VYSOKÁ ŠKOLA STROJNÍ A TEXTILNÍ V LIBERCI

Fakulta strojní

Katedra technické kybernetiky Školní rok: 1992/93

ZADÁNÍ DIPLOMOVÉ PRÁCE

pro Petra JÁNSKÉHO

obor 23-40-8 ASŘ výrobních procesů ve strojírenství

Vedoucí katedry Vám ve smyslu zákona č. 172/1990 Sb. o vysokých školách určuje tuto diplomovou práci:

Název tématu:
Počítačové zpracování barevných obrazů

Zásady pro vypracování:

1) Seznamte se se způsoby snímání a ukládání obrazové informace při jejím zpracování na číslicových počítačích
2) Navrhněte a sestavte aparaturu pro snímání obrazové informace osobním počítačem a její následný přenos na pracovní stanice DEC 5000/25. Řešte možnosti reprodukce barevných obrazů na monitorech VGA a DEC.
3) Navrhněte algoritmy a sestavte odpovídající programy (PASCAL, C) pro stanice DEC, které budou analyzovat jednoduché barevné scény.

Rozsah grafických prací:

Rozsah průvodní zprávy: 40 stran

Seznam odborné litaratury:

1) Šenke,M.-Hlaváč,V.: Vidění počítačem. V tisku

2) Manuály OS, jazyků, počítačů

Vedoucí diplomové práce: Ing. Petr Tůma

Konzultant: Ing.J.Buchta, Doc.Ing.V.Věchet,CSc.

Zadání diplomové práce: 31.10.1992

Termín odevzdání diplomové práce: 28.5.1993

L.S.

Vedoucí katedry

Doc.Ing.Vojtěch Konopa, CSc.

Děkan

Prof.Ing.Jaroslav Exner, CSc.

V Liberci dne 19.10. 1992

Vysoká škola stojní a textilní v Liberci

fakulta strojní

Petr Jánský

Počítačové zpracování barevného obrazu

DIPLOMOVÁ PRÁCE

1993

obor 31-40-8

ASŘ výrobních procesů ve strojírenství

Katedra technické kybernetiky.

Počítačové zpracování barevných obrazů

Jméno a příjmení autora: Petr Jánský

Vedoucí práce: ing. Petr Tůma, CSc.

Rozsah práce a příloh:

Počet stran: 32

Počet příloh: 2

Počet obrázků: 8

Počet tabulek: 1

Místopřísežně prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím uvedené literatury.

V Liberci dne 28. května 1993 .....

Obsah

Obsah 4
SEZNAM POUŽITÝCH ZKRATEK A SYMBOLŮ 5
1. ÚVOD 6
1.1.Cíl diplomové práce 6
2.Kolorimetrie 8
2.1.Základní poznatky z kolorimetrie 8
2.1.1.Určování barev 8
2.1.2.Mísení barev 8
2.2.Znázornění barev v souřadném systému 9
2.3.Snímání barevného obrazu 10
..... 11
2.4.Metrika v RGB prostoru 11
3.Identifikace barev 14
3.1.Standardní paleta 14
..... 14
3.1.1.Implicitně nastavená paleta 14
3.1.2.Ruční nastavení palety 15
3.2.Metoda shlukové analýzy 15
3.2.1.Základní pojmy shlukové analýzy 15
3.2.2.Shlukování 19
3.2.3.Přiřazení barev 19
4.Identifikace tvarů 21
4.1.Segmentace 22
..... 22
4.2.Rekonstrukce tvarů 22
4.2.1.Určování troj- a čtyř- úhelníků 22
4.2.2.Určování kruhových tvarů 24
5.Popis programu 26
5.1. Ovládání programu 27
..... 28
5.2. Běh hlavních procedur 28
..... 28
5.2.1. Nastavení palety 28
..... 29
5.2.2. Shluková analýza a Výpočet z VGA palety 29
5.2.3. Tvary 30
6. Závěr 31
..... 31
Použitá literatura: 32

Obsah

Obsah 4
SEZNAM POUŽITÝCH ZKRATEK A SYMBOLŮ 5
1. ÚVOD 6
1.1.Cíl diplomové práce 6
2.Kolorimetrie 8
2.1.Základní poznatky z kolorimetrie 8
2.1.1.Určování barev 8
2.1.2.Mísení barev 8
2.2.Znázornění barev v souřadném systému 9
2.3.Snímání barevného obrazu 10
2.4.Metrika v RGB prostoru 11
3.Identifikace barev 14
3.1.Standardní paleta 14
3.1.1.Implicitně nastavená paleta 14
3.1.2.Ruční nastavení palety 15
3.2.Metoda shlukové analýzy 15
3.2.1.Základní pojmy shlukové analýzy 15
3.2.2.Shlukování 19
3.2.3.Přiřazení barev 19
4.Identifikace tvarů 21
4.1.Segmentace 22
4.2.Rekonstrukce tvarů 22
4.2.1.Určování troj- a čtyř- úhelníků 22
4.2.2.Určování kruhových tvarů 24
5.Popis programu 26
5.1. Ovládání programu 27
5.2. Běh hlavních procedur 28
5.2.1. Nastavení palety 28
5.2.2. Shluková analýza a Výpočet z VGA palety 29
5.2.3. Tvary 30
6. Závěr 31
Použitá literatura: 32

SEZNAM POUŽITÝCH ZKRATEK A SYMBOLŮ

R,G,B ..... červená, zelená a modrá (Red, Green, Blue) složka bodu v obraze

DMA ..... Direkt Memory Access, přímý přístup do paměti

A/D ..... Analogově - číslicový převodník

KB ..... Kilobyte - 1024 B

1. ÚVOD

Pro získání obrazové informace existuje řada metod. Obrazy zpravidla bývají zaznamenány analogově na médium, jako je např.: magnetický pásek nebo film. Před zpracováním obrazové informace na počítači je nutno obraz po nasnímání digitalizovat. Zařízení pro digitalizaci obrazu se nazývá scanner v současné době se nejčastěji používají scannery ruční a stolní a barevné nebo černobílé.

Další možností jak získat obrazovou informaci je použití televizní kamery s vakuovou elektronkou, která snímá statický obraz nebo snímačů CCD, což jsou unipolární integrované obvody s maticí světlocitlivých buněk, jenž jsou obklopeny řídící elektronikou díky níž generují stejný TV signál jako kamera s vakuovou elektronkou. Protože tyto signály jsou analogové je nutno je dále diskretizovat například pomocí A/D převodníků.

Použité snímací zařízení instalované na KTK je kamera s analogovým výstupem připojená k počítači AT 286 přes kartu 6-bitovým paralelním A/D převodníkem, který umožňuje digitalizaci obrazu v matici 256x256 pixelů v 64 hodnotách jasu pro každý pixel. Přenos dat mezi snímačem a pamětí počítače je zajištěn pomocí DMA kanálu (kanál přímého přístupu do paměti s vyřazením procesoru).

