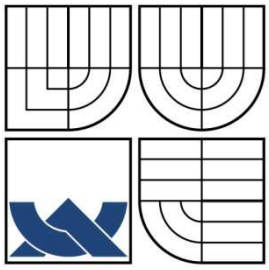


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

GRAFICKÝ EDITOR PRO OS ANDROID

GRAPHIC EDITOR FOR ANDROID OS

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — Bc. MATĚJ HRAZDÍRA
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN ROUPEC, Ph.D.
SUPERVISOR

BRNO 2013

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2012/13

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Matěj Hrazdíra

který/která studuje v **magisterském studijním programu**

obor: **Aplikovaná informatika a řízení (3902T001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Grafický editor pro OS Android

v anglickém jazyce:

Graphic editor for Android OS

Stručná charakteristika problematiky úkolu:

Cílem diplomové práce je vytvořit aplikaci pro práci s rastrovou grafikou pro mobilní zařízení s operačním systémem Android, která by využila výkonu moderních zařízení, zejména pak tabletů a umožnila použití vybraných pokročilých funkcí, které jsou standardní při práci na PC, jako je například práce s vrstvami, použití bodových transformací a filtrů, či geometrických transformací. Dále by vhodným způsobem využila výhod dotykového ovládání, které je pro práci s grafikou výhodné a pro tato zařízení přirozené, zatímco na PC musí být simulováno přídavnými zařízeními.

Cíle diplomové práce:

- Seznámit se s možnostmi vývoje mobilních aplikací pro zařízení s operačním systémem Android.
- Seznámit se s algoritmy používanými při úpravách a zpracování obrazu.
- Na základě získaných poznatků implementovat software pro práci s rastrovou grafikou pracující na zařízeních s operačním systémem Android.

Seznam odborné literatury:

1. MEIER, R.: Professional Android 4 application development.
Indianapolis: John Wiley & Sons, 2012. ISBN 9781118102275.
2. ECKEL, B.: Thinking in Java. 4th ed. Upper Saddle River, NJ: Prentice Hall, 2006. ISBN 0131872486.
3. BURGER, W.: Principles of digital image processing : fundamental techniques. London: Springer, 2009. ISBN 9781848001909.

Vedoucí diplomové práce:Ing. Jan Roupec, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2012/13.

V Brně, dne 11.12.2012

L.S.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA STROJNÍHO INŽENÝRSTVÍ DĚKANÁT

Ing. Jan Roupec, Ph.D.
Ředitel ústavu

prof. RNDr. Miroslav Doupovec, CSc., dr. h. c.
Děkan

Abstrakt

Diplomová práce se zabývá návrhem a implementací aplikace pro editaci rastrové grafiky pracující pod operačním systémem Android. Nejprve je uveden přehled dostupného software, a to jak pro platformu Android, tak PC. Na základě tohoto přehledu jsou formulovány požadavky na vytvořený software.

V následující kapitole jsou shrnuty potřebné teoretické znalosti, na základě kterých jsou jednotlivé funkce programu implementovány. Poté je představen operační systém Android se zaměřením na jeho nasazení a další specifické vlastnosti, které mají přímý vliv na vývoj aplikací pod ním pracujících.

Dále je popsána navržená aplikace z pohledu koncového uživatele. Zde je popsáno uživatelské rozhraní a jednotlivé funkce programu. V poslední kapitole je popsána samotná implementace vytvořeného software – rozdělení aplikace na jednotlivé moduly a odpovědnost jednotlivých tříd aplikace. Přílohy obsahují ukázky použití aplikace.

Klíčová slova

Android; grafický editor; zpracování obrazu

Summary

The proposed master thesis deals with design and implementation of application for editing of raster graphics for Android operating system. First of all, an overview of available software is shown. Based upon this overview, requirements for implemented software are formulated.

Required theoretical knowledge providing foundation for implemented functions is summarized in the next chapter. Further Android operating system is introduced with respect to its deployment and other specific characteristics, which have direct impact on application development.

The following part describes implemented application from end user perspective. User interface is described here together with application features. The last chapter is focused on implementation of created software – separation to single modules and class responsibilities. Appendices contain examples of application usage.

Keywords

Android; Graphic editor; Image processing

HRAZDÍRA, Matěj. *Grafický editor pro OS Android.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2013. 98 s. Vedoucí diplomové práce Ing. Jan Roupec, Ph.D.

Já, Matěj Hrazdíra, prohlašuji, že jsem diplomovou práci vypracoval samostatně a že jsem uvedl všechny použité prameny a literaturu.

V Brně dne 23. 5. 2013 .....

Děkuji vedoucímu mé práce za trpělivost, ochotu a především vstřícnost, se kterou mi byl nápomocen při psaní této práce. Dále bych rád poděkoval své rodině a přítelkyni za veškerou podporu, kterou mi poskytovali.

Obsah

Úvod

Elektronické zpracování obrazu má za sebou již poměrně dlouhou historii. Už v roce 1851 britský vynálezce Frederick Bakewell předvedl zařízení, které dokázalo přenášet kresby pomocí telegrafu. Vývoj pokračoval dál, až roku 1927 Philio T. Farnsworth představil první elektronickou televizi, ta se však dočkala komerčního využití až po druhé světové válce.

První digitální obraz, ve smyslu jak jej chápeme dnes, byl vytvořen roku 1957 Russelem Kirschem pomocí bubnového scanneru. V roce 1973 byly poprvé použity CCD senzory pro pořízení digitálního obrazu na astronomických teleskopech a byl tak položen základ dnešním digitálním fotoaparátům [1].

Rozvoj digitálního zpracování obrazu šel ruku v ruce s rozvojem výpočetní techniky. Vzhledem k výpočetní náročnosti byla tato disciplína zprvu vyhrazena pouze úzké skupině odborníků, kteří měli k dispozici nákladné vybavení, vždyť osobní počítače ještě na počátku 90. let neměly dostatečný výkon ani pro načtení typické fotografie z dnešního fotoaparátu do své paměti. S postupným nárůstem výkonu počítačů a poklesem jejich ceny se začala situace měnit. V okamžiku, kdy začala být dostatečně výkonná výpočetní technika široce dostupná, začal vznikat uživatelsky přívětivý software, který umožňoval upravovat digitální obraz i uživatelům s pouze základními znalostmi z této oblasti. Souběžně také probíhal vývoj digitálních fotoaparátů, které postupně začaly z trhu vytlačovat fotoaparáty na film. Původně digitální fotoaparáty konkurovaly klasickým především díky snadnější správě fotografií, ale postupem času našli digitální fotoaparáty uplatnění i v profesionální a umělecké fotografii, což dále podpořilo vývoj softwaru pro zpracování obrazu určeného pro širokou veřejnost.

Vzhledem k historickému vývoji je zřejmé, že většina softwaru pro zpracování obrazu je dostupná pro osobní počítače. V posledních několika letech však trh s výpočetní technikou zaznamenal výrazný nástup chytrých telefonů a počítačových tabletů. Jedná se o multifunkční zařízení, která poskytují dostatečný výkon, srovna-

telný s výkonem starších osobních počítačů, který je dostatečný pro mnoho běžných činností, jako je například prohlížení internetových stránek, psaní e-mailů, poznámek atd. Tato zařízení se však v mnoha ohledech liší od osobních počítačů. Jejich největší slabinou je nižší výpočetní výkon, který je však kompenzován především mobilitou – díky malým rozměrům, hmotnosti a dlouhé výdrži při práci na baterie je možné je nosit prakticky všude s sebou. Tato zařízení však představují mnohem více. Obvykle nabízejí prostředky pro bezdrátové připojení k počítačové síti, ale také řadu senzorů, jako je GPS čip, kompas, akcelerometr a v neposlední řadě digitální fotoaparát, který u lepších modelů může směle konkurovat samostatným kompaktním fotoaparátům.

Stejně jako narůstá výkon osobních počítačů, dramaticky roste i výkon těchto mobilních zařízení, která tak začínají být postupně vhodná i pro úkony náročnější na výkon.

Dříve bylo při práci s digitální fotografií nutné fotoaparátem pořídit fotografie, ty následně převést do počítače k dalšímu zpracování, kdy pro pohodlnou práci s grafikou bylo nutné použít speciální polohovací zařízení – tablet, který simuloval klasickou tužku a umožňoval přesnější a pohodlnější kreslení i výběr. Profesionálové obvykle ještě využívali kalibrovaných monitorů s rozšířeným gamutem.

Ačkoliv pro profesionální použití bude pravděpodobně tento přístup stále nenahraditelný, současná výkonná mobilní zařízení nabízí pro běžné uživatele alternativu. Umožňují jak pořídit fotografie v dobré kvalitě, tak poskytují dostatečný výkon pro pokročilé operace s obrazem. Jako bonus nabízejí dotykové ovládání, které je pro práci s obrazem výhodné – odpadá tedy nutnost používat externí tablet – jako polohovací zařízení lze použít jak vlastní prst, tak stylus, který lze pořídit za zlomek ceny tabletu. Vzhledem k dobré konektivitě má uživatel dále možnost fotografie okamžitě sdílet na internetu či poslat k vyvolání. Může se tak kompletně vyhnout veškeré práci na počítači.

Cílem této práce je navrhnout a následně implementovat software pro mobilní zařízení s operačním systémem Android, který umožní zkušeným uživatelům používat funkce známé ze softwaru pro osobní počítače, dále využít výhod dotykového ovládání, které je zde běžné, a umožnit jim tak pohodlně upravovat fotografie pořízené nejen těmito zařízeními.

1 Přehled dostupného software

Pro práci s počítačovou grafikou na PC existuje celá řada nástrojů, které se liší jednak svými možnostmi a jednak licencí, pod kterou jsou poskytovány. Kvalitní nástroje jsou dostupné jak komerčně, tak zdarma, včetně open-source řešení. Platforma Android je v tomto ohledu na jiné úrovni jednak díky svému nízkému věku a jednak díky faktu, že teprve nedávno vstoupila na trh dostupná, vysoce výkonná zařízení s tímto operačním systémem. Tato kapitola si klade za cíl podat přehled dostupného software pro tyto platformy a následně formulovat požadavky na software implementovaný při řešení této diplomové práce.

1.1 Software pro PC

Tato sekce velmi stručně představuje dva nejznámější představitele softwaru pro zpracování obrazu na PC, více pozornosti je však věnováno programům pro OS Android, neboť tato platforma je předmětem diplomové práce.

1.1.1 Adobe Photoshop

Adobe Photoshop lze označit za naprostou špičku mezi dostupným softwarem. Je vyvíjen na komerční bázi společností Adobe. Jedná se o vyspělý software, jehož vývoj začal již koncem 80. let 20. století a jehož první verze byla vydána začátkem roku 1990 [2]. Tato verze byla napsána především v jazyce Pascal a částečně v assembleru, její zdrojový kód je dnes již pro nekomerční účely volně ke stažení [3].

V poslední verzi umožňuje Adobe Photoshop provádět velké množství komplexních operací od jednoduchých úprav jasu a kontrastu, přes pokročilou práci s vrstvami a filtry až po 3D renderování, Samozřejmostí je také správa barev, podpora RAW formátů různých výrobců fotoaparátů a široké možnosti exportu. Podrobný výčet funkcí by byl značně nad rámec této práce, pouze uživatelská příručka poslední

verze, dostupná z [4] dosahuje rozsahu 751 stran. S nadsázkou lze říci, že Photoshop obsahuje téměř všechny v praxi užitečné algoritmy pro úpravy obrazu.

1.1.2 GIMP

GIMP je zkratkou z anglického GNU Image Manipulation Software. Jedná se o svobodný software, který poskytuje jak základní tak pokročilé funkce pro úpravu rastrové grafiky, včetně bodových úprav jasu, kontrastu, barev, práce s vrstvami, použití filtrů, geometrických transformací a dalších funkcí. Vzhledem k otevřenosti softwaru je navíc celá řada dalších funkcí dostupná ve formě doplňků aplikace od různých autorů. Za hlavní nevýhodu GIMPu oproti Photoshopu lze považovat jistou těžkopádnost a nejednotnost uživatelského rozhraní, která je patrná při používání stejných funkcí v rámci obou produktů.

1.2 Software pro Android

Tato sekce obsahuje podrobné srovnání dvou nejvýznamnějších komerčních programů pro práci s rastrovou grafikou pro platformu Android a udává také přehled volně dostupných alternativ. Vzhledem k tomu, že se jedná o segment trhu, který podléhá velmi rychlému vývoji, je dobré poznamenat, že tato sekce je aktuální k datu 25. 4. 2013.

1.2.1 Adobe Photoshop Touch

Adobe Photoshop Touch představuje odlehčenou mobilní verzi Photoshopu, dříve představeného v sekci pojednávající o softwaru pro PC, která ovšem obsahuje většinu jeho základních funkcí. Program je komerčně dostupný z oficiálního zdroje aplikací pro platformu Android – Google play [5], následuje přehled hlavních funkcí aplikace roztříděný dle kategorií.

- **Vrstvy**
  - Podpora rastrových vrstev s nastavitelnou průhledností.
  - Podporované módy: normální, ztmavit, násobit, zesvětlit, závoj, lineárně zesvětlit, překrýt, rozdíl, odečíst.
- **Kreslení**
  - Nástroje: kulatý štětec, sprej, klonovací razítko a guma.
  - Volby nástrojů: plynule nastavitelná velikost, tvrdost, tok a průhlednost.
- **Výběr**
  - Obdélníkový a eliptický výběr, laso a polygonální výběr, magická hůlka a výběr pomocí štětce.

- **Retušovací nástroje**
  - **Bodové operace**
    - Úprava jasu a kontrastu, teploty barev, vyvážení barev, nasycení, úrovní, křivek a další. Použitelné jak na jednu vrstvu, výběr, či jako štětec na konkrétní místo.
  - **Filtry**
    - Množství filtrů od rozmazání a zostření až po imitaci perokresby. Aplikovatelné na celou vrstvu či výběr. Jako lokální nástroj dostupné pouze rozmazání a rozostření.
  - **„Healing brush“**
    - Nástroj na poloautomatickou opravu částí obrazu.
- **Renderovací nástroje**
  - Výplň výběru, vykreslení šumu a oblak, odlesku objektivu.
- **Geometrické transformace**
  - Oříznutí a změna velikosti obrázku, otočení obrazu a volná transformace.

Aplikace umožňuje export obrazu do galerie ve formátu jpg a png, export do cloudového úložiště pak i ve formátu psd, který zachová vrstvy obrazu.

Za hlavní nedostatek do ještě nedávné doby patřila relativně malá velikost obrazu, která nedosahovala ani zdaleka rozlišení, jaké poskytují integrované fotoaparáty běžných zařízení. Původně bylo maximální rozlišení 1600×1600 pixelů, které bylo následně zvýšeno na 2048×2048 pixelů [6], což je stále méně, než je běžné rozlišení integrovaných fotoaparátů. V současné době je však již k dispozici podpora až 12 MPx bitmap.

1.2.2 Autodesk SketchBook Pro

Jedná se o komerčně dostupnou aplikaci, kterou je možno také získat z oficiálního obchodu Google play [7]. Tato aplikace se specializuje, jak již název napovídá, na kreslení a skicování, chybí tedy podpora pro bodové operace a filtry, zato je zde velmi propracovaná paleta štětců a podobných nástrojů. Následuje přehled vybraných funkcí.

- **Vrstvy**
  - Podpora rastrových vrstev s nastavitelnou průhledností.
  - Podporované módy: normální, násobit, lineárně zesvětlit, závoj.

- **Kreslení**
  - Nástroj štětec s velkým množstvím nastavení – plynule volitelná velikost a průhlednost, mezera stopy, tři přednastavené tvrdosti štětce, velké množství přednastavených štětců, které se liší jak svými parametry zmíněnými výše, tak tvarem hrotu.
  - Nástroj guma.
- **Výběr**
  - Není k dispozici, pro usnadnění kreslení jsou však k dispozici pravítka a nástroje pro kreslení rovných čar, obdélníků, elips.
- **Retušovací nástroje**
  - Nejsou k dispozici.
- **Renderovací nástroje**
  - Nejsou k dispozici.
- **Geometrické transformace**
  - Posun, otočení a změna velikosti vrstvy.

Autodesk SketchBook Pro disponuje pohodlnějším importem a exportem obrázků, než Adobe Photoshop Touch. Podporuje jak export, tak import souborů formátu jpg, png a psd a to i do cloudu i do lokálního úložiště.

Aplikace dříve vůbec nepodporovala nastavení velikosti obrazu, současná verze však podporuje obrázky až do rozlišení 6 MPx.

1.2.3 Volně dostupný software

K dispozici je také řada volně dostupného softwaru. Žebříček nejlépe hodnocených aplikací lze nalézt v obchodu Google play [8]. Volně dostupné aplikace jsou obvykle sponzorovány nějakou formou reklamy v aplikaci, která v závislosti na rozlišení displeje zařízení může být dosti rušivá a komplikovat práci.

Volně dostupný software lze rozdělit do dvou skupin. První skupina obsahuje aplikace, které pracují podobně jako Photoshop – úpravy jsou prováděny kombinací elementárních úkonů, nad kterými má uživatel vysoký stupeň kontroly, tyto aplikace jsou vhodné pro zkušené uživatele, kteří přesně vědí, čeho a jak chtějí dosáhnout.

Mezi volně dostupnými aplikacemi však převažují aplikace druhého typu, které mají přednastavené efekty pro obraz, jako je například starý vzhled fotografie, přidání rámečku atd. Uživatel má běžně na výběr z mnoha takovýchto operací, takže pro občasné uživatele, kteří chtějí vylepšit svoje momentky před publikováním na sociálních sítích, poskytují dostatek možností.

PicsArt – Photo Studio

Tento software patří k nejlepším volně dostupným programům, lze jej získat z [9]. Pohybuje se na rozhraní obou dříve zmíněných kategorií. Obsahuje jednak mnoho přednastavených efektů a nástrojů pro okamžitou stylizaci obrazu, ale také umožňuje standardní kreslení a práci s vrstvami, kdy pro štětce poskytuje standardní nastavení jako je krytí, velikost štětce, tvar hrotu štětce a další.

Hlavní nevýhodou je omezená velikost obrazu maximálně 2048×2048 pixelů a export pouze do formátů jpg a png, které nepodporují vrstvy. Program také neumí uložit rozpracovaný obraz s vrstvami – zvládá pouze export. Program zobrazuje reklamu, ovšem v aplikaci se dá zakoupit její vypnutí spolu s dalšími doplňky.

Pixlr Express a Pixlr-o-matic

Tyto dva zdarma dostupné programy, které lze získat z [10], [11] jsou dílem společnosti Autodesk a jsou typickými představiteli druhé kategorie. Poskytují velké množství přednastavených nástrojů pro rychlou stylizaci fotografie.

1.3 Shrnutí dostupného softwaru

V současné době je na trhu dostupná celá řada softwaru z této oblasti, na platformu Android však zatím příliš nepronikly aplikace, které by pracovali na principu programů pro PC, tedy poskytovali pokročilé funkce uživatelům, kteří jsou zvyklí s nimi pracovat a kombinovali je s výhodami dotykového ovládání. Jedinou aplikací, která nabízí tento přístup, je výše zmíněná aplikace Adobe Photoshop Touch, na které je ale znát, že je pouze mobilním doplňkem verze pro PC. Zejména co se týče práce s vrstvami, nenabízí tato aplikace ani zdaleka tolik možností co verze desktopová.

V oblasti automatických úprav obrazu existuje mnoho pokročilých aplikací, které jsou většinou dostupné zdarma, byť sponzorované reklamou, což je pravděpodobně dáno tím, že většinu na trhu drží chytré telefony, na kterých vzhledem k velikosti obrazovky není možné provádět komplexní úpravy, zatímco výběr přednastavené stylizace obrázku je rychlá a efektivní cesta pro běžného uživatele jak vylepšit své fotografie pořízené tímto zařízením.

1.4 Formulace požadavků na navrhovaný software

Na základě průzkumu dostupného software byly formulovány následující požadavky na software, jehož implementace je cílem této práce.

- **Prakticky neomezená velikost obrazu**
  - V době začátku tvorby práce neexistoval žádný software, který by umožnil práci s velkými bitmapami.

- **Poskytnout uživateli standardní nástroje známé z PC**
  - Cílem je poskytnout zkušeným uživatelům známé prostředky, které jim poskytnou naprostou kontrolu nad výsledným obrazem.
- **Vhodné využití dotekového ovládání**
  - Dotekové ovládání je zejména vhodné pro kreslení a přesný výběr oblastí obrazu, na PC musí být v naprosté většině simulováno přídavným tabletem.
- **Pokročilá práce s vrstvami**
  - V desktopových aplikacích vrstvy nepředstavují pouze bitmapy, koncept vrstev lze rozšířit prakticky na všechny operace s obrazem. Na zařízeních s OS Android také neexistuje aplikace, která by umožňovala aplikovat na vrstvy masku.
- **Ukládání obrazu včetně vrstev**
  - Tuto funkci by měl podporovat každý seriózní program.
- **Bezztrátový export obrazu do standardního formátu**
  - Tato funkce je nezbytná pro použití v reálném světě.

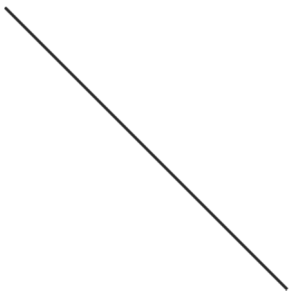
2 Teoretické základy zpracování obrazu

Obrazová data bývají v souvislosti se zpracováním na PC obvykle dělena do dvou velkých skupin - na data rastrová a vektorová.

V zásadě lze říci, že soubor s vektorovými daty obsahuje popis toho, co je na obrázku, vyjádřený pomocí elementárních entit, jakými jsou například úsečky, kružnice a křivky, které je následně příslušný program schopen vykreslit na obrazovku. Příkladem vektorového obrazového formátu je formát svg, který ukládá informace o obraze do xml souboru, který je dobře čitelný jak pro počítač, tak pro člověka. Následující kód převzatý z [12] například popisuje obraz, na kterém je jediná úsečka vedená z bodu $[0,0]$ do bodu $[200,200]$, která má černou barvu a tloušťku 2.

