výpočtem koeficientu pro lineární kombinaci α_t v kroku 3, jsou navrženy tak, aby byl v každém kroku učení minimalizován horní odhad chyby výsledného klasifikátoru.

Aktualizace vah v kroku 4 způsobí to, že váha špatně klasifikovaných měření se zvětší a váha dobře klasifikovaných měření se zmenší. V následujícím kroku tedy bude hledán slabý klasifikátor, který bude muset lépe klasifikovat doposud špatně klasifikovaná měření.

Největší výhodou algoritmu AdaBoost je, že dokazatelně a velmi rychle konverguje k hypotéze s nízkou chyba na trénovací sadě. Tato skutečnost je pravdivá za předpokladu, že učící algoritmus dokáže pokaždé nalézt slabou hypotézu, jejíž chyba je nižší než náhodné hádání nad aktuální distribucí D_t.

5.3 WaldBoost

Waldboost je metaalgoritmus, který bývá použit ve spojení s jinými algoritmy učení, aby zvýšil jejich rychlost. Právě algoritmus AdaBoost popsaný v předchozí kapitole, musí projít (vyhodnotit) vždy všechny slabé klasifikátory, což může mít velký vliv na rychlost zpracování, neboť těchto klasifikátorů může být velké množství. Klasifikátor natrénovaný touto metodou nemusí být vyhodnocen vždy celý. Pokud v průběhu klasifikace vzroste jistota, že vzorek bude klasifikován do určité třídy, vyhodnocení je možné ukončit.

Základem algoritmu WaldBoost je tedy algoritmus AdaBoost a dále metoda SPRT (Sequentional Probability Ratio Test). Tato metoda se označuje jako sekvenční rozhodovací strategie.

Výhodnocení v každém kroku probíhá tak, že se provede jedno měření (vyhodnotí se jeden slabý klasifikátor) a určí se, zda patří do jedné ze dvou tříd +1 nebo -1 na základě dvou prahů A, B metody SPRT. Pokud vyhodnocení R_t je menší nebo rovno B je klasifikován jako +1, pokud je vyšší nebo roven A je klasifikován jako -1. Pokud je R_t v intervalu mezi A a B, pak je nutné provést další měření.

$$S = \begin{cases} +1 & R_t \leq B \\ \# & B < R_t < A \\ -1 & R_t \geq A \end{cases}$$

kde # znamená, proveď ještě jedno měření a R_t je

$$R_t = \frac{p(x_1, \dots, x_t; y = -1)}{p(x_1, \dots, x_t; y = +1)}$$

Konstanty A a B jsou nastaveny v závislosti na dvou parametrech α(false negative rate) a β(false positive rate). V praxi se parametry A a B určují obtížně, proto se určují na základě odhadu.

$$A' = \frac{1-\beta}{\alpha}, B' = \frac{\beta}{1-\alpha}$$

Vstup algoritmu: $(x_1, y_1) \dots (x_m, y_m)$, $x \in X$, $y \in \{-1. +1\}$, míry chyb α, β

Inicializace: $D_1(i) = \frac{1}{m}$, $i = 1, \dots, m$; nastavení prahů $A = \frac{1-\beta}{\alpha}, B = \frac{\beta}{1-\alpha}$

Pro $t = 1, \dots, T$

1. Nalezení nejlepšího slabého klasifikátoru h_t na základě algoritmu AdaBoost
2. Odhad R_t

$$R_t = \frac{p(H_t(x); y = -1)}{p(H_t(x); y = +1)}$$

3. Nalezení prahů $\theta_A^{(t)}$ a $\theta_B^{(t)}$ na základě parametrů α, β.
4. Z množiny trénovacích vzorků odeber ty, pro které platí

   $H_T \leq \theta_A^{(t)}$ nebo $H_T \geq \theta_B^{(t)}$
5. Naplň trénovací sadu novými vzorky

Výstup: silný klasifikátor H_T a prahy $\theta_A^{(t)}$ a $\theta_B^{(t)}$.

Algoritmus 2: WaldBoost

Pro aplikace, které jsou kritické z hlediska rychlosti klasifikace, je možné natrénovat klasifikátory tím způsobem, že pro vzorky dat, která jsou jednoduše klasifikovatelná, vyhodnotíme pouze malé množství slabých hypotéz. Jednoduše klasifikovatelné vzorky dat, jsou takové, pro které klasifikátor v určitém kroku klasifikace rozhodne s dostatečnou důvěryhodností. Obecně je v úloze detekce objektů klasifikátory extrémně náročné skenovat celý obrázek se všemi možnými podokny a velikostmi oken na různých pozicích a případně rotacích. Z toho plyne velké množství vyhodnocování klasifikátoru a velmi záleží na efektivnosti výpočtu klasifikátoru. Typicky pro detekci obličeje se používají stovky slabých klasifikátorů, avšak průměrný počet vyhodnocení slabých hypotéz na jedno podokno může klesnout v průměru až na pět. Toto nemusí nutně vést ke snížení přesnosti klasifikace. Tohoto faktu, že některé vzorky dat jsou snadněji klasifikovatelné, se využívá také již při trénování. Této technice se říká „bootstrapping“ a je hojně využívána při strojovém učení. Jedním z možných zrychlení trénování je již představený algoritmus WaldBoost a další možností je využití kaskády slabých klasifikátorů pro zrychlení klasifikace. Tyto kaskády poprvé použily Viola a Jones ve své detekci obličejů v reálném čase. Jejich řešení bylo asi patnáctkrát rychlejší než

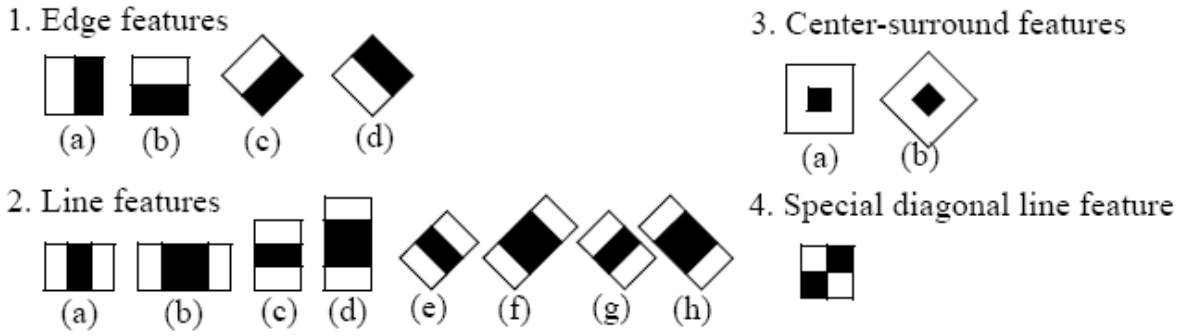
dosavadní nejrychlejší metody a stále drží statut nejvýkonnější detektor objektů v reálném čase. Hlavní ideou kaskádové klasifikace je snížení počtu nutných vyhodnocování slabých hypotéz nad jednotlivými podokny vyhodnocovaného obrazu. Slabé klasifikátory jsou rozděleny do posloupnosti, kde každý klasifikátor má definovanou hranici pro vyhodnocení, zda vzorek dat je negativní či pozitivní. Pokud je výsledek pozitivní, je předáno podokno dalšímu klasifikátor k vyhodnocení. Pokud takto vzorek projde všemi klasifikátory, je vzorek označen za hledaný objekt. Pokud v daném vyhodnocování je vzorek označen za negativní, je automaticky vyloučen z dalších výpočtu hypotéz. Tyto kaskády jsou trénovány za použití techniky „bootstrapping“ – následující klasifikátory jsou trénovány pouze za použití těch vzorků dat, které prošly předchozími stavy. Toto dovoluje trénovacímu procesu efektivně a přesně vyhodnotit slabé hypotézy i pro náročnější vzorky dat, které projdou do posledního stavu kaskády.

5.4 Slabé klasifikátory a příznaky

Algoritmus AdaBoost potřebuje pro svoji činnost slabé klasifikátory, ze kterých je nakonec vytvořen silný klasifikátor, jak již bylo zmíněno v popisu tohoto algoritmu. Slabým klasifikátorem může být jakákoliv funkce, která má lepší výsledky než náhodný výběr. Chceme-li například vytvořit klasifikátor pro rozdělení množiny lidí na muže a ženy, tak jedním z takových slabých klasifikátorů může být výška člověka (protože muži jsou v průměru vyšší než ženy). Při hledání objektů v obraze je nutné pracovat s obrazovými příznaky.

Obrazové příznaky můžeme rozdělit na spojité a diskrétní. Spojité příznaky vrací jako svoji hodnotu jakékoliv reálné číslo. Diskrétní pak jako výsledek vrací hodnoty z množiny celých čísel.