1.1.Cíl diplomové práce

Cílem diplomové práce je možnost reprodukce barevného obrazu

na monitorech VGA a DEC. K realizaci vytyčeného cíle byla pro získání obrazové informace použita kamera připojená k PC, která je k dispozici na KTK a program ZOI jenž umožňuje mimo jiné uložení nasnímaného obrazu do souboru. K získání složek barevného obrazu byly zapůjčeny barevné filtry z katedry fyziky.

2.Kolorimetrie

2.1.Základní poznatky z kolorimetrie

2.1.1.Určování barev

V kolorimetrii rozlišujeme dva způsoby určení barvy:

a) *chromatičnost*

b) *kolorita*

ad a) Přímé zdroje světla jako jsou např.: sluneční světlo, světélkující luminofory televizní obrazovky, hořící látky nebo světlo žárovky jejichž světelné paprsky dráždí přímo zrak, aniž přitom mění své spektrum tj. rozložení složek světla. Podle obsahu těchto složek je určena barva světelného zdroje (*chromatičnost*). Barvu světelného zdroje měníme pomocí různých barevných filtrů. Například červené sklo brzdových světel automobilu nedovolí průchod všem spektrálním složkám "bílého" světla žárovky krimě červené.

ad b) Barva předmětů (*kolorita*) je dána schopností předmětů odrážet určité složky barevného spektra. Jestliže pozorujeme barvu předmětu, vnímáme jen odražené složky původního zdroje světla. Například trávník zadního pole baseballového hřiště vnímáme jako zelený, protože dopadající bílé světlo jeho povrch pohltí až na zelenou složku, která se odrazí. Kdybychom jej ozářili červeným světlem, byl by zcela černý.

2.1.2.Mísení barev

a) *subtrakční*

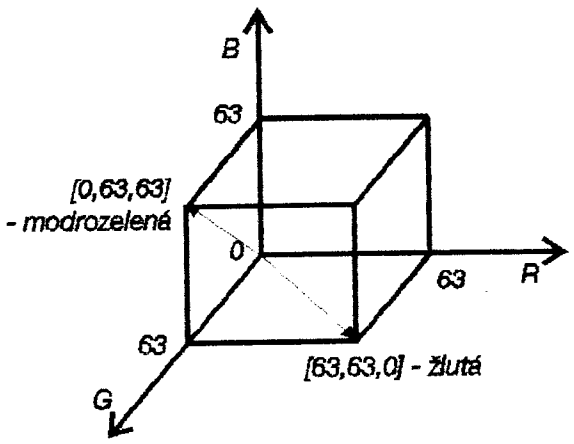
b) *adiční*

ad a) Tím, že odčítáme různé barevné složky z bílého (slunečního) světla, měníme poměr zbylých složek a získáváme tak jiné barvy. Toho lze dosáhnout buď průchodem přes filtry, nebo odrazem od povrchu barevného předmětu. Například mísení temperových barev: červené, modré a zelené, získáme černou barvu, stejného efektu dosáhneme, jestliže zařadíme za sebou před "bílé" světlo červený, modrý a zelený filtr. Tento způsob, kdy získáme novou barvu odčítáním nazýváme rozdílové (*subtrakční*) mísení barev.

ad b) Při současném dopadu dvou nebo více světel na bílou plochu vzniká nová barva. Intenzita zrakového vjemu se sčítá, jas se tedy zvětšuje. Při určitých energiích lze ze tří základních barev získat bílé světlo. Toto součtové (*adiční*) mísení se používá při reprodukci barevné scény barevnou obrazovkou.

2.2.Znázornění barev v souřadném systému

Pro součtové skládání barev lze použít většího počtu základních (primárních) barev k dosažení co největšího počtu barevných odstínů vyskytujících se v přírodě. V praxi se však z ekonomických důvodů používá pouze tří základních spektrálních barev. Různými vzájemnými poměry jasů těchto barev získáme většinu existujících barev.



obr. č. 1 - RGB prostor

Barevné světlo (jeho chromatičnost a jas) je určeno třemi trojbarvými součiniteli například R,G,B, jejichž velikostem lze přiřadit tři pravoúhlé souřadnicové směry (RGB prostor). Například žlutá barva má RGB souřadnice [63,63,0], černá [0,0,0] a bílá [63,63,63].

2.3.Snímání barevného obrazu

Pro získání úplné barevné informace je nutné každou barevnou scénu nasnímat třikrát. Po každé přes jeden ze tří barevných filtrů R,G,B. Použité filtry mají vlnové délky: červený (R) λ=610nm, zelený (G) λ=535nm, modrý (B) λ=472nm. Lze použít samozřejmě i filtry s jinou vlnovou délkou, ale použití těchto filtrů se však ukázalo výhodné ze dvou důvodů, a to:

1) vlnové délky 610, 535 a 472nm jsou v barevném spektru rozmístěny zhruba rovnoměrně a proto jejich kombinací lze nejlépe

obsáhnout celé barevné spektrum.

2) tyto tři barvy používají barevné monitory a procedura SetRGBColor Turbo Pascalu umožňuje definovat každou z 16 barev VGA palety pomocí tří parametrů, které představují hodnoty těchto barevných složek. Stejně lze použít i větší počet filtrů, ale potom se neúměrně zvětšují nároky na paměť a dobu výpočtu při zpracování na počítači.

Při snímání obrazu kamerou pomocí programu ZOI získáme kombinací 64 stupňů všech tří složek 262144 barevných odstínů. V programu ZOI byl pro uložení obrazové informace zvolen textový soubor s řádky dlouhými 256 znaků, kde každý znak má ASCII kód z intervalu od 20h do 5Fh (32 - 95 decimálně).

VGA monitor sice umožňuje zobrazit kterýkoli z 262144 odstínů, ale z této škály najednou pouze 16 barev (případně 256), proto je nutné barevnou informaci zredukovat na těchto 16 respektive 256 odstínů.