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
   <line x1="0" y1="0" x2="200" y2="200"
      style="stroke:rgb(0,0,0);stroke-width:2"/>
</svg>
```

Obraz generovaný tímto kódem je zobrazen na obr. 2.1, výstup byl pořízen programem Inkscape.



Obr. 2.1 Ukázka vektorové grafiky.

Je zřejmé, že takovýto popis je velmi vhodný pro obrazová data, která se z principu skládají z primitivních entit. Příkladem takovýchto dat jsou například technické výkresy, písma a nejrůznější loga. Vektorový popis je naopak poměrně nevýhodný pro obrazová data, jako jsou například fotografie, kdy ve většině případů nelze provést rozklad na primitivní entity při zachování odpovídající kvality obrazu. K vektorovým datům je ještě dobré poznamenat, že při vhodném použití zabírají takto uložené obrázky obvykle mnohem méně místa, než obrázky rastrové a jsou také obvykle lépe editovatelné. Ovšem v naprosté většině případů je vektorová data pro zobrazení uživateli nutné převést na data rastrová, se kterými pracuje většina výstupních zařízení na PC.

Vzhledem k tomu, že zpracování vektorových obrazových dat není předmětem této práce, nebudeme se jimi dále zabývat, i když mnohé z popsaných algoritmů mohou být aplikovány také na data vektorová.

2.1 Rastrová data

Rastrová data, někdy označovaná také jako bitmapy, na rozdíl od dat vektorových neobsahují informace o tom, co je na obrázku zobrazeno, ale pouze informace o vzhledu obrázku. Rastrová data tvoří základ digitální fotografie, ale také s nimi pracuje většina zobrazovacích zařízení, jako jsou například monitory, televize a tiskárny. Bitmapu (rastrový obraz) si můžeme představit jako matici hodnot, jejíž jednotlivé prvky drží informaci o vzhledu daného bodu obrazu (obrazový bod se obvykle označuje zkráceně jako pixel). Tato informace může mít více významů v závislosti na tom, o jakou bitmapu se jedná, což bude zřejmé z následujícího výkladu.

2.1.1 Velikost obrazu

Velikost obrazu může mít více výkladů. Může se jednat jednak o fyzickou velikost danou v centimetrech či jiné délkové jednotce, která udává například rozměr vytištěného obrázku. Dalším možným výkladem velikosti obrazu je velikost udávaná v obrazových bodech (pixelech), která souvisí s množstvím obrazových dat, které obrázek obsahuje.

2.1.2 Rozlišení obrazu

Rozlišení obrazu udává „jemnost obrázku“ a je dáno poměrem velikosti obrázku v pixelech ku fyzické velikosti obrázku. Obvykle bývá udáváno v DPI, což je zkratka z anglického Dots Per Inch – počet bodů na palec.

2.1.3 Počet kanálů obrazu

Počet kanálů udává, kolik složek má každý pixel. Pokud je počet kanálů větší než jedna, představuje hodnota pixelu vlastně vektor či pole. V běžné praxi se setkáváme s těmito počty kanálů:

Jednokanálové bitmapy

Nejčastější využití jednokanálových bitmap je pro uložení černobílých obrázků. Hodnota každého pixelu představuje jas daného obrazového bodu.

Tříkanálové bitmapy

Tříkanálové bitmapy se používají obvykle k reprezentaci barevných obrázků. Jednotlivé kanály odpovídají intenzitě dané složky barvy. Nejčastěji se barva rozkládá na červenou, zelenou a modrou složku. Mluvíme potom o barevném modelu RGB (z anglického Red Green Blue – Červená Zelená Modrá). Pro zpracování obrazu na počítači se ovšem používají i jiné barevné modely.

Čtyřkanálové bitmapy

Čtyřkanálové bitmapy obvykle reprezentují barevné obrázky, kdy čtvrtý kanál reprezentuje dodatečnou informaci o průhlednosti daného bodu. Toto téma bude podrobněji vysvětleno dále.

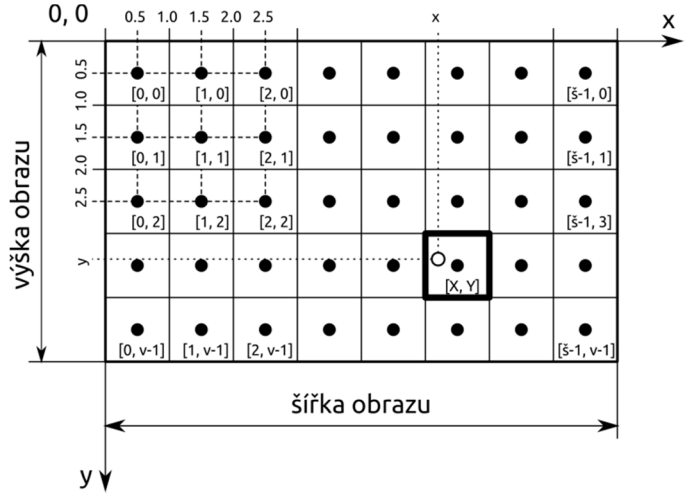
2.1.4 Hloubka kanálu

Hloubka kanálu souvisí s počtem hodnot, které je možno na obraze rozlišit. Lidské oko je schopno rozlišit až deset milionů barev [13]. Z tohoto pohledu by za dostatečnou hloubku kanálu šlo považovat 8 bitů, což dává $2^{3 \cdot 8} = 16\,777\,216$ barev pro tříkanálovou barevnou bitmapu. V praxi se ovšem používají i mnohem vyšší hloubky kanálů – například Adobe Photoshop podporuje kanály o hloubce 16 a 32 bit. Tyto informace, i když nejsou pouhým okem již rozlišitelné, se hodí při počítačovém zpracování obrazu.

2.2 Souřadný systém bitmapy

Pro identifikaci jednotlivých obrazových bodů je nutné na bitmapě definovat souřadný systém. Pro určování pozice na bitmapě obvykle stačí celočíselné souřadnice, ovšem v některých případech, potřebujeme-li obraz zvětšovat, zmenšovat nebo s obrazem provádět jiné geometrické transformace, potřebujeme spojité souřadnice. V této práci budeme dále striktně dodržovat souřadný systém, který je definován na obr. 2.2. Souřadný systém má počátek v levém horním rohu obrazu. Kladný směr osy x směřuje doprava a kladný směr osy y směřuje dolů. Uvažujeme-li celočíselné souřadnice, jsou jednotlivé pixely vzhledem ke konvenci, která se uplatňuje v mnoha

programovacích jazycích, indexované od nuly. Celočíselná souřadnice X tedy nabývá hodnot z intervalu $\langle 0, \text{šířka})$ a celočíselná souřadnice Y nabývá hodnot z intervalu $\langle 0, \text{výška})$. Pro reálné souřadnice x, y platí díky volbě souřadného systému stejná omezení. Převod reálných souřadnic na celočíselné lze v takto zvoleném souřadném systému snadno realizovat pouhým použitím celé části reálné souřadnice.



Obr. 2.2 Souřadný systém obrazu.

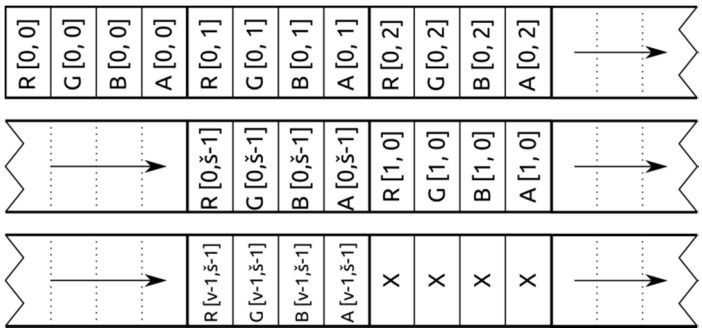
2.2.1 Reprezentace bitmapy v paměti

Vzhledem ke struktuře paměti počítače, není možné uložit matici takovým způsobem, jak je naznačeno na obr. 2.2, ale data je nutno uložit lineárně (paměť počítače si můžeme představit jako velké jednorozměrné pole, kde konkrétní místo v paměti určuje jeho adresa, která je jednoznačně určena číselnou hodnotou, která udává vzdálenost od počátku). Existuje více způsobů reprezentace rastrových dat, v dalším však budeme předpokládat následující popsaný, který je pro případ čtyřkanálové bitmapy naznačen na obr. 2.3.

Data jsou v paměti uložena po řádcích takovým způsobem, že poslední pixel daného řádku je okamžitě následován prvním pixelem řádku následujícího. Pokud se jedná o vícekanálovou bitmapu, pixel je tvořen jednotlivými jeho kanály, které jsou v daném pořadí uloženy v paměti těsně po sobě. Případ čtyřkanálové bitmapy ilustruje obr. 2.3, ze kterého je patrné i pořadí kanálů, tedy první je červený kanál, který je následován zeleným a modrým kanálem. Jako poslední je v pixelu uložen alpha kanál, který udává průhlednost pixelu.

Velikost pixelu je rovna součtu velikostí jeho kanálů a závisí tedy na jejich hloubce. Nejobvyklejší je hloubka 8 bit, což znamená, že kanál může nabývat celočíselných hodnot $0 - 255$. Při 16 bitové hloubce by to byly hodnoty $0 - 65535$ atd. Vzhledem k tomu, že operace nad danou bitmapou jsou shodné pro všechny bitové hloubky,

budeme dále pro účely výkladu jednotlivých operací s bitmapou uvažovat, že každý kanál může nabývat reálných hodnot z intevalu $\langle 0, 1 \rangle$.



Obr. 2.3 Uložení čtyřkanálové bitmapy v paměti PC

2.3 Barevné modely

Obraz v odstínech šedi lze jednoduše reprezentovat jednokanálovou bitmapou, kde hodnota kanálu určuje intenzitu (jas) daného bodu. V případě barevných obrázků je však situace složitější – barvu je nutno reprezentovat pomocí jejích složek a rozklad na tyto složky není jednoznačný. Historicky se pro účely reprezentace barev vytvořilo velké množství barevných modelů, které se vzájemně liší jak svou podstatou a vlastnostmi, tak oblastí použití (přenos analogového televizního signálu, komprese digitálních fotografií...). Tato sekce představuje pouze ty modely, které jsou potřebné pro další výklad, a to ryze prakticky a ve značně zjednodušené podobě.

2.3.1 Model RGB

Jedná se o nejznámější barevný model, kdy výsledná barva je reprezentována pomocí tří základních hodnot – intenzity červené, zelené a modré složky (odtud plyne název Red Green Blue). Výhodou tohoto modelu je jeho názornost. Hodnota 0 daného kanálu znamená, že složka není přítomna, naopak hodnota 1 udává její maximální hodnotu (jedná se o tzv. aditivní model). Pixel o hodnotách $(0, 0, 0)$ tedy reprezentuje černou barvu, pixel $(0, 0, 1)$ modrou a $(1, 1, 1)$ představuje barvu bílou.

Dále, pokud nebude uvedeno jinak, budeme vždy předpokládat, že barevné bitmapy využívají tento barevný model.

2.3.2 Model HCL

Tento model patří mezi méně obvyklé. Název je opět složen z prvních písmen anglického názvu modelu, tedy Hue (odstín) Chroma („barevnost“ vzhledem k podobně osvětlené bílé) Luminance.(„světlost“ či „jas“ dané barvy). Tento model je výhodný, protože umožňuje samostatně upravovat jednotlivé složky zvlášť – například odstín barvy bez změny jejího jasu či sytosti. Toto v modelu RGB nebylo možné jednoduše provést – úprava intenzity jednoho kanálu vždy ovlivnila celkový jas atd.

Podobné možnosti nabízí i další modely, jako je například model HSV (Hue – odstín, Saturation – sytost, Value – hodnota, jas) či HLS (Hue – odstín, Lightness – světlost, Saturation – sytost). Tyto modely však dostatečně nerespektují způsob, jakým lidské oko vnímá barvy. (2.1) udává postup převodu z modelu RGB do modelu HCL.

$$
\begin{aligned}
&M = max(R,G,B) \quad m = min(R,G,B) \quad C = M - m \\
&L = 0.30 \cdot R + 0.59 \cdot G + 0.11 \cdot B \\
&H' = \frac{1}{6} \cdot \begin{cases} \dfrac{G-B}{C} & M = R \\ \dfrac{B-R}{C} + 2 & M = G \\ \dfrac{R-G}{C} + 4 & M = B \end{cases} \quad H = \begin{cases} H' & H' \geq 0 \\ H' + 1 & H < 0 \end{cases}
\end{aligned} \tag{2.1}
$$

Opačný převod je uveden v (2.2). Vztahy jsou převzaty z [14].

$$
\begin{aligned}
&H' = 6 \cdot H \\
&X = C \cdot (1 - |H' \ mod 2 - 1|) \\
&(R',G',B') = \begin{cases} (C,X,0) & 0 \leq H' < 1 \\ (X,C,0) & 1 \leq H' < 2 \\ (0,C,X) & 2 \leq H' < 3 \\ (0,X,C) & 3 \leq H' < 4 \\ (X,0,C) & 4 \leq H' < 5 \\ (C,0,X) & 5 \leq H' < 6 \end{cases} \\
&m = L - (0.30 \cdot R' + 0.59 \cdot G' + 0.11 \cdot B') \\
&(R,G,B) = (R' + m, G' + m, B' + m)
\end{aligned} \tag{2.2}
$$

2.4 Alpha blending

Alpha blending je mechanismus, který slouží k vzájemnému skládání (prolínání) obrázků, které obsahují kromě informací o barvě obrazových bodů navíc hodnotu, která udává průhlednost daného pixelu (krytí, viditelnost). Tato hodnota se nazývá alpha a pokud uvažujeme hodnotu alpha z intervalu $\langle 0; 1 \rangle$, pak hodnota 0 znamená plně průhledný pixel a hodnota 1 plně viditelný pixel. Výsledný obraz, který vznikne složením dvou obrázků s různou hodnotou alpha, si při prvním přiblížení můžeme představit tak, jako když se podíváme přes dvě poloprůhledné fólie najednou.

Základy alpha blendingu byli položeny v práci [15], v počítačové grafice se však v současné době používá alpha blending nejen pro pouhé skládání poloprůhledných obrázků na sebe, ale využívají se například i různé kombinace barev skládaných bit-

map, výsledek je tedy obecně nějakou funkcí hodnoty kanálů pixelu první a druhé bitmapy.

Následuje popis běžně používaných módů pro alpha blending. V [16] můžeme nalézt základní popis jednotlivých módů. Podrobnější popis je obsažen v [17], a také v [18], kde je doplněn i o základní matematické vzorce. Výčet jednotlivých módů zde není však zdaleka kompletní. Výborným zdrojem je [19], ovšem místo popisu je tu k dispozici jen implementace vybraných módů. Co se týče matematického vyjádření většiny módů je dobrým zdrojem také [20]. Dále uváděný popis a vzorce jsou syntézou informací obsažených ve výše uvedených dokumentech. Jednotlivé vzorce však bylo třeba (mnohdy výrazně) upravit, aby zahrnovali jak průhlednost horní bitmapy, která se aplikuje na spodní, tak průhlednost spodní bitmapy. Definice jednotlivých módu je v různých pramenech navzájem různá nebo dokonce chybí úplně.

2.4.1 Alpha blending čtyřkanálových bitmap

V dalším uvažujeme vždy dvě čtyřkanálové bitmapy využívající barevného modelu RGB, přičemž na spodní bitmapu D aplikujeme horní bitmapu S v daném módu. Výsledek označme jako bitmapu R. Základní vzorce (2.3) a (2.4) jsou převzaty z [21].

Při provádění prolnutí bitmap je nutno určit průhlednost výsledné bitmapy. Ta se určí dle rovnice (2.3).

$$\alpha_R = \alpha_S + (1 - \alpha_S) \cdot \alpha_D \tag{2.3}$$

Poté pro jednotlivé kanály vypočítáme výslednou hodnotu dle (2.4)

$$V_R = \frac{(V_S \cdot \alpha_s + (1 - \alpha_S) \cdot \alpha_D \cdot V_D)}{\alpha_R} \tag{2.4}$$

Tyto rovnice prakticky popisují normální mód prolnutí, který bývá také označován jako „S nad D“, ale jsou uvedeny samostatně, vzhledem k tomu, že se používají i při ostatních módech. Výpočet výsledné hodnoty průhlednosti α_R je shodný téměř pro všechny módy. Výsledná hodnota kanálu se ale pro ostatní módy vypočítá tak, že se nejprve určí takovým způsobem, jako kdyby byly obě bitmapy plně viditelné, a poté se tento mezivýsledek aplikuje v normálním módu na bitmapu D s průhledností odpovídající α_S. Výsledná hodnota se tedy určí z rovnice (2.5), kdy jednotlivé módy definují funkci $f(V_S,\ V_D)$.

$$V_R = \frac{\left(V_S \cdot \alpha_s + (1 - \alpha_S) \cdot \alpha_D \cdot f(V_S,\ V_D)\right)}{\alpha_R} \tag{2.5}$$

Následuje popis běžně používaných módů prolnutí včetně matematického vyjádření. Úvodní slovní popis je převzat z [22], U matematického vyjádření je vždy uveden příslušný zdroj.

Normální mód (Normal)

Jedná se o základní mód, který pouze provede kompozici bitmap s respektováním jejich průhlednosti. Jeho určující funkce je dána rovnicí (2.6)

$$f(V_S, V_D) = V_S \tag{2.6}$$

Rozpustit (Dissolve)[1]

Upravuje nebo maluje každý z obrazových bodů na výslednou barvu. Výsledná barva je ale náhodným nahrazením obrazových bodů základní nebo míchanou barvou, v závislosti na krytí daného obrazového bodu.

Nejedná se o mód prolnutí v pravém slova smyslu, protože pro dosažení konzistentních výsledků nemůže být nahrazení obrazových bodů náhodné, ale musí záviset na souřadnici daného pixelu. V opačném případě by při každém prolnutí dvou obrazů docházelo k jinému efektu, což by v konečné implementaci způsobovalo „sněžení“ při každém překreslení výsledného obrazu.

Zezadu (Behind)

Upravuje nebo maluje pouze na průhledné části vrstvy. Působí podobně, jako byste malovali na zadní stranu průhledné fólie.

Výsledek je v tomto případě totožný jako při použití normálního módu, jen s tím rozdílem, že se dolní a horní bitmapa navzájem zamění.

Vymazat (Clear)

Upravuje nebo maluje jednotlivé obrazové body tak, že je zprůhlední.

Operace odpovídá odečtení masky, které bude popsáno dále v sekci pojednávající o blendingu jednokanálových bitmap. Jako maska se v tomto případě použije alpha kanál horní bitmapy.

Ztmavit (Darken)

Vezme barevnou informaci v jednotlivých kanálech a vybere tmavší z míchané a základní barvy jako výslednou barvu. Obrazové body světlejší než míchaná barva se nahradí a obrazové body tmavší než míchaná barva se nezmění.

Určující funkcí módu je funkce uvedená v (2.7), převzato z [18].

$$f(V_S, V_D) = min(V_S, V_D) \tag{2.7}$$

[1] Tento mód nebyl jako jediný v rámci této práce implementován.

Násobit (Multiply)

Porovná barevné informace v jednotlivých kanálech a vynásobí základní barvu míchanou barvou. Výsledkem je vždy tmavší barva. Násobení libovolné barvy s černou vytvoří černou barvu. Násobení libovolné barvy s bílou nechá barvu beze změny. Malujete-li jinou barvou než černou nebo bílou, vedou následné tahy nástrojem pro malování k postupně tmavším a tmavším barvám. Efekt je podobný vícenásobné malbě popisovačem (fixem) přes sebe.

Určující funkcí módu je funkce uvedená v (2.8), převzato z [18].

$$f(V_S, V_D) = V_S \cdot V_D \tag{2.8}$$

Ztmavit barvy (Color Burn)

Porovná informace o barvě v jednotlivých kanálech a zvýšením kontrastu mezi nimi ztmaví základní barvu, aby odpovídala smíchané barvě. Míchání s bílou nechá barvu beze změny.

Určující funkcí módu je funkce uvedená v (2.9), převzato z [20].

$$f(V_S, V_D) = 1 - \frac{(1 - V_D)}{V_S} \tag{2.9}$$

Lineárně ztmavit (Linear Burn)

Porovná barevné informace v jednotlivých kanálech a zmenšením jasu ztmaví základní barvu, aby odpovídala míchané barvě. Míchání s bílou nechá barvu beze změny.

Určující funkcí módu je funkce uvedená v (2.10), převzato z [20].

$$f(V_S, V_D) = V_S + V_D - 1 \tag{2.10}$$

Tmavší barva (Darker Color)

Porovná součet hodnot všech kanálů pro míchanou a základní barvu a zobrazí barvu s nižší hodnotou. Režim Tmavší barva nevytvoří třetí barvu, která by mohla být výsledkem prolnutí Ztmavit, protože v tomto režimu se pro výslednou barvu zvolí nižší hodnota jednotlivých kanálů ze základní a míchané barvy.

Tento mód je totožný s normálním módem s jediným rozdílem – jako horní pixel se použije buď příslušný pixel horní bitmapy, nebo příslušný pixel bitmapy spodní, a to v závislosti na tom, který má nižší hodnotu součtu všech svých kanálů udávajících jeho barvu. Formálně je toto pospáno rovnicí (2.11). R, G a B představují po řadě hodnoty červeného, zeleného a modrého kanálu příslušné bitmapy.

$$S' = \begin{cases} S & R_S + G_S + B_S < R_D + G_D + B_D \\ D & R_S + G_S + B_S \geq R_D + G_D + B_D \end{cases} \qquad (2.11)$$
$$f(V_S, V_D) = V_{S'}$$

Zesvětlit (Lighten)

Porovná barevné informace v jednotlivých kanálech a vybere světlejší z míchané a základní barvy jako výslednou barvu. Obrazové body tmavší než míchaná barva se nahradí a obrazové body světlejší než míchaná barva se nezmění.

Určující funkcí módu je funkce uvedená v (2.12), převzato z [18].

$$f(V_S, V_D) = max(V_S, V_D) \qquad (2.12)$$

Závoj (Screen)

Porovná barevné informace v jednotlivých kanálech a vynásobí inverzní hodnotu míchané a základní barvy. Výsledkem je vždy světlejší barva. Závoj s černou barvou nechá barvu beze změny. Závoj s bílou barvou vytvoří bílou barvu. Efekt je podobný, jako když promítáte více diapozitivů přes sebe.

Určující funkcí módu je funkce uvedená v (2.13), převzato z [18].

$$f(V_S, V_D) = 1 - (1 - V_S) \cdot (1 - V_D) \qquad (2.13)$$

Zesvětlit barvy (Color Dodge)

Porovná informace o barvě v jednotlivých kanálech a snížením kontrastu mezi nimi zesvětlí základní barvu, aby odpovídala smíchané barvě. Mícháním s černou nevzniká žádná změna.

Určující funkcí módu je funkce uvedená v (2.14), převzato z [20].

$$f(V_S, V_D) = \frac{V_D}{(1 - V_S)} \qquad (2.14)$$

Lineárně zesvětlit (Linear Dodge / Add)

Porovná barevné informace v jednotlivých kanálech a zvýšením jasu zesvětlí základní barvu, aby odpovídala míchané barvě. Mícháním s černou nevzniká žádná změna.

Určující funkcí módu je funkce uvedená v (2.15), převzato z [20].

$$f(V_S, V_D) = V_S + V_D \qquad (2.15)$$

Světlejší barva (Lighter Color)

Porovná součet hodnot všech kanálů pro míchanou a základní barvu a zobrazí barvu s vyšší hodnotou. Režim Světlejší barva nevytvoří třetí barvu, která by mohla

být výsledkem prolnutí Zesvětlit, protože v tomto režimu se pro výslednou barvu zvolí vyšší hodnota jednotlivých kanálů ze základní a míchané barvy.

Mód je podobně jako „tmavší barva“ totožný s normálním módem s jediným rozdílem – jako horní pixel se použije buď příslušný pixel horní bitmapy, nebo příslušný pixel bitmapy spodní, a to v závislosti na tom, který má vyšší hodnotu součtu všech svých kanálů udávajících jeho barvu. Formálně je toto pospáno rovnicí (2.16). R, G a B představují po řadě hodnoty červeného, zeleného a modrého kanálu příslušné bitmapy.

$$S' = \begin{cases} S & R_S + G_S + B_S > R_D + G_D + B_D \\ D & R_S + G_S + B_S \leq R_D + G_D + B_D \end{cases} \qquad f(V_S, V_D) = V_{S'} \tag{2.16}$$

Překrýt (Overlay)

Násobí nebo závojem změní barvy v závislosti na základní barvě. Vzorky nebo barvy překryjí existující obrazové body, přičemž se zachovají světla a stíny základní barvy. Základní barva se nenahrazuje, ale smíchá se s míchanou barvou tak, aby odpovídala světlosti nebo tmavosti původní barvy.

Určující funkcí módu je funkce uvedená v (2.17), převzato z [18].

$$f(V_S, V_D) = \begin{cases} 2 \cdot V_S \cdot V_D & V_D < 0.5 \\ 1 - 2 \cdot (1 - V_S) \cdot (1 - V_D) & V_D \geq 0.5 \end{cases} \tag{2.17}$$

Měkké světlo (Soft Light)

Ztmaví nebo zesvětlí barvy v závislosti na míchané barvě. Efekt je podobný osvětlení obrazu rozptýleným světlem. Pokud je míchaná barva (světelný zdroj) světlejší než 50 % šedá, obraz se zesvětlí. Pokud je míchaná barva tmavší než 50 % šedá, obraz se ztmaví. Při malování zcela černou nebo bílou vznikne výrazně tmavší nebo světlejší plocha, ale ne zcela černá nebo bílá.

Určující funkcí módu je funkce uvedená v (2.18), převzato z [18].

$$f(V_S, V_D) = \big((1 - V_D) \cdot V_S + 1 - (1 - V_S) \cdot (1 - V_D)\big) \cdot V_D \tag{2.18}$$

Tvrdé světlo (Hard Light)

Násobí nebo závojem změní barvy podle míchané barvy. Efekt je podobný osvětlení obrazu ostrým bodovým světlem. Pokud je míchaná barva (světelný zdroj) světlejší než 50 % šedá, obraz se zesvětlí, jako v režimu Závoj. To se hodí pro přidávání světel do obrazu. Pokud je míchaná barva tmavší než 50 % šedá, obraz se ztmaví, jako v režimu Násobit. To se hodí pro přidávání stínů do obrazu. Při malování zcela černou nebo bílou vznikne čistá černá nebo bílá.

Určující funkcí módu je funkce uvedená v (2.19), převzato z [18].

$$f(V_S, V_D) = \begin{cases} 2 \cdot V_S \cdot V_D & V_D < 0.5 \\ 1 - 2 \cdot (1 - V_S) \cdot (1 - V_D) & V_D \geq 0.5 \end{cases} \tag{2.19}$$

Jasné světlo (Vivid Light)

Ztmaví nebo zesvětlí barvy pomocí zvětšení nebo zmenšení kontrastu v závislosti na míchané barvě. Pokud je míchaná barva (světelný zdroj) světlejší než 50 % šedá, obraz se zesvětlí zmenšením kontrastu. Pokud je míchaná barva tmavší než 50 % šedá, obraz se ztmaví zvětšením kontrastu.

Určující funkcí módu je funkce uvedená v (2.20), převzato z [20].

$$f(V_S, V_D) = \begin{cases} 1 - \dfrac{(1 - V_D)}{2 \cdot V_S} & V_S < 0.5 \\ \dfrac{V_D}{2 \cdot (1 - V_S)} & V_S \geq 0.5 \end{cases} \tag{2.20}$$

Lineární světlo (Linear Light)

Ztmaví nebo zesvětlí barvy pomocí zmenšení nebo zvětšení jasu v závislosti na míchané barvě. Pokud je míchaná barva (světelný zdroj) světlejší než 50 % šedá, obraz se zesvětlí pomocí zvětšení jasu. Pokud je míchaná barva tmavší než 50 % šedá, obraz se ztmaví pomocí zmenšení jasu.

Určující funkcí módu je funkce uvedená v (2.21), převzato z [19].

$$f(V_S, V_D) = \begin{cases} 2 \cdot V_S + V_D - 1 & V_S < 0.5 \\ 2 \cdot (V_S - 0.5) + V_D & V_S \geq 0.5 \end{cases} \tag{2.21}$$

Bodové světlo (Pin Light)

Nahrazuje barvy v závislosti na míchané barvě. Pokud je míchaná barva (světelný zdroj) světlejší než 50 % šedá, obrazové body tmavší než míchaná barva se nahradí a obrazové body světlejší než míchaná barva se nezmění. Pokud je míchaná barva tmavší než 50 % šedá, obrazové body světlejší než míchaná barva se nahradí a obrazové body tmavší než míchaná barva se nezmění. To se hodí pro přidávání speciálních efektů do obrazu.

Určující funkcí módu je funkce uvedená v (2.22), převzato z [19].

$$f(V_S, V_D) = \begin{cases} min(2 \cdot V_S, V_D) & V_S < 0.5 \\ max(2 \cdot (V_S - 0.5), V_D) & V_S \geq 0.5 \end{cases} \tag{2.22}$$

Tvrdé míchání (Hard Mix)

Přidá hodnoty červeného, zeleného a modrého kanálu míchané barvy k hodnotám RGB základní barvy. Pokud je výsledný součet pro kanál 255 nebo větší, dostane hodnotu 255; pokud je menší než 255, hodnotu 0. Proto mají všechny prolnuté obra-

zové body hodnoty červeného, zeleného a modrého kanálu buď 0 nebo 255. Tím dojde ke změně všech obrazových bodů na primární aditivní barvy (červenou, zelenou nebo modrou), bílou nebo černou.

Určující funkcí módu je funkce uvedená v (2.23), převzato z [19].

$$f(V_S, V_D) = \begin{cases} 0 & V_S < 1 - V_D \\ 1 & V_S \geq 1 - V_D \end{cases} \tag{2.23}$$

Rozdíl (Difference)

Porovná barevné informace v jednotlivých kanálech a odečte buď míchanou barvu od základní barvy nebo základní barvu od míchané barvy, podle toho, která má vyšší hodnotu jasu. Míchání s bílou invertuje hodnoty základní barvy; mícháním s černou nevzniká žádná změna.

Určující funkcí módu je funkce uvedená v (2.24), převzato z [18].

$$f(V_S, V_D) = \begin{cases} V_S - V_D & V_D < V_S \\ V_D - V_S & V_D \geq V_S \end{cases} \tag{2.24}$$

Vyloučit (Exclusion)

Vytváří podobný efekt jako režim Rozdíl, ale méně kontrastní. Míchání s bílou invertuje hodnoty základní barvy. Mícháním s černou nevzniká žádná změna.

Určující funkcí módu je funkce uvedená v (2.25), převzato z [20].

$$f(V_S, V_D) = V_S + V_D - 2 \cdot V_S \cdot V_D \tag{2.25}$$

Odečíst (Subtract)

Porovná barevné informace v jednotlivých kanálech a odečte míchanou barvu od základní barvy. V 8bitových a 16bitových obrazech se jakékoli výsledné záporné hodnoty oříznou na nulu.

Určující funkcí módu je funkce uvedená v (2.26), převzato z [18].

$$f(V_S, V_D) = \begin{cases} 0 & V_D < V_S \\ V_D - V_S & V_D \geq V_S \end{cases} \tag{2.26}$$

Rozdělit (Divide)

Porovná barevné informace v jednotlivých kanálech a podělí základní barvu míchanou barvou.

Určující funkcí módu je funkce uvedená v (2.27), převzato z [18].

$$f(V_S, V_D) = \frac{V_D}{V_S} \tag{2.27}$$

Odstín (Hue)

Vytváří výslednou barvu se světlostí a sytostí základní barvy a s odstínem míchané barvy.

Pro dosažení požadovaného výsledku je nutné převést data do jiného barevného modelu, protože model RGB neumožnuje samostatnou úpravu odstínu, světlosti a sytosti. Dobrých výsledků v tomto ohledu dosahuje barevný model HCL popsaný dříve. Samotné prolnutí probíhá v normálním módu, ovšem hodnotu pixelu bitmapy S je nutno složit z komponent pixelu S a D v modelu HCL. Celý proces popisuje (2.28), kde symbol c představuje příslušnou převodní funkci.

$$\begin{aligned} &(H_S, C_S, L_S) = c_{RGB\to HCL}(R_S, G_S, B_S) \quad (H_D, C_D, L_D) = c_{RGB\to HCL}(R_D, G_D, B_D) \\ &S' = c_{HCL\to RGB}(H_S, C_D, L_D) \\ &f(V_S, V_D) = V_{S'} \end{aligned} \tag{2.28}$$

Sytost (Saturation)

Vytváří výslednou barvu se světlostí a odstínem základní barvy a se sytostí míchané barvy. Malujete-li s použitím tohoto režimu v oblasti s nulovou sytostí (šedá), nedochází k žádným změnám.

Tento mód opět vyžaduje převod do modelu HCL, vše popisuje (2.29).

$$\begin{aligned} &(H_S, C_S, L_S) = c_{RGB\to HCL}(R_S, G_S, B_S) \quad (H_D, C_D, L_D) = c_{RGB\to HCL}(R_D, G_D, B_D) \\ &S' = c_{HCL\to RGB}(H_D, C_S, L_D) \\ &f(V_S, V_D) = V_{S'} \end{aligned} \tag{2.29}$$

Barva (Color)

Vytváří výslednou barvu se světlostí základní barvy a s odstínem a sytostí míchané barvy. Zachovají se tím úrovně šedi v obraze. Tento režim je vhodný pro kolorování černobílých obrazů a tónování barevných obrazů.

Tento mód opět vyžaduje převod do modelu HCL, vše popisuje (2.30).

$$\begin{aligned} &(H_S, C_S, L_S) = c_{RGB\to HCL}(R_S, G_S, B_S) \quad (H_D, C_D, L_D) = c_{RGB\to HCL}(R_D, G_D, B_D) \\ &S' = c_{HCL\to RGB}(H_S, C_S, L_D) \\ &f(V_S, V_D) = V_{S'} \end{aligned} \tag{2.30}$$

Světlost (Luminosity)

Vytváří výslednou barvu s odstínem a sytostí základní barvy a se světlostí míchané barvy. Tento režim vytvoří opačný efekt než režim Barva.

Tento mód opět vyžaduje převod do modelu HCL, vše popisuje (2.31).

$$\begin{aligned} &(H_S, C_S, L_S) = c_{RGB\to HCL}(R_S, G_S, B_S) \quad (H_D, C_D, L_D) = c_{RGB\to HCL}(R_D, G_D, B_D) \\ &S' = c_{HCL\to RGB}(H_D, C_D, L_S) \\ &f(V_S, V_D) = V_{S'} \end{aligned} \tag{2.31}$$

2.4.2 Alpha blending s jednokanálovými bitmapami

Pro alpha blending jednokanálových bitmap nepřipadají výše zmíněné módy v úvahu, zato pokud uvažujeme jednokanálovou bitmapu jako masku průhlednosti, tedy že její hodnoty dodatečně upravují hodnoty průhlednosti jiné bitmapy, připadá v úvahu několik možností použití takové masky. Pro jednoduchost uvažujme masku průhlednosti s normalizovanými hodnotami z intervalu $\langle 0, 1\rangle$. Pokud je maska aplikována na čtyřkanálovou bitmapu, dále popsané operace se aplikují pouze na alpha kanál této bitmapy.

Přiřazení masky průhlednosti

Nejzákladnější operací s maskou průhlednosti je její přiřazení k dané bitmapě. Tato operace spočívá v prostém vynásobení odpovídajících si hodnot, jak je naznačeno v (2.32). Výpočet koresponduje s očekávanými výsledky, tedy aplikace plně viditelné masky odpovídá násobení jedničkou a hodnota se nemění, naopak aplikace plně průhledné masky odpovídá násobení nulou a celá bitmapa je neviditelná.

$$\alpha_R = \alpha_S \cdot \alpha_D \tag{2.32}$$

Přičtení masky

Přičtení masky průhlednosti je analogické k určení výsledné průhlednosti u blendingu čtyřkanálových bitmap, výsledný vztah udává (2.33).

$$\alpha_R = \alpha_S + (1 - \alpha_S) \cdot \alpha_D \tag{2.33}$$

Odečtení masky

Odečtení masky průhlednosti lze odvodit logickou úvahou z očekávaných výstupů pro krajní hodnoty a uvědomění si podobnosti se vztahy, které platí pro mohutnost výsledků množinových operací sjednocení a rozdílu, kdy přičtení masky je analogické k sjednocení množin, zatímco odečtení k odečtení. Formulujeme-li tedy požadavek, že odečítání průhledné masky neovlivňuje hodnotu, zatímco odečítání plně viditelné masky dává plně neviditelný výsledek, je vidět, že dané podmínky dobře splňuje vztah (2.34).

$$\alpha_R = \alpha_D - \alpha_S \cdot \alpha_D \tag{2.34}$$

2.5 Bodové operace

Bodové operace provádí modifikaci hodnoty pixelu, aniž by měnily velikost, geometrii nebo lokální strukturu obrazu. Hodnota každého pixelu závisí jen a pouze na předchozí hodnotě pixelu na stejné pozici.[23]

Pro každý pixel tedy platí vztah (2.35), kde $V_R[X,Y]$ označuje výslednou hodnotu pixelu na pozici X a Y, která je funkcí jen a pouze původní hodnoty pixelu na pozici X, Y označené jako $V_S[X,Y]$. Je dobré si uvědomit, že funkce $f(V_S[X,Y])$ nezávisí nijak na poloze daného pixelu.

$$V_R[X,Y] = f(V_S[X,Y]) \tag{2.35}$$

Mezi základní bodové operace patří například úpravy jasu a kontrastu, gama korekce, ale také komplexnější úpravy, jakými je například převod obrazu do odstínů šedi, posterizace, či téměř obecné úpravy intenzit pomocí tzv. křivek.

V dalším textu bude uveden popis bodových operací, které byly v rámci řešení diplomové práce implementovány. V literatuře je obvykle uveden pouze popis těchto operací pro jednokanálové černobíle bitmapy, zobecnění na barevné bitmapy je však nasnadě – operace se postupně aplikují na jednotlivé kanály obsahující informace o barvě daného pixelu.

Bodové operace jsou jednoznačně určeny svým funkčním předpisem. Obor hodnot těchto funkcí však může zasahovat i mimo dovolené hodnoty obrazových bodů, v takovém případě je třeba hodnoty ořezat. V dalším textu implicitně předpokládáme, že pokud je funkční hodnota předepsané funkce vyšší, než je maximální dovolená hodnota pixelu (vzhledem k normalizovaným hodnotám z intervalu $\langle 0, 1\rangle$ je touto hodnotou jednička), bude hodnota nastavena na nejvyšší dovolenou. Analogicky pokud je hodnota menší, než nejmenší dovolená, bude nastavena na tuto.

2.5.1 Úprava jasu

Úpravy jasu obrazu lze dosáhnout například metodou popsanou v [23], kdy platí vztah (2.36). Dle tohoto vztahu dosáhneme úpravy jasu pouhým přičtením konstanty k k hodnotě každého pixelu.

$$V_R[X,Y] = f(V_S[X,Y]) = V_S[X,Y] + k \tag{2.36}$$

Tento přístup však není v souladu s výsledky, úpravy jasu v běžně dostupných grafických editorech. Tato závislost by se dala spíše označit za úpravu expozice obrázku, která se sice projeví zvýšením celkového jasu, ale způsobí vznik přeexponovaných oblastí, kde ztratíme všechny informace o obraze. Software GIMP implementuje funkční závislost danou rovnicí (2.37), která již nezpůsobuje vznik takto přeexponovaných oblasti.