Příznak $f_i(x)$ je náhodná proměnná popisující některé vybrané vizuální charakteristiky obrazu jako například skladba nízkých frekvencí v obraze. Například v úloze hledání obličejů v obraze je to nalezení takových příznaků, které odliší obličej od ostatních částí obrazu. V tomto příkladě se používají Haarovy příznaky, protože jsou díky integrální reprezentaci obrazu velmi rychlé na výpočet. Hlavní nevýhodou Haarových příznaků a jiných typů příznaků založených na „waveletech“ je, že je nutné tyto vzorky dat normalizovat, abychom získaly vzorek dat invariantní vůči jasovým změnám. Typicky se používá invariance standardní odchylkou intenzity vzorku. Haarovy příznaky jsou odvozeny od Haarových vlnek (waveletů). Haarovy vlnky jsou nejjednodušším možným typem vlnek. Haarovy příznaky rozšiřují vlnky do 2D. Nejjednodušším typem Haarových příznaků jsou dvě příléhající oblasti obdélníkového tvaru stejné velikosti osově souměrné. Výsledkem je pak rozdíl hodnoty intensity těchto dvou oblastí. Existují ovšem i komplikovanější Haarovy příznaky. Příklady často používaných příznaků můžeme vidět na obrázku 5.2.



Obrázek 5.1 Haarovy příznaky – pixely ležící uvnitř bílé části jsou odečteny od pixelů ležících v tmavé části. Tyto příznaky jsou schopné detekovat některé jednoduché visuální vlastnosti jako hrany, přímky nebo rohy. [18]

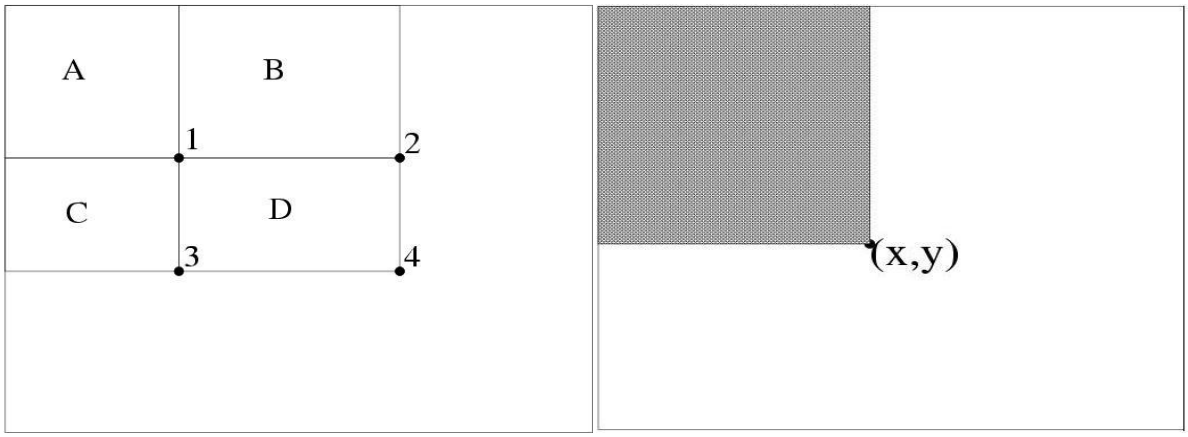
Počet všech možných Haarových příznaků v obraze o rozměrech 40x40 pixelů je mnohem vyšší než dimenze (počet pixelů v obrazu). Každý takovýto příznak popisuje nějaký vizuální atribut obrazu.

Hodnota Haarova příznaku je rozdíl sum šedotónových hodnot pixelů bílých a černých částí obdélníkové oblasti, viz obrázek 5.2.

Vyjádření obrazu pomocí *integrální reprezentace* dat má velké výhody při počítání s hodnotami pixelů v nějaké oblasti obrazu. Hlavní výhodou je rychlost výpočtu např. rozdílu sum pixelů dvou oblastí v obrazu, neboť s velikostí oblasti, v které počítáme sumu pixelů, roste i doba potřebná k výpočtu.

V pravé části obrázku 5.3 vidíme souřadnice bodu v obrazu. V integralním obrazu bude v tomto místě hodnota rovna sumě hodnot pixelů v původním obrázku nalevo nahoru od tohoto bodu.

Pokud tedy budeme chtít znát sumu pixelů v nějaké čtvercové oblasti, viz levá část obrázku 5.3, tak výsledná hodnota bude pro oblast D: $ii(4) + ii(1) - ii(2) - ii(3)$



Obrázek 5.2 Integrální obraz

5.5 Knihovna pro vytváření dat pro trénování klasifikátoru

Úloha, kterou se tato práce zabývá, je za pomoci algoritmu AdaBoost nalézt vhodné klasifikátory pro detekci korespondujících bodů v obraze. Jak již bylo v kapitole o AdaBoost popsáno, je nutné vytvořit trénovací a validační sadu dat, na které se budou nové klasifikátory vytvářet. Takováto data obsahují jednak pozitivní vzorky, které patří do hledané množiny a negativní vzorky, které by s hledanou množinou vzorků měly mít co nejméně společných znaků. Pokud například budeme trénovat klasifikátor na rozpoznávání psaných číslic, musíme vytvořit velké množství obrazových dat napsaných číslic. Zde je to poněkud složitější než v případě hledání obličejů, kde jsou pouze dvě množiny jedna pozitivní (obličeje) a negativní (vše ostatní). Je tedy nutné mít velký počet (desetitisíce) obrázků obličejů a obrovský počet (statisíce) negativních obrázků. Takovéto obrázky jsou v podobě sad výřezů předány na vstup pro trénování algoritmem AdaBoost. Definují se, jak budou vypadat hledané příznaky, např. Haarovy, a na základě definic jejich parametrů se spustí trénování klasifikátorů.

Pro hledání korespondujících bodů v obraze je potřeba vytvořit vhodnou množinu trénovacích dat. Představme si, že hledáme korespondující body v sekvenci snímků pořízené kamerou. Snímky jsou řazeny za sebou, tak jak je kamera vytváří. Charakteristické je to, že mezi jednotlivými snímky nedochází k velkým rozdílům v obrazu. Z pohledu jednotlivých pixelů obrazu dochází k malým rotacím, translacím, změně jasu. Z pohledu oblasti pixelů dochází ke změně měřítka.

V předchozích kapitolách byly popsány způsoby určení významných bodů v obraze. Nalezení korespondencí těchto významných bodů je z časového hlediska náročné a je zapotřebí hledat takové metody, které budou umět nalézt korespondující oblast v kratším časovém úseku, než existující metody. Pro účely zpracování obrazu v reálném čase, např. videosekvence, je tento požadavek prvotní.

Vstupem do trénovacího algoritmu jsou pozitivní a negativní sady obrázků. Každá z těchto sad je navíc ještě rozdělena na trénovací a validační sadu. Pro trénovací a validační pozitivní množinu je tedy nutné vytvořit takovou sadu obrazových dat, která bude tvořena dvojicemi snímků, které si vzájemně odpovídají, ale jeden z nich je nepatrně transformovaný.

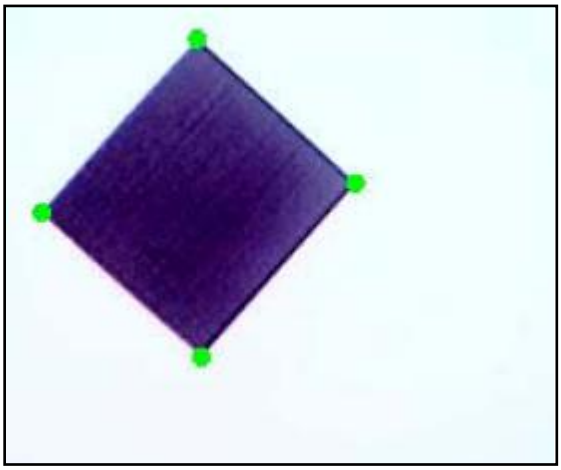
Cílem knihovny je tedy vytvořit takovouto sadu obrazových dat. Pro negativní vzorky je důležité, aby dvojice snímků byla z různých částí obrazu, tzn. aby si vzájemně dvojice snímků odpovídaly co nejméně.

5.5.1 Implementace

Knihovna je implementována za pomoci knihovny OpenCV, která obsahuje velké množství optimalizovaných funkcí pro práci s obrazem. Aplikace je implementována jako konzolová aplikace, která na vstupu očekává parametry. Vstupním parametrem aplikace je název konfiguračního souboru, v němž jsou popsány pomocí parametrů potřebné informace pro vytvoření sady výřezů.

Knihovnu pro generování výřezů můžeme rozdělit na dvě části. První část generuje výřezy pro pozitivní sadu a druhá část generuje výřezy pro negativní sadu. Pozitivní sadu výřezů totiž tvoří výřezy z oblasti obrázků, které jsou nějakým způsobem zajímavé z hlediska obrazové informace. Takovým místem může být například hrana či roh, jak bylo popsáno v části 2.

Pro nalezení těchto významných bodů využívá aplikace funkci knihovny OpenCV GoodFeaturesToTrack [13], která nalezne rohy s velkými vlastními čísly, viz popis Harrisova detektoru v části 2 tohoto textu. Funkce nejdříve spočítá vlastní čísla pro všechny pixely obrázku a následně vybere pouze maximální hodnoty v lokální oblasti 3x3. V dalším kroku jsou odstraněny rohy, jejichž vlastní čísla jsou menší než minimální nastavená hodnota. Dále se kontroluje, zda jsou nalezené rohy dostatečně vzdálené, pokud nejsou, jeden z dvojice se odstraní.