2.4.Metrika v RGB prostoru

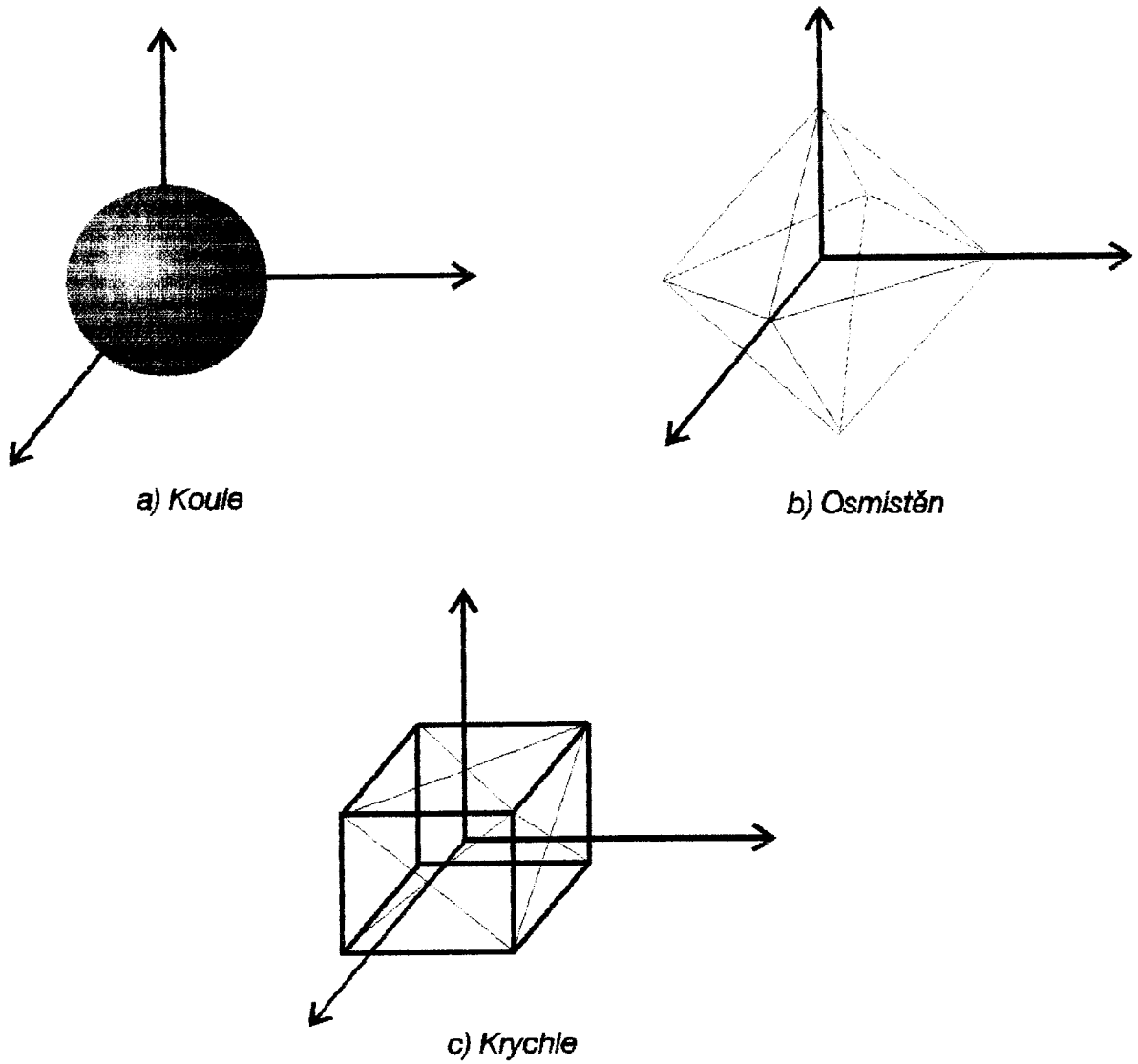
Důležitou mírou, jak v RGB prostoru, tak v dvojrozměrném prostoru obrazové mřížky, je vzdálenost, která může být definována třemi způsoby:

a) Pythagorovou větou

$$Vzd = \sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (G_1 - G_2)^2}$$

b) vzdálenost je dána součtem absolutních hodnot rozdílů jednotlivých složek souřadnic

$$Vzd = |R_1 - R_2| + |G_1 - G_2| + |B_1 - B_2|$$



obr. č. 2 - Jednotkové prostory

c) vzdálenost je dána maximem z absolutních hodnot z jednotlivých rozdílů všech tří složekch složek souřadnic v RGB prostoru

$$Vzd = \max(|R_1 - R_2|, |G_1 - G_2|, |B_1 - B_2|)$$

3.Identifikace barev

Při identifikaci barev byly použity dvě metody: pomocí standardní palety, kterou lze také nastavit ručně a metoda shlukové analýzy.

3.1.Standardní paleta

3.1.1.Implicitně nastavená paleta

Touto metodou se jednotlivým pixelům přiřadí na základě jeho polohy od *etalonu* (*etalon* značí každou z 16-ti barev palety) v RGB prostoru index barvy. Program porovná vzdálenosti bodů od etalonu a bodu přiřadí index barvy nejbližšího *etalonu*. Zde se z důvodu zkrácení doby výpočtu ukázalo výhodné použít pro výpočet vzdáleností vztah z 2.4. c)

	R	G	B
0	0	0	0
1	0	0	39
2	0	39	0
3	0	39	39
4	39	0	0
5	39	39	0
6	39	19	0
7	39	39	39
8	19	19	19
9	19	19	63
10	0	63	0
11	0	63	63
12	63	19	19
13	63	0	63
14	63	63	0
15	63	63	63

tab.1 (tab RGB palety)

3.1.2.Ruční nastavení palety

V tomto případě probíhá výpočet podle stejného algoritmu jako v 3.1.1. s tím rozdílem, že před vlastním výpočtem lze nastavit složky *RGB* etalonů (viz tab.1) podle originální předlohy.

3.2.Metoda shlukové analýzy

Touto metodou lze dosáhnout lepších výsledků než metodou nalezení nejmenší vzdálenosti od etalonu, protože shlukováním příbuzných barevných odstínů optimalizuje paletu. Což je ovšem vykoupeno větším požadavkem na výpočetní čas.

3.2.1.Základní pojmy shlukové analýzy

Rozdělení metod shlukování:

Shlukovací metody - hierarchické - aglomerativní
- divizivní
- nehierarchické - optimalizační
- analýzy modů

Pro většinu používaných shlukovacích metod je spíše charakteristické to, že jejich algoritmy směřují k realizaci nějakých intuitivních představ o geometrických modelech hledaných shluků. Výsledky aplikace shlukovacího algoritmu na danou množinu objektů *O* jsou nazývány *shluky*. Přitom výsledné shluky tvoří systém, který je buď hierarchickým, nebo nehierarchickým shlukováním.

Hierarchickým shlukováním nazveme systém navzájem různých neprázdných podmnožin množiny *O*, v němž průnikem každých dvou podmnožin je buď jedna z nich, nebo prázdná množina a v němž existuje alespoň jedna dvojice podmnožin, jejichž průnikem je jedna z nich.