$$V_R[X,Y] = f(V_S[X,Y]) = \begin{cases} (1+k)\cdot V_S[X,Y] & -1 \leq k \leq 0 \\ (1-k)\cdot V_S[X,Y] + k & 0 < k \leq 1 \end{cases} \tag{2.37}$$

2.5.2 Úprava kontrastu

Funkce pro úpravu kontrastu je opět uvedena v základní podobě například v [23] jako (2.38), ovšem ani tento přístup spočívající v pouhém násobení hodnoty pixelu konstantou k není nejvhodnější, protože jak uvádí [13], při této úpravě dochází ke změně celkového jasu obrázku, což většinou není vhodné, proto je dobré tuto změnu jasu alespoň nějakým způsobem kompenzovat.

$$V_R[X,Y] = f(V_S[X,Y]) = k \cdot V_S[X,Y] \tag{2.38}$$

Rovnice (2.39) uvádí výhodnější funkci, která je opět implementována například v programu GIMP a korekci jasu již řeší.

$$V_R[X,Y] = f(V_S[X,Y]) = k \cdot (V_S[X,Y] - 0.5) + 0.5 \quad k \geq 0 \tag{2.39}$$

2.5.3 Gama korekce

Pojem gama má původ v klasické fotografii a souvisí se závislostí mezi množstvím dopadajícího světla na film a výsledné barvě zachycené na filmu, která je přibližně logaritmická. Pokud uvažujeme intenzitu světla v logaritmických souřadnicích, je tato závislost přibližně lineární a sklon této závislosti udává hodnotu gama daného fotografického materiálu. [23]

V digitální fotografii se gama korekce používá ke kompenzaci nelinearit, které vznikají v důsledku nelineární závislosti intenzity světla, které dopadá na senzor fotoaparátu, a jeho výstupního napětí, ale také nelinearit, ke kterým může docházet na straně výstupních zařízení, jako jsou monitory a tiskárny. Měřením barev, zajištěním toho, aby barvy skutečné scény odpovídaly těm zachyceným na fotografii a tyto barvy byly dále reprodukovatelné na výstupních zařízeních, jako jsou monitory a tiskárny, se zabývá kolorimetrie. Úvod do problematiky je možné nalézt například v [24], výklad tohoto tématu je ovšem již nad rámec této práce.

Pro praktickou aplikaci gama korekce lze využít funkce (2.40) uvedené v [23]

$$V_R[X,Y] = f(V_S[X,Y]) = (V_S[X,Y])^{\gamma} \tag{2.40}$$

2.5.4 Křivky

Lepší programy pro práci s rastrovou grafikou umožňují provádět bodové operace také nástrojem běžně označovaným jako křivky. Tento nástroj umožňuje uživateli předepsat převodní funkci $f(V_S[X,Y])$ téměř libovolně, a to jak pro výslednou hodnotu pixelu, tak pro jednotlivé jeho kanály samostatně, což dovoluje uživateli kontrolovat i zabarvení obrázku. Uživatel zadává body, kterými má požadovaná funkce pro-

cházet a výsledná funkce se určí interpolací těchto bodů. V této práci použité řešení využívá pro interpolaci kubický spline. Pro samotný výpočet interpolace je použit algoritmus uvedený v [25].

2.5.5 Efektivita bodových operací

Pokud je převodní funkce $f(V_S[X,Y])$ složitější, může být výhodné si její funkční hodnoty předpočítat a uložit si je do takzvané LUT tabulky (z anglického Look Up Table – „náhledová tabulka“). Toto řešení vyžaduje jistý výpočetní čas ještě před samotnou aplikací operace na obraz, ovšem v případě složitých funkčních předpisů vede ke zvýšení efektivity provádění bodových operací, protože místo vyčíslování funkční hodnoty v daném bodě pouze načteme funkční hodnotu z tabulky.

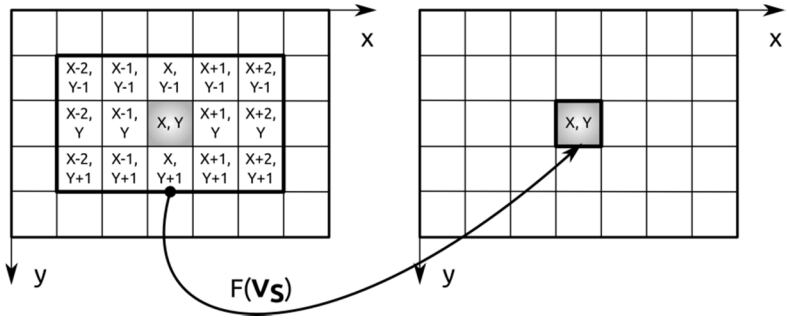
2.6 Filtry

Společnou vlastností bodových operací a filtrů je, že nemění geometrii obrazu (provádí mapování $1:1$ – velikost obrazu se nemění). Filtry však na rozdíl od výše zmíněných bodových operací určují výslednou hodnotu daného pixelu z hodnot více pixelů původního obrazu. Formálně lze tedy filtr definovat například jako (2.41), kde $\boldsymbol{V_S}$ označuje množinu bodů původního obrazu.

$$V_R[X,Y] = F(\boldsymbol{V_S}) \tag{2.41}$$

Funkce $F(\boldsymbol{V_S})$ může být zcela obecná a brát v úvahu všechny body původního obrazu, obvykle však uvažuje jen jisté okolí pixelu na souřadnici X, Y, které je obvykle obdélníkového nebo čtvercového tvaru a střed tohoto obdélníka je právě bod X, Y. $\boldsymbol{V_S}$ lze tedy formálně popsat tak, jak je naznačeno na (2.42). Princip funkce takového filtru je naznačen na obr. 2.4 pro $m = 2$ a $n = 1$. Jak je patrné, pro výpočet hodnoty jednoho pixelu musíme brát v úvahu $(2 \cdot m + 1) \cdot (2 \cdot n + 1)$ hodnot.

$$\boldsymbol{V_S} = \{V_S[X+i, Y+j]\} \quad \begin{matrix} i = -m \ldots m \\ j = -n \ldots n \end{matrix} \tag{2.42}$$



Obr. 2.4 Princip funkce filtru

Problém nastává, jestliže požadovaná data nejsou pro výpočet k dispozici – k tomu dochází v případě, že počítáme výslednou hodnotu v blízkosti hranice obrazu. Existuje několik způsobů, jak se s touto situací vypořádat, následuje přehled několika běžně používaných.

- **Vynechání okrajové oblasti**
  - Toto řešení spočívá v tom, že pro okrajové pixely se neprovádí výpočet a ponechá se zde původní hodnota. Pro kvalitní výstup je toto řešení nepřijatelné.

- **Uvažování konstantní okrajové oblasti**
  - Toto řešení může poskytovat v mnoha případech dobré výsledky. Při tomto řešení přiřadíme pixelům mimo oblast obrazu hodnotu nejbližších krajních pixelů, tedy například $V_S[-1,3] \coloneqq V_S[0,3]$, $V_S[-2,3] \coloneqq V_S[0,3]$ atd.

- **Zrcadlení původní oblasti.**
  - Toto řešení je výpočetně nejnáročnější, ale obvykle poskytuje nejlepší výsledky a při řešení této práce bylo právě toto implementováno. Principem je doplnění obrazových bodů mimo obraz zrcadlovým obrazem původního obrazu, tedy $V_S[-1,3] \coloneqq V_S[0,3]$, $V_S[-2,3] \coloneqq V_S[1,3]$ atd.

2.6.1 Lineární filtry

Velkou skupinu v praxi používaných filtrů tvoří filtry lineární, pro které při uvažování obdélníkové oblasti platí (2.43).

$$V_R[X,Y] = F(\boldsymbol{V_S}) = \sum_{i=-m}^{m} \sum_{j=-n}^{n} h_{i,j} \cdot V_S[X+i, Y+j] \tag{2.43}$$

Tyto filtry se nazývají lineární, protože výsledná hodnota je dána lineární kombinací pixelů v daném okolí. Takovýto lineární filtr je jednoznačně určen maticí koeficientů $\boldsymbol{H} = |h_{i,j}|$.

2.6.2 Separabilita lineárních filtrů

S použitím matematického aparátu uvedeného v [23] lze odvodit, že pokud jsou prvky matice $\boldsymbol{H}$ dány funkcí $H_{x,y}(i,j)$ dvou proměnných, která lze vyjádřit jako součin dvou funkcí jedné proměnné (2.44), lze dosáhnout postupnou filtrací s maticemi odpovídajícími těmto funkcím stejného výsledku, jako filtrací původní maticí $\boldsymbol{H}$. Výhoda tohoto postupu plyne z toho, že matice odpovídajících funkcí mají jeden rozměr roven jedné, což výrazně snižuje výpočetní nároky.

$$H_{x,y}(i,j) = \big(H_x \otimes H_y\big)(i,j) = H_x(i) \cdot H_y(j) \tag{2.44}$$

2.6.3 Vybrané lineární filtry

Lineární filtry lze dělit do dvou velkých skupin. První skupinou jsou filtry typu dolní propust, které potlačují vysoké frekvence v obraze a mají vyhlazující efekt. Druhou skupinou jsou filtry typu horní propust, které naopak zvýrazňují detaily obrazu. Následuje přehled vybraných lineárních filtrů, které byly v této diplomové práci implementovány. Jsou zde zmíněni zástupci obou skupin.

Filtry typu dolní propust

Gaussovské rozostření

Jedná se o filtr typu dolní propust, který je separovatelný [23]. Vzhledem k diskrétní podstatě digitálních filtrů a tvaru funkce určující hodnotu prvků matice filtru je pro dosažení optimálních výsledků vhodné zajistit její dostatečnou velikost. Doporučuje se $m \approx 3 \cdot \sigma_x$ a $n \approx 3 \cdot \sigma_y$. Funkční předpis pro jednotlivé prvky matice $\boldsymbol{H}$ je na (2.45) [13].

$$\boldsymbol{H} = \left[h_{i,j}\right]_{i=-m\ldots m, j=-n\ldots n} \quad h_{i,j} = k \cdot e^{-\left(\frac{i^2}{2\cdot\sigma_x^2}+\frac{j^2}{2\cdot\sigma_y^2}\right)} \tag{2.45}$$

Separovatelná verze je pak dána v souladu s [23] dle (2.46).

$$\begin{aligned} \boldsymbol{H_x} &= \left[h_{i,j}\right]_{i=-m\ldots m,\ \ j=0} \quad h_{i,j} = k_x \cdot e^{-\left(\frac{i^2}{2\cdot\sigma_x}\right)} \\ \boldsymbol{H_y} &= \left[h_{i,j}\right]_{i=0,\ \ j=-n\ldots n} \quad h_{i,j} = k_y \cdot e^{-\left(\frac{j^2}{2\cdot\sigma_y}\right)} \end{aligned} \tag{2.46}$$

Konstanty k, k_x a k_y dle [13] volíme tak, aby součet všech prvků v dané matici byl roven jedné, což vychází z předpokladu, že výsledná hodnota je vlastně vážený průměr hodnot pixelů v daném okolí.

Filtry typu horní propust

Existuje celá řada filtrů typu dolní propust, následuje popis těch, které byly implementovány v rámci řešení této diplomové práce.

Tyto filtry již obecně nejsou separovatelné, opět uplatňujeme podmínku, že součet všech prvků matice $\boldsymbol{H}$ musí být roven jedné. Popsané funkce, které generují prvky matice, mají však obecně tento součet rovný 0 (pro dostatečný počet prvků). Abychom dosáhnuli součtu rovnému jedné, postupujeme následovně. Vypočteme prvky matice $\boldsymbol{H}$ dle generující funkce. Sečteme hodnoty všech prvků $h_{i,j}$, čímž dostaneme rozdíl, který musíme korigovat. Hodnotu tohoto rozdílu rovnoměrně rozdělíme mezi kladné prvky $h_{i,j}$, čímž dostaneme součet rovný nule. Pokud chceme dodatečně upravit sílu filtru, vynásobíme nyní jednotlivé prvky matice $\boldsymbol{H}$ odpovídajícím koeficientem. Nakonec formálně přičteme ke středovému prvku jednotku (jedná se o prvek $h_{0,0}$).

Laplacián

Jedná se o základní filtr, jeho matice není dána generující funkcí, ale pevně danými hodnotami (2.47). Jedná se o aproximaci aplikace Laplaceova operátoru $\nabla^2 V_S$ [26].

$$\boldsymbol{H} = \left[h_{i,j}\right]_{i=-m\ldots m, j=-n\ldots n} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{2.47}$$

Gaussovské zostření

Jedná se o velmi jednoduchý filtr typu horní propust, který je zkonstruován postupem popsaným v [27], který spočívá v odečtení matice filtru typu dolní propust od matice filtru, který nemění obraz, tedy je dán maticí, která má samé nuly na všech prvcích kromě prostředního, který je roven jedné. Jako dolní propust zde bylo zvoleno Gaussovské rozostření. Vztah pro jednotlivé prvky matice $\boldsymbol{H}$ je uveden v rovnici (2.48), je dobré si všimnout, že zde již nemůžeme ignorovat hodnotu konstanty k, jako v případě rovnice (2.45), ale musíme použít známý Gaussův vztah s přesným vyjádřením této konstanty.

Navzdory své jednoduchosti dává tento filtr velmi dobré výsledky.

$$\boldsymbol{H} = \left[h_{i,j}\right]_{i=-m\ldots m, j=-n\ldots n} = \begin{cases} h_{i,j} = 1 - \dfrac{1}{\sqrt{2\cdot\pi\cdot\sigma^2}} \cdot e^{-\left(\frac{i^2+j^2}{2\cdot\sigma^2}\right)} & i = 0,\ j = 0 \\ h_{i,j} = -\dfrac{1}{\sqrt{2\cdot\pi\cdot\sigma^2}} \cdot e^{-\left(\frac{i^2+j^2}{2\cdot\sigma^2}\right)} & i \neq 0,\ j \neq 0 \end{cases} \tag{2.48}$$

Laplacián Gaussova vztahu (LoG)

Tento filtr představuje přesné vyčíslení funkce aproximované filtrem označeným dříve jako „Laplacián“. Vztah lze snadno odvodit aplikací operátoru ∇^2 na Gaussův vztah, jak jen naznačeno v (2.49) (výpočet byl proveden programem Maple), ale lze jej nalézt i v [28].

$$\begin{aligned} G(x,y) &= \frac{1}{\sqrt{2\cdot\pi\cdot\sigma^2}} \cdot e^{-\left(\frac{x^2+y^2}{2\cdot\sigma^2}\right)} \\ LoG(x,y) = \nabla^2 G(x,y) &= \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2} = \cdots \\ &= \frac{e^{-\left(\frac{x^2+y^2}{2\cdot\sigma^2}\right)} \cdot (x^2 + y^2 - 2\cdot\sigma^2)}{2\cdot\pi\cdot\sigma^6} \end{aligned} \tag{2.49}$$

Výsledný vztah pro prvky matice $\boldsymbol{H}$ je tedy na (2.50).

$$\boldsymbol{H} = \left[h_{i,j}\right]_{i=-m\ldots m, j=-n\ldots n} = \frac{e^{-\left(\frac{i^2+j^2}{2\cdot\sigma^2}\right)} \cdot (i^2 + j^2 - 2\cdot\sigma^2)}{2\cdot\pi\cdot\sigma^6} \tag{2.50}$$

Rozdíl Gaussovských rozostření (DoG)

Rozdíl dvou funkcí daných Gaussovým vztahem s různým parametrem σ se obvykle používá k aproximaci $LoG(x, y)$. Tento filtr oproti filtru LoG však nabízí více možností parametrizace. Vztah lze snadno odvodit, jak jen naznačeno v (2.51) (výpočet byl proveden programem Maple), ale lze jej nalézt i v [29].

$$\begin{gathered} G1(x,y) = \frac{1}{\sqrt{2\cdot\pi\cdot\sigma^2}} \cdot e^{-\left(\frac{x^2+y^2}{2\cdot\sigma^2}\right)} \quad G2(x,y) = \frac{1}{\sqrt{2\cdot\pi\cdot(k\cdot\sigma)^2}} \cdot e^{-\left(\frac{x^2+y^2}{2\cdot(k\cdot\sigma)^2}\right)} \\ DoG(x,y) = G1(x,y) - G2(x,y) = \cdots = \frac{k^2\cdot e^{-\left(\frac{x^2+y^2}{2\cdot\sigma^2}\right)} - e^{-\left(\frac{x^2+y^2}{2\cdot(k\cdot\sigma)^2}\right)}}{2\cdot\pi\cdot(k\cdot\sigma)^2} \end{gathered} \tag{2.51}$$

Výsledný vztah pro prvky matice $\boldsymbol{H}$ je tedy na (2.52).

$$\boldsymbol{H} = \left[h_{i,j}\right]_{i=-m\ldots m, j=-n\ldots n} = \frac{k^2\cdot e^{-\left(\frac{i^2+j^2}{2\cdot\sigma^2}\right)} - e^{-\left(\frac{i^2+j^2}{2\cdot(k\cdot\sigma)^2}\right)}}{2\cdot\pi\cdot(k\cdot\sigma)^2} \tag{2.52}$$

2.7 Geometrické transformace

Geometrické transformace slouží k úpravě geometrie obrazu. Běžným příkladem geometrických transformaci jsou operace jako například zvětšování a zmenšování obrazu, otáčení, či jeho zkosení. Základem je takzvaná mapovací funkce (2.53), která přiřazuje každému bodu obrazu $\boldsymbol{p} = [x, y]$ nové souřadnice. Výsledný bod má tedy souřadnice $\boldsymbol{p}' = [x', y']$. Tento proces je formálně popsán na (2.54) a (2.55) [24].

$$T = \mathbb{R}^2 \to \mathbb{R}^2 \tag{2.53}$$

$$\boldsymbol{p}' = T(\boldsymbol{p}) \tag{2.54}$$

$$x' = T_x(x, y) \quad y' = T_y(x, y) \tag{2.55}$$

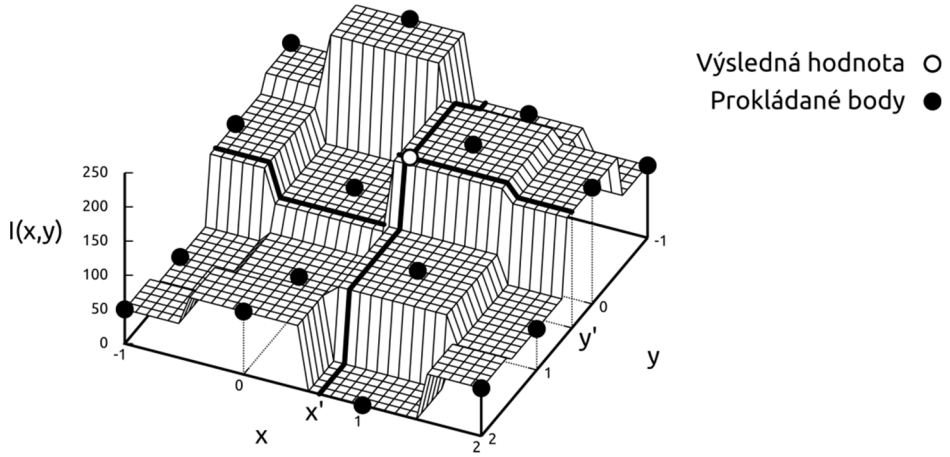
Pro dosažení uspokojivých výsledků je třeba uvažovat spojité obrazové souřadnice definované na obr. 2.2. V takto uvažovaném souřadném systému je hodnota pixelu uložena přesně v jeho středu (pokud se tedy dopustíme zjednodušení a budeme pixel považovat za malý čtverec), ale my potřebujeme znát odhad hodnoty i mimo střed pixelu, k čemuž slouží interpolační techniky popsané v následující kapitole.

2.7.1 Interpolace

Abychom získali odhad hodnoty na souřadnici $[x, y]$, je třeba zavést funkci, která bere v úvahu hodnoty pixelů v okolí dané souřadnice. Jednotlivé interpolační metody se navzájem liší velikostí tohoto okolí. Obecně platí, že velikost tohoto okolí zvyšuje výpočetní náročnost, ale zlepšuje kvalitu interpolace.

Nejbližší soused (Nearest Neighbour)

Interpolace metodou nejbližšího souseda pokládá hodnotu na souřadnicích $[x, y]$ rovnou přesně nejbližší známé hodnotě. V námi definovaném souřadném systému spočívá v použití hodnoty pixelu uložené na celočíselných souřadnicích $[X, Y]$, které dostaneme použitím celé části reálných souřadnic $[x, y]$. Tato interpolační metoda je nejjednodušší a nejrychlejší, ovšem poskytuje značně neuspokojivé výsledky. Výsledná interpolační funkce je nespojitá, příklad jejího průběhu je na obr. 2.5. Pro snadné porovnání interpolačních metod je zde však použit jiný souřadný systém.



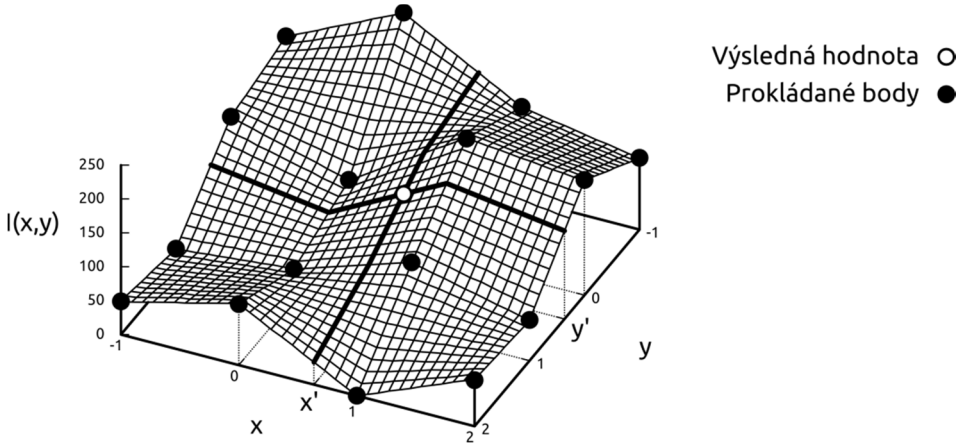
Obr. 2.5 Průběh interpolační funkce pro metodu nejbližší soused.

Bilineární interpolace

Bilineární interpolaci získáme rozšířením známé lineární interpolace do dvourozměrného prostoru, k výpočtu hodnoty budeme tedy potřebovat hodnoty čtyř bodů. Bilineární interpolace je dána formálně rovnicí (2.56), kde $a_{i,j}$ představuje čtyři neznámé koeficienty, které určíme z hodnot okolních bodů.

$$I(x, y) = \sum_{i=0}^{1} \sum_{j=0}^{1} a_{i,j} \cdot x^i \cdot y^j = a_{0,0} + a_{1,0} \cdot x + a_{0,1} \cdot y + a_{1,1} \cdot x \cdot y \tag{2.56}$$

Výsledná interpolační funkce je spojitá, ovšem má nespojité první derivace. Interpolace touto metodou je rychlá a poskytuje uspokojivé výsledky. Příklad průběhu interpolační funkce ukazuje obr. 2.6.



Obr. 2.6 Průběh bilineární interpolační funkce.

Pro určení hodnot neznámých koeficientů musíme řešit soustavu rovnic (2.57), kterou dostaneme z okrajových podmínek. Hodnoty $p_{i,j}$ představují známé hodnoty.

$$\begin{matrix} I(0,0) = a_{0,0} = p_{1,1} & I(1,0) = a_{0,0} + a_{1,0} = p_{2,1} \\ I(0,1) = a_{0,0} + a_{0,1} = p_{1,2} & I(1,1) = a_{0,0} + a_{1,0} + a_{0,1} + a_{1,1} = p_{2,2} \end{matrix} \tag{2.57}$$

Řešení této soustavy rovnic bylo provedeno s pomocí softwaru Maple, vztahy pro jednotlivé koeficienty jsou uvedeny v (2.58).

$$\begin{matrix} a_{0,0} = p_{1,1} & a_{1,0} = -p_{1,1} + p_{2,1} \\ a_{0,1} = -p_{1,1} + p_{1,2} & a_{1,1} = p_{1,1} - p_{1,2} - p_{2,1} + p_{2,2} \end{matrix} \tag{2.58}$$

Bikubická interpolace.

Bikubická interpolace je dána rovnicí (2.59). Oproti bilineární interpolaci vyžaduje hodnoty šestnácti okolních bodů, což odpovídá šestnácti neznámým koeficientům $a_{i,j}$.

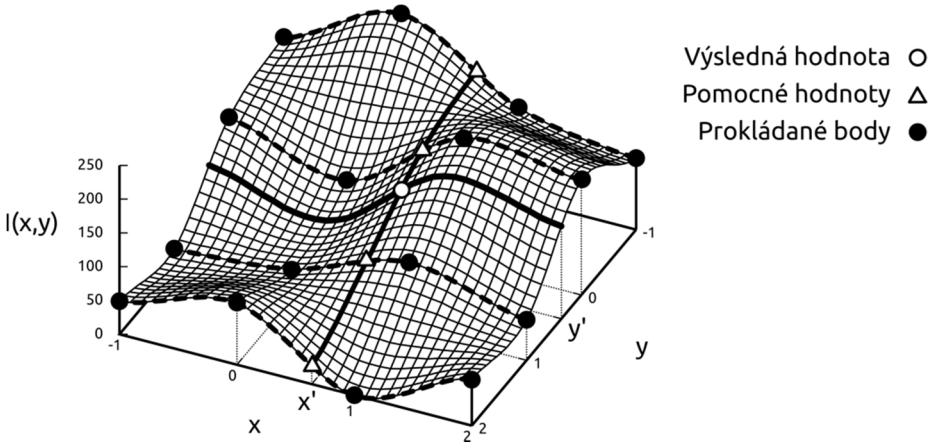
$$I(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{i,j} \cdot x^i \cdot y^j \tag{2.59}$$

Použití tohoto přístupu však není výpočetně vhodné, výpočet lze však, jak je uvedeno v [24], rozdělit na dva kroky, kdy se provádí pouze kubická interpolace, která je dána rovnicí (2.60), má jen 4 neznámé koeficienty, které dostaneme řešením soustavy (2.61) dané z okrajových podmínek, kde $p_0 = I(-1)$ a $p_3 = I(2)$. V prvním kroku interpolujeme pouze řádky, v kroku druhém interpolujeme sloupec, který je dán výsledky interpolace řádků.

$$I(x) = \sum_{i=0}^{3} a_i \cdot x^i = a_0 + a_1 \cdot x + a_2 \cdot x^2 + a_3 \cdot x^3 \tag{2.60}$$

$$I(0) = a_0 = p_1 \qquad I(1) = a_0 + a_1 + a_2 + a_3 = p_2$$
$$\frac{dI(x)}{dx}\Big|_{x=0} = a_1 = \frac{p_2 - p_0}{2} \qquad \frac{dI(x)}{dx}\Big|_{x=1} = a_1 + 2 \cdot a_2 + 3 \cdot a_3 = \frac{p_3 - p_1}{2} \tag{2.61}$$

Interpolační funkce daná kubickou (a také bikubickou) interpolací je spojitá a má také spojité první derivace. Příklad průběhu bikubické interpolační funkce ukazuje obr. 2.7, včetně znázornění průběhu výpočtu.



Obr. 2.7 Průběh bikubické interpolace – nejprve interpolujeme podle řádků (naznačeno čárkovaně), poté interpolujeme sloupec pomocných hodnot.

Ostatní interpolační metody

Existuje množství dalších interpolačních metod, ty však nebyli v rámci této práce implementovány.

2.7.2 Lineární transformace

Lineární transformace jsou speciální podskupinou geometrických transformací, kdy mapující funkce $\boldsymbol{T}$ je lineární. Do této skupiny transformací patří běžně používané transformace, jako je posun, otočení, změna velikosti, zkosení, ale i projektivní transformace. Výhodou této skupiny transformací je, že při zavedení homogenních souřadnic (potřebná teorie je popsána například v [24]) je lze popsat jednoduše maticovým počtem, jak je naznačeno v (2.62), kdy transformace je jednoznačně určena maticí $\boldsymbol{M}$, x a y představuje souřadnice v obraze.

$$\boldsymbol{p'_h} = \boldsymbol{M} \cdot \boldsymbol{p_h} = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{2.62}$$

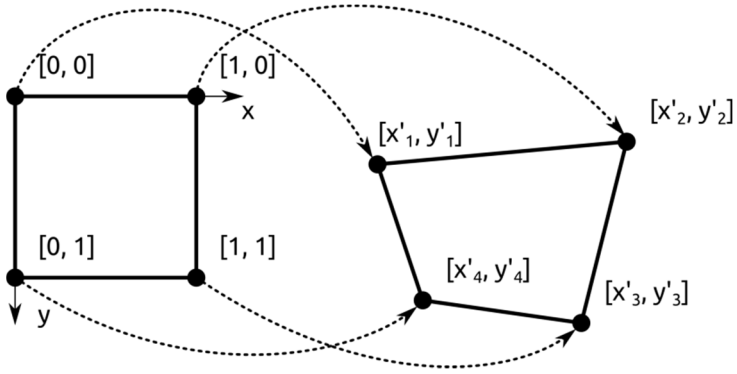
Jak je uvedeno v [13], lineární transformace lze jednoduše skládat vynásobením jejich matic, pokud chceme na obraz například aplikovat transformace určené maticemi $\boldsymbol{M_1}$, $\boldsymbol{M_2}$ a $\boldsymbol{M_3}$ v tomto pořadí, nemusíme provádět jednotlivé transformace postupně, ale stačí vynásobit transponované matice transformací v pořadí, v jakém jsou apliko-

vány, jak je naznačeno v rovnici (2.63), a poté aplikovat na obraz transformaci určenou výslednou maticí $\boldsymbol{M}$.

$$\boldsymbol{M}^T = \boldsymbol{M_1}^T \cdot \boldsymbol{M_2}^T \cdot \boldsymbol{M_3}^T \tag{2.63}$$

Projektivní transformace

V rámci řešení diplomové práce byla implementována nejobecnější lineární transformace, která mapuje obecný čtyřúhelník na jiný čtyřúhelník. Postup pro určení matice transformace je převzatý z [24].



Obr. 2.8 Transformace jednotkového čtverce na libovolný čtyřúhelník.

Postup se skládá z několika kroků. V prvním kroku určíme matici $\boldsymbol{M_1}$, která popisuje transformaci výchozího čtyřúhelníku na jednotkový čtverec. Poté určíme matici $\boldsymbol{M_2}$, která transformuje tento jednotkový čtverec na konečný čtyřúhelník. Pro určení matice $\boldsymbol{M_1}$ využijeme známého předpisu pro matici transformující jednotkový čtverec na libovolný čtyřúhelník, který je uveden v (2.64) [23] a faktu, že opačná transformace je určena maticí k této matici inverzní, což je zřejmé z podstaty homogenních souřadnic. Matici $\boldsymbol{M_2}$ určíme přímo dle (2.64). Význam souřadnic je zřejmý z obr. 2.8. Výslednou matici transformace určíme složením dle (2.63).

$$m_{3,1} = \frac{(x_1' - x_2' + x_3' - x_4') \cdot (y_4' - y_3') - (y_1' - y_2' + y_3' - y_4') \cdot (x_4' - x_3')}{(x_2' - x_3') \cdot (y_4' - y_3') - (x_4' - x_3') \cdot (y_2' - y_3')}$$

$$m_{3,2} = \frac{(y_1' - y_2' + y_3' - y_4') \cdot (x_2' - x_3') - (x_1' - x_2' + x_3' - x_4') \cdot (y_2' - y_3')}{(x_2' - x_3') \cdot (y_4' - y_3') - (x_4' - x_3') \cdot (y_2' - y_3')} \tag{2.64}$$

$$m_{1,1} = x_2' - x_1' + m_{3,1} \cdot x_2' \quad m_{1,2} = x_4' - x_1' + m_{3,2} \cdot x_4' \quad m_{1,3} = x_1'$$
$$m_{2,1} = y_2' - y_1' + m_{3,1} \cdot y_2' \quad m_{2,2} = y_4' - y_1' + m_{3,2} \cdot y_4' \quad m_{2,3} = y_1'$$

3 Operační systém Android

Android je moderní operační systém speciálně určený pro mobilní zařízení, ale v žádném případě se neomezuje jen na ně. Dle [30] lze Android charakterizovat jako systém, který je:

- **Robustní**
  - Jedná se o robustní, bezpečnou a jednoduše upgradovatelnou platformu, která nabízí dobře definovaná rozhraní pro vývojáře, kteří tak mohou vytvářet aplikace dobře integrované do celého systému. Android také nabízí certifikační programy pro výrobce zařízení, kteří tak mohou vyrábět vysoce kompatibilní hardware.

- **Otevřený**
  - Celá platforma je vyvíjena a poskytována jako open source software pod Apache licencí, jenž umožňuje zejména výrobcům zařízení přidávat či ubírat funkce systému a přijít tak na trh s produktem, který se odlišuje od ostatních.
  - Další významnou vlastností je to, že Android nerozlišuje mezi aplikacemi poskytnutými výrobcem zařízení a aplikacemi od ostatních vývojářů – všechny aplikace mají rovnocenný přístup k zařízení.

- **Zdarma**
  - Vývojáři aplikací nemusí platit žádné licenční, registrační či jiné poplatky. Veškeré nástroje potřebné pro vývoj aplikací včetně dokumentace, jsou poskytovány zdarma a jsou dostupné pro většinu hlavních operačních systémů.

3.1 Historie

Historie operačního systému Android se začala psát roku 2003, kdy stejnojmenný startupový projekt začal vyvíjet mobilní operační systém. Tento projek byl později,

v roce 2005, odkoupen společností Google, jako součást její strategie pro vstup na trh s mobilními zařízeními. [30]

Google převzal celý projekt s cílem vytvořit otevřenou platformu pro vývoj mobilních aplikací, která by byla postavena na technologiích Google, které budou moci využívat jak uživatelé, tak vývojáři. V roce 2007 bylo založeno konsorcium Open Handset Alliance (OHA) s cílem vytvořit systém, který by představoval otevřený standard pro mobilní zařízení. Svoji činnost OHA zahájilo právě ohlášením systému Android. OHA nyní sdružuje desítky společností po celém světě, mezi které patří například [31]:

- **Mobilní operátoři**
  - Vodafone, T-Mobile, Telefónica...
- **Výrobci výpočetní techniky**
  - Samsung, Asus, HTC, Garmin...
- **Výrobci polovodičových součástek**
  - Intel, Qualcom, ARM...
- **Další společnosti z oblasti IT**
  - Google, eBay, Accenture...

První beta verze OS Android byla vydána roku 2007 a systém se od té doby neustále vyvíjí. Pokud pomineme rychlý počáteční vývoj, za největší změny lze označit zavedení značně přepracovaného API ve verzi 3.0, které zavádí podporu pro počítačové tablety, které se oproti chytrým telefonům vyznačují zejména velkou obrazovkou s vyšším rozlišením. Android 3.x však byl určen pouze pro počítačové tablety, jednotný systém tedy zavedl až příchod systému Android 4.0.1 [30].

3.2 Nasazení

V současnosti je Android operační systém, který není zdaleka limitován na chytré telefony. Zařízení, na kterých můžeme nalézt OS Android, jsou například chytré telefony, tablety, čtečky, elektronických knih, netbooky, MP4 přehrávače a chytré televize.

Tato zařízení se navzájem velmi liší, a to v mnoha aspektech. Jedním z rozdílů je způsob ovládání, kdy v případě chytrých telefonů a tabletů je k dispozici ve většině případů dotyková obrazovka s podporou jedno-, ale i vícedotykového ovládání, zatímco jinde je k dispozici standardní klávesnice, dálkové ovládání, touchpad či připojitelná myš.

Dalším podstatným rozdílem je velikost a rozlišení obrazovky, která se pohybuje od několika palců s rozlišením 320×480, jako je tomu například u levných telefonů,

přes 10″ tablety, až po televize s více než 40″ obrazovkou a full HD rozlišením. Platforma si navíc musí poradit s nejrůznějšími kombinacemi velikosti a rozlišení, protože uživatelské rozhraní musí vypadat jinak na full HD televizoru s 40″ obrazovkou a full HD telefonu s 5″ obrazovkou.

V neposlední řadě se výrazně liší výpočetní výkon a dostupný hardware. Zařízení bývají osazena nejrůznějšími procesory s různou architekturou a různými instrukčními sadami. Systém tedy musí být schopný pracovat jak na telefonu s jednojádrovým 800 MHz procesorem a 512 MB RAM, tak na zařízení s čtyřjádrovým procesorem o taktu 1,9 GHz a 2 GB RAM. Dále je omezena dostupnost některých HW senzorů, kterými je zařízení osazeno - jedná se třeba o akcelerometr, kompas, GPS chip, kameru, a mnohé další.

3.3 Architektura OS

Operační systém Android lze rozdělit na pět oblastí, které jsou uspořádány do čtyř vrstev, jak je naznačeno na obr. 3.1. Zde následuje popis jednotlivých částí, který je kompilací informací uvedených na oficiálních stránkách projektu [33] a v literatuře [34], [35]

- **Linuxové jádro**
  - Linuxové jádro (verze 2.6) poskytuje základní vrstvu abstrakce mezi aplikací a hardwarem. Zde jsou obsaženy ovladače pro hardware a zde také dochází ke správě paměti a spuštěných procesů. Android spoléhá na Linuxové jádro také v otázce zabezpečení.

- **Knihovny**
  - Knihovny běží přímo nad Linuxovým jádrem a starají se například o grafické operace a zpracování multimédií, přístup ke sdílené SQLite databázi...

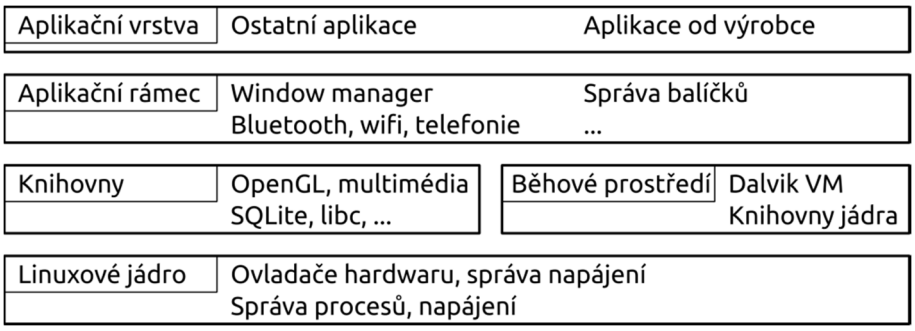
- **Běhové prostředí Android**
  - Tato část se stará o to, aby OS Android byl něco více, než jen mobilní implementací Linuxu a skládá se ze dvou částí:

    - **Dalvik VM**
      - Jedná se o virtuální stroj, který je speciálně optimalizovaný tak, aby více jeho instancí mohlo běžet současně. Navzdory zažitému mínění se nejedná o tradiční Java VM. Ikdyž jsou aplikace pro Android psány v Javě, soubory zkompilované překladačem Javy jsou dále přeloženy pro Dalvik VM a teprve potom jsou spustitelné.
      - Dalvik VM se spouští pro každou běžící aplikaci samostatně v samostatném procesu a pro nízkoúrovňové operace jako je správa paměti a vláken spoléhá na Linuxové jádro.

- **Knihovny jádra**
  - Tyto knihovny se starají o to, aby Dalvik VM poskytoval většinu funkcí, které poskytuje standardní Java VM. Také poskytují funkcionalitu specifickou pro Android.

- **Aplikační rámec**
  - Aplikační rámec poskytuje obecný přístup k hardwaru zařízení a stará se o uživatelské rozhraní aplikace.

- **Aplikační vrstva**
  - V aplikační vrstvě běží jak výrobcem poskytnuté aplikace, tak aplikace třetích stran.



Obr. 3.1 Architektura OS Android.

3.4 Aplikace v OS Android

Aplikace pro OS Android se nazývají Aktivity (Activity). Základní filozofie jejich práce se poněkud liší od práce aplikací na PC, kde uživatel běžně pracuje s několika spuštěnými aplikacemi, které dle potřeby přepíná, zapíná a vypíná. Situace kdy jedna aplikace spouští jinou aplikaci od úplně jiného vývojáře je méně obvyklá.

V případě zařízení s OS Android je situace jiná. Jednotlivé aplikace jsou více integrovány do systému a mají propracovaný životní cyklus. Uživatel nemá ve většině případů (bez použití speciálních nástrojů) kontrolu nad tím, kdy, a zda vůbec, bude aplikace ukončena. Aplikace obvykle běží na pozadí nebo je operační systém udržuje v paměti pro případ rychlejšího opětovného spuštění. Uživatel tedy běžně aplikace neukončuje, ale pouze je opouští, přičemž k ukončení dojde až v okamžiku, kdy to operační systém uzná za vhodné – většinou tomu tak je v případě nedostatku volné operační paměti.

Výhodou tohoto přístupu je, že systém se uživateli jeví pružnější – aplikace startují rychle. Nevýhodou jsou zvýšené požadavky na vývojáře, který musí vývoj aplikace přizpůsobit jejímu životnímu cyklu, zejména s ohledem na ukládání dat, protože pokud uživatel opustí aplikaci s úmyslem se k ní opět vrátit, operační systém může

kdykoliv rozhodnout o jejím ukončení, které může být i násilné, pokud nastane vážný nedostatek systémových prostředků. Aplikace může být tedy kdykoliv, když není v přímé interakci s uživatelem, ukončena systémem bez jakéhokoliv varování.

Dalším specifikem pro aplikace v OS Android je omezení velikosti paměti, kterou může aplikace alokovat pro své účely. Velikost této paměti závisí na výrobci zařízení a mezi zařízeními se výrazně liší. Vzhledem k použití Dalvik VM, kde prostředky uvolňuje automaticky garbage collector, může být toto omezení problém, zejména při časté alokaci velkých objektů, jakými jsou například bitmapy. Nutno ovšem poznamenat, že toto omezení se vztahuje pouze na alokace, které probíhají na heapu VM, pokud aplikace volá nativní knihovny, paměť alokovaná těmito knihovnami na fyzickém heapu se do limitu nezapočítává.

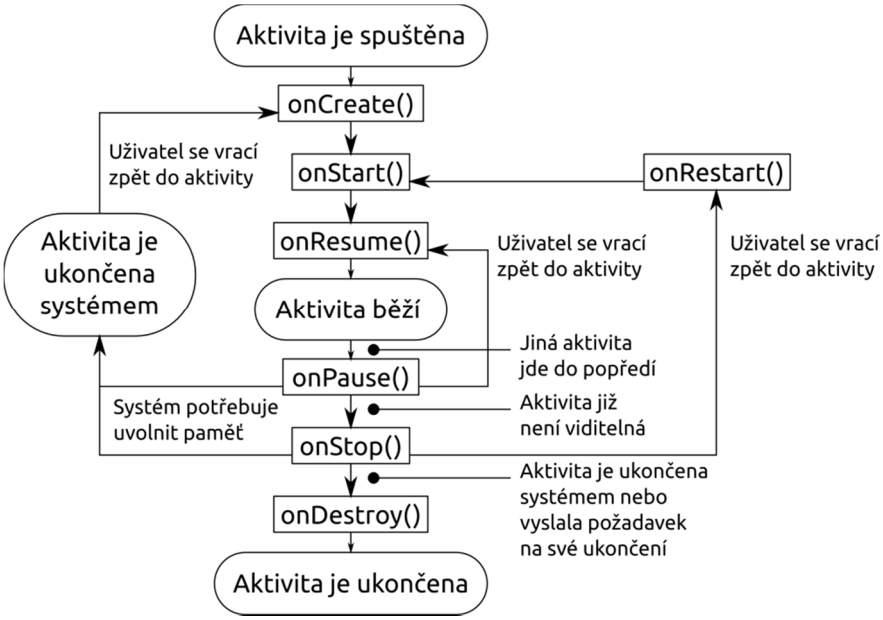
Základním mechanismem, který umožňuje vzájemnou kooperaci jednotlivých aplikací, je implementace tzv. „Záměrů“ (Intents). Jedná se o zprávy, které posílá aplikace systému s požadavkem na provedení nějaké operace. Tyto požadavky mohou být jak konkrétní – spuštění konkrétní aplikace či služby na pozadí, tak naprosto obecné – například aplikace žádá o odeslání e-mailu. V případě obecných požadavků systém sám zvolí aplikaci, která je schopná požadovanou akci provést. Pokud je takových aplikací více, dá uživateli na výběr. Může ale nastat i situace, kdy není k dispozici žádná aplikace, která by mohla požadavek splnit.

Tento systém dává vývojáři možnost vyvíjet aplikace určené jen na provedení konkrétního specializovaného úkolu a pro běžné záležitosti využívat služeb systémových aplikací dodaných výrobcem zařízení. Příkladem tohoto přístupu je například využití předinstalovaného e-mailového klienta pro odeslání zpětné vazby autorovi aplikace. Tento mechanismus ovšem umožňuje vývojáři také vytvořit aplikaci, která nahradí výchozí systémovou aplikaci – například lepšího e-mailového klienta, který bude nahrazovat původního a kterého budou moci využívat ostatní aplikace, pro splnění požadované akce.

Mechanismus „Záměrů“ je asynchronní a požadovaná akce nemusí, ale může vracet výsledek, nebo jakoukoliv relevantní informaci o svém provedení.

3.5 Životní cyklus aplikace

Jak již bylo zmíněno, aplikace má jistý životní cyklus, jehož popis shrnuje obr. 3.2. Obrázek je volně převzatý z [36], kde je možné nalézt další informace k danému tématu. Zde budou zmíněny jen nejdůležitější etapy životního cyklu. Každá etapa životního cyklu je představována konkrétní metodou hlavní třídy aplikace, která je volána systémem.



Obr. 3.2 Životní cyklus aktivity

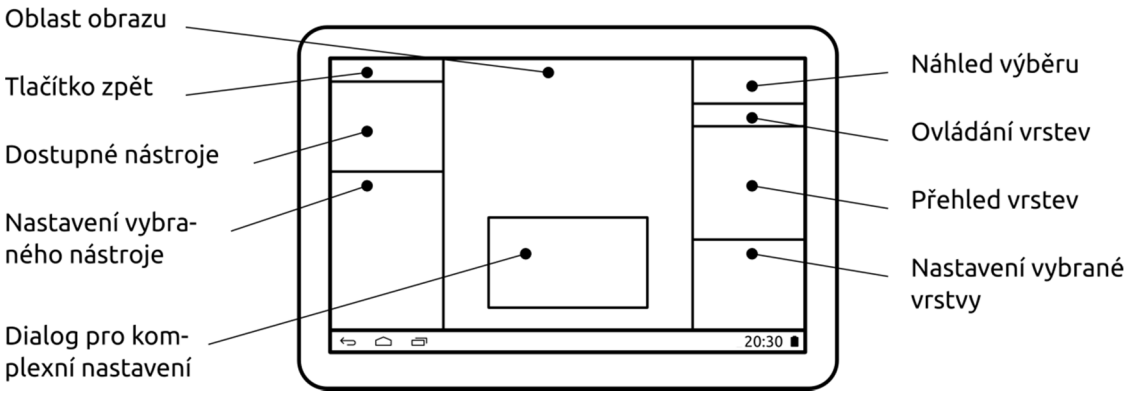