Obrázek 5.3 Ukázka detekce rohů pomocí funkce GoodFeaturesToTrack

Takto získaná množina je následně použita pro generování výřezů. Detekované rohy jsou středem výřezu. Sadu výřezů tvoří dvojice snímků. První výřez je původní a druhý z dvojice je vytvořen transformací prvního výřezu. S výřezem může být provedeno několik transformací zároveň. Jaké transformace budou použity, je definováno v konfiguračním souboru. Mezi podporované transformace patří rotace, změna měřítka, posunutí. Parametry jednotlivých transformací jsou na základě vstupních parametrů náhodně generovány. U každé transformace je definována střední hodnota a směrodatná odchylka a na základě tohoto, jsou parametry pro každý výřez náhodně generovány.

Negativní sada výřezů je generována tak, že se opět najdou rohové body a v jejich okolí se vytvoří výřezy. Tyto výřezy se pak zkombinují způsobem „každý s každým“, takže z jednoho obrázku

vznikne velká sada negativních výřezů. Tímto způsobem můžeme jednoduše dosáhnout požadavku pro velké množství negativních snímků pro trénování klasifikátoru.

Program na výstupu zobrazuje průběh generování výřezů. Výsledná sada výřezů je uložena jako raw soubor, jehož hlavička je tvořena informací o tom, kolik je v souboru výřezů a rozměry těchto výřezů.

Formát hlavičky raw souboru je následující:

```
typedef struct tHeader {
  char ID[4];
  unsigned int Cnt;
  unsigned int sizeX;
  unsigned int sizeY;
};
```

ID – je řetězcový identifikátor o délce 3 znaky, pro zajištění kompatibility formátu raw se stejným formátem jež využívá knihovna pro trénování algoritmem AdaBoost, byla zvolena hodnota „S2D“.

Cnt – počet výřezů, které jsou v raw souboru obsaženy

sizeX – horizontální velikost výřezu, v tomto případě je to šířka dvojice výřezů

sizeY – vertikální velikost výřezu

Funkce knihovny

Knihovna je napsána za použití jazyka C. Obsahuje několik důležitých funkcí.

`int parseInputFile(const char *fileName)`

- tato funkce slouží pro zpracování konfiguračního souboru, jejím vstupem je název konfiguračního souboru
- zpracovává postupně jednotlivé parametry konfiguračního souboru a inicializuje proměnné definující, jaké transformace budou s obrazem provedeny. Pokud ve vstupním souboru narazí na chybný parametr, ohlásí chybu a skončí.

`int makeCut(const IplImage *src, IplImage *dst, int x, int y, int width, int height)`

- funkce vytvoří výřez v obrázku uložený v proměnné src na souřadnici x, y o šířce width a výšce height a uloží jej do proměnné dst

`void makeTransfM(IplImage *dst, const IplImage *src)`

- funkce provede transformace definované v konfiguračním souboru. Na základě parametrů se náhodně vygenerují parametry transformace. Protože jsou zde použity pouze lineární transformace, můžeme s výhodou použít reprezentaci bodů pomocí homogenních souřadnic.

Díky tomu můžeme vyjádřit transformace v jediné matici. Skládání transformací se totiž v tomto kontextu realizuje jako násobení matic.

- **Posunutí**

  Transformace posunutí nebo také translace bodu P je určena vektorem posunutí $\vec{p} = (X_t, Y_t)$. Matice posunutí má tvar

$$\mathbb{T}(X_t, Y_t) = \begin{bmatrix} 1 & 0 & X_t \\ 0 & 1 & Y_t \\ 0 & 0 & 1 \end{bmatrix} \tag{24}$$

  pro posun bodu $P(x, y) = [x, y, 1]$ platí $\overline{P}[x + X_t, y + Y_t] = \mathbb{T}(X_t, Y_t) \cdot P(x, y)$

- **Rotace**

  Matice transformace otáčení mají tvar

$$\mathbb{R}(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{25}$$

- **Změna měřítka**

  Změna měřítka ovlivňuje současně polohu i velikost transformovaného objektu ve směru souřadnicových os. Pokud je absolutní hodnota koeficientu měřítkování v intervalu (0,1), dochází ke zmenšení a přiblížení transformovaného objektu k počátku souřadnic. Je-li absolutní hodnota koeficientu větší než jedna, dojde k prodloužení. Je-li znaménko koeficientu záporné, dochází k prodloužení či zmenšení v opačném směru. Příslušná transformační matice je

$$\mathbb{S}(s_x, s_y) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{26}$$

  Kde s_x je koeficient změny měřítka ve směru souřadnicové osy x a s_y je koeficient změny měřítka ve směru souřadnicové osy y.

  Při změně měřítka, se nastavuje střed souřadnic do středu výřezu, tudíž nedochází k posunu transformovaného výřezu. Provádí se to tím způsobem, že se transformovaná část obrazu posune a poté aplikuje změna měřítka a následně se znova posune zpět.

- **Zkosení**

Knihovna také podporuje transformaci zkosení a to zkosení ve směru osy x. Transformační matice pro tento typ zkosení má tvar

$$\mathbb{S}_{hx}(sh_x) = \begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{27}$$

`void makeRandomVars()`

- funkce naplní parametry transformace náhodnými hodnotami podle definovaných vstupních parametrů. Parametry jsou vygenerovány náhodně s gaussovským rozložením. To znamená, že u každého parametru transformace je definována střední hodnota a odchylka.
- Pro gaussovské rozložení pravděpodobnosti se používá známý vzorec $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, σ – rozptyl, μ – střední hodnota

`CvPoint2D32f* makeCornerPoints(const int pocetRohu, int pocetDetekovanych)`

- funkce vytvoří pole souřadnic detekovaných rohů a v proměnné pocetDetekovanych vrátí počet detekovaných rohů, ten totiž nemusí být stejný jako počet požadovaných detekcí, když se v daném obraze vyskytuje málo vhodných rohových oblastí.
- Funkce také vyřadí ty oblasti, které jsou blízko okrajů obrazu. S těmi by mohl být problém při jejich transformaci
- v této funkci se pro výpočet souřadnic rohových bodů využije funkce goodFreaturesToTrack, popsané v předchozím textu

`void makeCutToFileCorner(const char *fileName,const IplImage *transfImage, const IplImage *origImage)`

- Tato funkce je jádrem celého programu. Na základě definovaného počtu výřezů vytvoří sadu, která je následně uložena do souboru. Uspořádání výřezů je vždy takové, že pro jeden výřez se vytvoří dvojice originál a transformovaný výřez. Tyto jsou pak umístěny vedle sebe. Je náhodně zvoleno, zda originální bude vlevo nebo vpravo. Dvojice výřezů jsou umístěny pod sebou.

5.5.1.1 Zajištění nezávislosti datové sady

Pro trénování klasifikátorů je důležité, aby trénovací algoritmus nenašel nějakou závislost na původních obrázcích, aby se nenatrénoval jen na určitý druh obrazových dat. Proto je součástí knihovna také program *shuffle*, který náhodně promíchá vygenerované výřezy z různých snímků.
Na vstup tohoto programu vchází seznam raw souborů s již vygenerovanými výřezy. Tyto výřezy se mezi jednotlivými soubory náhodně promíchají. Výstupem je opět sada raw souborů, které již obsahují promíchané výřezy.

5.5.2 Použití knihovny pro generování výřezů

Ukázka vstupních parametrů v konfiguračním souboru pro jeden vstupní obrázek:

```
i IMG_0040.bmp
o onegIMG_0040.raw
w 16
h 16
sr 1.0 0.3
rr 5 2
p 5000
```

V následující tabulce jsou popsány možné vstupní parametry

parametr	popis parametru
i	název vstupního souboru s obrázkem
o	název výstupního souboru se sadou výřezů
w	šířka výřezu
h	výška výřezu
sr	parametry změny měřítka, střední hodnota a směrodatná odchylka
rr	parametry rotace, střední hodnota a směrodatná odchylka
txr	parametry translace ve směru osy x, střední hodnota a směrodatná odchylka
txy	parametry translace ve směru osy y, střední hodnota a směrodatná odchylka
j	parametr pro změnu jasu v procentech
z	zkosení ve směru osy x
p	počet generovaných výřezů

Počet vygenerovaných výřezů nemusí souhlasit s počtem požadovaných výřezů. Je to dáno vlastnostmi funkce pro detekci rohových bodů v obraze goodFeaturesToTrack, které se nemusí podařit nadetekovat takové množství, které je požadováno. Počet generovaných rohových bodů je dán vlastnostmi vstupního obrazu a také parametry této funkce.

Obrázek 5.4 Ukázka výstupu generátoru výřezů pro pozitivní dvojice

6 Trénování klasifikátorů

Pro vlastní trénování klasifikátorů byl použit program pro trénování metodou AdaBoost, který je vyvíjen v rámci výzkumné skupiny zpracování obrazu a videa na Fakultě informačních technologií. Program je možné jednoduchým způsobem doplňovat o nové typy příznaků, které jsou na základě vstupních parametrů vygenerovány nad trénovací sadou dat.