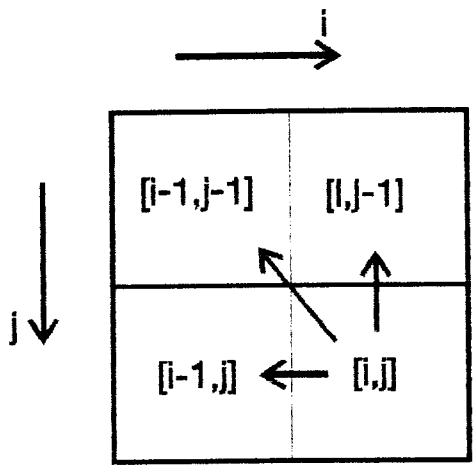
Nehierarchickým shlukováním nazveme systém navzájem různých neprázdných podmnožin množiny *O*, v němž průnikem každých dvou podmnožin není žádná z nich. Z toho je zřejmé, že se podmnožiny tvořící nehierarchické shlukování mohou částečně překrývat.

V tomto konkrétním případě byla použita metoda aglomerativní, která vychází z jednotlivých samostatných objektů a jejich postupným seskupováním docházíme z počátečního stavu, kdy je každý z 65536 bodů rastru 256x256 pixelů chápán jako samostatný shluk, až ke konečnému stavu - spojení všech objektů do jedné množiny v tomto případě do 16-ti množin.

Objekt

Hovoříme-li o objektech, máme obvykle na mysli nějakou množinu předmětů nebo jevů. Každý konkrétní *objekt* musí být popsán *znaky*.

V případě identifikace barev je *objekt* definován jako bod v barevném obrazu a *znak* určuje jeho polohu v RGB prostoru.



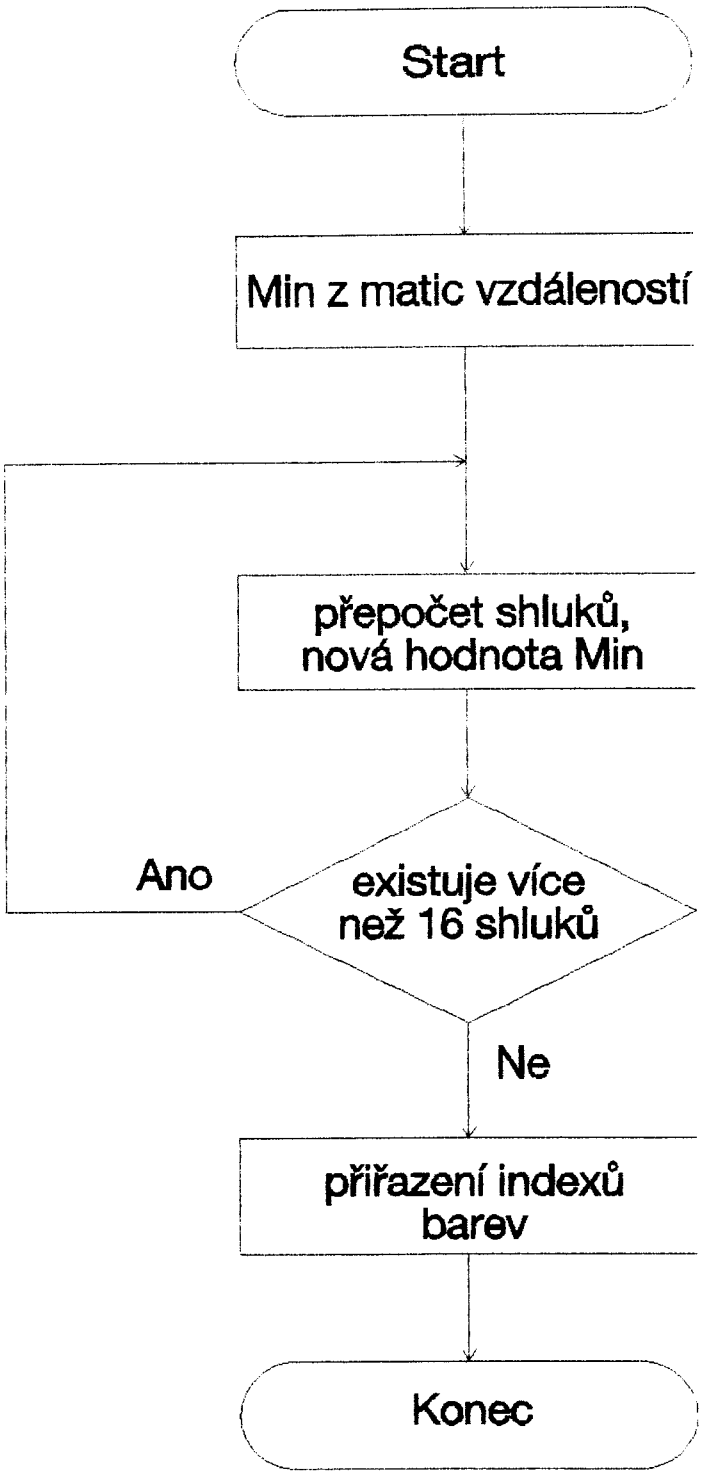
obr. č. 3

Matice dat

Matice dat je zde definována jako matice o rozměrech (3x256x256), kde objekty tvoří matici (256x256) a znaky (hodnoty R,G,B) jsou jako její třetí rozměr. Geometrický model *matice dat* si lze představit jako zobrazení *objektů* v RGB prostoru.

1	1	1	1	1
1	1	1	1	1
1	259	259	259	259
1	1	1	259	259
1	261	1	259	259

obr. č. 4 - Matice shluků



obr. č. 5 - Shlukování

3.2.2.Shlukování

V tomto případě bylo použito metody nejbližšího souseda (v RGB prostoru). Zde je nutné poznamenat, že musíme rozlišit souseda v *RGB prostoru* a souseda bodu (*pixelu*) v obrazové funkci (v obrázku). Tato metoda spočívá na následujícím principu: v prvním kroku cyklu se provede výpočet vzdálenosti mezi každým bodem [i,j] a jeho sousedy (v obrázku) s indexy [i,j-1], [i-1,j], [i-1,j-1] (horním, levým a levým horním - viz obr.3) a uloží do tří matic vzdáleností. Kde každý směr od bodu [i,j] je reprezentován vždy jednou maticí: VzdN - [i,j-1], VzdL - [i-1,j], VydS - [i-1,j-1].

Jako další krok je nalezení minimální hodnoty matic vzdáleností. Podle nalezeného minima se sloučí sousední body, jejichž vzdálenost v RGB prostoru je rovna nebo menší než minimum a jako poslední krok cyklu je nalezení nového minima. Cyklus probíhá dokud je počet shluků větší než požadovaný počet (zde 16).

3.2.3.Přiřazení barev

Jestliže metodou shlukové analýzy získáme konečný počet shluků (záleží na požadovaném barevném rozlišení), pak každý shluk má své číslo z intervalu 0..65535 a je tedy nutné tyto shluky přečíslovat do intervalu 0..15, aby bylo možné každému ze shluků přiřadit index barvy.