Nejdůležitější část životního cyklu je skryta v metodě `onCreate()`, která musí být implementována. Tato metoda je volána vždy při spuštění aplikace a obvykle zde probíhá veškerá inicializace, a to včetně inicializace uživatelského rozhraní aplikace. Další významnou částí životního cyklu je metoda `onResume()`, která je volána v okamžiku, kdy aplikace začíná interagovat s uživatelem. Toto je vhodné místo pro spouštění pracovních vláken aplikace. Opakem metody `onResume()` je metoda `onPause()`, která je volána v okamžiku, kdy uživatel již neinteraguje s aplikací. Toto je obvykle první náznak toho, že uživatel aktivitu opouští, a je to jediná metoda, u které je garantováno, že proběhne při ukončování aplikace. V této metodě je vhodné pozastavit práci na pozadí, uvolnit systémové zdroje a uložit data, která uživatel předpokládá, že budou do jeho návratu uložena, protože aplikace může být mezitím zcela ukončena operačním systémem.

4 Popis navržené aplikace

Tato kapitola se zabývá popisem aplikace jednak z pohledu koncového uživatele – shrnuje uživatelské rozhraní a dostupné nástroje – ale také zdůvodňuje jednotlivá rozhodnutí, která bylo během návrhu nutno učinit. Pro popis uživatelského rozhraní jsou z důvodů názornosti použity schematické obrázky. Větší množství snímků obrazovky z výsledné aplikace je možno nalézt v příloze, kde je k dispozici ukázka jejího použití v reálných situacích.

4.1 Uživatelské rozhraní

Na obr. 4.1 Je naznačeno základní schéma uživatelského rozhraní. Rozvržení jednotlivých ovládacích prvků vychází z obvyklého rozvržení používaného jak v programech pro PC, tak pro jiné platformy. Nalevo je panel s nástroji a napravo panel s vrstvami. Oba panely je možné klepnutím na jejich hranici minimalizovat.



Obr. 4.1 Základní schéma uživatelského rozhraní aplikace.

Nahoře na panelu nástrojů je umístěno tlačítko zpět, které umožňuje vrátit poslední modifikaci obrazových dat.

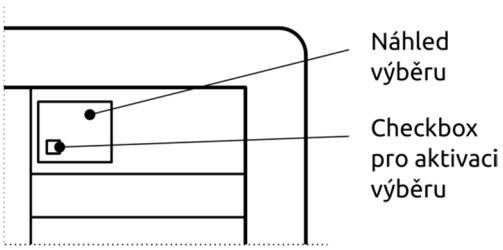
Po vybrání nástroje se pod seznamem dostupných nástrojů objeví jeho nastavení. Pokud je nastavení příliš komplexní nebo vyžaduje více prostoru, než mu může poskytnout vyhrazený prostor, objeví se nastavení uprostřed obrazovky v místě označeném jako Dialog pro komplexní nastavení. Toto je případ například nástroje křivky, který pro pohodlnou editaci vyžaduje více prostoru, ale také případ nástrojů, které ovlivňují celý obraz – v tomto případě je výhodné, aby bylo viditelné pouze jejich nastavení a uživatel měl možnost skrýt postranní panely.

Pravý panel s vrstvami poskytuje celkový přehled o obraze a také ovládací prvky sloužící k modifikaci jeho struktury. Úplně nahoře je náhled výběru. Systém, s jakým pracuje výběr, se poněkud liší od systému používaného v desktopových aplikacích a bude vysvětlen dále.

Pod náhledem výběru je ovládání vrstev. Zde je přístup k vytváření vrstev, ale také k nastavení průhlednosti vybrané vrstvy, protože toto nastavení je společné pro všechny druhy vrstev. Pod ním je umístěn panel s přehledem vrstev, který slouží k jejich správě. Úplně vespod je nastavení vybrané vrstvy, pokud je k dispozici.

4.2 Výběr

Výběr obrazových bodů je implementován pomocí jednokanálové bitmapy jako maska. Výběr tedy může mít měkké okraje a v kombinaci s nástrojem Alpha brush, který bude popsán dále, umožňuje rychlý a přesný výběr i složitých oblastí s proměnlivým krytím. Výběr lze libovolně aktivovat a deaktivovat, přičemž poslední použitý výběr zůstává vždy uložen pro další použití. Detail ovládání je naznačen na obr. 4.2.

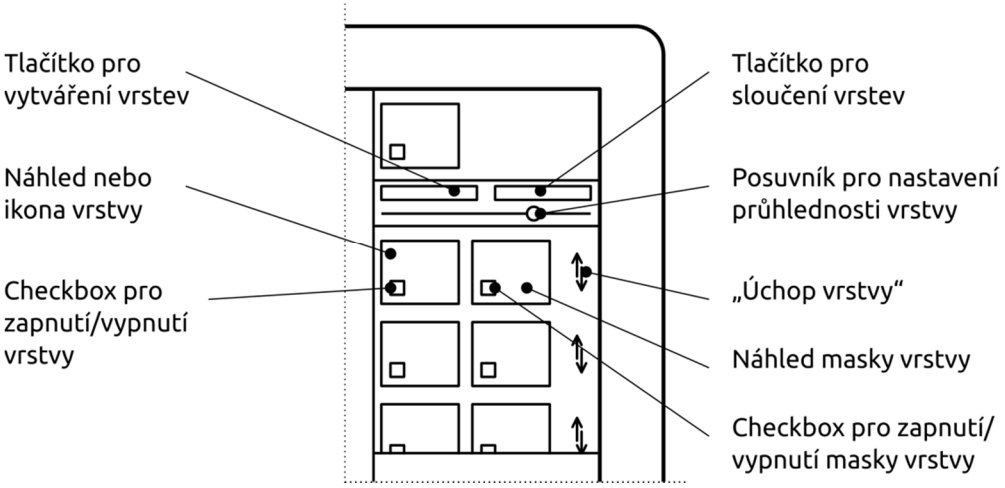


Obr. 4.2 Schéma ovládání výběru.

4.3 Práce s vrstvami

Pro navržený program je stěžejní koncept práce s vrstvami. Každou vrstvu lze zapnout, vypnout, nastavit její průhlednost a také jí přiřadit masku (ve formě jednokanálové bitmapy), kterou lze libovolně aktivovat a deaktivovat. Správa vrstev se realizuje pomocí ovládacích prvků na položce v přehledu vrstev, která reprezentuje danou vrstvu. Odpovídající elementy uživatelského rozhraní ukazuje obr. 4.3. Úchop vrstvy

po klepnutí zobrazí nabídku pro duplikování a odstranění vrstvy a po dlouhém přidržení umožní její přesun nahoru a dolů přetažením.



Obr. 4.3 Schéma popisující prvky uživatelského rozhraní sloužící k ovládání vrstev.

Náhled vrstvy a náhled masky (včetně masky výběru) zobrazí po dlouhém stisku nabídku, která umožňuje provádět s rastrovými daty daného elementu další operace. Jejich přehled udává tab. 4.1.

Tab. 4.1 Přehled operací s rastrovými daty vybraného elementu.

Operace	Obraz	Maska	Popis
Kopírování	✓	✓	Zkopíruje rastrová data elementu do schránky.
Vložení	✓	✓	Vloží rastrová data ze schránky do daného elementu.
Smazání	✓	✓	Smaže rastrová data elementu.
Výplň	×	✓	Nastaví všechny pixely masky na maximální hodnotu (stejný význam jako příkaz „vyber vše“).
Inverze	×	✓	Invertuje hodnotu všech pixelů (vytvoří „doplněk výběru“).
Přičtení	✓	✓	Přičte k datům elementu hodnotu rastrových dat ze schránky (analogie k „sloučení výběru“). Pokud je voláno na obraze, ovlivní pouze alpha kanál. Ve schránce musí být jednokanálová data.
Odečtení	✓	✓	Odečte od dat elementu hodnotu rastrových dat ze schránky (analogie k „odečtení výběru“). Pokud je voláno na obraze, ovlivní pouze alpha kanál. Ve schránce musí být jednokanálová data.
Prolnutí	✓	×	Prolne data ze schránky s daty obrazu v normálním režimu. Ve schránce musí být čtyřkanálová data.

Rastrová data lze takto kopírovat mezi jednotlivými vrstvami a maskami, a to buď jako celou bitmapu, nebo jen její část, která je určena maskou výběru, pokud je tato aktivní. Zapnutá maska výběru také určuje oblast, která bude požadovanou operací ovlivněna. Ze zřejmých důvodů lze kopírovat rastrová data masky pouze mezi maskami a data obrazu pouze mezi rastrovými vrstvami.

Koncept vrstvy je v podání navrženého programu značně abstraktní, zobecněný na většinu operací s obrazem. Jako vrstva tedy již nevystupuje jen bitmapa (jak je tomu u programu GIMP), ale také překrytí barvou či bodové operace (jak je tomu u programu Photoshop). Jako vrstvu lze však použít také jakýkoliv implementovaný filtr, což je dle dostupných informací v současné době unikátní funkce. Tento koncept umožňuje provádět s obrazem složité úpravy a prakticky vždy přitom nabízí možnost změnit nastavení operací provedených na začátku. Na rozdíl od vrácení se zpět a nutnosti celý proces opakovat, můžeme celý proces parametrizovat a modifikovat jednotlivé jeho kroky. Vzhledem k tomu, že jednotlivé druhy vrstev ve většině odpovídají nástrojům pro modifikaci obrazu, popis jejich funkce bude uveden až v sekci zabývající se právě nástroji. Zde je uveden pouze seznam dostupných vrstev s krátkým komentářem.

- **Rastrová vrstva**
  - Tato vrstva reprezentuje bitmapu, lze jí nastavit mód prolnutí. Na výběr je z módů popsaných v sekci 2.4.1, Alpha blending čtyřkanálových bitmap.

- **Vrstva barvy**
  - Vrstva představuje výplň konstantní barvou, její síla vynikne především v kombinaci s maskami vrstev. U této vrstvy lze nastavit barva a mód prolnutí.

- **Vrstvy bodových operací**
  - Jako vrstvu je možné použít libovolnou implementovanou bodovou operaci, k dispozici jsou bodové operace popsané v sekci 2.5, Bodové operace, tedy úprava jasu, kontrastu, gama korekce a křivky.

- **Vrstvy filtrů**
  - Jako vrstvu lze použít i libovolný implementovaný filtr, k dispozici jsou filtry popsané v sekci 2.6, Filtry.

4.4 Práce s maskami a výběrem

Cílem návrhu bylo, aby práce s maskami byla co nejpřirozenější a nejpohodlnější, proto je způsob práce s maskami vrstev bližší práci v programu GIMP než Photoshop, kde je jejich přiřazování a úprava poněkud těžkopádnější.

Pro zahájení úpravy masky vrstvy stačí kliknout na ikonu reprezentující její náhled. Z důvodů efektivity je maska vrstvy vytvořena až při prvním použití. Nyní již můžeme masku vrstvy jednoduše upravovat nástrojem Alpha brush.

Vybraná oblast je zvýrazněna poloprůhlednou červenou barvou, jak je zvykem při práci s rychlou maskou ve Photoshopu. Právě tato vybraná oblast bude z vrstvy viditelná, pokud se jedná o rastrovou vrstvu či vrstvu barvy. Pokud se jedná o vrstvu úprav, této oblasti se budou požadované úpravy týkat. Vzhledem k tomu, že maska vrstvy není pouze dvouhodnotová, ale nabízí plynulý výběr, lze takto kontrolovat i intenzitu zvolených úprav na daném místě obrazu.

4.5 Popis nástrojů a funkcí

4.5.1 Operace se soubory (File operations)

Tento nástroj slouží k vytváření, otevírání, ukládání a exportu souborů. Ovládání je značně intuitivní, o možnostech exportu detailněji pojednává následující sekce, která poskytuje přehled o dostupných formátech a jejich možnostech.

4.5.2 Navigace (Navigation)

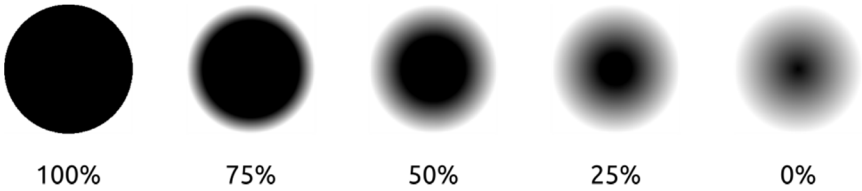
Tento nástroj slouží k ovládání posunu a zvětšení/zmenšení zobrazení obrazu. Program nabízí pro tyto funkce standardní ovládání gesty – posun tažením prstem, posun a zvětšování/zmenšování tažením dvěma prsty. Dále nástroj nabízí přednastavené možnosti pro přizpůsobení velikosti obrazu obrazovce a vycentrování obrazu při nastaveném měřítku 1:1, kdy jeden pixel displeje odpovídá jednomu pixelu obrazu.

Na rozdíl od ostatních programů pro úpravu obrazu na platformě Android nabízí navržený software navigaci jako samostatný nástroj, protože přístup, kdy tažení jedním prstem je používáno ke kreslení, zatímco dvoudotykové gesto je využíváno k posunu má nežádoucí vedlejší účinky. Umožňuje uživateli neúmyslně modifikovat obraz, pokud se nedotkne oběma prsty zároveň, čehož si nemusí ani všimnout. Aby k této situaci nedošlo, musí se uživatel příliš soustředit na samotné posouvání, místo toho, aby se mohl plně soustředit na proces úprav obrazu. Dále je nevýhodou neustálé zvětšování/zmenšování obrazu, ke kterému dochází při dvoudotykovému gestu, které nemusí být pro uživatele žádoucí. Tento přístup navíc umožňuje rychlejší reakci na dotyk uživatele při kreslení ostatními nástroji – není třeba prvně detekovat gesto.

4.5.3 Štětec (Brush)

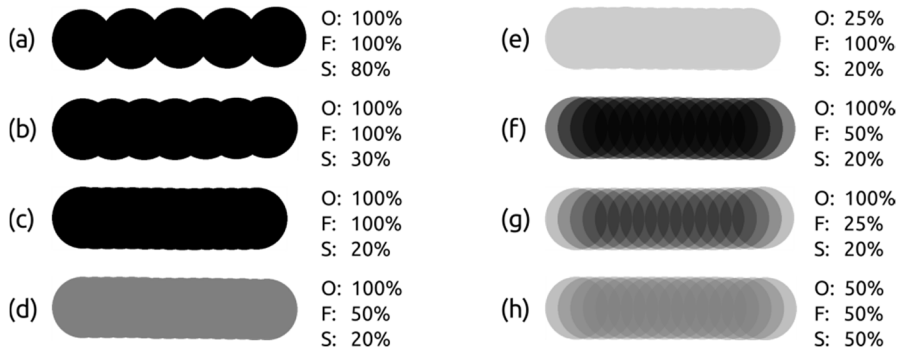
Jedná se o dobře známý nástroj prakticky ze všech programů pro práci s grafikou. Implementován byl standardní kulatý štětec, u kterého je možné nastavit jak barvu, tak mód prolnutí, ve kterém bude daná barva aplikována.

Dále je plynule nastavitelná jak velikost štětce (size), tak jeho tvrdost (hardness). Vliv těchto nastavení ilustruje obr. 4.4. Velikost představuje průměr štětce v pixelech, tvrdost pak udává, kolik procent z poloměru štětce bude použito pro postupný přechod z plné do průhledné barvy.



Obr. 4.4 Vliv nastavení tvrdosti na hrot štětce.

Pro nastavení stopy je dále nastavitelné krytí (opacity), tok (flow) a mezery mezi otisky hrotu štětce (spacing). K dispozici živý náhled, který ukazuje, jak se bude vybraný nástroj chovat. Vliv jednotlivých možností je patrný z obr. 4.5. Krytí udává maximální viditelnost stopy jako celku, zatímco tok udává průhlednost jednotlivých otisků hrotu štětce.



Obr. 4.5 Vliv jednotlivých nastavení stopy štětce. Značení je následující: krytí – O; tok – F; mezery otisku hrotu – S.

Tato nastavení nabízí uživateli dostatek prostoru, aby mohl zvolit nástroj vhodný jak pro jemné stínování, tak pro výplň velkých oblastí. Výběr jednotlivých parametrů je realizován pomocí posuvníků. Tam, kde je to vhodné, byla použita exponenciální stupnice (příkladem je velikost štětce, kdy při výběru velkého štětce obvykle nezáleží na tom, jestli je velikost na pixel přesná, ovšem tato přesnost volby je vyžadována u štětců malých.

4.5.4 Alpha štětec (Alpha brush)

Jedná se o štětec, jehož funkce se liší v závislosti na tom, jestli je používán na obraz či masku. Podobně jako u klasického štětce je zde nastavitelná velikost a tvrdost hrotu štětce, krytí, tok a mezery stopy. Dále je na výběr ze dvou módů jeho použití – přičítání a odečítání, které pracuje na principu popsaném v sekci 2.4.2, Alpha blen-

ding s jednokanálovými bitmapami. V módu odečítání pracuje jako klasická guma jak pro masky tak obraz. V módu přičítání pracuje jako štětec při použití na masku, zatímco při použití na obraz vrací zpět oblasti umazané odečítáním.

4.5.5 Štětec bodových operací (Point op. brush)

Tento štětec slouží k lokální aplikaci všech implementovaných bodových operací, které jsou posané v sekci 2.5, Bodové operace. Hrot a stopa štětce jsou nastavitelné obvyklým způsobem.

4.5.6 Geometrické transformace (Geometric transformation)

Pod položkou nástroje geometrické transformace se ve skutečnosti skrývají tři nástroje.

Ořez obrazu

První umožňuje ořez obrazu. Ovládání je intuitivní – na obrazovce se objeví obdélník s uchopitelnými rohy, které určují ořez. Odříznutá oblast je ztmavena.

Zmenšení/zvětšení obrazu

Druhý nástroj umožňuje změnu velikosti obrazu – uživatel může obraz zvětšovat i zmenšovat jak volně, tak se zamknutým poměrem stran. K dispozici je několik interpolačních metod – jedná se o metody popsané v sekci 2.7.1, Interpolace. Při zmenšování je pro zamezení aliasingu prováděno odfiltrování vysokých frekvencí v obraze Gaussovským rozostřením s volitelnou silou. Uživatel tak může díky náhledu výsledného obrazu kontrolovat výsledek a u obrazů, které nejsou náchylné k negativním efektům aliasingu, vyhlazení potlačit. Naopak u obrazů, pro které je aliasing citlivé téma, může více odfiltrovat vysoké frekvence.

Projektivní transformace

Posledním nástrojem je projektivní transformace, která umožňuje provádět s vybranou vrstvou obecnou lineární transformaci, která je popsána v sekci 2.7.2, Lineární transformace. Transformace se ovládá pomocí vrcholů čtyřúhelníku zobrazeného na obrazovce, jejichž posun určuje posun odpovídajících obrazových bodů. Pro přehled zůstává na obrazovce zešedlý původní čtyřúhelník.

K dispozici je živý náhled, který lze zapnout/vypnout. Pro vykreslení náhledu je z důvodů rychlosti použita interpolace metodou nejbližší soused, při potvrzení se ovšem použije bikubická interpolace.

Tato funkce je zejména vhodná pro korekci perspektivy fotografie. Za tímto účelem je k dispozici přepínač deformace/opravy (Deform/Repair). Pokud uživatel pracuje v módu deformace, odpovídá přetažení vrcholů čtyřúhelníku deformaci, pokud je

však zapnut mód opravy, transformace je inverzní. Uživatel tedy zadává, jaký obecný čtyřúhelník se má transformovat na obdélník, naznačený zešedlým čtyřúhelníkem.

4.5.7 Filtr (Filter)

Nástroj filtr umožňuje použití jednoho z filtrů popsaných v sekci 2.6, Filtry, na vybranou vrstvu. Pokud je aktivní výběr, použije se jako maska operace, filtr tedy bude aplikován pouze na vybranou oblast. Nastavení jednotlivých filtrů je dáno jejich volitelnými parametry, jak byli popsány dříve.

Pro všechny filtry je k dispozici volitelný živý náhled.

4.5.8 Bodové operace (Point operation)

Tento nástroj pracuje podobně jako filtry – s jediným rozdílem – na vrstvu je aplikována vybraná bodová operace ze sekce 2.5, Bodové operace. Za zmínku stojí nástroj křivek, který umožňuje předepsat funkci určující bodovou operaci jako kubický spline. Uživatel zadává jednotlivé body, kterými spline prochází kliknutím do oblasti s křivkami. Dvojklik na existující bod jej odstraní. Je možné určit jak společnou křivku pro kanály RGB, tak křivku pro každý kanál zvlášť, včetně alpha kanálu.

4.6 Import a export dat

Aplikace zvládá import obrázků z galerie, které mají formát podporovaný platformou Android, tedy jpg a png. Do galerie umožňuje také export, a to v obou dříve zmíněných formátech, kdy u formátu jpg umožňuje nastavit úroveň komprese.

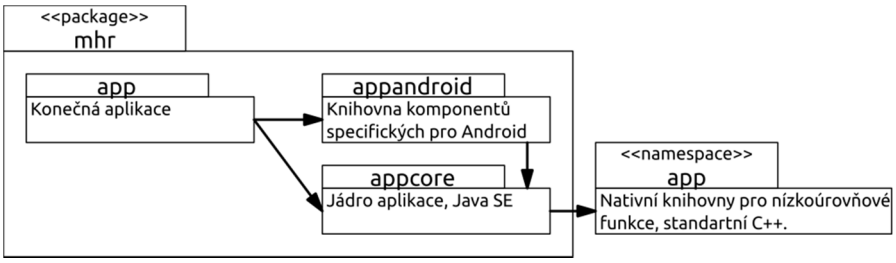
Výše zmíněné formáty souborů nedovolují uložit vrstvy obrazu. Tento nedostatek odstraňuje možnost uložit obraz, včetně vrstev, do operačním systémem vyhrazeného umístění. Každý takto uložený obraz má vlastní složku odpovídající názvu obrázku, který uživatel při ukládání zvolil. Tato složka obsahuje hlavní soubor xml s popisem obrazu a veškerá rastrová data pro jednotlivé vrstvy, masky vrstev a výběr ve formátu png, který je bezeztrátový a přesně zachová uživatelská data. Uživatel má přístup k takto uloženým datům, takže v případě potřeby může jednoduše získat přístup k obrazovým datům jednotlivých vrstev pro použití v jiné aplikaci.

5 Implementace

Aplikace byla implementována pomocí oficiálně doporučovaných nástrojů, tedy s využitím vývojového prostředí Eclipse a v programovacím jazyku Java. Vzhledem k výpočetní náročnosti operací spojených se zpracováním obrazu bylo nutné implementovat část aplikace jako nativní knihovnu v jazyce C++. Vývoj nativních aplikací pro platformu Android není doporučován, přesto existuje oficiální sada nástrojů pro tento účel určených, která se neustále rozrůstá.

5.1 Struktura aplikace

Základní struktura aplikace je naznačena na obr. 5.1. Architektura aplikace je zvolena tak, aby jednotlivé ucelené součásti byli pokud možno platformě nezávislé a portovatelné.



Obr. 5.1 Organizační schéma aplikace.

Všechny třídy na straně Javy jsou součástí balíčku `mhr`, který je rozdělen na tři základní části.

- **Balíček `app`**
  - Tento balíček obsahuje třídy samotné aplikace a využívá komponenty platformy Android a komponenty obsažené v balíčcích `appandroid` a `appcore.`

- **Balíček `appandroid`**
  - V balíčku jsou obsaženy třídy, které zapouzdřují funkcionalitu specifickou pro platformu Android. Jedná se jednak o vlastní části uživatelského rozhraní, ale také o třídy implementující rozhraní definovaná v balíčku `appcore`.

- **Balíček `appcore`**
  - Balíček využívá pouze funkcionality tříd dostupných ve standardní verzi Javy (Java SE), přístup k veškerým funkcím, které mohou být závislé na konkrétní platformě (především práce se soubory a grafickým uživatelským rozhraním) je realizován pomocí rozhraní, která je třeba pro danou platformu implementovat. V případě této aplikace jsou součástí balíčku `appandroid`.
  - Balíček má dvě základní funkce. Zaprvé zprostředkovává přístup k nízkoúrovňovým funkcím nativní knihovny pomocí JNI (Java Native Interface) a zadruhé se stará o vysokoúrovňové záležitosti, jako je například synchronizace vláken.

O nízkoúrovňové operace se stará C++ knihovna, jejíž všechny třídy jsou součástí jmenného prostoru `app`. Použití této knihovny je výhodné zejména ze dvou důvodů – prvním z nich je rychlost zpracování – dle vlastních testů je její rychlost při provádění základních obrazových operací (jako je zvýšení jasu) na emulátoru i reálném zařízení více než desetkrát vyšší oproti kódu napsanému v Javě. Druhým důvodem je obejití limitu pro maximální velikost dat alokovaných aplikací. Nevýhodou je naopak nutnost manuální správy alokované paměti a jen obtížné ladící možnosti na zařízeních s OS Android. Implementovaná knihovna ovšem využívá pouze standardní C++, takže ji lze zkompilovat, používat a ladit na téměř libovolné platformě.

5.2 Paralelní zpracování

Vzhledem k tomu, že většina moderních zařízení disponuje vícejádrovými procesory, aplikace byla navržena tak, aby pokud možno co nejvíce využívala paralelního zpracování dat, které při správném využití umožňuje dosáhnout podstatného zvýšení výkonu, ovšem za cenu náročnějšího vývoje a testování aplikace samotné. Obr. 5.2 schematicky znázorňuje základní rozdělení odpovědnosti mezi jednotlivá vlákna aplikace při zpracování vstupu uživatele.

Obr. 5.2 Základní schéma práce jednotlivých vláken.

5.2.1 Vlákno uživatelského rozhraní

Vlákno uživatelského rozhraní zpracovává události související se vstupem uživatele. Ovládání samotné aplikace probíhá asynchronně. Na vlákně s uživatelským rozhraním se vytvoří objekt zapouzdřující provedení požadované akce a ten je přidán do fronty příkazů. Zároveň je o skutečnosti, že byl přidán nový příkaz, informováno vlákno provádějící modifikace obrazu.

5.2.2 Pracovní vlákno

Na tomto vlákně probíhá naprostá většina modifikací obrazu. Vlákno postupně odebírá příkazy z fronty a spouští jimi zapouzdřené akce. Provedení příkazu mnohdy vyžaduje ještě před samotným vykonáním modifikace obrazu pozastavení renderovacího vlákna, protože pracovní a renderovací vlákno paralelně přistupují ke stejným datům instance obrazu. Zastavování renderovacího vlákna je výhradně v komperenci pracovního vlákna.

Jakmile je provedena požadovaná úprava obrazu, pracovní vlákno zadá požadavek na překreslení změněné oblasti.

5.2.3 Renderovací vlákno

Toto vlákno se stará o kompozici výsledného obrazu z jeho jednotlivých vrstev. Pokud obraz obsahuje více vrstev, může být tento proces značně časově náročný, proto probíhá v samostatném vlákně.

Jakmile toto vlákno dokončí renderování požadované oblasti, zadá požadavek na zobrazení obrazu, který obsluhuje vykreslovací vlákno.

5.2.4 Vykreslovací vlákno

Jedná se o vlákno, které má na starosti vykreslování výstupního obrazu na obrazovku. Toto vlákno je ovládáno jak z renderovacího vlákna, tak z vlákna uživatelské-

ho rozhraní. Příkazy týkající se posunu a zvětšování/zmenšování výstupního obrazu neprochází procesem na obr. 5.2, ale nastavují přímo parametry vykreslování tohoto vlákna.

5.3 Popis procesu modifikace obrazu

Tato sekce obsahuje popis toho, jak probíhá aplikace jednotlivých nástrojů používaných k modifikaci obrazu, a jak probíhá práce s vrstvami. Zde uvedený text slouží k vysvětlení abstrakce jednotlivých nástrojů a synchronizace vláken popsaných v předchozí sekci, soustředí se tedy na podstatu věci a upouští od výkladu na konkrétních třídách a metodách.

5.3.1 Abstrakce obrazu

Obraz se skládá z jednotlivých vrstev, které mohou být různého typu, jak bylo zmíněno dříve v sekci 4.3, Práce s vrstvami. Každá vrstva obrazu má nastavitelnou svoji průhlednost a masku, která může být zapnutá či vypnutá. Každá vrstva je také aplikovatelná na bitmapu. Abychom získali výsledný obraz jako jedinou bitmapu, postupně aplikujeme všechny vrstvy v daném pořadí na prázdnou cílovou bitmapu. Na konci procesu bude tato obsahovat výsledný obraz.

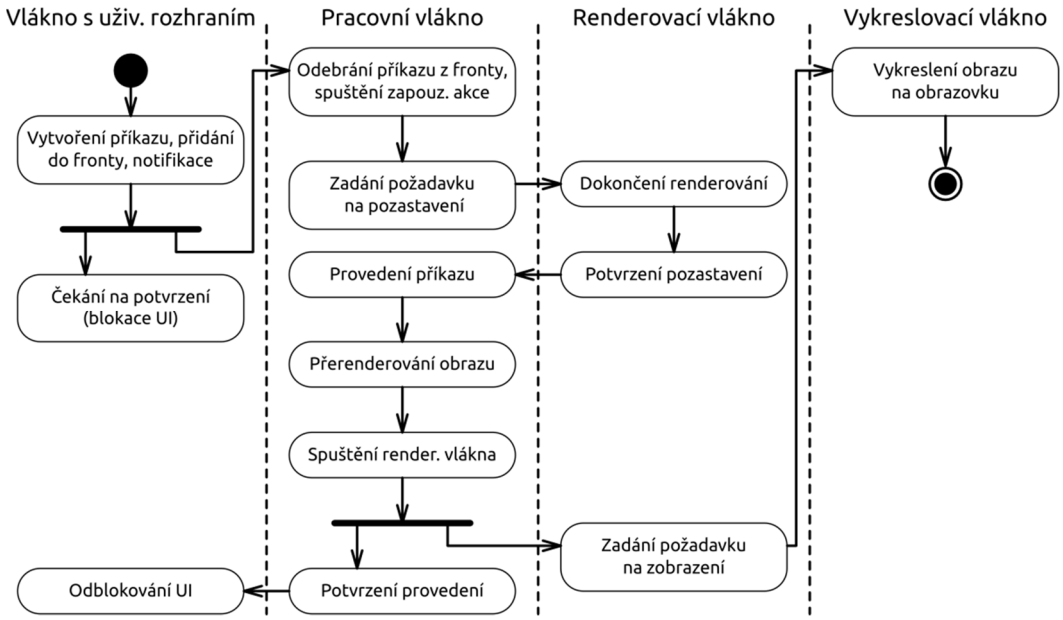
Pro účely kreslení má obraz dále ve vlastní režii bitmapu reprezentující masku výběru, dočasnou bitmapu zvanou plátno (canvas), na kterou se aplikují jednotlivé nástroje a dočasnou rastrovou vrstvu, která během kreslení zastupuje vybranou vrstvu, a jejíž maska při kreslení zastupuje masku vybrané vrstvy či výběr, pokud je modifikován jeden z těchto elementů obrazu.

Obraz má dále v režii bitmapu záplaty, která slouží k vracení jednoho kroku modifikace rastrových dat na libovolném elementu obrazu. Tato záplata se vytváří vždy při dokončení aplikace nástroje, jak bude popsáno dále.

Odpovědností obrazu je také udržování informace o nástrojem pozměněné oblasti, která ještě nebyla překreslena. Tato informace je popisována obdélníky, které se postupně ukládají do fronty. Při renderování se pak jednotlivé obdélníky odebírají a slučují tak dlouho, dokud je toto slučování výhodné, tedy do okamžiku, kdy plocha sloučené oblasti (oblast určená obdélníkem, který obklopuje všechny dosud odebrané obdélníky z fronty) je menší nebo rovna součtu ploch všech odebraných obdélníků. Tento postup je použit s ohledem na to, že pozměněné oblasti se obvykle značně překrývají, je tedy zřejmé, že postupné renderování by nebylo efektivní. Za zvážení stojí také použití dalších přístupů, ovšem výše uvedená metoda dosahuje dobrých výsledků sama o sobě.

5.3.2 Modifikace struktury obrazu

Mezi modifikace struktury obrazu patří operace vytváření, mazaní a změny pořadí vrstev, stejně jako nastavení jejich parametrů, mezi které patří viditelnost, průhlednost a aplikace masky. Všechny tyto operace vyžadují zastavení renderovacího vlákna, protože renderovací vlákno by mohlo přistupovat k objektům, které již byly uvolněny. Dále je požadováno překreslení celého obrazu, a aby uživatelské rozhraní zabránilo posílání dalších příkazů, protože ty by se také mohli týkat neplatných objektů a způsobit havárii programu. Celý proces shrnuje obr. 5.3.



Obr. 5.3 Průběh modifikace struktury obrazu.

5.3.3 Abstrakce nástroje

Základní vlastností nástroje je jeho schopnost být aplikován na obraz (většinou na konkrétní vrstvu, ale to obecně není podmínkou). Způsob aplikace se řídí parametry, které jsou nástroji během aplikace poskytnuty.

Jako další úroveň abstrakce lze nástroje rozdělit do dvou skupin. První skupinou jsou nástroje podobné štětci, které se aplikují na oblast obrazu, kterou uživatel určuje pomocí štětce, který má nastavené parametry hrotu a stopy. Jakmile uživatel oblast vybere, nástroj je aplikován bez jakéhokoliv náhledu či odsouhlasení. Tyto nástroje obvykle ovlivňují obraz inkrementálně po malých oblastech, které jsou dány otisky hrotu štětce. Charakteristický je relativně dlouhý výběr ovlivněné oblasti.

Druhou skupinou jsou nástroje, které lze označit za „nástroje s náhledem“. Pro tyto nástroje je charakteristické, že jsou aplikovány na celý obraz, nebo jen vybranou oblast, která je ale předem známá. Uživatel v tomto případě postupně pomocí náhle-

du nastavuje parametry nástroje a až po odsouhlasení těchto parametrů dojde k jeho použití.

Vzhledem k paralelnímu zpracování obrazu byl zvolen dále popsaný koncept aplikace nástroje na obraz, který se liší v závislosti na skupině, do které nástroj patří. I v nejobecnější abstrakci však nástroj musí podporovat následující:

- **Být aplikovatelný na obraz**
  - Tento bod odpovídá aplikaci nástroje na obraz, která byla vyvolána v souvislosti se vstupem uživatele. Zde jsou nástroji předány dodatečné parametry a vše probíhá na pracovním vlákně. Konkrétní akce závisí na implementaci nástroje a předaných parametrech.
  - Při zahájení aplikace si nástroj zamkne obraz tím, že požádá o plátno, kam si může ukládat data spojená se svou aplikací. Plátno si udržuje provedené změny do té doby, než je odesláno obrazu. Nástroj by měl v této fázi modifikovat pouze plátno.
  - V tomto bodě je nezbytně nutné, aby nástroj zaregistroval oblast, kterou svou aplikací změnil (pokud existuje), aby mohla být dále překreslena a později její změny provedeny.

- **Renderovat provedené změny**
  - Odpovídající metoda je volána na renderovacím vlákně v okamžiku, kdy byl vláknu zadán požadavek na přerenderování změněné oblasti. Slouží k zobrazení náhledu operace v průběhu aplikace nástroje.
  - Po nástroji je požadováno, aby v okamžiku, kdy je k tomu vyzván, upravil danou oblast v dočasné vrstvě (přesněji bitmapě, která je součástí dočasné vrstvy) tak, aby data v této oblasti dočasné vrstvy byla shodná s daty, která by obsahovala vybraná vrstva, kdyby byl nástroj aplikován na ni.
  - K dispozici nástroj dostává informace o renderované oblasti, bitmapu vybrané a dočasné vrstvy, bitmapu reprezentující masku výběru a plátno. Dále je k dispozici i odkaz na obraz samotný.

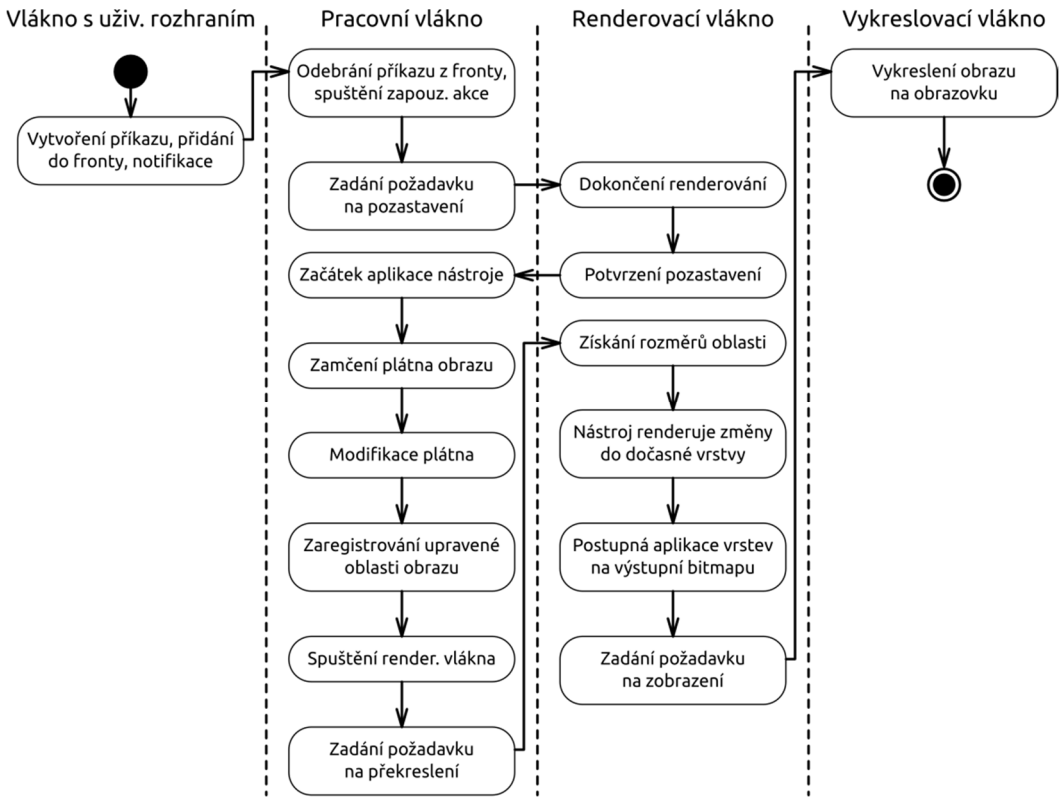
- **Provést změny přímo na vrstvě**
  - Odpovídající metoda je volána na pracovním vlákně v okamžiku, kdy nástroj dokončil modifikaci obrazu a odeslal plátno. Data, která dostává nástroj k dispozici, jsou shodná s těmi v předchozím bodě, jen s tím rozdílem, že tentokrát má nástroj provést změny na vybrané vrstvě. Informace o oblasti představují oblast, která zahrnuje sjednocení oblastí, které nástroj dříve zaregistroval jako pozměněné.

Aplikace nástroje probíhá ve třech fázích, které budou dále podrobně popsány, protože tvoří základ práce navržené aplikace. Jednotlivé fáze se liší především prací s plátnem a požadavky na synchronizaci vláken.

Začátek aplikace nástroje

Aplikace nástroje začíná právě touto fází. Průběh této operace je naznačen na obr. 5.4. Na začátku aplikace nástroje je vhodné, aby renderovací vlákno bylo pozastaveno, protože při zamykání plátna se vlastně mění struktura obrazu. Je třeba vyčistit plátno. Dále se od této chvíle při renderování nebude používat vybraná vrstva, ale místo ní bude použita vrstva dočasná, do které bude vybraný nástroj renderovat provedené změny.

K této fázi dochází buď v okamžiku, kdy uživatel začne kreslit (dotkne se poprvé obrazovky), nebo když se zobrazuje první náhled pro nástroj, který tuto funkci podporuje.

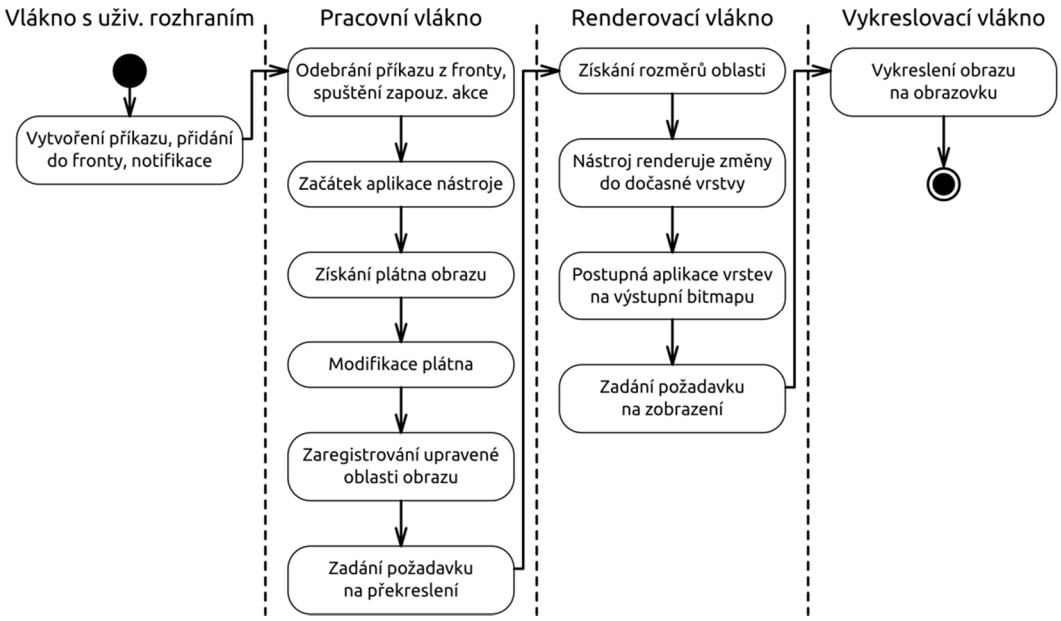


Obr. 5.4 Průběh začátku aplikace nástroje na obraz.

Průběžná aplikace nástroje

Tato fáze odpovídá buď kreslení u nástroje podobného štětci, nebo náhledu pro „nástroj s náhledem“. Celý proces nyní již nevyžaduje synchronizaci a může probíhat plně paralelně. Průběh procesu je naznačen na obr. 5.5 a i když to není z diagramu

na první pohled zřejmé, k modifikaci plátna dochází několikanásobně častěji než k renderování, obzvláště u obrazů s více vrstvami. Renderování probíhá inkrementálně ve smyčce, dle postupu popsaného v sekci 5.3.1, Abstrakce obrazu.



Obr. 5.5 Průběh aplikace nástroje na obraz (průběžný).

Dokončení aplikace nástroje

Jedná se o poslední fázi, která nastává v okamžiku, kdy se uživatel přestane dotýkat obrazovky (pro nástroj podobný štětci), nebo při potvrzení/zrušení akce pro nástroj s náhledem. Tato fáze vyžaduje ze zřejmých důvodů opět synchronizaci a je schematicky naznačena na obr. 5.6.

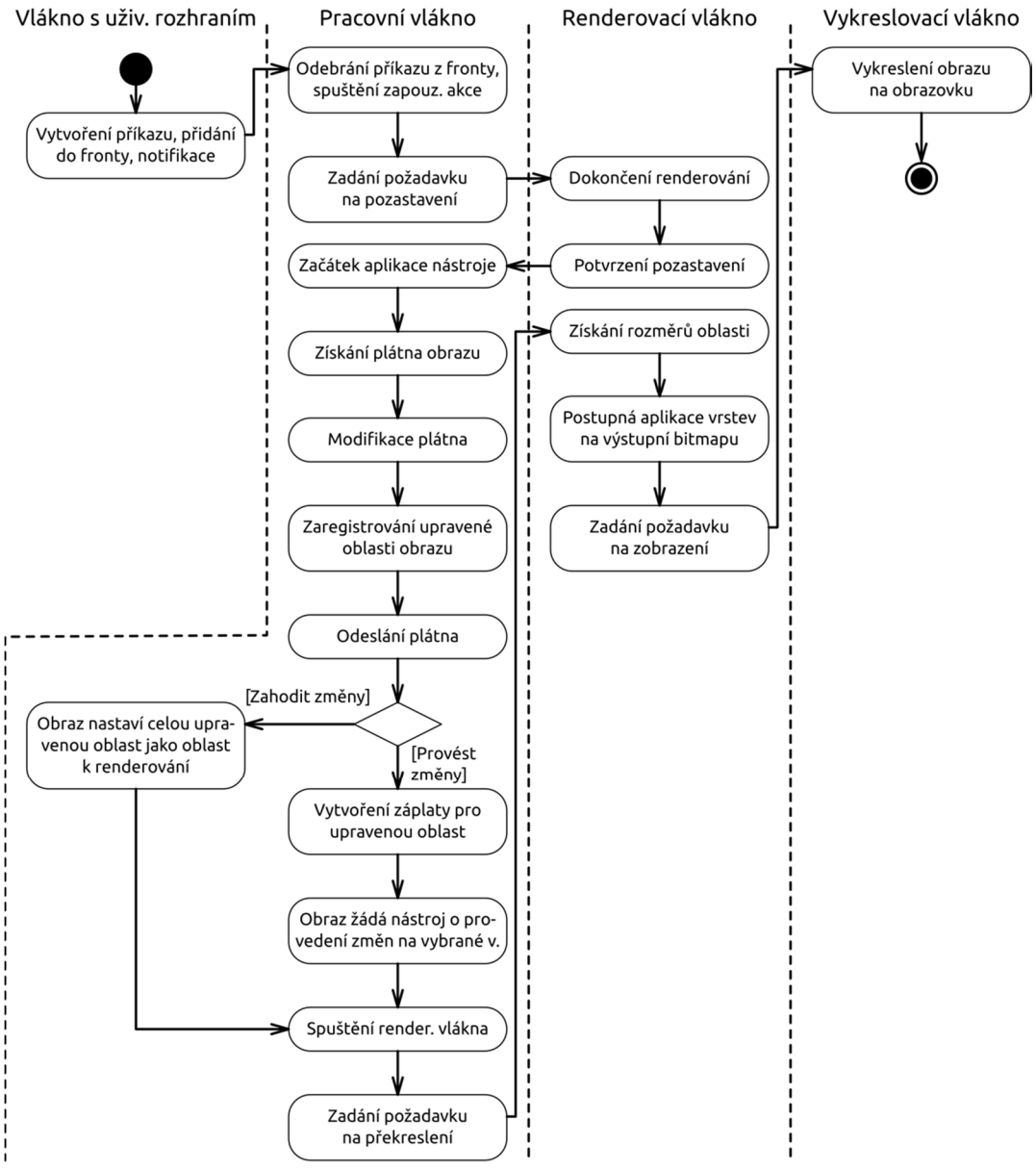
U nástroje podobného štětci jsou změny aplikovány na vybranou vrstvu, u nástroje s náhledem mohou nastat dvě varianty. Zaprvé uživatel operaci potvrdí, čímž umožní, aby se provedené změny promítnuly na vybranou vrstvu. Zadruhé uživatel operaci zruší. V tom případě jsou změny prostě zahozeny.

Od okamžiku, kdy nástroj odešle plátno, se již k renderování používá vybraná vrstva místo té dočasné. Pokud se mají změny použít, je nástroj zpětně požádán obrazem o promítnutí změn do vybrané vrstvy, v opačném případě obraz určí celou dříve upravenou oblast, která byla označena v průběhu aplikace nástroje, jako oblast k přerenderování.

V této fázi je také vytvářena záplata, která umožní uživateli vzít zpět provedenou úpravu obrazu. Toto je vhodné například v případě, že k úpravě došlo v důsledku neúmyslného dotyku obrazovky. Záplatu vytváří obraz a záplata obsahuje jednak původní rastrová data vybrané vrstvy (konkrétně oblasti, která má být upravena ná-

strojem), ale také informace identifikující element obrazu, pro který je záplata vytvořena, a dále informace o umístění této záplaty v daném elementu.

Při vracení provedené akce jsou data záplaty jednoduše zkopírována zpět do upraveného elementu. Vzhledem k tomu, že i toto vrácení zpět by mělo být vratné, je nutné opět vytvořit záplatu pro daný element, která bude obsahovat upravená data.



Obr. 5.6 Průběh ukončení aplikace nástroje na obraz.

5.4 Popis C++ knihovny

Veškeré operace s obrazem jsou prováděny s využitím vlastní C++ knihovny. Dále následuje stručný popis jednotlivých tříd v knihovně, rozdělený do kategorií dle

odpovědnosti jednotlivých tříd. Detailní interaktivní dokumentace je součástí příloh, a to ve formátu html, generována systémem Doxygen.

Všechny třídy jsou součástí jmenného prostoru `app`, který nebude dále pro jednoduchost uváděn. Většina tříd je tvořena šablonami (template), což umožňuje snadný přechod k bitmapám s vyšší bitovou hloubkou. V současné době jsou však plně podporovány pouze bitmapy s 8 bity na kanál.

Pro základní definici datových typů slouží soubor typedefs.hpp, kde jsou definovány datové typy pro reprezentaci pixelů a kanálů. Pro definici je využito typů z hlavičkového souboru `<cstdint>` ze standardu C++11, který umožňuje definovat celočíselné typy o přesné velikosti. Obsah souboru typedefs.hpp je následující:

```
#include <cstdint>
typedef uint32_t px_4x8bit;
typedef uint8_t px_1x8bit;
```

5.4.1 Reprezentace obrazu

Třída `TBitmap`

Tato třída slouží k reprezentaci bitmapy a je deklarovaná jako:

```
template <typename TPIXEL, typename TCHANNEL> class TBitmap;
```

Parametr `TPIXEL` určuje celočíselný typ, který pojme všechny kanály bitmapy, parametr `TCHANNEL` určuje celočíselný typ, který odpovídá jednomu kanálu bitmapy. Pro reprezentaci jednokanálové bitmapy (8 bitů na kanál) je tedy nutno vytvořit instanci třídy `TBitmap<px_1x8bit, px_1x8bit>` a pro reprezentaci čtyřkanálové instanci třídy `TBitmap<px_4x8bit, px_1x8bit>`.

Třída udržuje adresu paměti, kde jsou uložena obrazová data a dále informace o jejich interpretaci – rozměry bitmapy, barevný model, informaci o přednásobení alpa kanálem.

Třídu je možné vytvořit jednak takovým způsobem, že se sama stará o alokaci paměti pro obrazová data (a tuto paměť pak uvolňuje), ale také jako „obal“ pro existující oblast paměti reprezentující obraz, kdy více instancí může sdílet stejná data. Druhý přístup je potřebný při modifikaci bitmap vytvořených mimo oblast působení knihovny (zde bitmapy reprezentované třídami platformy Android).

Třída umožňuje kromě přístupu k informacím o obraze provádět tyto operace:

- Měnit přednásobení bitmapy alpha kanálem.
- Extrahovat kanál bitmapy jako jednokanálovou bitmapu.
- Kopírovat vybranou oblast bitmapy do jiné bitmapy.
- Vyplnit vybranou oblast bitmapy konstantní barvou.
- Vymazat bitmapu (ve smyslu nastavení všech pixelů na hodnotu `0x0`).

5.4.2 Alpha blending bitmap

Třída `Blender`

Třída tvoří jednu z mála výjimek, protože není definována jako šablona. Obsahuje pouze statické metody pro alpha blending jednokanálových a čtyřkanálových bitmap.

Tato třída představuje základ celé aplikace, protože se podílí na většině operací souvisejících s modifikací obrazu a jeho vykreslováním. Vzhledem k velkému počtu módů prolnutí a nutnosti poskytnout veškeré výše popsané operace pro každý z nich, byly při implementaci ve velké míře využity makra preprocesoru jako kompromis mezi čitelností kódu, jeho zbytečnou duplikací a efektivitou. Používání maker preprocesoru je obecně považováno za nebezpečnou techniku, ovšem zde jsou makra použita pouze izolovaně a klient třídy s nimi nepřichází do styku.

Třída umožňuje provádět operace, které vychází z módů prolnutí popsaných v sekci 2.4, Alpha blending. Spodní bitmapu budeme dále označovat jako cílovou a horní bitmapu jako zdrojovou.

Čtyřkanálové bitmapy

Třída umožňuje provádět s čtyřkanálovými bitmapami následující operace.

- Prolnout na vybranou oblast cílové bitmapy konstantní barvu v daném módu s možností uplatnění dodatečné průhlednosti (tedy zmenšení krytí zdrojové barvy vynásobením jejího alpha kanálu dodatečnou hodnotou alpha).
- Prolnout konstantní barvu do cílové bitmapy přes masku. V tomto případě se určí oblast masky a její umístění v cílové bitmapě, hodnoty pixelů masky určují dodatečnou průhlednost barvy. Opět je možno aplikovat dodatečnou průhlednost.
- Prolnout vybranou oblast zdrojové bitmapy na vybrané místo v cílové bitmapě s možností dodatečného zvýšení průhlednosti zdroje.
- Prolnout zdrojovou bitmapu na cílovou přes masku. V tomto případě se určí oblast masky a její umístění v cílové bitmapě. Maska určuje dodatečnou průhlednost zdrojové bitmapy. Dále je nutno specifikovat zarovnání zdrojové bitmapy s cílovou. Opět je možno dodatečně zvýšit globální průhlednost zdroje
- Přičtení/odečtení vybrané oblasti jednokanálové bitmapy k aplha kanálu cílové bitmapy, včetně určení dodatečné hodnoty krytí zdrojové jednokanálové bitmapy.
- Přičtení/odečtení jednokanálové bitmapy k alpha kanálu cílové bitmapy přes masku. Parametry operace odpovídají parametrům při prolínání dvou bitmap přes masku, jak je popsáno výše.

Jednokanálové bitmapy

Pro jednokanálové bitmapy jsou k dispozici tyto operace:

- Přičtení a odečtení vybraných oblastí jednokanálových bitmap včetně specifikace dodatečné hodnoty alpha.
- Přičtení a odečtení vybraných oblastí jednokanálových bitmap přes masku, opět s možností určení dodatečné hodnoty alpha.

5.4.3 Bodové operace

Třída `TLUT`

Třída reprezentuje náhledovou tabulku pro obecné provádění bodových operací. Třída je definovaná opět jako šablona s následující deklarací:

```
template<typename TPIXEL, typename TCHANNEL> class TLUT;
```

Význam parametrů je obvyklý. Třída má k dispozici dynamicky alokovaná pole pro každý kanál obrazu. Při použití na obraz se pro každou hodnotu kanálu nalezne v poli jeho nová hodnota, která je dána položkou na pozici odpovídající původní hodnotě kanálu.

Třída umožňuje aplikovat bodové operace na čtyřkanálové bitmapy a to jedním z těchto způsobů:

- Aplikovat bodovou operaci na vybranou oblast.
- Aplikovat bodovou operaci na vybranou oblast s dodatečnou hodnotou alpha. V tomto případě je výsledek takový, jako bychom si uložili výsledek aplikace bodové operace do samostatné bitmapy a tuto na původní bitmapu prolnuli s dodatečnou průhledností alpha.
- Aplikovat bodovou operaci na bitmapu přes vybranou oblast masky. Opět výsledek odpovídá prolnutí výsledku bodové operace do původního obrazu přes masku. Lze použít i s dodatečnou hodnotou alpha.

Třída `TLUTGenerator`

Šablona třídy je dána následující deklarací:

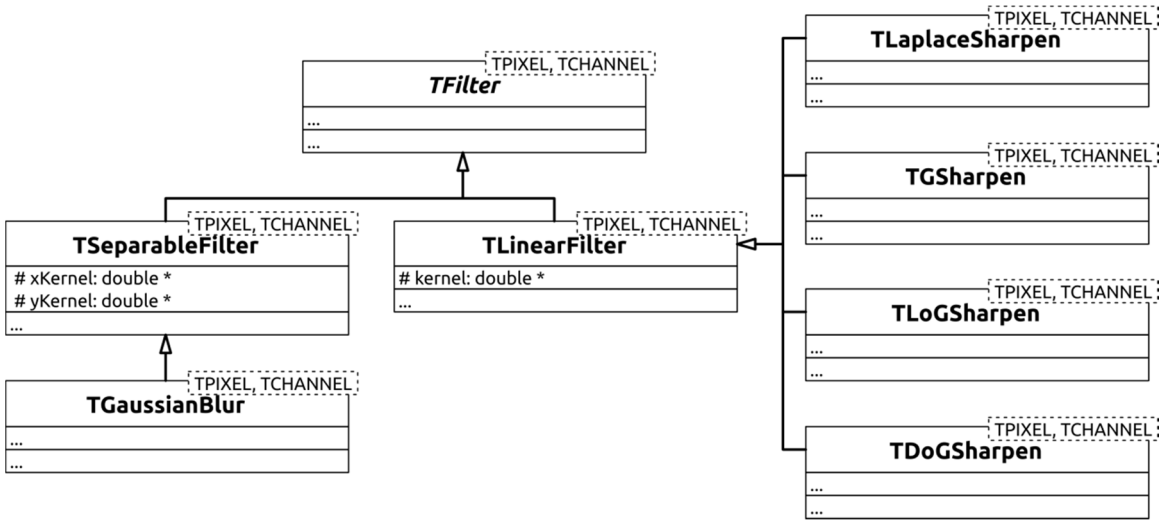
```
template<typename TPIXEL, typename TCHANNEL>
   class TLUTGenerator;
```

Třída obsahuje pouze statické metody, které souží k vytváření instancí třídy `TLUT` či modifikaci těchto instancí – pokud je již instance `TLUT` vytvořena, stačí změnit hodnoty v tabulce pro jednotlivé kanály. Toto zmenšuje potřebu synchronizace při paralelním zpracování klientem a celkově zefektivňuje běh programu.

K dispozici je generování tabulek pro bodové operace popsané v sekci 2.5, Bodové operace.

5.4.4 Filtry

O realizaci filtrů se stará skupina tříd, která zobrazena na obr. 5.7. Všechny tyto třídy jsou realizovány jako šablony.



Obr. 5.7 Třídy starající se o realizaci filtrů.

Všechny filtry musí nabízet stejné možnosti použití jako bodové operace, tedy každý filtr musí být aplikovatelný na vybranou oblast, na vybranou oblast s dodatečnou hodnotou průhlednosti alpha, dále aplikovatelný přes masku a to i s možností specifikace dodatečné hodnoty alpha. Podobně jako u bodových operací je výsledek aplikace filtru s dodatečnou hodnotou alpha shodný s výsledkem, který dostaneme, pokud výsledek použití filtru prolneme odpovídajícím způsobem na původní bitmapu.

Třída `TFilter`

Tato třída má poskytovat pouze společné rozhraní pro jednotlivé filtry, přesto však u všech požadovaných metod nabízí výchozí implementaci, která ovšem neprovádí žádné úpravy, pouze vrací chybový kód. Třída je ryze virtuální díky neimplementované metodě informující o úspěšné inicializaci, jejíž implementace je úkolem odvozených tříd.

Třída `TSeparableFilter`

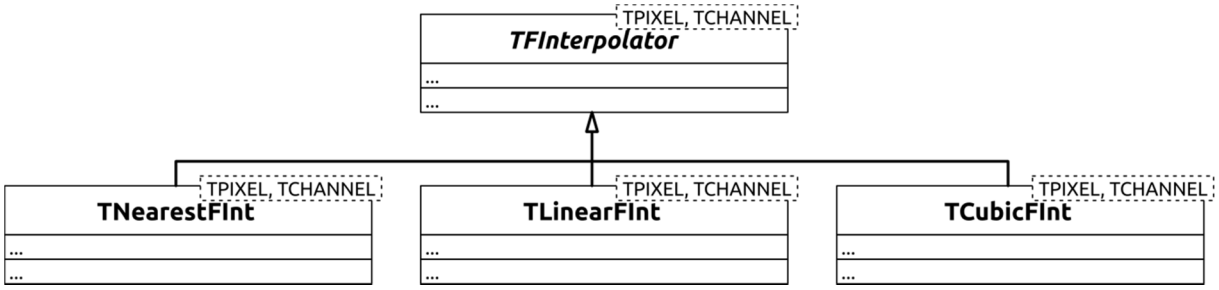
Třída poskytuje režijní strukturu potřebnou pro aplikaci separovatelného filtru. Odvozené třídy se v konstruktoru pouze starají o inicializaci jádra filtru v jednotlivých směrech. Tuto třídu využívá filtr Gaussovské rozostření implementovaný pomocí třídy `TGaussianBlur`.

Třída `TLinearFilter`

Třída implementuje režijní strukturu, která umožňuje aplikaci libovolného lineárního filtru. Povinností odvozených tříd je opět inicializace matice filtru v konstruktoru. Tuto třídu využívají filtry Laplacián (`TLaplaceSharpen`), Gaussovské zostření (`TGSharpen`), LoG (`TLoGSharpen`) a DoG (`TDoGSharpen`).

5.4.5 Interpolace

O zajištení interpolace se stará skupina tříd zobrazených na obr. 5.8. Všechny interpolátory mají jako bázovou třídu ryze virtuální třídu `TFInterpolator`.



Obr. 5.8 Třídy zajišťující interpolaci.

Třída `TFInterpolator` je ryze virtuální a zajišťuje jednotné rozhraní pro ostatní třídy interpolátoru, které určují konkrétní metodu interpolace. Všechny interpolátory se vytváří nad instancí bitmapy a poskytují tyto možnosti interpolace:

- Převzorkování celé bitmapy, nad kterou je interpolátor vytvořen, do jiné bitmapy s využitím dané metody interpolace.
- Převzorkování celé bitmapy do jiné s využitím filtru dolní propust (Gaussovské rozostření) pro zabránění aliasingu při zmenšování obrazu. V tomto případě je nastavitelná síla rozostření.
- Získání interpolované hodnoty, která je na zadaných reálných souřadnicích. Pokud je požadovaná hodnota mimo oblast obrazu, vrací `0x0`.

5.4.6 Lineární transformace

O provádění lineární transformace se stará třída definovaná následovně:

```
template<typename TPIXEL, typename TCHANNEL,
   typename TINTERPOLATOR> class TTransform;
```

Třída umožňuje transformovat zdrojovou bitmapu do cílové pomocí matice lineární transformace implementované pomocí třídy `Matrix3x3`, která poskytuje také základní maticové operace jako je transpozice, násobení a inverze matice. Třída `TTransform` také nabízí metody pro vytvoření matice popisující translaci, rotaci, změnu velikosti, zkosení a obecnou projektivní transformaci.

5.4.7 Pomocné třídy

Knihovna dále obsahuje pomocné třídy, které lze rozdělit do několika kategorií.

Geometrie

Sem patří třídy `Point`, `Vect` a `Rect`, které popisují po řadě bod, vektor v rovině a obdélník s celočíselnými souřadnicemi. Třída `Rect` umožňuje základní operace s obdélníkem jako je výpočet jeho výšky, šířky, ale také posun, změnu velikosti a především počítání průniků obdélníků a jejich porovnávání. Tyto třídy se používají při určování oblasti na bitmapě.

Třída `PointerArea`

Tato třída využívá tříd `Point`, `Vect` a `Rect`. Jak bylo zmíněno dříve v sekci 2.2.1, Reprezentace bitmapy v paměti, bitmapa je dvojrozměrný objekt, ale v paměti je uložena jako jednorozměrná posloupnost svých bodů. Tato třída slouží k počítání potřebných offsetů pro pohodlnou práci s různými jejími obdélníkovými výřezy. Třída vyžaduje při vytváření znalosti o počáteční adrese oblasti, rozměrech celkové oblasti (v počtu prvků), počtu bajtů, které každý prvek v této oblasti zabírá a volitelně také rozměr vybrané oblasti. Třída umožňuje určit kromě offsetů a adres významných bodů (začátek, konec...) také „společnou oblast“ dvou a tří oblastí, což je potřeba při prolínání několika bitmap na sebe.

Mezi další operace patří výplň dané oblasti v paměti a kopírování dané oblasti na určené místo jiné oblasti.

Třída `MirroredReindexer`

Jedná se o malou pomocnou třídu, která vytváří náhledovou tabulku pro indexy pole. Pro přístup k prvku pole se použije hodnota daná touto tabulkou. Mapování indexů je takové, jako kdyby se pole neustále zrcadlově opakovalo. Použití je při určování hodnoty pixelů mimo oblast obrazu pro aplikaci filtrů.

Třída `CubicSpline`

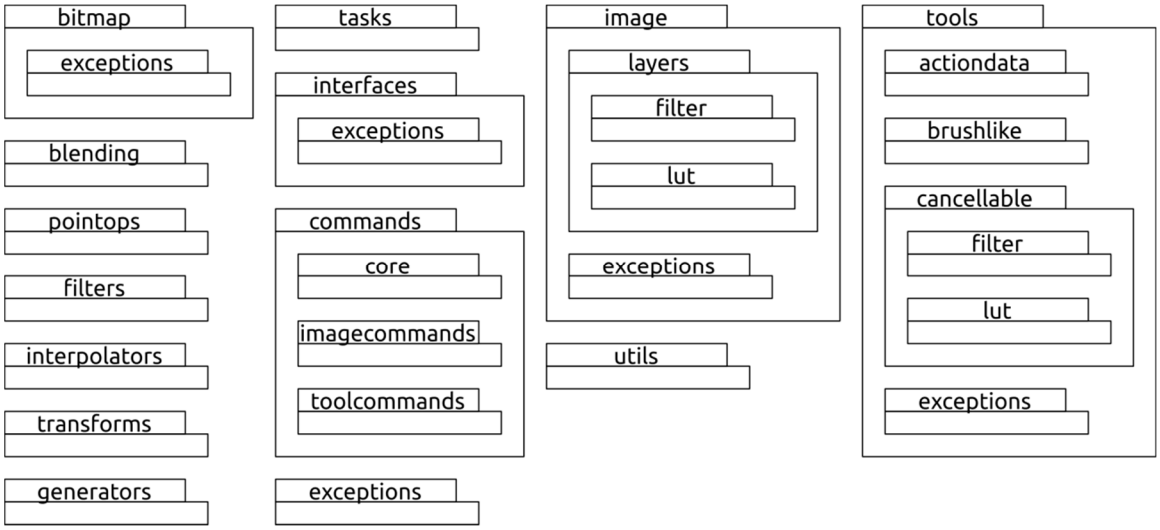
Třída poskytuje interpolaci kubickým spline a využití nachází při bodových operacích – úpravě křivek. Implementace s drobnými úpravami vychází z kódu dostupného z [25].

Třída `TBrushGenerator`

Jak již název napovídá, tato třída slouží ke generování štětců. Implementováno je generování kulatého štětce s volitelnou tvrdostí.

5.5 Popis jádra aplikace

Jádro aplikace je implementováno pouze za využití Java SE, je tedy přenositelné na mnoho systémů. Propojení s C++ knihovnou zajišťuje JNI. Jádro aplikace obsahuje řídící mechanismy pro ovládání vláken aplikace a přístup k nástrojům pro modifikaci obrazu z výše popsané knihovny. Popis jádra aplikace bude členěn dle jednotlivých balíčků (přehled balíčků udává obr. 5.9, všechny balíčky jsou součástí balíčku `mhr.appcore`). Vzhledem k velkému počtu tříd, budou podrobně popsány pouze ty nejvýznamnější. Podrobná dokumentace je opět dostupná ve formátu html v rámci příloh.



Obr. 5.9 Organizační schéma jádra aplikace.

Balíčky lze rozdělit do několika skupin. První skupinou jsou balíčky s třídami pro přístup k nativní knihovně, ty jsou na obr. 5.9 v levém sloupci. Druhý sloupec představuje balíčky pro režijní strukturu aplikace. Zde jsou třídy potřebné pro práci jádra zajišťující ovládání vláken atd. Ve třetím sloupci jsou pak balíčky, které se přímo vztahují k obrazu, a čtvrtý sloupec obsahuje balíčky s třídami nástrojů.

5.5.1 Třídy pro přístup k nativní knihovně.

Balíček `bitmap`

Balíček obsahuje třídu `NBitmap`, která zprostředkovává přístup k metodám třídy `TBitmap` nativní knihovny. Dále jsou zde definované výčtové typy `ChannelCount`, `ColorMode`, `Depth` a `NativeType`, které popisují vlastnosti bitmapy. Tyto výčtové typy explicitně určují konstanty typu `int`, které se používají pro komunikaci s nativní knihovnou. Pro uložení komplexních informací o bitmapě slouží třída `BitmapInfo`.

Balíček `blending`

Tento balíček obsahuje výčtový typ `BlendMode`, který určuje mód prolnutí při alpha blendingu a definuje explicitní konstanty typu `int` pro komunikaci přes JNI. Dále je zde třída `Blender`, která obsahuje pouze statické metody, které odpovídají metodám nativní třídy `Blender`.

Balíček `pointops`

Balíček poskytuje jedinou třídu `LUT`, která zapouzdřuje provádění bodových operací – její odpovědností je jednak vytvoření nativní LUT tabulky dané bodové operace, ale také její aplikaci na obraz.

Balíček `filters`

Zde je obsažena jediná třída `Filter`, která slouží k jednotnému vytváření a aplikaci filtrů nativní knihovny.

Balíček `interpolators`

Balíček definuje přístup k interpolátorům nativní knihovny, respektive k převzorkování celé bitmapy do jiné pomocí třídy `Interpolator`. K určení typu interpolace slouží zde definovaný výčet `InterpolatorType`.

Balíček `transforms`

Obsahem balíčku je třída `Transform`, která umožňuje provádět s obrazem libovolné lineární transformace. Při aplikaci lze vybrat typ interpolátoru, který se má pro transformaci použít. Matice transformace je definována třídou `TM`, která je zde také definována a umožňuje základní maticové operace.

Balíček `generators`

Tento balíček obsahuje třídu `BrushGenerator`, která zpřístupňuje funkce stejnojmenné nativní třídy.

5.5.2 Třídy režijní struktury

Třída `AppCore`

Přístup k jádru aplikace poskytuje třída `AppCore`, která je umístěna přímo v balíčku `mhr.appcore`. Klientská aplikace vytváří instanci této třídy, která se dále sama stará o spouštění a zastavování pracovního a renderovacího vlákna, přidávání příkazů do fronty a export a ukládání obrazu. Třída dále umožňuje nastavit posluchače, který bude informován o výjimkách, ke kterým došlo při vykonávání příkazu (například při pokusu na kreslení do vrstvy, která tuto operaci nepodporuje). Dále lze nastavit třídu, která bude zpracovávat neodchycené výjimky pracovních vláken.

Balíček `interfaces`

Balíček definuje rozhraní pro poskytnutí platformě závislé funkcionality a standardní výjimky, které mají používat.

Rozhraní `ImageFile`

Jedná se o nejobsáhlejší rozhraní, které slouží k ukládání obrazu včetně vrstev. Jádro aplikace požaduje, aby třída implementující toto rozhraní dokázala poskytnout instanci třídy implementující `org.w3c.dom.Document`, která bude obsahovat xml popis čteného či ukládaného souboru. Dále rozhraní specifikuje požadavek na schopnost načtení a zapsání jedno a čtyřkanálové bitmapy, vytvoření bitmapy dle zadané specifikace a smazání souboru s obrazem pro případ nepovedeného zápisu.

Rozhraní `PDBitmap`

Rozhraní specifikuje požadavky na platformě závislou bitmapu. Tato musí být schopna zamknout svoje data a zpřístupnit jejich nativní adresu, poskytnout svůj popis ve formě instance třídy `BitmapInfo`, být schopna vytvořit svoji kopii a také vytvořit novou bitmapu dle zadaných požadavků.

Rozhraní `PDDisplayer`

Toto rozhraní specifikuje požadavky na třídu zajišťující výstup obrazu. Prvním požadavkem je schopnost nastavit platformě závislou bitmapu jako zdroj výstupních dat, druhým požadavkem je schopnost na požádání překreslit výstup.

Rozhraní `PDExceptionFeedback`

Prostřednictvím třídy implementující toto rozhraní vrací instance `AppCore` nefatální výjimky, ke kterým došlo během vykonávání příkazu.

Rozhraní `PDFeedback`

Instance třída implementující toto rozhraní může být připojena k libovolnému příkazu a její odpovídající metoda bude spuštěna po jeho provedení.

Třída `PDImageDataPresentation`

Slouží k definici požadavků na třídu, která bude platformě závislým způsobem reprezentovat strukturu obrazu. Definuje větší množství metod, které přibližně odpovídají metodám na obraze, jako je například přidání či odebrání vrstvy. Potomek této třídy slouží k asynchronní výměně dat mezi vláknem uživatelského rozhraní a pracovním vláknem.

Balíček `Commands`

Tento balíček zapouzdřuje příkazy, které určují operace probíhající na pracovním vlákně.

Třída `AppCommand`

Jedná se o abstraktní třídu, která je výchozí pro všechny ostatní třídy příkazů. Tyto třídy později přepisují metodu `action()`, která je spuštěna na pracovním vlákně. K příkazu je možno připojit instanci třídy implementující rozhraní `PDFeedback`, které je oznámeno úspěšné provedení příkazu, a také jinou instanci této třídy, které bude oznámeno, dojde-li při vykonávání k libovolné výjimce `RuntimeException`.

Balíček `corecommands`

Balíček obsahuje příkazy týkající se přímo jádra aplikace.

Balíček `imagecommands`

Tento balíček obsahuje třídy reprezentující příkazy pro modifikaci struktury obrazu, jako je vytváření a mazání vrstev. Jejich detailní popis viz html dokumentace.

Balíček `toolcommands`

Obsahem balíčku jsou příkazy související s aplikací jednotlivých nástrojů a výběrem nástroje.

Balíček `exceptions`

Tento balíček obsahuje několik základních výjimek, které se používají napříč jádrem aplikace. Jedná se jednak o `AllocationException`, která značí problém s alokací nativního objektu a `AlreadyDisposedException`, která nastává v okamžiku, kdy se některá komponenta snaží využít jinou, již uvolněnou komponentu. Dále jsou zde definované následující dvě výjimky, ze kterých jsou odvozeny všechny další výjimky.

Výjimka `AppFatalException`

Tato výjimka značí, že došlo k problému, ze kterého se již nelze jednoduše vzpamatovat a obvykle znamená ukončení aplikace, například v důsledku nedostatku paměti RAM pro další práci.

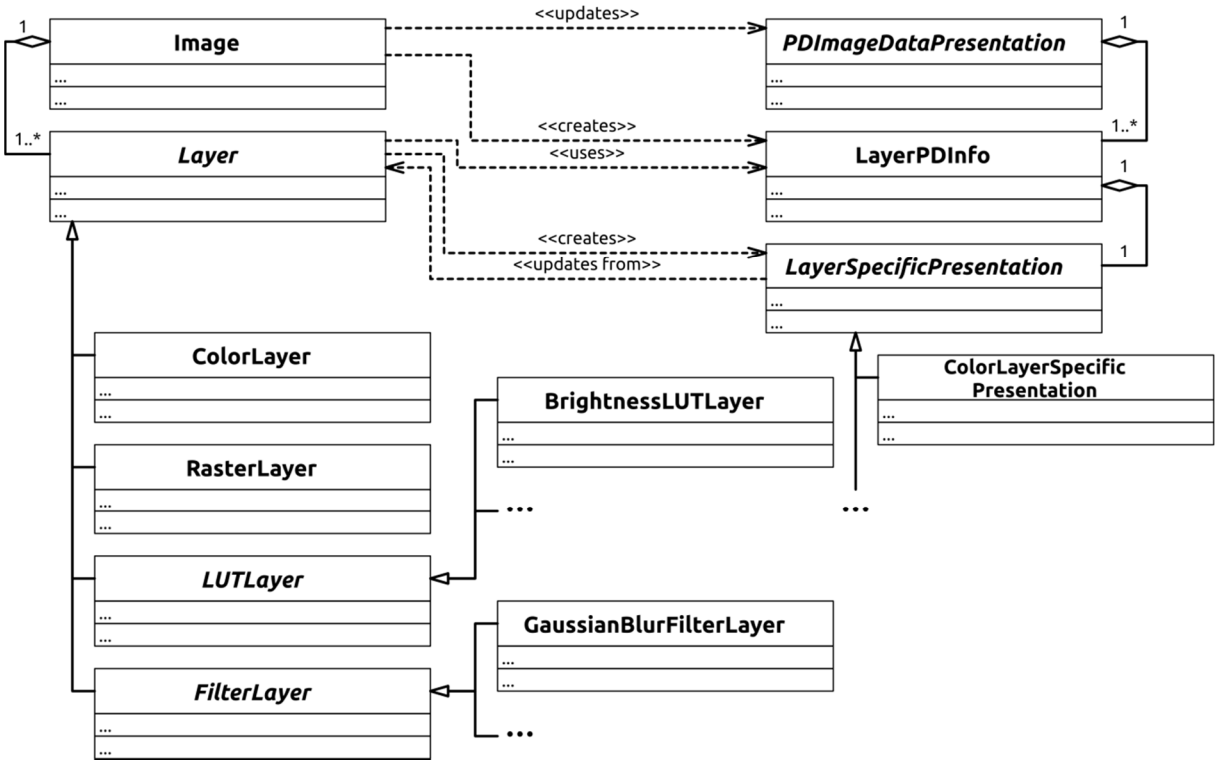
Výjimka `AppNonFatalException`

K této výjimce dochází, pokud nastal problém, který vyžaduje zastavení všech probíhajících operací, ale aplikace po provedení úklidu může nadále fungovat. Takovým příkladem je například úprava masky vrstvy nástrojem, který k tomu není určený – odpovídající výjimka, odvozená od `AppNonFatalException`, je vrácena klientovy prostřednictvím `PDExceptionFeedback`, jsou odstraněny všechny příkazy z fronty a jakmile klient informuje jádro aplikace o tom, že již situaci napravil, je jádro aplikace opět spuštěno.

5.5.3 Třídy obrazu

Sem patří třídy obsažené v balíčku image. Jedná se o třídy, které jsou zobrazeny na obr. 5.10 (S výjimkou `PDImageDataPresentation`, která je součástí balíčku `interfaces`). Obraz se skládá z jednotlivých vrstev, jejichž hierarchie dědičnosti je patrná z obrázku (z prostorových důvodů zde nejsou zobrazeny všechny).

Z obrázku je dále patrné, jak probíhá komunikace třídy `Image` a klientské aplikace. Klient předá obrazu instanci, která je potomkem `PDImageDataPresentation`. Obraz poté vytvoří v této prezentaci pro každou vrstvu odpovídající záznam (instanci `LayerPDInfo`), která obsahuje informace specifické konkrétní vrstvě jako instanci třídy odvozené od `LayerSpecificPresentation` (kterou na požádání vytvoří vrstva). Tato specifická prezentace slouží jednak k poskytnutí informace o vrstvě klientovy, ale také k předávání informací opačným směrem – klient upravuje specifickou prezentaci dané vrstvy a poté žádá obraz, aby danou vrstvu dle této prezentace upravil. Tímto mechanismem se také předávají náhledy vrstev. Implementace potomka `PDImageDataPresentation` má obsahovat mechanismus, který umožní obrazu informovat klienta o provedených změnách. Princip práce s vrstvami a jejich funkce byli popsány dříve, zejména v sekci 5.3, Popis procesu modifikace obrazu.



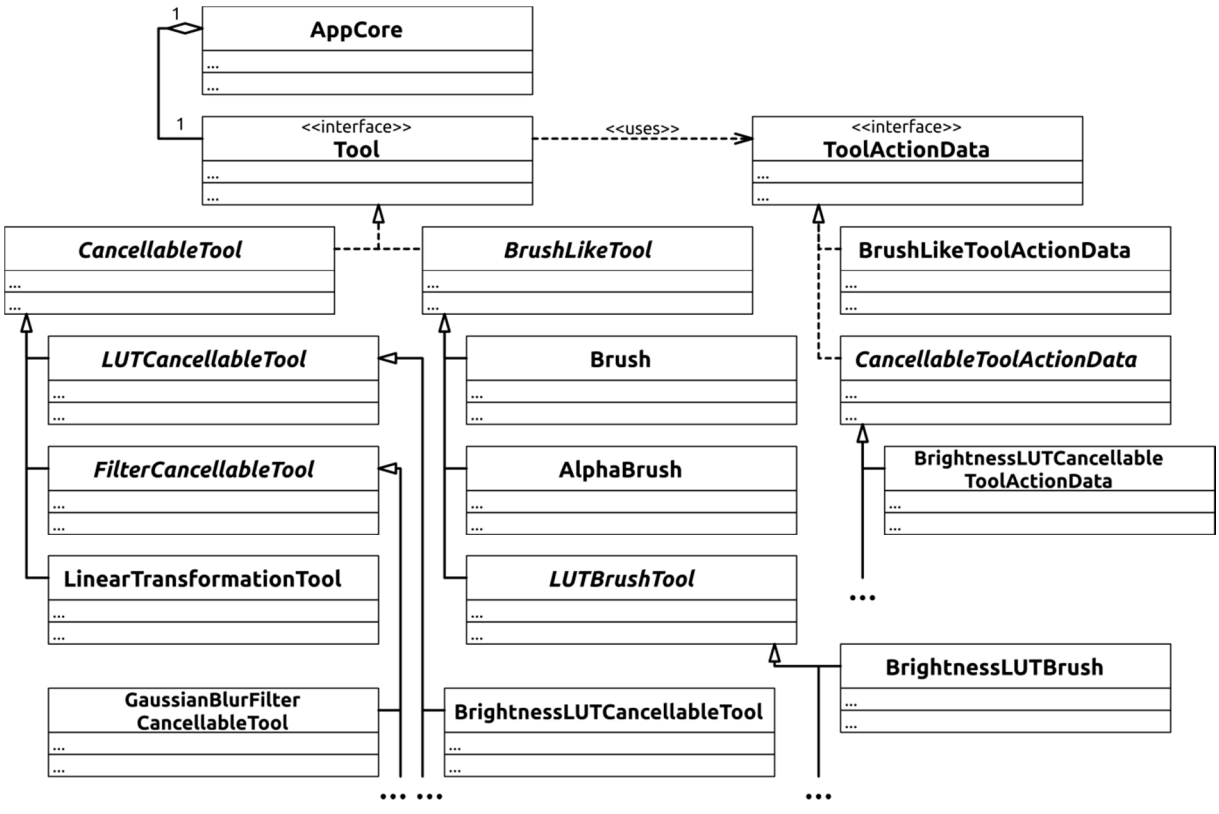
Obr. 5.10 Hierarchie tříd obrazu a vrstev.

Třídy nástrojů

Třídy reprezentující nástroje jsou obsaženy v balíčku `tools`. Hierarchie tříd je patrná z obr. 5.11. (Opět z prostorových důvodů nejsou uvedeny všechny, ale systém je zřejmý).

Jádro aplikace si drží instanci aktuálně vybraného nástroje. Každý Nástroj implementuje rozhraní `Tool`, které specifikuje požadavky na nástroj uvedené v sekci 5.3.3, Abstrakce nástroje. Výběr nástroje, stejně jako jeho použití se ovládá asynchronně pomocí příkazů zaslaných jádru aplikace. Během aplikace jsou nástroji předána data, která musí implementovat výchozí rozhraní `ToolActionData`. Význam těchto dat se liší v závislosti na vybraném nástroji. Obecně každý „nástroj s náhledem“ (tyto nástroje mají společnou bázovou třídu `CancellableTool`) má vlastní třídu pro reprezentaci svých dat, protože během aplikace nástroje se nastavují jeho parametry.

Oproti tomu nástroje podobné štětci (tyto nástroje mají společnou bázovou třídu `BrushLikeTool`) mají společná data popisující prováděnou akci, protože pro tuto skupinu nástrojů je charakteristické, že jakmile jsou vytvořeny, jejich parametry se již nemění. Data nástroje v tomto případě udávají typ události – fázi aplikace nástroje a pozici, na kterou má být aplikován. Toto platí i při rozšíření nástroje například o podporu změny parametrů v závislosti na přítlaku – přidá se pouze informace o velikosti přítlaku, o modifikaci parametrů se poté postará sám nástroj.



Obr. 5.11 Hierarchie tříd nástrojů.

5.6 Popis komponent specifických pro OS Android

Komponenty specifické pro operační systém Android představují zejména elementy uživatelského rozhraní a také třídy implementující rozhraní, jenž jsou součástí balíčku `mhr.appcore.interfaces`. Tyto třídy jsou umístěny v balíčcích, které jsou součástí `mhr.appandroid`.

Za zmínku stojí především třídy obsažené v balíčku `mhr.appandroid.views`, respektive v balíčcích, které jsou jeho součástí. Zde jsou obsaženy vlastní třídy uživatelského rozhraní, které zajišťují například pokročilejší vykreslování bitmap, než jaké umožňují výchozí komponenty poskytované platformou Android (k tomuto účelu slouží třídy `SimpleBitmapView` a `MaskView`). Dále jsou zde třídy, které zapouzdřují formulářové prvky pro komplexní vstup uživatele – například element pro výběr barvy pomocí RGB a HSV hodnot s průběžným náhledem (`ColorPickerView` a `SVPickerView`) či formulářový prvek zajišťující zadání hodnot pro vyvažování barev pomocí křivek (`CurvesView`) a třídy pro výběr parametrů štětce a stopy (`RoundBrushPickerView` a `PathPickerView`). Kompletní výčet tříd je obsažen v přiložené html dokumentaci, jejich význam je však vždy intuitivně zřejmý již z názvu.

Hlavní třída aplikace (`AppMainActivity`), stejně jako veškeré její formuláře jsou součástí balíčku `mhr.app`.

Závěr

Cílem práce bylo vytvořit editor rastrové grafiky pro zařízení s operačním systémem Android. Na základě přehledu aktuálně dostupného softwaru pro PC a Android byly formulovány požadavky na navrhovanou aplikaci tak, aby se odlišovala od ostatních aplikací dostupných pro tuto platformu a přinášela funkce známé ze softwaru pro PC. Zejména pak elementární operace s obrazem a pokročilou práci s vrstvami, poskytující zkušeným uživatelům dostatek tvůrčí svobody.

Teoretický úvod poskytuje shrnutí znalostí potřebných pro zpracování obrazu. Zaměřuje se však zejména na konkrétní, praktické poznatky významné pro samotnou implementaci. Jsou zde shrnuty jak běžně v literatuře uváděné teoretické znalosti týkající se bodových operací, filtrů a geometrických transformací obrazu, tak ryze praktické informace, které popisují chování jednotlivých v praxi používaných bodových operací, filtrů a interpolačních metod. Ryze praktická je také část o alpha blendingu (prolínání, skládání nebo také míchání bitmap s rozdílnou průhledností). Toto téma není obecně v literatuře zabývající se zpracováním obrazu příliš popsáno, přesto však existuje celá řada metod míchání bitmap, které nachází široké uplatnění při úpravách fotografií a obrazu obecně a tvoří základ práce s vrstvami.

Na základě popsaných teoretických znalostí byl pomocí C++ a Javy implementován software, který umožňuje provádět obvyklé operace s obrazem, jako je úprava jasu, a kontrastu či gama korekce, ale také provádět téměř libovolné bodové úpravy obrazu pomocí nástroje křivky, a to jak pro všechny kanály současně, tak pro každý samostatně. Dále jsou k dispozici základní filtry pro rozostření a zostření obrazu. Samozřejmě nechybí ani nástroj štětec s mnoha nastavitelnými parametry. Hrot štětce má nastavitelnou velikost a tvrdost, stopa štětce pak krytí, tok a mezery otisku hrotu. Pro klasický štětec je k dispozici 28 módů prolnutí. Dále je k dispozici speciální štětec pro modifikaci alpha kanálu, masek vrstev a výběru či štětec pro přesnou lokální aplikaci bodových operací.