Pro každý průběh trénování je potřeba vytvořit konfigurační soubor. V tomto souboru je definována řada parametrů ovlivňující průběh trénování, které nakonec vede k vygenerování výstupních hodnot. Mezi důležité parametry patří cesta k obrazovým souborům, počet kroků učení, parametr alfa, definující přesnost vygenerovaného klasifikátoru, počet pozitivních a negativních samplů, které budou použity během trénování, maximální velikost alokované paměti pro negativní a pozitivní samply, které je důležité nastavit správně, aby se veškerá data vešla do paměti a nemuselo docházet během průběhu výpočtu k odkládání dat na pevný disk, což samozřejmě vede k rapidnímu zpomalení výpočtu klasifikátorů. Dále je v konfiguračním souboru definováno, jaké typy příznaků se budou používat pro výpočet klasifikátorů. U jednotlivých typů příznaků je nutné definovat parametry ovlivňující to, jak budou příznaky generovány, s jakými parametry, např. minimální / maximální výška / šířka bloku. Jak bylo řečeno, vstupem je také parametr udávající umístění souboru s parametry definující, která obrazová data budou použita pro trénování. V tomto souboru jsou odlišena negativní a pozitivní data a dále u těchto skupin data pro trénovací cyklus, validační cyklus a testovací cyklus trénovacího algoritmu.

Pro trénování klasifikátorů je nutné použít velké množství trénovacích dat. Pro pozitivní sadu výřezů bylo vygenerováno 130000 dvojic výřezů. Pro negativní sadu bylo použito 3,5mil. dvojic.

6.1 Implementované klasifikátory

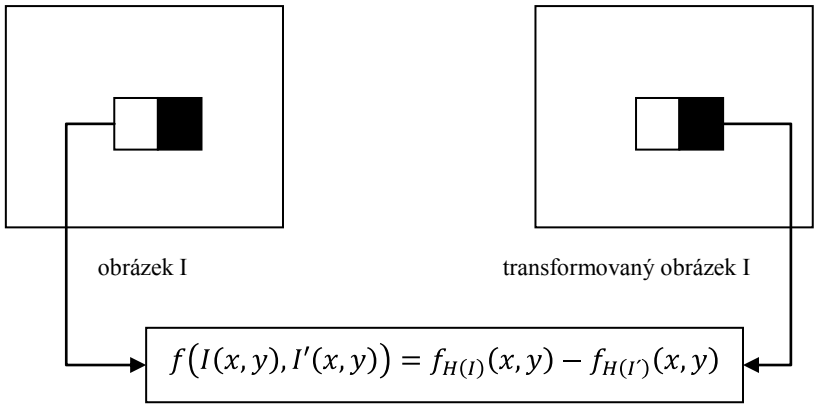
Z vlastností vygenerovaných pozitivních vzorků je patrné, že obsahují podobné obrazové informace. Tudíž jasová informace u takové dvojice výřezů se nebude příliš lišit. Možným řešením tohoto problému by mohlo být použití klasifikátoru, který bude odečítat jasové složky ploch, které jsou ve dvojici výřezů umístěny ve stejném místě o stejné velikosti. Toto řešení je ovšem velmi citlivé na vstupní pozitivní a negativní obrazová data. Pokud budeme mít dvojici negativních výřezů z různých oblastí obrázku, ovšem s podobným rozložením intenzit, pak bude tento klasifikátor selhávat.

6.1.1 Využití Haarových příznaků

Pokud se na vygenerované výřezy podíváme jako na oblast pixelů, zjistíme, že kromě intenzit ve snímcích je podobná struktura obrazu, tzn. v obou snímcích se díváme na velmi podobné pixely (což je v důsledku intenzita pixelů). Cílem úkolu je tedy nějakým způsobem zachytit podobnost dvojice výřezů z podobných snímků. Pro zachycení informace o vnitřní struktuře obrazu je možné použít výše zmiňované Haarovy příznaky. S řešením podobného problému se můžeme setkat v [11], kde Viola a Jones využívají Haarových příznaků k identifikaci osoby, tzn. nalezení podobného snímku osoby v databázi snímku k aktuálnímu snímku osoby.

6.1.2 Diference Haarových příznaků

Podobnostní funkce dvou snímků může být realizována pomocí rozdílu Haarových příznaků ve dvojici snímků. Tento rozdíl nám dává mnohem lepší informaci o struktuře snímku, než pouhé odečtení intenzit dvojic ve stejných oblastech dvojice snímků.



Obrázek 6.1 Funkce rozdílu Haarových příznaků ve dvojici snímků. $\boldsymbol{f_{H(I)}(x,y)}$ je Haarův příznak v obrázku I na pozici x, y

Obraz druhého z dvojice snímku bude obvykle nějakým způsobem transformován. V předchozí úvaze jsme do druhého výřezu umisťovali okno vyhodnocující Haarův příznak na stejnou pozici jako v prvním výřezu. Pokud ovšem bude druhý snímek transformován, pixely tohoto snímku budou oproti původnímu snímku posunuty (transformovány). Tuto transformaci můžeme postihnout pomocí posunu okna vyhodnocujícího Haarův příznak ve druhém výřezu snímku oproti původní pozici o jeden či více pixelů všemi směry. Takovéto rozšíření počtu vygenerovaných příznaků nám může dát lepší informaci o tom, kam budou transformovány pixely prvního výřezu. Nutno ovšem podotknout, že takto zvýšený počet příznaků (přibližně 7,5krát pro posun o jeden pixel ve všech směrech), znatelně zpomalí trénování klasifikátorů.

6.1.3 Popis implementace diference Haarových příznaků

Implementované příznaky využívají Haarových příznaků. Zdrojové kódy jsou umístěny v souborech *HaarAreaDifference.cpp* a *HaarAreaDifference.h*. Pro každý typ příznaku je vytvořena třída. Jsou tedy vytvořeny následující třídy:

- *THaarAreaDifferenceFeaturesH* – definuje rozdíl dvojice horizontálních Haarových příznaků
- *THaarAreaDifferenceFeaturesV* – definuje rozdíl dvojice vertikálních Haarových příznaků
- *THaarAreaDifferenceFeaturesHT* – definuje rozdíl dvojice horizontálních ternálních Haarových příznaků
- *THaarAreaDifferenceFeaturesVT* – definuje rozdíl dvojice vertikálních ternálních Haarových příznaků
- *THaarAreaDifferenceFeaturesDiag* – definuje rozdíl dvojice diagonálních Haarových příznaků
- *THaarAreaDifferenceFeaturesSurround* – definuje rozdíl dvojice surround Haarových příznaků

Pro každý typ příznaku, který má být použit k učení klasifikátoru, tzn. je definován v konfiguračním souboru, se vygeneruje na základě parametrů množina všech možných příznaků. Tyto příznaky vstupují do učícího procesu.

Každá třída obsahuje důležitou funkci *evaluate*, která pro daný příznak provede jeho vyhodnocení nad daným vzorkem dat.

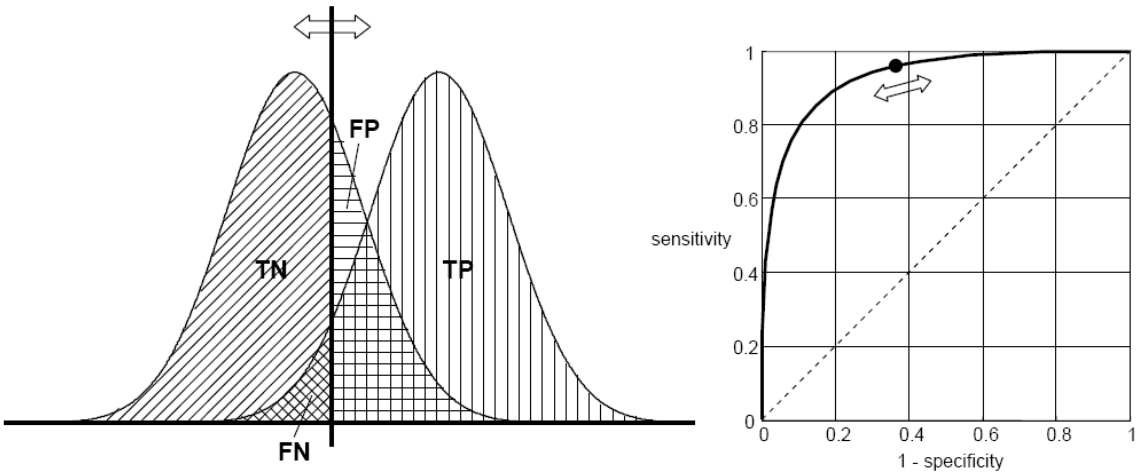
6.2 ROC

ROC (Receiver Operator Characteristics) křivky slouží k hodnocení a grafickému znázornění chování klasifikátorů při klasifikaci do dvou tříd. Na rozdíl od tradičně užívaných měr kvality, jako jsou např. celková správnost (accuracy) nebo celková chyba (misclassification error), netrpí zkreslením, k němuž dochází při nevyváženém zastoupení jednotlivých tříd v populaci, proto se s nimi lze setkat v širokém spektru aplikací od medicíny, přes bankovnictví, až po humanitní vědy.

ROC patří mezi nejvíce rozšířené a používané metody interpretace výsledků. ROC analýza je vhodná v těch případech, kdy používáme klasifikaci do dvou tříd. V této situaci můžeme rozlišit 4 případy. Pro vzory patřící první třídě, které byly správně klasifikovány, se užívá označení „true positive“ (TP). Pro vzory z první třídy, které byly chybně klasifikovány do druhé třídy, se užívá označení „false negative“ (FN). Další dva případy pro negativní vzorky se označují obdobně. („true negative“, „false positive“).