Aby bylo zobrazení barev co nejvěrnější je nutné změnit barevnou paletu. Tu změníme pomocí příkazu Turbo Pascalu *SetRGBPalette(I,R,G,B)*, kde *I* je index barvy a *R,G,B* jsou hodnoty palety. Optimální hodnoty *R,G* a *B* se vypočtou jako těžiště každého shluku v RGB prostoru a v konečné fázi je přiřazen každému prvku shluku odpovídající index barvy *I*.

4.Identifikace tvarů

Pro identifikaci tvarů jsou důležité některé pojmy shodné se zpracováním monochromatické obrazové informace a ve výsledcích dokonce lepší, protože při barevném zobrazení v 16-ti barevné paletě se v menší míře objevují odstíny stejné barvy.

Obrazová funkce

Obrazová funkce je funkce dvou argumentů (plošných souřadnic x,y). V tomto případě má obrazová funkce definiční obor diskrétní, přičemž jednotlivým hodnotám jsou přiřazeny indexy barev.

Oblast

Oblast v obraze je taková množina obrazových bodů, jejichž každé dva prvky jsou vzájemně souvislé, tj. přes ostatní prvky množiny existuje mezi nimi cesta.

Hranice

Hranice oblasti je taková její podmnožina, kde každý prvek má alespoň jednoho souseda, který do oblasti nepatří.

Hrana

Hrana v obraze je označení pro místo, kde dochází k velké změně hodnoty obrazové funkce. Vypočte se pomocí gradientu jasové funkce. *Hrana* je určena svou velikostí a směrem. Směr *hrany* je kolmý na směr gradientu. Protože zde každá úroveň jasu představuje index jedné barvy na rozdíl od černobílého obrazu, kde každá úroveň jasu představuje odstín šedi, není nutné určovat

gradient jasové funkce.

4.1.Segmentace

Cílem *segmentace* je rozčlenění obrazu do oblastí.Rozdělení obrazu do částí (oblastí) lze získat prahováním.

Prahování

Prahování je nejjednodušším postupem *segmentace*, která slouží k oddělení objektů od pozadí (zde i od sousedících nebo překrytých objektů). Výsledkem je binární obraz, kde je oddělený obraz připraven k rekonstrukci jeho původního tvaru.

4.2.Rekonstrukce tvarů

Pro rekonstruování jednoduchých geometrických tvarů je potřeba nejprve provést segmentaci, která nám jednoznačně určí plochu zaplněnou danou barvou.

4.2.1.Určování troj- a čtyř- úhelníků

Dále je nutné z důvodu nerovností hranic jednoduchých geometrických obrazců, jako jsou trojúhelníky, obdélníky, lichoběžníky a kosodélníky, určit směrnice jejich stran a vrcholy těchto útvarů.

Určování vrcholů lze provést následujícím způsobem:

Prohledáváním obrazu ve všech směrech tj. od bodu [0,0] do bodu [255,255], od [0,255] do [255,0], od [255,0] do [0,255], od [255,255] do [0,0], nalezneme první bod s indexem barvy rekonstruovaného tvaru. Tvar lze pak zrekonstruovat prostým spojením vrcholů úsečkami a vyplněním vnitřní plochy útvaru.

Další způsob jak rekonstruovat tvar je algoritmus pro tvorbu oblastí na základě neúplných hranic:

1) Pro každou buňku x^{-} hranice hledejme, až do jisté vzdálenosti, protější hraniční element. Pokud je nalezena hraniční buňka, která není protější, ukončeme prohledávání a postupme na vyhodnocení další hraniční buňky. Každou buňku ležící na spojnici protějších hraničních obrazových elementů označme jako potenciální buňku oblasti.

2) Pro každou buňku obrazu určeme, kolikrát ležela na spojnici protějších hraničních buněk, počet výskytů buňky x' na některé spojnici označme $b(x'^{-})$.

3) Vážený počet výskytů B(x') určíme podle vztahu:

$$B(x'^{\rightarrow}) = \begin{cases} 0 \text{ pro } b(x'^{\rightarrow}) = 0 \\ 0{,}1 \text{ pro } b(x'^{\rightarrow}) = 1 \\ 0{,}2 \text{ pro } b(x'^{\rightarrow}) = 2 \\ 0{,}5 \text{ pro } b(x'^{\rightarrow}) = 3 \\ 1{,}0 \text{ pro } b(x'^{\rightarrow}) = 4 \end{cases}$$

Věrohodnost, že buňka $x^{\rightarrow}$ je buňkou oblasti, je určována jako součet $\sum_{i} B_i$ v 3x3 okolí buňky $x^{\rightarrow}$. Je-li věrohodnost, že buňka $x_{x^{\rightarrow}}$ je buňkou oblasti, větší nebo rovna jedné, je buňka $x^{\rightarrow}$ označena jako buňka oblasti. V opačném případě je označena jako buňka pozadí.

4.2.2.Určování kruhových tvarů

Z analytické geometrie je známo, že k jednoznačnému určení kružnice je nutné znát buď střed a poloměr nebo tři body na obvodu. V tomto konkrétním případě bude výhodné najít tři body na obvodu a z nich vypočítat střed a poloměr. Z důvodu co možná nejpřesnější rekonstrukce tvaru kruhu musíme správně určit hranici oblouku. K tomuto účelu lze použít *Houghovu transformaci*.

Původní *Houghova transformace* byla navržena pro detekci křivek. V úlohách segmentace je využitelná pro detekci oblastí, jsou-li známy rovnice jejich hraničních křivek. Přitom nejsou

kladeny žádné požadavky na znalost polohy oblasti. Při hledání kruhových oblastí je rovnice hledané křivky

$$\sqrt{(x_1 - a)} + \sqrt{(x_2 - b)} = r^2,$$

souřadnice bodu (a,b) jsou souřadnice středu kružnice o poloměru r. Uvažujeme hledání kruhové hranice oblasti dané barvy, zde hledáme oblouk kružnice s konstantním poloměrem, proto: $r=R$. Bez uvažování informace o směru hran inkrementujeme v parametrickém prostoru všechny hodnoty $A(a,b)$, jejichž (a,b) leží na kružnici se středem x, y. Vezmeme-li v úvahu i směr hran, stačí inkrementovat čítače jen v malé části této kružnice. Vzhledem k diskretisaci směru hran do osmi hodnot vyhodnocujeme jen 1/8 této kružnice uvažujeme-li chybu směru velikosti $\pi/4$, vyhodnocujeme 3/8 této kružnice. Parametry a,b lze při uvažování směru hran určit podle vztahů:

$$a = x_1 - R*\cos(\Phi(x^{\rightarrow}) \pm \Delta\Phi),$$
$$b = x_2 - R*\sin(\Phi(x^{\rightarrow}) \pm \Delta\Phi).$$

$\Phi(x^{\rightarrow})$ je směr hrany v buňce $x^{\rightarrow}$, $\Delta\Phi$ je maximální předpokládaná chyba určení směru hrany. Při inkrementaci datové struktury uvažujeme jen ty hodnoty (a,b), vyhovující rovnici uvedené výše.

5.Popis programu

Zdrojový text programu PASTELKY.EXE se skládá z několika programových jednotek (units) většina z nich je součástí standardních knihoven Turbo Pascalu jako např.: *Crt, Dos, Graph, Objects, Memory, Drivers, App, Dialogs, StdDlg, Buffers, Menus, MsgBox a Views*. U některých byly přeloženy texty a hlášky do češtiny. Vlastní uživatelské prostředí je naprogramováno pomocí knihovny Turbo Vision, která umožňuje vytváření menu, stavového řádku, dialogových oken, přepínačů a dalších.

PASTELKY.PAS - hlavní program - obsahuje definici uživatelského prostředí, menu a dialogových a vlastní procedury identifikace barev a rekonstrukce tvarů.

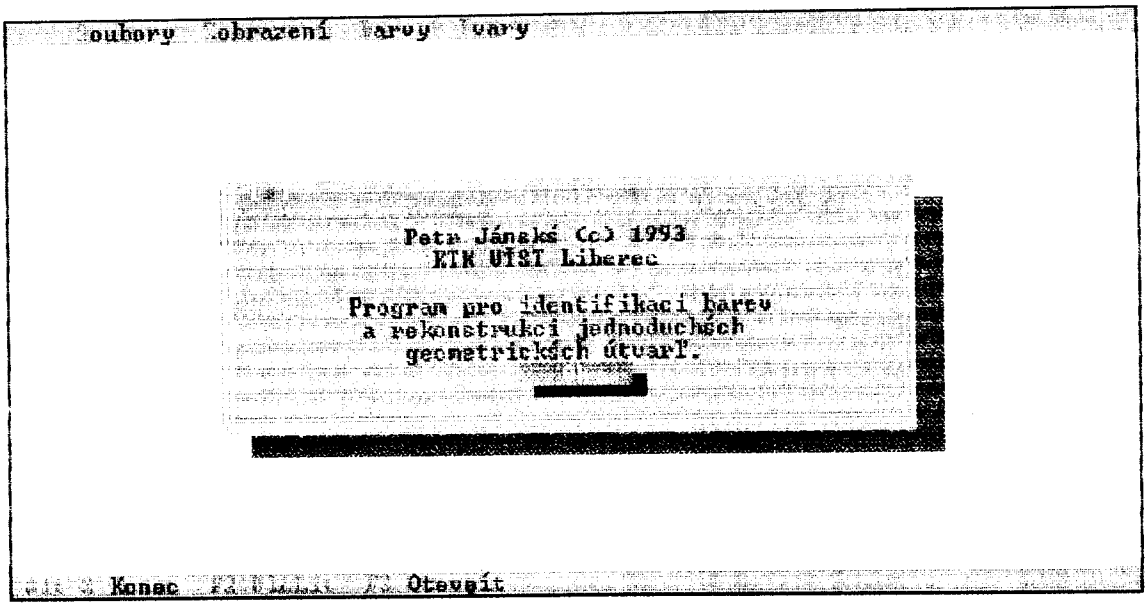
CSFR.PAS - programová jednotka s deklarací hlaviček procedur a funkcí ze souboru CSFR.OBJ, které umožňují zobrazit na monitoru znaky s českou diakritikou.

KRYSA.PAS - programová jednotka s deklarací procedur pro definici tvaru kursoru myši její ovládání v grafickém prostředí.

MATRIX.PAS - programová jednotka pro práci s maticemi, umožňuje práci s polemi většími než 64 KB pomocí dynamických datových struktur, tato pole může ukládat do konvenční paměti, EMS paměti, nebo na disk.

5.1. Ovládání programu

Po spuštění programu (PASTELKY.EXE <Enter>) se objeví dialogové okno se sdělením jaký program jsme vlastně spustili.



obr.č.6 (úvodní okno)

Program lze ovládat jak kursorem myši, tak i pomocí klávesnice (pomocí tzv. horkých kláves).

menu Soubory

- Otevřít - otevře všechny tři soubory jednotlivými složkami R, G a B (nebo horká klávesa F3)
- Uložit - uloží na disk barevný obraz (tento příkaz je vykonatelný až po identifikaci barev) - klávesa

F2

- Změnit adresář - změní adresář disku z něho bude program číst soubory a na nějž je bude ukládat.

- Konec - ukončí program - Alt X

Zobrazení - zobrazí obrázek ze souboru - lze zvolit barevné nebo černobílé zobrazení

menu Barvy

- Shluková analýza - identifikace barev metodou shlukové analýzy

- podmenu VGA paleta - Nastavení palety - ruční nastavení hodnot R, G a B

- Výpočet - vlastní identifikace metodou standardní palety

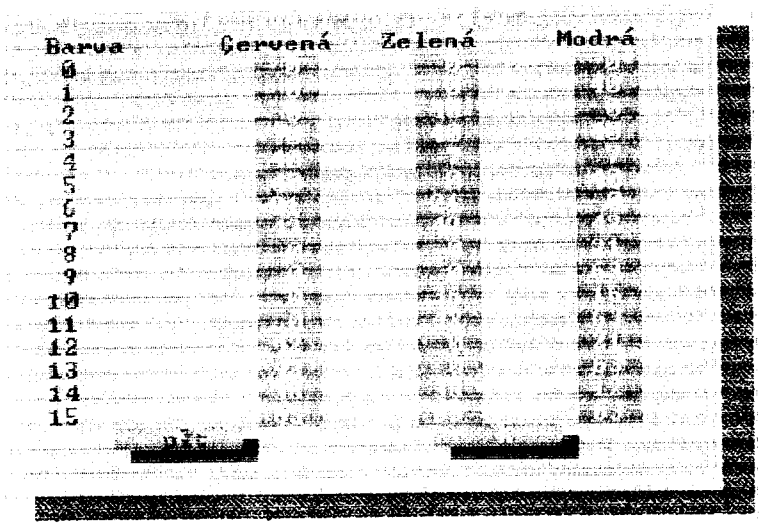
Tvary - rekonstrukce tvarů.