ROC je relací mezi správně klasifikovanými vzorky první třídy a chybně klasifikovanými vzorky druhé třídy. Na vertikální osu se tedy promítají hodnoty TP nebo jejich pravděpodobnostní rozložení. Tyto hodnoty jsou někdy nazývány jako sensitivity (citlivost). Na horizontální ose jsou hodnoty FP nebo jejich pravděpodobnostní rozložení.

Díky ROC křivce můžeme získat celkový přehled o charakteristice klasifikátoru. Čím rychleji se křivka blíží hodnotě jedna na vertikální ose, tím lepší je klasifikátor. Diagonální osa označuje oblast, kde jsou výsledky klasifikátoru rovny náhodné volbě.



Obrázek 6.2 Dvě různé interpretace ROC křiky

6.3 Experimenty s trénováním klasifikátorů

6.3.1 Datové sady

Pro účely testování generování různých klasifikátorů bylo vytvořeno několik sad dat.

První sada dat obsahuje šedotónová data s pozitivními a negativními vzorky za použití rohového detektoru pro nalezení významných oblastí v obraze. Při generování byl druhý ze snímků ve výřezu transformován rotací a změnou měřítka, v rozsahu $\langle 0°, 4° \rangle$ pro rotaci a $\langle 0.8, 1.2 \rangle$ pro změnu měřítka.

Druhá sada dat byla generována s použitím Laplaceova operátoru pro detekci hran se stejnými parametry transformace. Ve třetí sadě byla použita data pro pozitivní sadu stejná jako v první datové sadě plus pozitivní data, jejichž pozice ve snímku nebyla nalezena rohovým detektorem, ale náhodně vybrána.

Nakonec čtvrtá sada dat obsahuje více transformací, oproti první sadě obsahuje navíc ještě translaci v rozsahu $\langle -2px, 2px \rangle$.

Zdrojem obrazových dat byly snímky z různých scenérií. Zdrojová data obsahují snímky přírody i města.

Každá obrazová sada je rozdělena do několika souborů, aby bylo možno přidělit tato data k jednotlivým částem trénování klasifikátorů. Jsou to oddíly pro vlastní trénování, validaci a testování.

Protože výřezy byly generovány z různých snímků, bylo nutné zajistit, aby byly výřezy rozděleny rovnoměrně (náhodně z hlediska toho z jakých původních snímků byly vygenerovány). Toto je zajištěno náhodným promícháním vygenerovaných výřezů.

6.3.2 Konfigurace parametrů pro trénování klasifikátorů

Trénování klasifikátorů je zdlouhavý a náročný proces. Čím více kroků a vstupních dat bude použito, tím déle trénování trvá, a vyžaduje proto náležitou přípravu parametrů samotných testů před vlastním trénováním.

V konfiguračních souborech každého testu můžeme definovat spoustu parametrů. Jedním z nich je počet kroků trénování. V každém kroku trénování je vždy vyhodnocen nejlepší klasifikátor, který je posléze přidán k ostatním slabým klasifikátorům z předchozích kol. Tento parametr můžeme v konfiguračním souboru nalézt pod atributem *hypothesesToLearn*

```
    …
<TWaldBoostLearner hypothesesToLearn="80"
    …
```

Dále je možné definovat různé hodnoty parametru *alfa*, kterým vlastně určujeme rychlost a přesnost výsledného klasifikátoru. Čím je hodnota parametru alfa nižší, tím přesnější, ale pomalejší klasifikátor vygenerujeme. Čím je hodnota parametru alfa vyšší, tím méně přesný, ale rychlejší bude

výsledný klasifikátor. Parametr alfa nalezneme v konfiguračním souboru ihned za parametrem hypothesesToLearn.

Dalším parametrem určujeme nastavení počtu vzorků dat, která budou pro trénování načtena. Veškerá tato nastavení jsou v konfiguračním souboru v uzlu atributu *bootstrapper*. Je zde definován počet negativních a pozitivních vzorků dat načítaných ve fázi trénování a validace. Tento parametr je velmi důležitý z hlediska rychlosti trénování klasifikátorů. Pokud zařízení, na kterém trénovací algoritmus poběží, nemá dostatek prostředků (především operační paměti), bude trénování velmi zdlouhavé, protože paměť potřebná pro počet vzorků dat definovaných pro trénování je větší než fyzická paměť zařízení.

Důležitými parametry v tomto uzlu jsou *targetPositive/NegativeSetSize*, které mají význam počtu vzorků dat, která se mají nahrát z datové sady. Parametry *minPositive/NegativeSetSize* znamenají, že pokud počet samplů v datové sadě algoritmu poklesne pod tuto hodnotu, pak bootstrapping končí a nahrají se nová data.

Dalším významným parametrem je velikost posunu okna s Haarovým přiznakem ve druhé části výřezu. Hodnota tohoto parametru se nastavuje v konfiguračním souboru každého testu u každého typu Haarova příznaku.

```
…
<DecisionTreeLearner epsilon="0.001" maxDepth="2" maxLeafCount="8">
      <TCont2DiscFeatures minValue="-4.0" maxValue="4.0"
      numberOfBins="100">
      <HaarAreaDifferenceFeaturesV>
            <shift value="2" />
            …zde následují další parametry pro Haarovy příznaky
      </HaarAreaDifferenceFeaturesV>
</DecisionTreeLearner>
…
```

Z předchozího výpisu konfiguračního souboru je patrné, že každý typ Haarova příznaku, který chceme použít během trénování, je nutné definovat.

Pro experimenty bylo použito šest typů Haarových příznaků. Podle obrázku 5.2 jsou to tyto typy: 1a, 1b, 2a, 2c, 3a, 4.

V konfiguračním souboru jsou tyto příznaky označeny následovně:

typ příznaku	označení v konfiguračním souboru
1a	HaarAreaDifferenceFeaturesH
1b	HaarAreaDifferenceFeaturesV
2a	HaarAreaDifferenceFeaturesHT
2c	HaarAreaDifferenceFeaturesVT

3a	HaarAreaDifferenceFeaturesDiag
4	HaarAreaDifferenceFeaturesSurround

Jsou odvozeny od základních Haarových příznaků viz kapitola 5.6.1.2.

Pro každý typ Haarova příznaku se také kromě posunu okna nad druhým výřezem definují další potřebné parametry. Mezi tyto parametry patří:

minimální šířka bloku Haarova příznaku:

```
<minBlockWidth value="1"/>
```

minimální výška bloku Haarova příznaku:

```
<minBlockHeight value="1"/>
```

maximální šířka bloku Haarova příznaku:

```
<maxBlockWidth value="16"/>
```

maximální výška bloku Haarova příznaku:

```
<maxBlockHeight value="16"/>
```

krok při posunu okna Haarova příznaku ve směru osy X:

```
<shiftStepX value="1" />
```

krok při posunu okna Haarova příznaku ve směru osy Y:

```
<shiftStepY value="1" />
```

krok s jakým se bude zvětšovat šířka okna Haarova příznaku:

```
<blockWidthStep value="1" />
```

krok s jakým se bude zvětšovat výška okna Haarova příznaku:

```
<blockHeightStep value="1" />
```

6.3.3 Popis konfigurace jednotlivých testů

Označení jednotlivých testů odpovídá adresářům s výsledky testů na přiloženém cd.

testy č. 2

datová sada: pozitivní datová sada s rotací, změnou měřítka a posunem.

typy příznaků a jejich parametry: všechny typy Haarových příznaků zvlášť pro alfa=0.005, 0.01, 0.02. Všechny typy Haarových příznaků dohromady pro alfa=0.02. Počet kroků trénování 60.

testy č. 5

datová sada: pozitivní datová sada s rotací, změnou měřítka.

typy příznaků a jejich parametry: všechny typy Haarových příznaků zvlášť pro alfa=0.005, 0.01, 0.02. Všechny typy Haarových příznaků dohromady pro alfa=0.02. Počet kroků 80.

testy č. 6

datová sada: pozitivní datová sada s rotací, změnou měřítka + některé výřezy vybírány náhodně, bez pomoci rohového detektoru.
typy příznaků a jejich parametry: všechny typy Haarových příznaků zvlášť pro alfa=0.005, 0.01, 0.02. Všechny typy Haarových příznaků dohromady pro alfa=0.02. Počet kroků 80.

testy č. 7
datová sada: pozitivní datová sada s rotací, změnou měřítka, na obrazová data použit Laplaceův operátor
typy příznaků a jejich parametry: všechny typy Haarových příznaků zvlášť pro alfa=0.005, 0.01, 0.02. Všechny typy Haarových příznaků dohromady pro alfa=0.02. Počet kroků 80.

testy č. 8
datová sada: pozitivní datová sada s rotací, změnou měřítka. Okna Haarových příznaků posunuta ve dvojici snímků s krokem o ±1px.
typy příznaků a jejich parametry: všechny typy Haarových příznaků dohromady pro alfa=0.001, 0.02, 0.05. Počet kroků 200.

testy č. 10
datová sada: stejná jako u testů č. 8
typy příznaků a jejich parametry: podobně jako testů č. 8, ale okna Haarových příznaků posunuta ve dvojici snímků s krokem o ±2px, ±3px. Všechny typy Haarových příznaků dohromady pro alfa=0.02 a počet kroků 200.