5.2. Běh hlavních procedur

5.2.1. Nastavení palety

- po zavolání tohoto příkazu z podmenu VGA paleta se zobrazí dialogové okno (viz obr.č.7), kde je možné změnit všechny tři hodnoty *R*, *G* a *B* každé z 16-ti barev VGA palety. Zde je nutné upozornit, že v případě, kdy se nepodaří správně nastavit paletu,

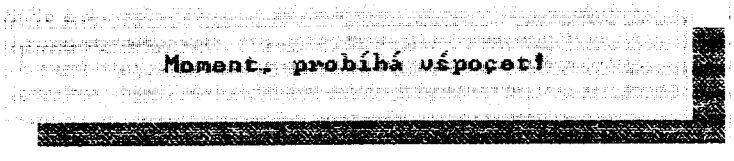
program zidentifikuje barvy špatně a nebo v horším případě je nezidentifikuje. Např.: ve chvíli, kdy se podaří vyřadit z barevné palety barvu, která se vyskytuje v originále, nebo změníme její odstín natolik, že se body, které k ní normálně patří, přiřadí k barvě zcela jiné.



obr.č.7 (dialog pro nastavení RGB palety)

5.2.2. Shluková analýza a Výpočet z VGA palety

- po spuštění těchto procedur se zobrazí okno s hláškou: ''Moment, probíhá výpočet!'' po jeho vymazání z obrazovky je výpočet skončen, a je možné naidentifikovaný obraz uložit na disk.



obr.č.8

5.2.3. Tvary

- po zavolání se procedura zeptá na jméno souboru pro rekonstrukci tvarů, pak je možné si vybrat rekonstruovaný obrazec (trojúhelník, čtyřúhelník nebo kruh). Poté se procedura přepne do grafického režimu se barevně zobrazí obrázek v dolní části obrazovky se objeví barevná paleta, z níž je nutné si vybrat barvu rekonstruovaného obrazce a poté lze spustit tlačítkem "*Rekonstrukce*", které se nachází nad paletou vpravo nahoře, vlastní rekonstrukci. Poté jej lze stejně jako obraz po identifikaci barev opět uložit do souboru na disk a poté opět zobrazit na display počítače.

6. Závěr

V úvodu předkládané práce jsou nastíněny způsoby snímání a přenosu obrazové informace na počítač.

V dalších kapitolách se zabývá základními informacemi zoboru kolorimetrie, které jsou potřebné pro snímání a reprodukci barevné obrazové informace a dále pak o metodách identifikace barev, především pak o metodách shlukové analýzy a o metodách rozlišování geometrických tvarů a v případě, že se překrývají, i o jejich rekonstrukci.

Hlavním cílem této diplomové práce bylo umožnit zobrazení nasnímaného barevného obrazu v reálných barvách na monitor počítače. Tento cíl se podařilo splnit jen částečně, pouze zobrazení na VGA monitoru, z toho důvodu, že nebyla včas k dispozici grafická knihovna pro pracovní stanice DEC 5000/25.

Přínos tohoto tématu je především v tom, že doposud byla na KTK zpracovávána pouze černobílá obrazová informace, a tato práce se poprvé zabývá zpracováním barevného obrazu. Konkrétní praktické využití řešení tohoto problému je hlavně ve využití algoritmů pro řešení dalších problémů spojených se zpracováním barevného obrazu, jako je například v současnosti velmi perspektivní bezkontaktní měření pomocí televizní kamery.

Použitá literatura:

[1] Hlaváč,V.-Šonka,V : Počítačové vidění.
SNTL. Praha 1992.

[2] Ing. Vít,V. a kolektiv : Televizní technika.
SNTL,Alfa. Praha 1979.

[3] RNDr. Lukasová,A.-RNDr. Šarmanová,J. : Metody shlukové analýzy.
SNTL. Praha 1985.

{ Příloha číslo 1 }

{ Výpočet pole vzdáleností }

```
rocedure Vzdalenosti;
ar
  Ad,A,Moc      : byte;
  I,J           : word;
  Bod1,Bod2     : TPix;

function Vzdal( P1,P2 : TPix) : byte;
begin
  Vzdal:=abs(P1.R-P2.R) + abs(P1.G-P2.G) + abs(P1.B-P2.B);
end;

egin
Moc:=255;
for I:=1 to 256 do
  for J:=1 to 256 do
  begin
    with Bod1 do
    begin
      PoleR^.Get(I,J,R);
      PoleG^.Get(I,J,G);
      PoleB^.Get(I,J,B);
    end;
    if J>1 then
    begin
      with Bod2 do
      begin
        PoleR^.Get(I,J-1,R);
        PoleG^.Get(I,J-1,G);
        PoleB^.Get(I,J-1,B);
      end;
      A:=Vzdal(Bod1,Bod2);
      VzdN^.Put(I,J,A);
    end
    else VzdN^.Put(I,J,Moc);
    if (J>1) and (I>1) then
    begin
      with Bod2 do
      begin
        PoleR^.Get(I-1,J-1,R);
        PoleG^.Get(I-1,J-1,G);
        PoleB^.Get(I-1,J-1,B);
      end;
      A:=Vzdal(Bod1,Bod2);
      VzdS^.Put(I,J,A);
    end
    else VzdS^.Put(I,J,Moc);
    if I>1 then
    begin
      with Bod2 do
      begin
        PoleR^.Get(I-1,J,R);
        PoleG^.Get(I-1,J,G);
```

```
      A:=Vzdal(Bod1,Bod2);
      VzdL^.Put(I,J,A);
    end
    else VzdL^.Put(I,J,Moc);
  end;
nd;
{------------------------------------------------------------------------}

                { Funkce pro zjištění počtu shluků }

unction Pocitej(var M : TPoleSh):word;
ar
    P     : array [0..255,0..31] of byte;
    S     : word;
    I,J,K : word;
    L,R,N : byte;
    O,Q   : byte;

begin
  for I:=0 to 255 do
    for J:=0 to 31 do
      P[I,J]:=0;
  for I:=1 to 128 do
  begin
    for J:=1 to 128 do
    begin
      S:=M.P1^[I,J];
      L:=S div 256;
      R:=S mod 256;
      N:=R div 8;
      O:=R mod 8;
      Q:=Ex2(O);
      P[L,N]:=P[L,N] or Q;
      S:=M.P2^[I,J+128];
      L:=S div 256;
      R:=S mod 256;
      N:=R div 8;
      O:=R mod 8;
      Q:=Ex2(O);
      P[L,N]:=P[L,N] or Q;
      S:=M.P3^[I+128,J];
      L:=S div 256;
      R:=S mod 256;
      N:=R div 8;
      O:=R mod 8;
      Q:=Ex2(O);
      P[L,N]:=P[L,N] or Q;
      S:=M.P4^[I+128,J+128];
      L:=S div 256;
      R:=S mod 256;
      N:=R div 8;
      O:=R mod 8;
      Q:=Ex2(O);
      P[L,N]:=P[L,N] or Q
```

```
  end;
  S:=0;
  for I:=0 to 255 do
    for J:=0 to 31 do
    begin
       K:=1;
       while K<255 do
       begin
         if (P[I,J] and K)<>0 then Inc(S);
         K:=K*2
       end
    end;
    Pocitej:=S
end;
```

{ Shluková analýza (MaxPocShl-kon.poç.shlukû) }

```
procedure Shluk(MaxPocShl : word);

var
    S,L,N           : byte;
    M,Min,StMin     : byte;
    Sh,PocSh,Shl    : word;
```

{ Zjištění minima }

```
 procedure Minim(B : boolean);
 var
       I,J      : word;
       A        : byte;

 begin
   Min:=255;
   for I:=1 to 256 do
     for J:=1 to 256 do
     begin
       VzdN^.Get(I,J,A);
       if (A<=Min) and (A>=StMin) then Min:=A;
       VzdS^.Get(I,J,A);
       if (A<=Min) and (A>=StMin) then Min:=A;
       VzdL^.Get(I,J,A);
       if (A<=Min) and (A>=StMin) then Min:=A;
       if B and ((Min-StMin)=1) then exit;
     end;
 end;

egin
 StMin:=0;
 Minim(false);
 StMin:=1;
 for I:=0 to 127 do
   for J:=0 to 127 do with PoleShl do
```

```
    P1^[I,J]:=I*256+J;
    P2^[I,J+128]:=I*256+J+128;
    P3^[I+128,J]:=(I+128)*256+J;
    P4^[I+128,J+128]:=(I+128)*256+J+128;
  end;
PocSh:=Pocitej(PoleShl);
PocSh:=65535;
while PocSh>MaxPocShl do
begin
  for I:=0 to 255 do
    for J:=0 to 255 do
    begin
      VzdS^.Get(I+1,J+1,S);
      VzdL^.Get(I+1,J+1,L);
      VzdN^.Get(I+1,J+1,N);
      with PoleShl do
      begin
        if (S<=Min) and (I>0) and (J>0) then
        begin
          if I<128 then if J<128 then
                                    P1^[I,J]:=P1^[I-1,J-1]
                                  else if J=128 then
                                                P2^[I,J]:=P1^[I-1,J-1]
                                              else
                                                P2^[I,J]:=P2^[I-1,J-1]
                       else if I=128 then if J<128 then
                                                P3^[I,J]:=P1^[I-1,J-1]
                                              else if J=128 then
                                                           P4^[I,J]:=P1^[I
                                                         else
                                                           P4^[I,J]:=P2^[I
                                  else if J<128 then
                                                P3^[I,J]:=P3^[I-1,J-1]
                                              else if J=128 then
                                                           P4^[I,J]:=P3^[I
                                                         else
                                                           P4^[I,J]:=P4^[I
        end;

        if (L<=Min) and (I>0) then
        begin
          if I<128 then if J<128 then
                                    P1^[I,J]:=P1^[I-1,J]
                                  else
                                    P2^[I,J]:=P1^[I-1,J]
                       else if I=128 then if J<128 then
                                                P3^[I,J]:=P1^[I-1,J]
                                              else
                                                P4^[I,J]:=P2^[I-1,J]
                                  else if J<128 then
                                                P3^[I,J]:=P3^[I-1,J]
                                              else
                                                P4^[I,J]:=P4^[I-1,J]
        end;
        if (N<=Min) and (J>0) then
```

```
        begin
          if I<128 then if J<128 then
                                    P1^[I,J]:=P1^[I,J-1]
                                  else if J=128 then
                                                P2^[I,J]:=P1^[I,J-1]
                                              else
                                                P2^[I,J]:=P2^[I,J-1]
                     else if J<128 then
                                    P3^[I,J]:=P3^[I,J-1]
                                  else if J=128 then
                                                P4^[I,J]:=P3^[I,J-1]
                                              else
                                                P4^[I,J]:=P4^[I,J-1]
        end;
      end;
    end;
  PocSh:=Pocitej(PoleShl);
  StMin:=Min;
  Minim(true);
end;
writeln;
nd;

rocedure InitVzd;                    { Inicialisace polí vzdálenosti }
ar I,J,M : word;

egin
VzdN:=New(PMatrix,Init(256,256,mt_Byte,mv_Ems));
if VzdN=nil then
begin
  writeln('Nedostatek mista pro matici VzdN');
  halt(1);
end;
VzdS:=New(PMatrix,Init(256,256,mt_Byte,mv_Ems));
if VzdS=nil then
begin
  writeln('Nedostatek mista pro matici VzdS');
  halt(1);
end;
VzdL:=New(PMatrix,Init(256,256,mt_Byte,mv_Ems));
if VzdL=nil then
begin
  writeln('Nedostatek mista pro matici VzdL');
  halt(1);
end;

M:=255;
VzdL^.Put(1,1,M);
VzdN^.Put(1,1,M);
VzdS^.Put(1,1,M);
for I:=1 to 256 do
begin
  VzdL^.Put(I,1,M);
  VzdN^.Put(I,1,M);
  VzdS^.Put(I,1,M);
  VzdL^.Put(1,I,M);
```

```
    VzdN^.Put(I,I,M);
    VzdS^.Put(I,I,M);
  end;
nd;

rocedure DoneVzd;                    { Uvolnění polí vzdálenosti z paměti }

egin
 dispose(VzdS,Done);
 dispose(VzdL,Done);
 dispose(VzdN,Done);
nd;

rocedure InitRGB;                    { Inicialisace polí RGB }

egin
 PoleR:=New(PMatrix,Init(256,256,mt_Byte,mv_Disc));
 if PoleR=nil then
 begin
   writeln('Nedostatek mista pro matici PoleR');
   halt(1);
 end;
 PoleG:=New(PMatrix,Init(256,256,mt_Byte,mv_Disc));
 if PoleG=nil then
 begin
   writeln('Nedostatek mista pro matici PoleG');
   halt(1);
 end;
 PoleB:=New(PMatrix,Init(256,256,mt_Byte,mv_Disc));
 if PoleB=nil then
 begin
   writeln('Nedostatek mista pro matici PoleB');
   halt(1);
 end;
nd;

rocedure DoneRGB;                    { Uvolnění polí RGB z paměti }

egin
 dispose(PoleR,Done);
 dispose(PoleG,Done);
 dispose(PoleB,Done);
nd;

rocedure InitShl;                    { Inicialisace polí RGB }
egin
 New(PoleShl.P1);
 New(PoleShl.P2);
 New(PoleShl.P3);
 New(PoleShl.P4);
nd;


rocedure DoneShl;                    { Uvolnění pole shluků z paměti }
egin
 Dispose(PoleShl.P1);
```

```
    Dispose(Pulse3Ptr.P2);
    Dispose(Pulse3Ptr.P3);
    Dispose(Pulse3Ptr.P4);
end;
```