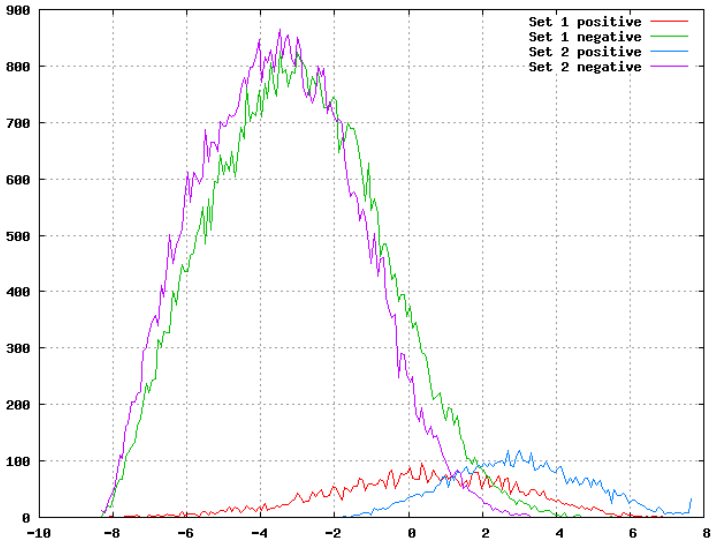
6.3.4 Výsledky experimentů

V následujících odstavcích budou popsány výsledky experimentů. Aplikace pro trénování klasifikátorů provádí také kromě trénování klasifikátorů i generování kontrolních grafů během průběhu trénování a po skončení všech kroků trénování. Díky nim je možno získat o natrénovaném klasifikátoru bližší informace a zjistit, jak je daný klasifikátor kvalitní.

testy č. 2
U těchto testů hrálo velkou roli to, že dvojice snímků v pozitivních datech byla vzájemně „hodně“ transformována. Tím pádem byl mezi dvojicí snímků velký rozdíl a pro použité Haarovy příznaky to znamenalo problém, resp. pro klasifikační funkci. Výsledky tomu také odpovídají. Algoritmus nedokáže rozlišit mezi pozitivními a negativními vzorky. U pozitivních snímků je díky velké

transformaci značný rozdíl. Proto byla v dalších testech použita pouze rotace a změna měřítka pro pozitivní datovou sadu (posun je vlastně v těchto transformacích již také obsažen).

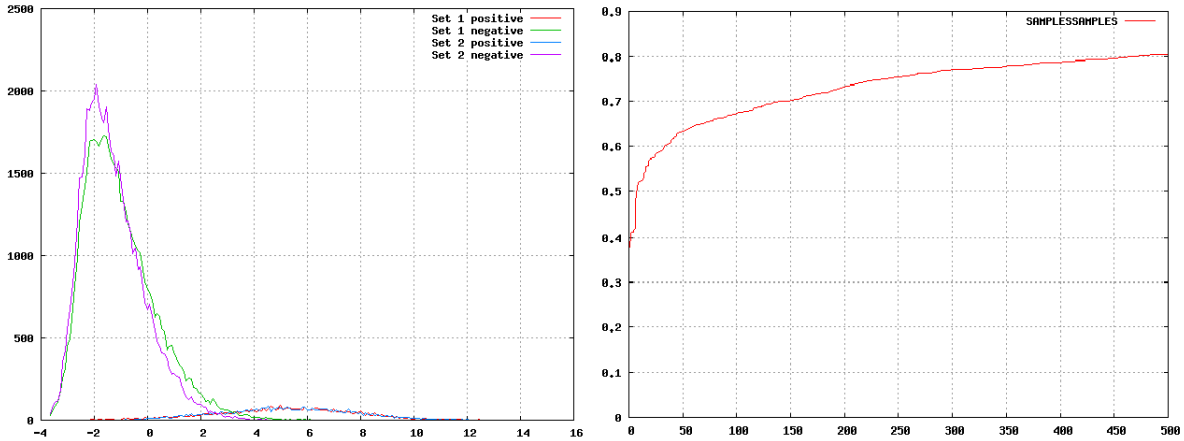
Výsledky ukazují, že klasifikační funkce nedokáže správně vyhodnotit dvojici snímků, které jsou vzájemně hodně transformované, a podobnost mezi nimi je nízká. Takovýto výsledek bylo možno očekávat, protože pozice okna Haarova příznaku v druhém snímku není vůči prvnímu snímku nijak transformována.



Obrázek 6.3 - ROC křivka pro testy č. 2 se všemi typy Haarových příznaků dohromady. Z grafu je patrné že algoritmus není schopen odlišit pozitivní a negativní vzorky (křivky pro negativní a pozitivní vzorky se překrývají)

testy č. 5

Výsledky těchto testů ukazují jisté zlepšení oproti předchozím testům. Vzorky nejsou zatíženy tolika transformacemi a algoritmus dokáže lépe rozlišit pozitivní vzorky od negativních.

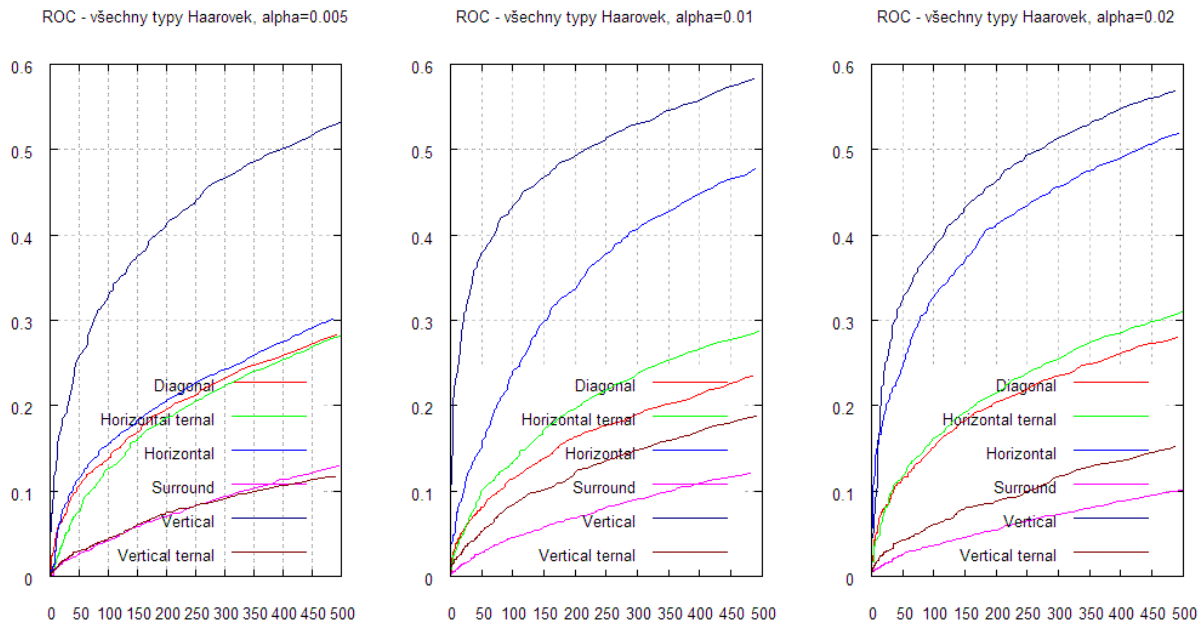


Obrázek 6.4 - výsledky experimentu testy č. 5 se všemi Haarovými příznaky

Z grafů na obrázku 5.9 je patrné zlepšení klasifikační funkce. Trénování klasifikátoru bylo provedeno, jak se všemi Haarovými příznaky dohromady, tak i s jednotlivými Haarovými příznaky

zvlášť. Na grafech obrázku 5.9 můžeme vidět rozdíl odezvy algoritmu u jednotlivých typů Haarových příznaků. Nejlepší výsledky dávají diference dvojic horizontálních a vertikálních Haarových příznaků.

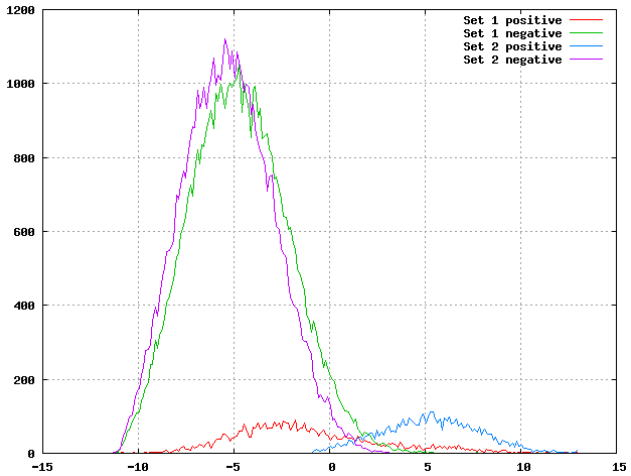
Výsledky potvrdily, že pokud dvojice snímků nejsou příliš transformovány, pak lze najít klasifikační funkci, která klasifikuje lépe než funkce u testů č. 2.



Obrázek 6.5 - výsledky experimentu test5 s jednotlivými Haarovými příznaky zvlášť. Tři různé grafy odpovídají třem různým nastavením hodnoty alfa.

testy č. 6

U tohoto souboru testů byly navíc součástí pozitivní datové sady dvojice výřezů, které byly vybrány náhodně, tzn. bez pomoci rohového detektoru, jak tomu je v ostatních případech. Díky tomu došlo ke zvýšení různorodosti pozitivních dat.

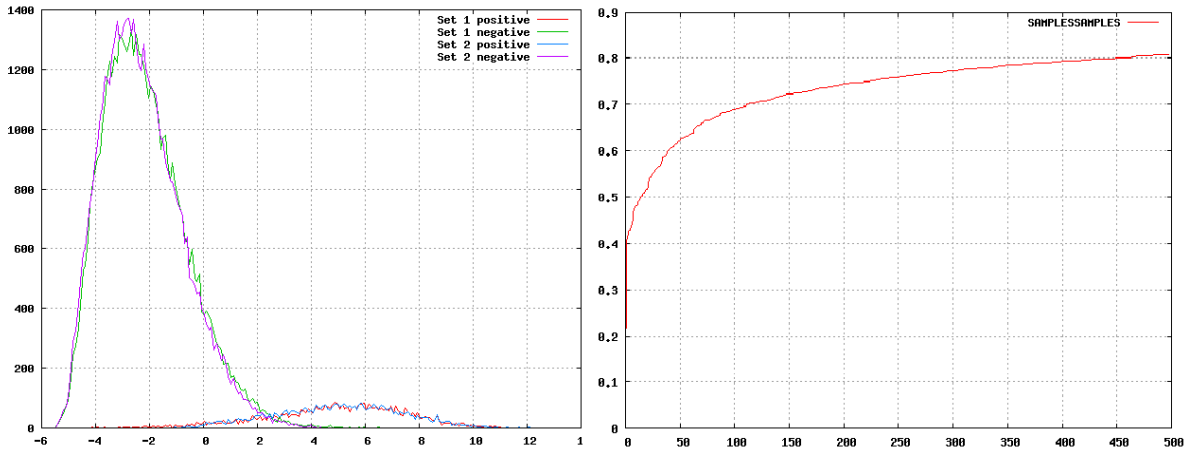


Obrázek 6.6 - ROC křivka pro testy č. 6 pro všechny typy Haarových příznaků použitých při trénování dohromady. Z grafu je vidět, že algoritmus nedokáže rozlišit mezi negativní a pozitivní datovou sadou

Z výsledků na grafu obrázku 5.11 vyplývá, že toto zvýšení různorodosti dat použitým Haarovým příznakům vůbec neprospívá a proto bylo v dalších experimentech upuštěno od používání této datové sady. Dále z výsledku experimentu vyplývá, že obrazové datové sady použité pro trénování klasifikátoru mohou být velmi různorodé a volba parametrů při generování datových sad hraje velmi důležitou roli ovlivňující vlastnosti datové sady.

testy č. 7

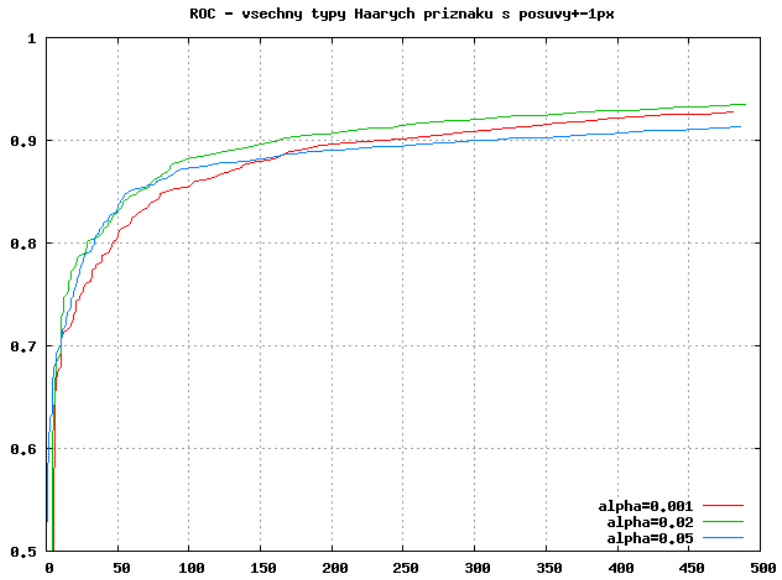
Při použití Laplaceova operátoru dojde v obrázku ke zvýraznění hran. Tento experiment by mohl přinést více informací pro trénovací algoritmus. Porovnáme-li výsledky s testy č. 5, zjistíme, že došlo k nepatrnému zlepšení. Provedení dalších experimentů s Laplaceovým operátorem, zejména s více kroky učení, by ověřilo smysl použití tohoto operátoru. Výsledky experimentu, v němž jsou použity všechny typy Haarových příznaků, jsou vidět na grafech obrázku 5.12.



Obrázek 6.7 - ROC křivky pro sadu experimentů testy č. 7. Grafy ukazují křivky při použití všech typů Haarových příznaků

testy č. 8, testy č. 10

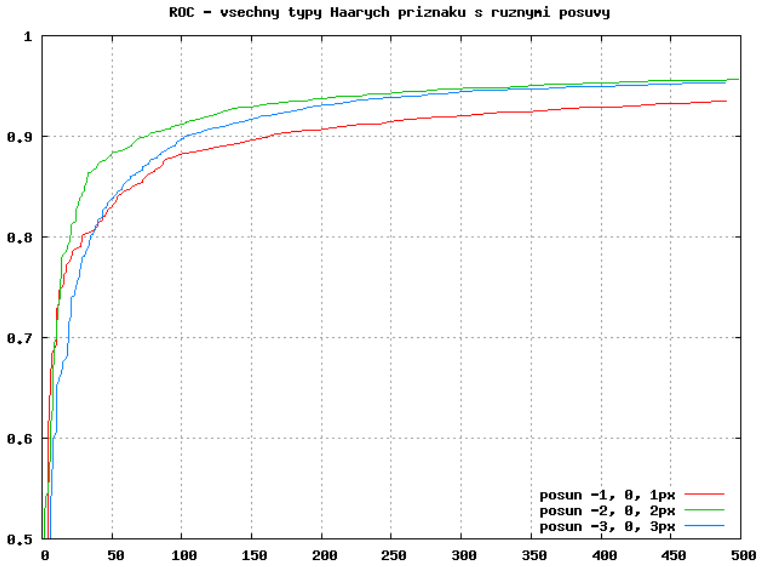
Experimenty v této sadě testů využily poznatků z výsledků předchozích testů a zaměřily se na použití takových parametrů, které v předchozích testech měly největší úspěch. Hlavní snahou těchto testů je zjistit, jak kvalitní klasifikátor se natrénuje při posunu okna Haarových příznaků u druhého výřezu o 1-3px. Vznikla tedy sada tří testů s posunem okna -1,0,1; -2,0,2; -3,0,3px. Díky tomu se vytvořilo pro jednu pozici okna Haarova příznaku v prvním snímku větší počet oken Haarových příznaků v druhém snímku, tzn. počet příznaků vzrostl. Díky tomu může trénovací algoritmus vybírat z více možností, který příznak bude v daném kroku trénování nejlepší.



Obrázek 6.8 - ROC křivka pro testy s posunem okna o ±1px s různou hodnotou alfa

Z grafu na obrázku 5.13 můžeme vidět, že nejlepších výsledků je dosaženo při použití hodnoty 0,02 parametru alfa. Pro další test byla zvolena právě tato hodnota.

Cílem posledního testu bylo zjistit, jak kvalitní klasifikátor lze natrénovat, za použití rozdílu Haarových příznaků nad dvojicí snímků s možností posunu okna Haarového příznaku u druhého snímku o různý počet pixelů. Cílem testu je tedy zjistit, pro kterou hodnotu parametru posunu okna je ROC křivka klasifikační funkce nejlepší.

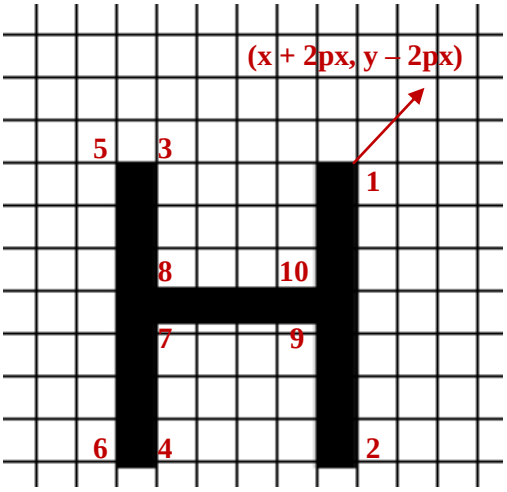


Obrázek 6.9 - ROC křivka klasifikační funkce pro různé parametry posunu okna Haarových příznaků v druhém snímku

Výsledky jsou patrné na grafu obrázku 5.14. Nutno dodat, že použité parametry při generování obrazové datové sady těmto výsledkům odpovídají a algoritmu se nejlépe daří klasifikovat vstupní snímky při použití posunu okna o ±2px.

6.3.4.1 Porovnání s existujícím algoritmem

Navržený algoritmus je vhodné srovnat s nějakou již existující metodou. Pro srovnání posloužila Lucas-Kanadeho metoda [12] estimace pohybu ve dvojici snímků. Jeho implementace je k dispozici v knihovně OpenCV. Pro srovnávací test byl použit nejlépe natrénovaný silný klasifikátor, který je výsledkem testů č. 10. Vstupem do obou testovaných algoritmů je množina pozic významných bodů prvního snímku. Oba algoritmy se pak snaží nalézt ty samé body v druhém snímku. Postup detekování v navrženém algoritmu je následující: Z prvního snímku se vytvoří výřez se středem v pozici detekovaného významného bodu. V druhém snímku, který je nějakým způsobem transformovaný se z 6x6px okolí této pozice (předpokládáme maximální posun 3px všemi směry) vytvoří sada výřezů. Každý výřez z této sady druhého snímku vytvoří s výřezem z prvního snímku dvojici, která je ohodnocena klasifikátorem. Nejlépe ohodnocená dvojice je vybrána a hodnota pozice výřezu druhého snímku je hledaný výsledek.



Obrázek 6.10 - Schéma znázorňující posun sledovaného objektu s označenými detekovanými významnými body

Na obrázku 6.10 je vidět tvar písmene H, které posloužilo jako sledovaný objekt pro srovnávací testy. Pro srovnání tedy byly vytvořeny dva snímky, z nichž ve druhém byl objekt posunut různými směry.

Na deseti testovaných bodech objektu vykázal navržený algoritmus podobné výsledky jako druhý algoritmus. Přesné detekce polohy bylo dosaženo u dvou bodů. V tabulce můžeme porovnat odchylky u jednotlivých bodů při posunu o 2px ve směru osy x a o -2px ve směru osy y, druhý sloupec u odchylek ukazuje detekovanou odchylku při posunu o -3px ve směru x a -3px ve směru y, třetí sloupec ukazuje posun o 5,-5px.

Velikosti posunů pixelů z výstupů obou algoritmů odpovídají velikosti posunům objektů v sekvencích snímků zachycujících pohybující se objekty.

	navržený algoritmus						srovnávací algoritmus					
	Δx			Δy			Δx			Δy		
skutečný posun	2px	-3px	5px	-2px	-3px	-5px	2px	-3px	5px	-2px	-3px	-5px
bod												
1	1	-3	4	-1	-2	-4	1	-4	4	-3	-3	-6
2	3	-1	4	-3	-2	-4	1	-4	4	-2	-4	-5
3	2	-3	5	-1	-2	-4	1	-4	4	-3	-4	-6
4	3	-1	5	-1	-2	-4	1	-4	4	-2	-3	-5
5	**2**	**-3**	**5**	**-2**	**-3**	**-5**	2	-4	4	-3	-3	-5
6	3	-2	**5**	-2	-3	**-5**	**2**	-4	4	**-2**	-4	-6
7	**2**	-3	5	**-2**	-3	1	2	-3	5	-3	-4	-6
8	3	-2	5	-1	-2	-4	1	-4	4	-3	-3	-5
9	2	-3	5	-1	-2	-4	2	-3	5	-3	-4	-6
10	2	-3	5	-1	-2	-4	1	-4	4	-2	-3	-5
průměr	2,3	-2,4	4,8	-1,5	-2,3	-3,7	1,4	-3,8	4,2	-2,6	-3,5	-5,5
průměrná odchylka	0,3	0,6	-0,2	0,5	0,7	1,3	-0,6	-0,8	-0,8	-0,6	-0,5	-0,5

Tabulka 1 – V tabulce jsou detekované hodnoty posunu objektu. První sloupec u Δx je pro posun o 2px, druhý pro -3px, třetí pro 5px, podobně je to u sloupce Δy.

Z výsledků v tabulce 1 je patrné, že navržený algoritmus dokáže detekovat posun zkoumaného bodu s přesností na jednotky pixelů. Důkazem tohoto faktu je průměrná odchylka ve všech detekovaných bodech, která nepřesahuje jeden pixel. Pouze u bodu 7 došlo při detekci posunu o -5px ve směru y k velké chybě, která přesahuje 6px. V porovnání s odchylkami u srovnávacího algoritmu vychází navržený algoritmus lépe. Ovšem odchylky u jednotlivých pixelů jsou v rámci každého detekovaného bodu zvlášť menší u druhého algoritmu. Komplikací navržené testovací metody může být větší posun, kdy je nutné generovat výřezy pro větší okolí hledaného bodu, což může být paměťově náročné.

7 Závěr

V této práci byly shrnuty některé poznatky z oblasti počítačového vidění, konkrétně z oblasti hledání významných znaků v obraze a vyhledání korespondujících významných znaků ve dvojici snímků.
V kapitolách o hledání významných znaků v obraze byly vytyčeny důležité typy znaků, které bývají základem pro hledání významných bodů v obraze. Asi nejpoužívanější metodou pro nalezení významných znaků se zdá být detekce rohů v obraze Harrisovým detektorem, který najde významný bod tam, kde je v jeho okolí velká změna jasu ve všech směrech od tohoto bodu.
Ve 3. a 4. kapitole bylo popsány používané metody pro hledání korespondujících bodů. V této oblasti dochází stále k bouřlivému vývoji. V současné době se také objevuje využití nových deskriptorů SIFT a zejména jeho novější a rychlejší varianta SURF.

Cílem práce bylo zjistit, zda je možné pomocí nějakého algoritmu založeného na strojovém učení, nalézt ve druhém obraze korespondující místo s místem v prvním obraze. Metoda využívá Haarových příznaků pro popis lokálních vlastností snímku, přesněji řečeno, využívá rozdílu intenzitních obrazů dvojice Haarových příznaků ve dvojici snímků. Na velkém množství experimentů a jejich výsledcích bylo zjištěno, že do budoucna má smysl se touto metodou rozpoznávání korespondujících bodů zabývat. Ovšem nutno podotknout, že bude nutné provést spoustu dalších experimentů, které užitečnost této metody podpoří či vyvrátí.
Z výsledků v šesté kapitole je vidět, že naučený klasifikátor dokáže rozpoznat pozitivní vzorky obrazových dat od negativních vzorků. Výhodou takovéhoto způsobu rozpoznávání je jeho rychlost. Pro zvýšení přesnosti bude určitě nutné vytvořit klasifikátor natrénovaný alespoň v tisíci krocích a použít posuny oken Haarových příznaků v rozmezí -3px až +3px s krokem 1px, což zajistí velké množství příznaků, ze kterých může trénovací algoritmus vybírat.
Závěr šesté kapitoly ukázal srovnání navrženého algoritmu s již existujícím algoritmem a výsledky ukazují, že navržený algoritmus dosahuje srovnatelných výsledků s již existujícím algoritmem.
Pro další experimenty by bylo také vhodné vygenerovat spoustu obrazových sad s různými parametry transformací, ke zjištění limitů popisované metody.
Hledání korespondujících bodů ve dvojici snímků nachází uplatnění na řadě míst. Může jím být sledování pohybu objektů či osob, tzv. trasování, nebo při 3D rekonstrukci snímané scény za použití dalšího matematického aparátu.

8 Literatura

[1] Bílek, P., Významné body v obraze: detekce, korespondence a lokalizace ve 3D, [Bakalářská práce], ČVUT Praha, 2007

[2] Lindeberg, T., Edge detection and ridge detection with automatic scale selection, Proc. CVPR'96, San Francisco, California, pages 465--470, 1996

[3] Interest point detection, 2007, http://en.wikipedia.org/wiki/Interest_point_detection

[4] Bay H., Tuytelaars T., Van Gool L., SURF: Speeded Up Robust Features, ETH Zurich, 2006

[5] Kaněčka P., Vyhledávání význačných bodů v obraze, [Diplomová práce], VUT Brno, 2007

[6] Horčík R., Korespondence mezi obrazy, [Diplomová práce], ČVUT Praha, 2001

[7] Harris C. , Stephens M. A combined corner and edge detector. Proceedings of the 4th Alvey Vision Conference: s. 147—151, 1988

[8] Freund Y., Schapire R., A Decision-theoretic Generalization of On-line Learning and an aplication to Boosting, Journal of Computer and System Sciences, 1997

[9] Viola P., Jones M., Robust Real Time Object Detection, SCTV, Vancouver, Canada, 2001

[10] Šochman J., Matas J., WaldBoost - Learning for Time Constrained Sequential Detection, Center for Machine Perception, Dept. of Cybernetics, Faculty of Electrical Engineering, ČVUT Praha, 2005

[11] Viola P., Jones M., Face Recognition Using Boosted Local Features. Mitsubishi Electric Research Laboratories, Inc., 2003

[12] Carlo Tomasi and Takeo Kanade. Detection and Tracking of Point Features. Carnegie Mellon University Technical Report CMU-CS-91-132, April 1991.

[13] Jianbo Shi and Carlo Tomasi. Good Features to Track. IEEE Conference on Computer Vision and Pattern Recognition, pages 593-600, 1994.

[14] *Fotometrická interpolace*, http://measure.feld.cvut.cz/groups/edu/osv/osv_dsp.html

[15] *Konvoluce*, http://cs.wikipedia.org/wiki/Konvoluce

[16] Lindeberg, T., Scale-space Theory in Computer Vision, Springer, 1994

[17] *Principal curvature*, http://en.wikipedia.org/wiki/Principal_curvatures

[18] Lienhart R., Kuranov A., Pisarevsky V., Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection, Microprocessor Research Lab, Intel Labs