Pro celý program je charakteristické využití konceptu vrstev, který je značně rozšířen. Každá vrstva má nastavitelnou jak viditelnost, tak plynule nastavitelnou průhlednost a je možno jí přiřadit masku. Jako vrstva může vystupovat nejen bitmapa, ale také například výplň barvou či bodová operace. Unikátní funkcí je možnost vytvoření filtru jako vrstvy, což neumožňuje žádný dostupný software.

K dispozici jsou dále geometrické transformace obrazu, které umožňují jak provádět běžné operace, jako je změna velikosti a ořez obrazu, tak provádět obecnou lineární transformaci pro korekci perspektivy.

Samozřejmostí je možnost exportu obrazu do standartních formátů a bezztrátové uložení obrazu, včetně jeho jednotlivých vrstev, pro pozdější práci. Program dále nabízí možnost využití schránky, a to nejen pro kopírování, ale také pro další operace s rastrovými daty.

Aplikace představuje silný nástroj především pro editaci fotografií a kresbu, který oproti ostatním aplikacím, které jsou pro Android dostupné, vyniká především pokročilou prací s vrstvami. Vrstvy výplně, bodových operací a filtrů zde žádný jiný software nenabízí, stejně jako nikde jinde nejsou k dispozici masky vrstev. Dále se aplikace výrazně odlišuje od ostatních díky velkému množství implementovaných módů prolnutí, kde významně předstihuje konkurenci, zejména pak díky pokročilým módům, které nevyužívají barevného prostoru RGB.

Stále však existuje mnoho oblastí, ve kterých se může aplikace dále vyvíjet. Jednou z nich je optimalizace kódu, která je vždy aktuální pro software, který provádí velké množství výpočtů. Dále by bylo vhodné vylepšit uživatelské rozhraní, a to jednak ve smyslu grafického návrhu, ale také čistším návrhem tříd, které je vytváří. Další vylepšení uživatelského rozhraní spočívá v zavedení lepší podpory pro malé obrazovky, aby aplikace hladce fungovala nejen na počítačových tabletech, ale i chytrých telefonech.

Ačkoliv navržená aplikace se snaží využívat pokud možno paralelního zpracování, které lze na moderních vícejádrových procesorech s výhodou využít ke zvýšení výkonu, stále jsou zde možnosti pro jeho masivnější nasazení – především v oblasti renderování výsledného obrazu, které představuje výpočetně nejnáročnější proces.

Další vývoj by se měl také zaměřit na implementaci ostatních algoritmů z oblasti zpracování obrazu. Ačkoliv většina základních byla již implementována, existuje stále mnoho užitečných neimplementovaných. Jako příklad uveďme bodové operace nevyužívající barevného modelu RGB či warp a zkapalnění pro nelineární geometrické transformace. Praktické využití by také určitě našli algoritmy pro korekci deformace způsobené objektivem fotoaparátu, ale i automatické funkce pro retušování nedokonalostí obrazu. V budoucí verzi by rozhodně neměla chybět možnost importu dalšího obrazu jako nové vrstvy. Uživatelé by dále jistě ocenily možnost většího výběru hrotů pro štětce, stejně jako import vlastních hrotů.

Seznam použitých zdrojů

[1] TRUSSELL, H. J. VRHEL, M. J. *Fundamentals of Digital Imaging.* Cambridge: Cambridge University Press, 2008. 532 s. ISBN 978-0-511-45518-6

[2] WEST, Angela. 20 *Years of Adobe Photoshop* [online]. 2010-02-01 [cit. 2013-04-25] <http://www.webdesignerdepot.com/2010/02/20-years-of-adobe-photoshop/>

[3] SCHUSTEK, Len. *Adobe Photoshop Source Code* [online]. 2013-02-13 [cit. 2013-04-25] <http://www.computerhistory.org/atchm/adobe-photoshop-source-code/>

[4] ADOBE. *Adobe Photoshop: Help and tutorials.* [online]. [cit. 2013-0425] <helpx.adobe.com/pdf/photoshop_reference.pdf>

[5] GOOGLE PLAY. *Adobe Photoshop Touch* [online]. [cit. 2013-04-25] <https://play.google.com/store/apps/details?id=air.com.adobe.pstouch>

[6] TOMKINS, Mike. *Photoshop Touch 1.2 relaxes image size limits, but with a catch* [online]. 2012-05-15 [cit. 2013-04-25] <http://www.imaging-resource.com/news/2012/05/15/photoshop-touch-1.2-relaxes-image-size-limits-but-with-a-catch>

[7] GOOGLE PLAY. *Autodesk SketchBook Pro* [online]. [cit. 2013-04-25] <https://play.google.com/store/apps/details?id=com.adsk.sketchbookhd>

[8] GOOGLE PLAY. *Nejlepší bezplatné v kategorii fotografie* [online]. [cit. 2013-04-25] <https://play.google.com/store/apps/category/PHOTOGRAPHY/collection/topselling_free>

[9] GOOGLE PLAY. *PicsArt – Photo Studio* [online]. [cit. 2013-04-25] <https://play.google.com/store/apps/details?id=com.picsart.studio>

[10] GOOGLE PLAY. *Pixlr Express* [online]. [cit. 2013-04-25] <https://play.google.com/ store/apps/details?id=com.pixlr.express>

[11] GOOGLE PLAY. *Pixlr-o-matic* [online]. [cit. 2013-04-25] <https://play.google.com/store/apps/details?id=pixlr.OMatic>

[12] W3Schools. *SVG Line* [online]. [cit. 2013-04-25] <http://www.w3schools.com/svg/svg_line.asp>

[13] MARTIŠEK, Dalibor. *Matematické principy grafických systémů.* Brno: Littera Brno, 2002. 296 s. ISBN 80-85763-19-2

[14] SINGH, AMAR. *HSL/HSV/HSI Blog* [online]. 2013-02-20 [cit. 2013-05-13]. <http://amarsinghhpcs.blogspot.cz/2013/02/hslhsvhsi-blog.html>

[15] PORTER, T. DUFF, T. Compositing Digital Images. *SIGGRAPH Comput. Graph.* Jul 1984, vol. 18, no. 3, s.253-259. Dostupné také z <http://doi.acm.org/10.1145/964965.808606>

[16] ADOBE. *Adobe Photoshop - Blending Modes* [online]. [cit. 2013-04-25] <http://help.adobe.com/en_US/photoshop/cs/using/WSfd1234e1c4b69f30ea53e41001031ab64-77eba.html>

[17] CHASTAIN, Sue. *Blending Mode Introduction* [online]. [cit. 2013-04-25] <http://graphicssoft.about.com/od/glossary/ig/Blending-Modes/Blending-Mode-Introduction.htm>

[18] GIMP. *Layer Modes* [online]. [cit. 2013-04-25] <http://docs.gimp.org/en/gimp-concepts-layer-modes.html>

[19] INLAND STUDIOS. Photoshop Blend Mode Math [online]. 2010-11-17 [cit. 2013-04-25] <http://inlandstudios.com/en/?p=851>

[20] JENSEN, Kevin. *Let's Learn Math: Photoshop Blend Modes* [online]. 2012-04-01 [cit. 2013-04-25] <http://www.venture-ware.com/kevin/coding/lets-learn-math-photoshop-blend-modes/>

[21] SMITH, Alvy Ray. Image Compositing Fundamentals. *Microsoft Technical Memo 4*. 15 Aug 1995. Dostupné z <http://alvyray.com/Memos/CG/Microsoft/4_comp.pdf>

[22] ADOBE. *Adobe Photoshop – Režimy prolnutí* [online]. [cit. 2013-04-25] <http://help.adobe.com/cs_CZ/photoshop/cs/using/WSfd1234e1c4b69f30ea53e41001031ab64-77eba.html>

[23] BURGER, Wilhelm. BURGE, Mark J. *Principles of Digital Image Processing: Fundamental Techniques*. Londýn: Springer-Verlag, 2009. 260 s. ISBN 978-1-84800-190-9

[24] BURGER, Wilhelm. BURGE, Mark J. *Principles of Digital Image Processing: Core Algorithms*. Londýn: Springer-Verlag, 2009. 327 s. ISBN 978-1-84800-194-7

[25] PRESS, William H. Teukolsky, Saul A. VETTERLING, William T., FLANNERY, Brian P. *Numerical Recipes in C: The Art Of Scientific Computing*. 2nd ed. Cambridge: Cambridge University Press, 1992. 994 s. Dostupné také z <http://www.it.uom.gr/teaching/linearalgebra/NumericalRecipiesInC/> ISBN 0-521-43108-5

[26] RUSS, John C. *The Image Processing Handbook.* Boca Raton: CRC Press, 2011. 838 s. ISBN-13 978-1-4398-4063-4

[27] WANG, Ruye. *Sharpening* [online]. 2012-11-15 [cit. 2013-04-23] <http://fourier.eng.hmc.edu/e161/lectures/gradient/node1.html>

[28] WANG, Ruye. *Laplacian Of Gaussian (LoG)* [online]. 2012-11-15 [cit. 2013-04-23] < http://fourier.eng.hmc.edu/e161/lectures/gradient/node9.html>

[29] WANG, Ruye. *Difference Of Gaussian (DoG)* [online]. 2012-11-15 [cit. 2013-04-23] <http://fourier.eng.hmc.edu/e161/lectures/gradient/node1.html>

[30] CINAR, Onur. *Android Apps with Eclipse.* Berkeley: Apress, 2012. 357 s. ISBN 978-1-4302-4434-9

[31] Open Handset Alliance. *Alliance Members* [online]. [cit. 2013-04-25] <http://www.openhandsetalliance.com/oha_members.html>

[32] GARGENTA, Marko. *Learning Android.* 1st ed. Sebastopol, California: O'Reily, 2011. 245 s. ISBN 978-1-449-39050-1.

[33] Android.com. *App Framework* [online]. [cit. 2013-04-25] <http://developer.android.com/about/versions/index.html>

[34] LEE, Wei-Meng H. *Beginning android 4 application development.* Indianapolis: Wiley pub., 2012. 533s. ISBN 978-1-118-19954-1.

[35] MEIER, Reto. *Professional Android 4 application development.* Indianapolis: John Wiley & Sons, 2012. 817 s. ISBN 978-1-118-10227-5.

[36] Android.com *Activities* [online]. [cit. 2013-04-25] <http://developer.android.com/guide/components/activities.html>

Seznam použitých zkratek a symbolů

Symbol/ zkratka	Význam
$c_{HCL \to RGB}$	Převodní funkce z HCL modelu do RGB modelu.
$c_{RGB \to HCL}$	Převodní funkce z RGB modelu do HCL modelu.
D	Označení dolní bitmapy při míchání.
DoG	Rozdíl Gaussovských rozostření (Difference of Gaussian).
$f(V_S[X,Y])$	Funkce představující bodovou operaci.
$f(V_S, V_D)$	Charakteristická funkce módu prolnutí.
$F(\boldsymbol{V_S})$	Funkce představující filtr.
$\boldsymbol{H}$	Matice určující lineární filtr.
HCL	Barevný model Hue Chroma Luminance (Odstín Barevnost Světlost).
HSL	Barevný model Hue Lightness Saturation (Odstín Světlost Sytost).
HSV	Barevný model Hue Saturation Value (Odstín Sytost Jas).
$I(x,y)$	Funkce představující interpolaci na souřadnicích x, y.
JNI	Java Native Interface, rozhraní Javy pro volání nativních funkcí.
LoG	Laplacián Gaussova vztahu (Laplacian of Gaussian).
$\boldsymbol{M}$	Matice určující lineární transformaci.
R	Označení výsledné bitmapy při míchání.
RGB	Barevný model Red Green Blue (Červená Zelená Modrá).
S	Označení horní bitmapy při míchání.
T	Geometrická transformace.
V	Hodnota kanálu pixelu.
$\boldsymbol{V}$	Množina hodnot stejného kanálu více pixelů.
VM	Virtuální Stroj (Virtual Machine).
X, Y	Celočíselné obrazové souřadnice.
x, y	Reálné obrazové souřadnice.
α	Alpha, hodnota průhlednosti pixelu nebo masky.

Seznam příloh

Samostatné přílohy

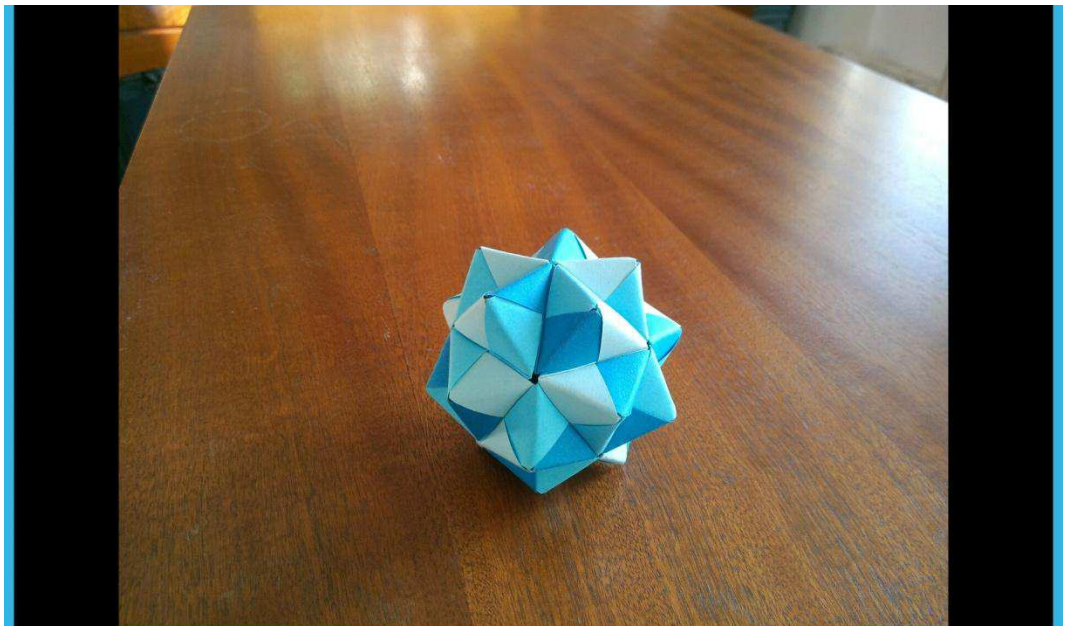
A Ukázky použití aplikace

Tato příloha obsahuje několik ukázek použití aplikace. Úpravy byly prováděny na zařízení ASUS TF700T se systémem Android 4.0.3, které bylo také použito k pořízení upravovaných fotografií. Všechny fotografie byly pořízeny v rozlišení 8 MPx.

A.1 Origami

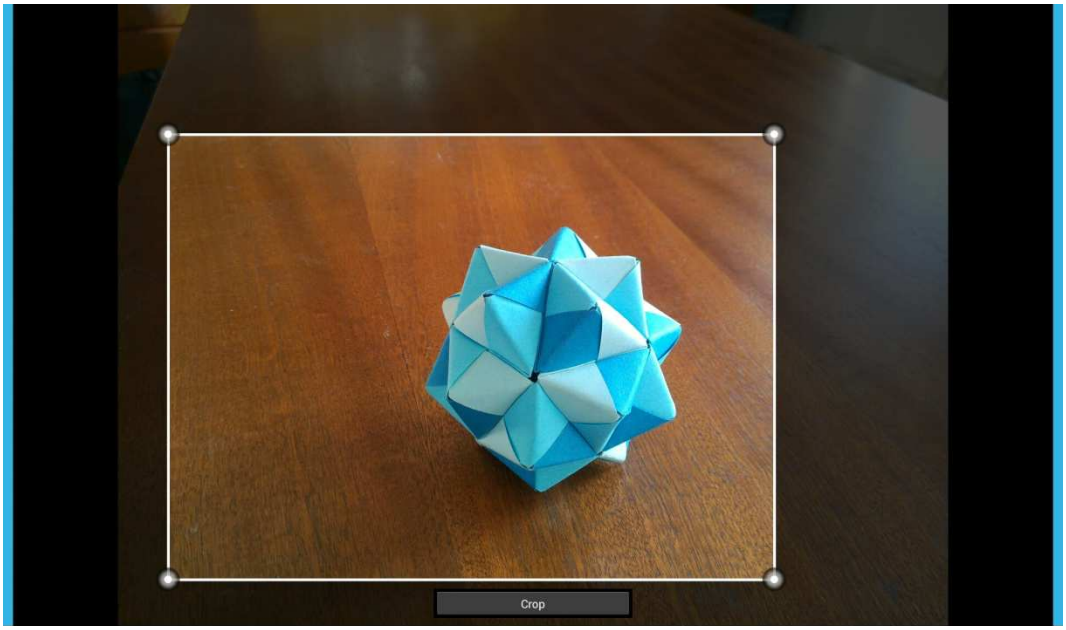
Tato ukázka představuje některé možnosti aplikace na úpravě fotografie origami „kostky“ z papíru. Cílem je zvýraznění fotografovaného objektu a případně jeho další oživení. Celý proces je proveden tak, že všechny jeho kroky jsou dále parametrizovatelné.

Prvním krokem je načtení fotografie z galerie pomocí nástroje „File operations“ a volby „Import image from gallery“. Načtená fotografie viz obr. A.1.



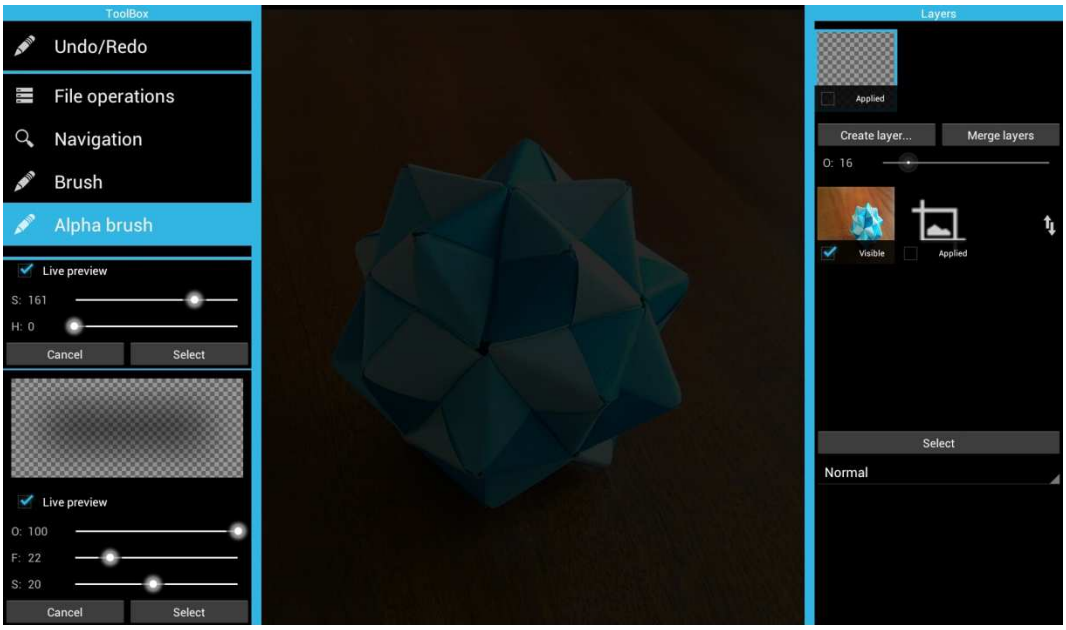
Obr. A.1 Načtená fotografie.

V dalším kroku provedeme ořez. K tomuto účelu je k dispozici nástroj „Crop“ dostupný z nabídky nástroje „Geometric transformation“. Příslušná akce je zobrazena na obr. A.2. Oblast ořezu určíme přetažením rohů obdélníka a potvrdíme stiskem tlačítka „Crop“.

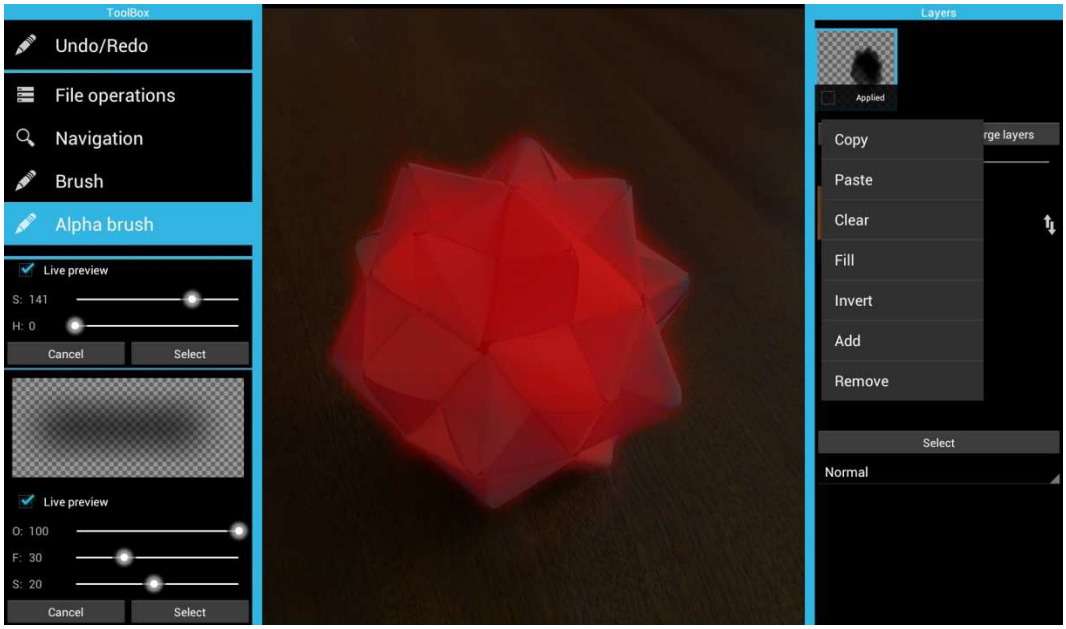


Obr. A.2 Ukázka ořezu obrazu.

V dalším kroku potlačíme pozadí obrázku rozmazáním. Abychom dodali fotografii hloubku, budeme chtít aplikovat rozmazání v menší míře i na vzdálenější části kostky. Za tímto účelem nejprve vybereme kostku (abychom si usnadnili práci, zmenšíme krytí fotografie (obr. A.3) – výběr je poté výraznější). K výběru použijeme nástroj „Alpha brush“ s měkkým hrotem a malým krytím a tokem, který nám umožní dobře kontrolovat intenzitu výběru. Ve skutečnosti je snazší vybrat to, co má zůstat ostré. Vybereme tedy kostku a pomocí kontextové nabídky výběr invertujeme (obr. A.4).

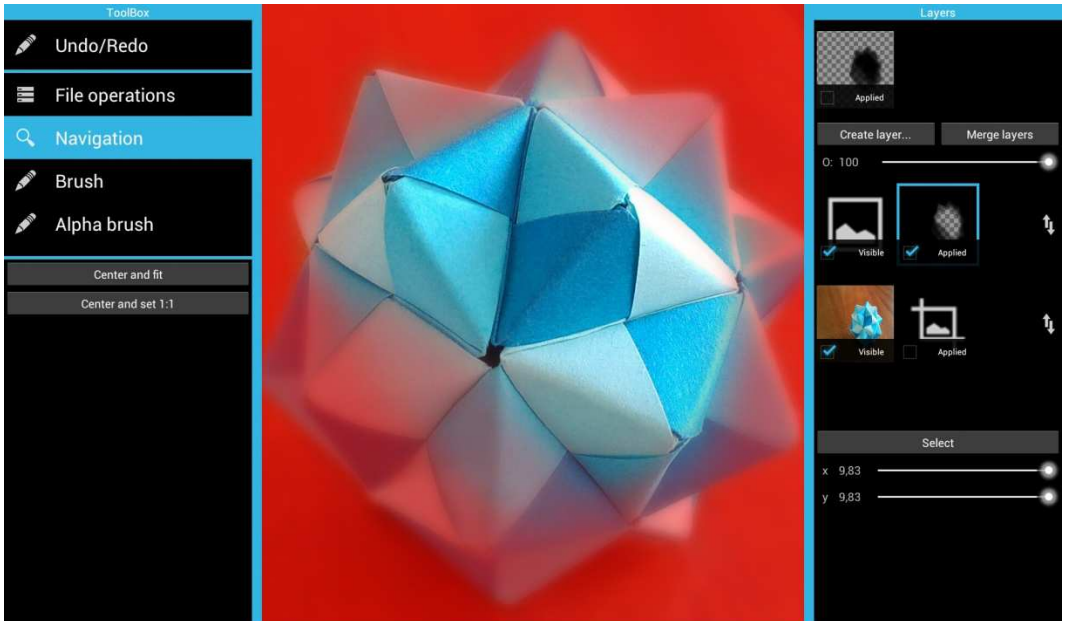


Obr. A.3 Ztmavení obrazu pomocí krytí vrstvy pro lepší viditelnost masky.



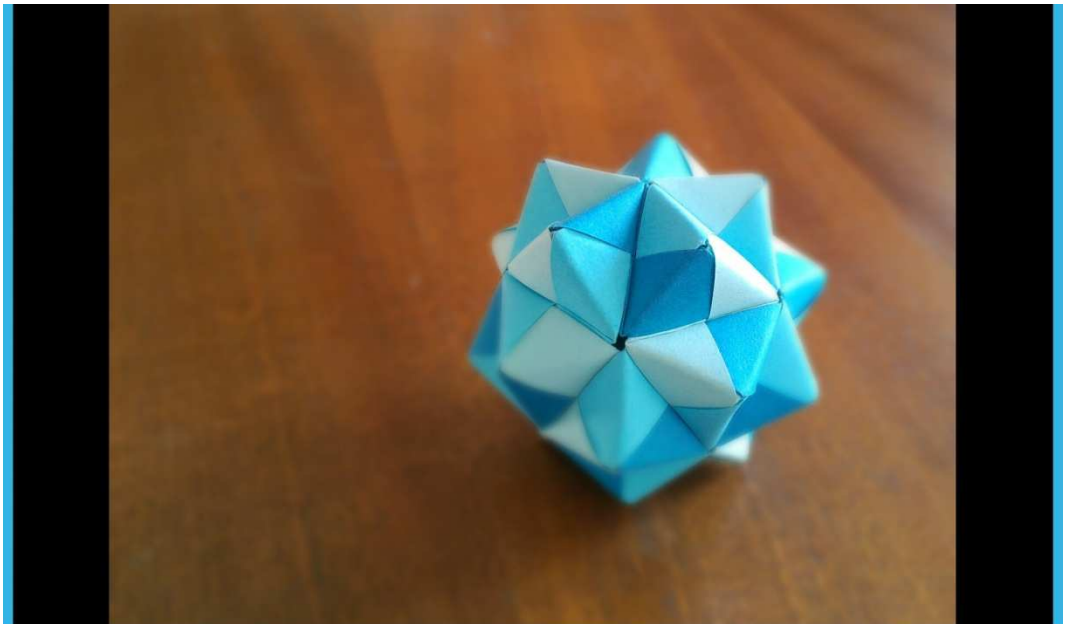
Obr. A.4 Vybraná kostka s ukázkou kontextové nabídky výběru.

Aplikace filtru přes výběr je ovšem nevhodná, protože bychom znehodnotili původní obraz a nebyli dále schopni měnit parametry a rozsah rozostření. Vytvoříme tedy vrstvu pro filtr Gaussovského rozostření a pomocí kontextové nabídky zkopírujeme masku výběru do masky vrstvy. Masku pak dále upravujeme, dokud nám rozsah rozostření nevyhovuje (obr. A.5).



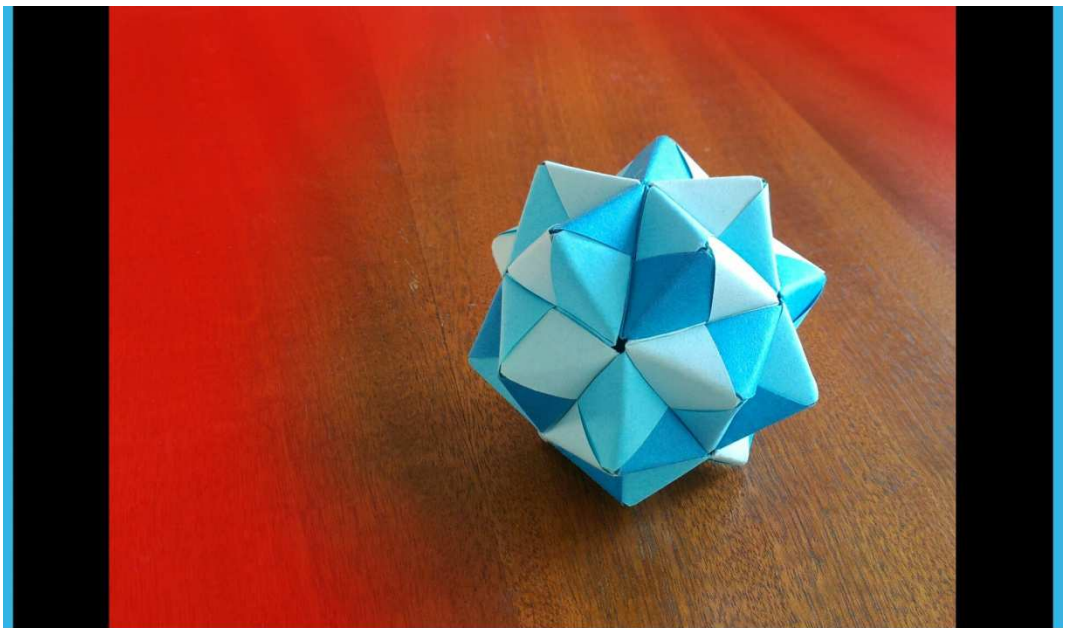
Obr. A.5 Konečná podoba masky pro rozostření.

Výsledek námi vytvořeného rozostření je na obr. A.6.

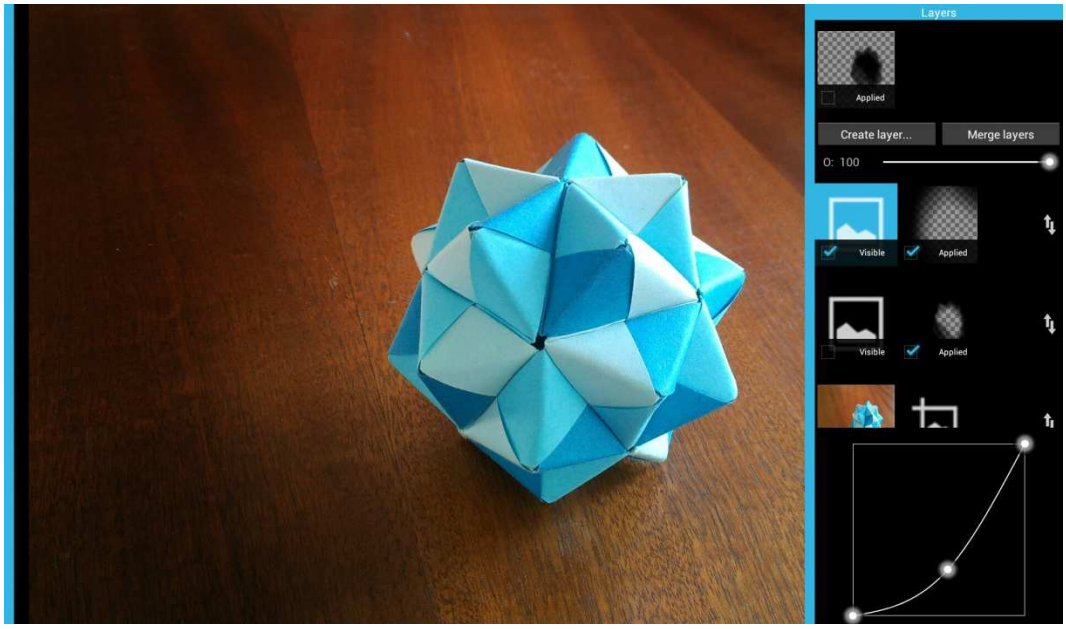


Obr. A.6 Výsledek aplikace Gaussovského rozostření s aplikovanou maskou.

Obraz takto dostal jistou hloubku, ale kostka stále nevystupuje dostatečně do popředí. Abychom obraz ještě vylepšili, vytvoříme vrstvu pro úpravu křivek, pomocí které ztmavíme okraje fotografie. Opět požadujeme plynulý přechod, proto přiřadíme vrstvě vhodnou masku známým způsobem (obr. A.7). Výsledek je na obr. A.8.

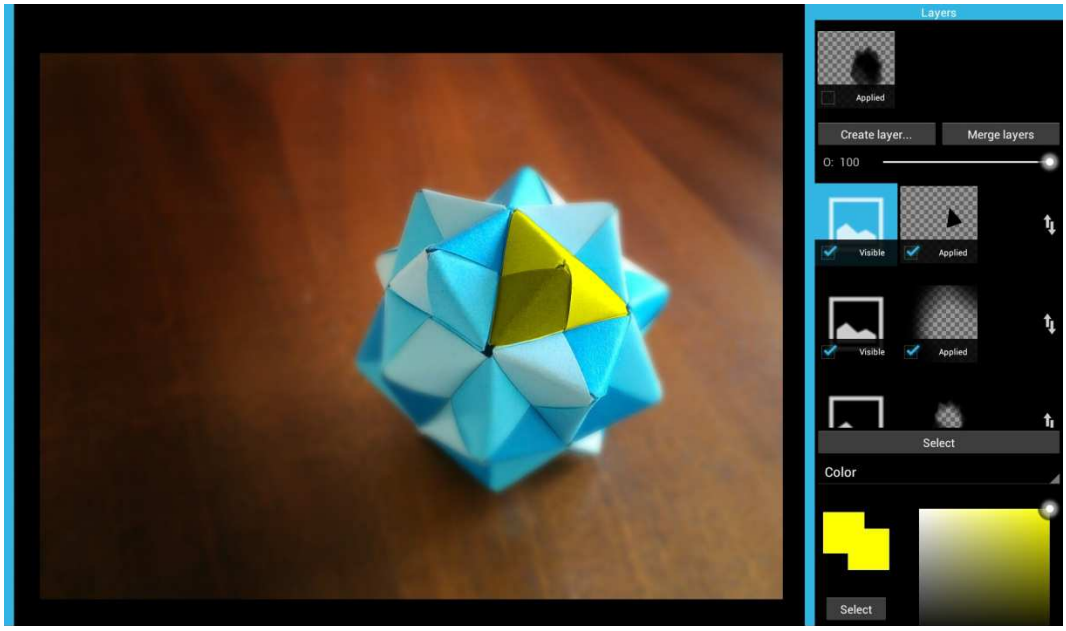


Obr. A.7 Maska pro ztmavení okrajů fotografie.



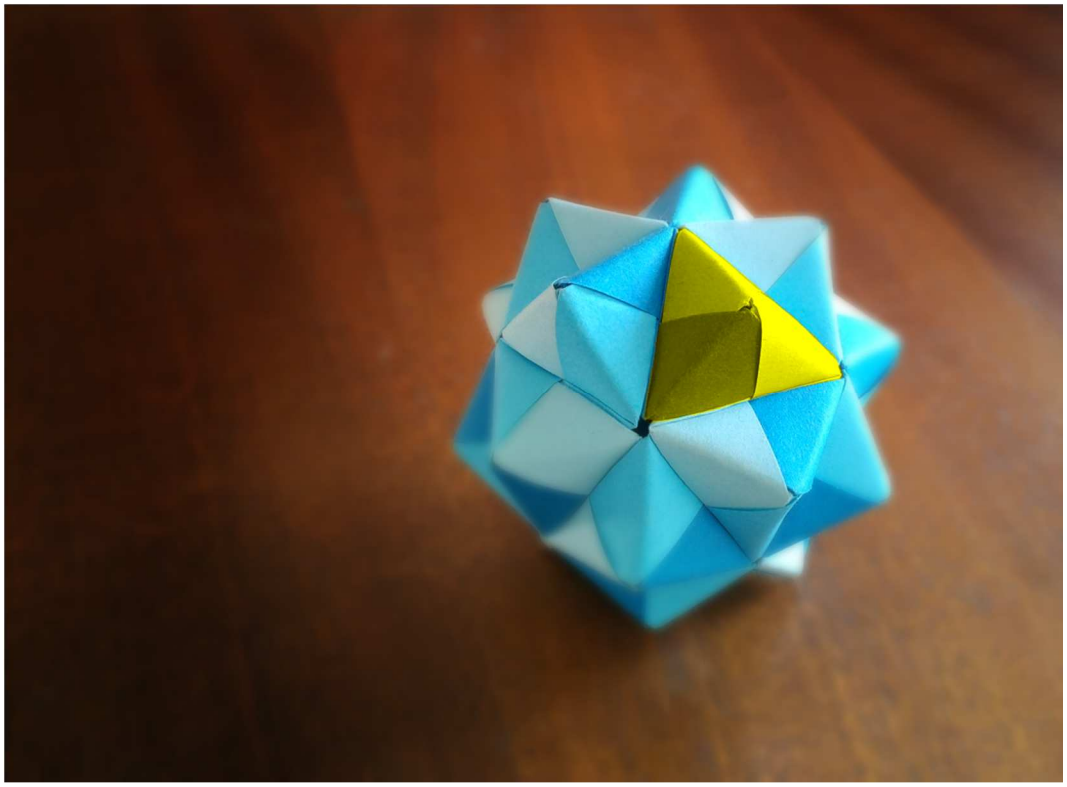
Obr. A.8 Nastavení vrstvy křivek a výsledek její aplikace na obraz.

Nyní ještě pro oživení fotografie můžeme přebarvit jeden roh kostky na jinou barvu. Tohoto efektu díky pokročilým módům prolnutí (konkrétně módu Barva (Color)) dosáhneme relativně snadno, a to pomocí vrstvy výplně a masky vrstvy, která určuje ovlivněnou oblast (obr. A.9).

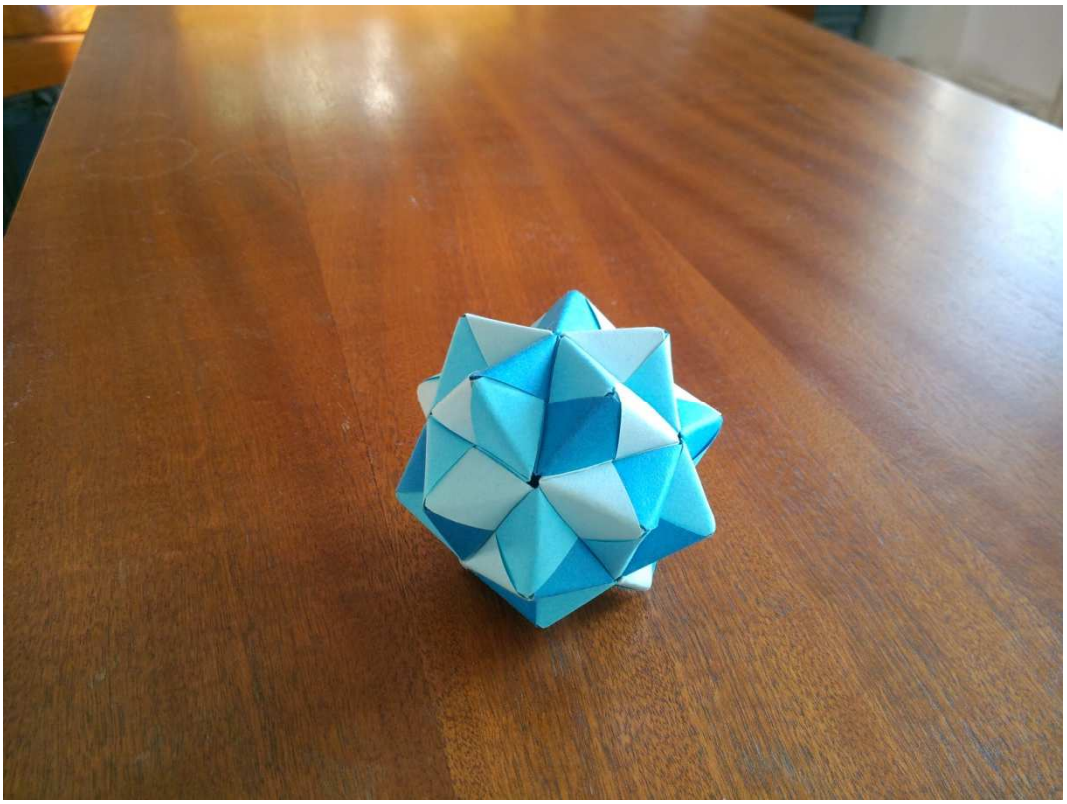


Obr. A.9 Výsledek aplikace vrstvy výplně v módu Barva (Color) na roh kostky.

Konečný výsledek ukazuje obr. A.10. Na obr. A.11 je pro porovnání původní fotografie.



Obr. A.10 Výsledek úprav.



Obr. A.11 Původní fotografie.

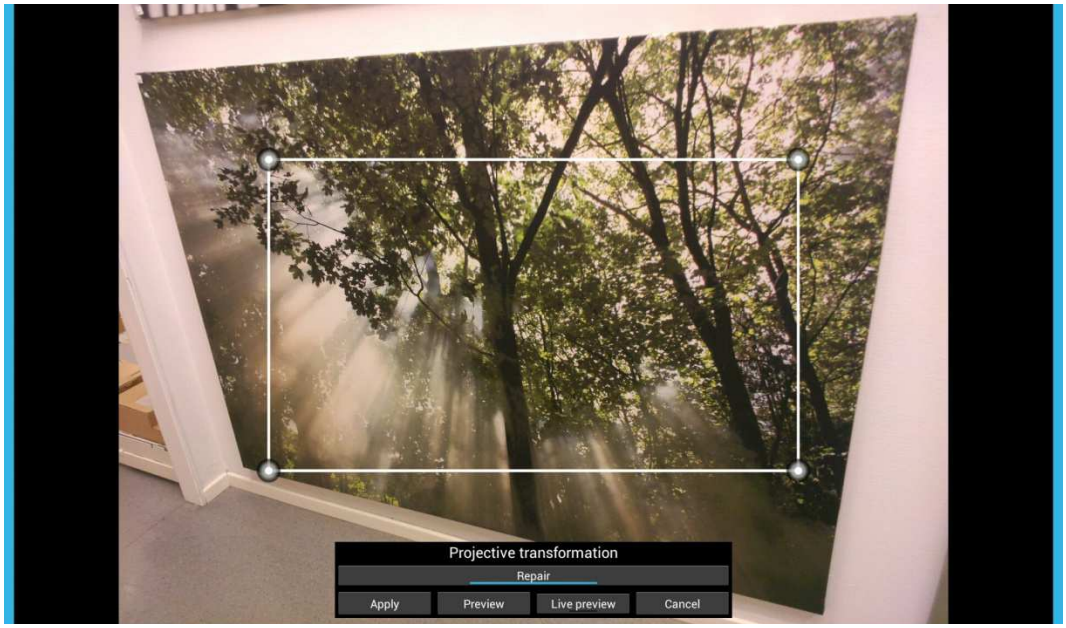
A.2 Fotografie obrazu

Tato ukázka demonstruje především použití nástrojů pro geometrické transformace a nástroje křivky. Pro jednoduchost nepoužijeme křivky jako vrstvu, ale jako nástroj. Bylo by však samozřejmě možné, pokud bychom chtěli provádět rozsáhlejší modifikace a k jejich aplikaci se vracet, použít křivky jako samostatnou vrstvu.

Budeme vycházet z fotografie obrazu, která byla pořízena v obchodním domě Ikea. Obraz byl umístěn v úzké uličce a byl široký dva metry, nebylo jej tedy možné vyfotit zepředu tak, aby se na fotografii vešel celý. Načtená fotografie je na obr. A.12. Obraz je na fotografii značně zdeformovaný a jeho barvy díky špatnému osvětlení také příliš neodpovídají reálným.

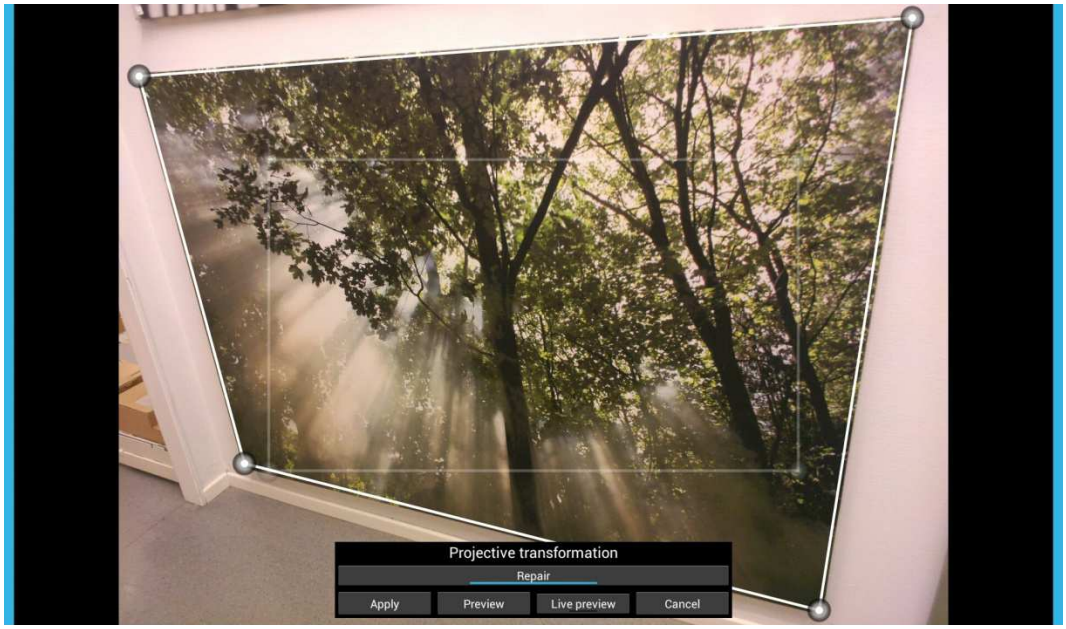
Obr. A.12 Načtená fotografie.

Pro odstranění deformace použijeme nástroj „Geometric transformation“, konkrétně volbu „Projective transform“. Po vybrání nástroje se na obrazovce objeví nabídka, kterou ukazuje obr. A.13. Tento nástroj provádí obecnou lineární transformaci, kterou určíme pomocí přetažení vrcholů čtyřúhelníku, který vidíme na obrazovce.



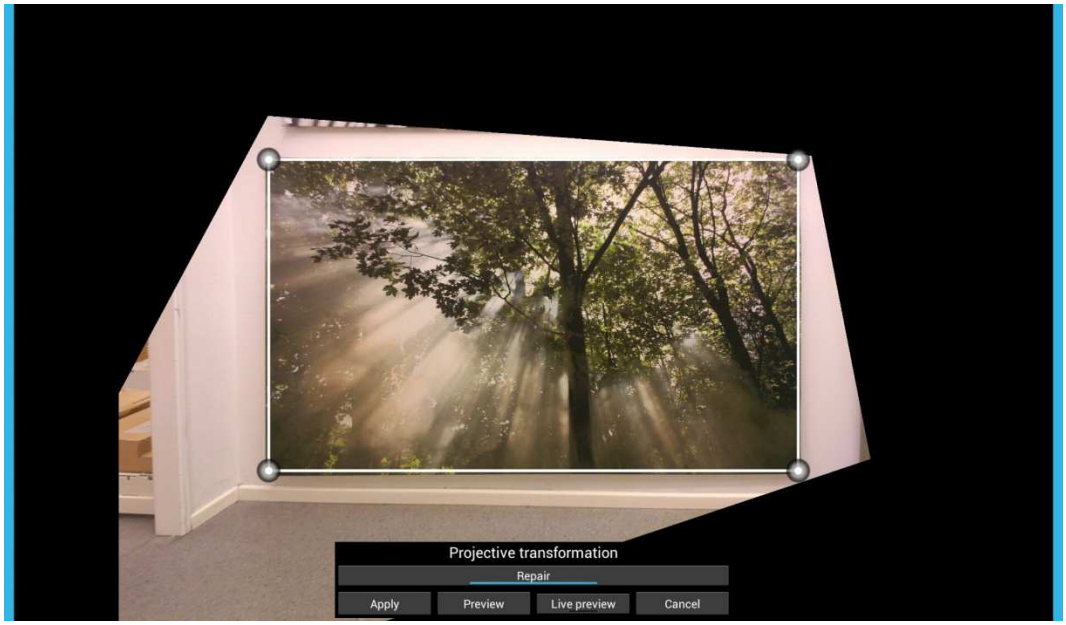
Obr. A.13 Nabídka pro nástroj „Projective transformation“.

Vzhledem k tomu, že chceme deformaci opravit, zapneme přepínač „Repair“. Nyní se bude námi vybraný čtyřúhelník transformovat na původní obdélník. Tažením tedy přesuneme rohové body čtyřúhelníku tak, aby odpovídali rohovým bodům obrazu, jak je naznačeno na obr. A.14 a operaci potvrdíme.



Obr. A.14 Výběr oblasti obrazu pro transformaci.

Výsledek operace ukazuje obr. A.15.

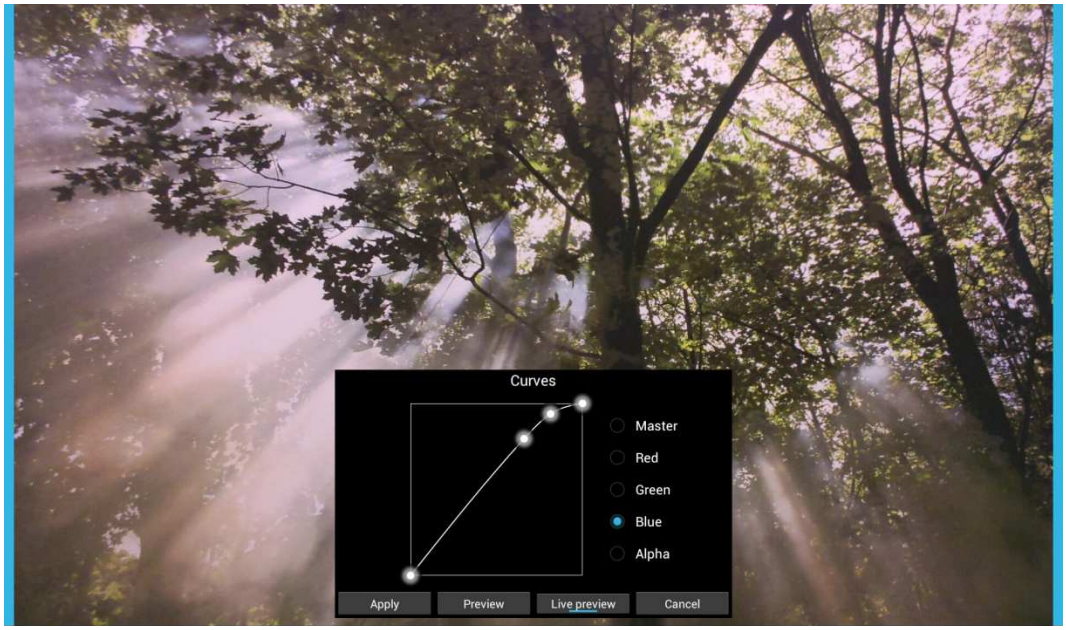


Obr. A.15 Výsledek transformace.

Obraz ořízneme obvyklým způsobem (obr. A.16). Poté budeme ještě chtít upravit barvy obrazu (požadovaný výsledek je modrá obloha a odstranění sépiového nádechu). Pro dosažení požadovaného výsledku použijeme nástroj „Point operation“, konkrétně volbu „Curves“.

Obr. A.16 Ořez obrazu.

Nejprve zvýšíme intenzitu modré barvy ve světlech (obr. A.17).

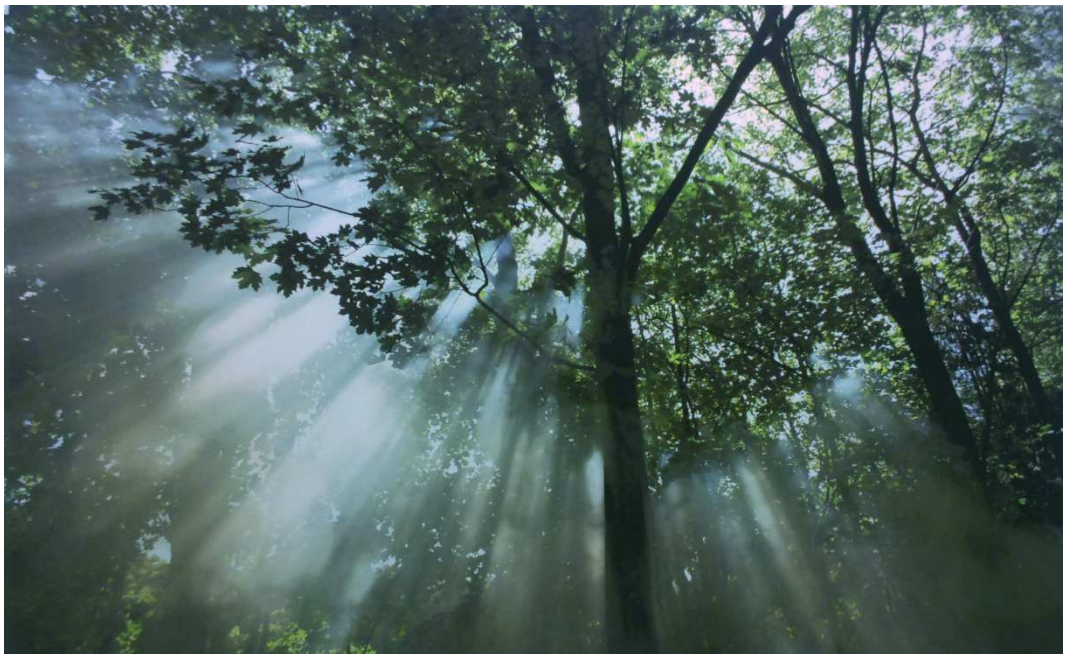


Obr. A.17 Nastavení křivek pro modrý kanál.

Poté pro odstranění růžového nádechu snížíme intenzitu červené, především pak ve světlech (obr. A.18).

Obr. A.18 Nastavení křivek pro červený kanál.

Výsledný obraz je na obr. A.19, pro porovnání je na obr. A.20 uveden i obraz původní.



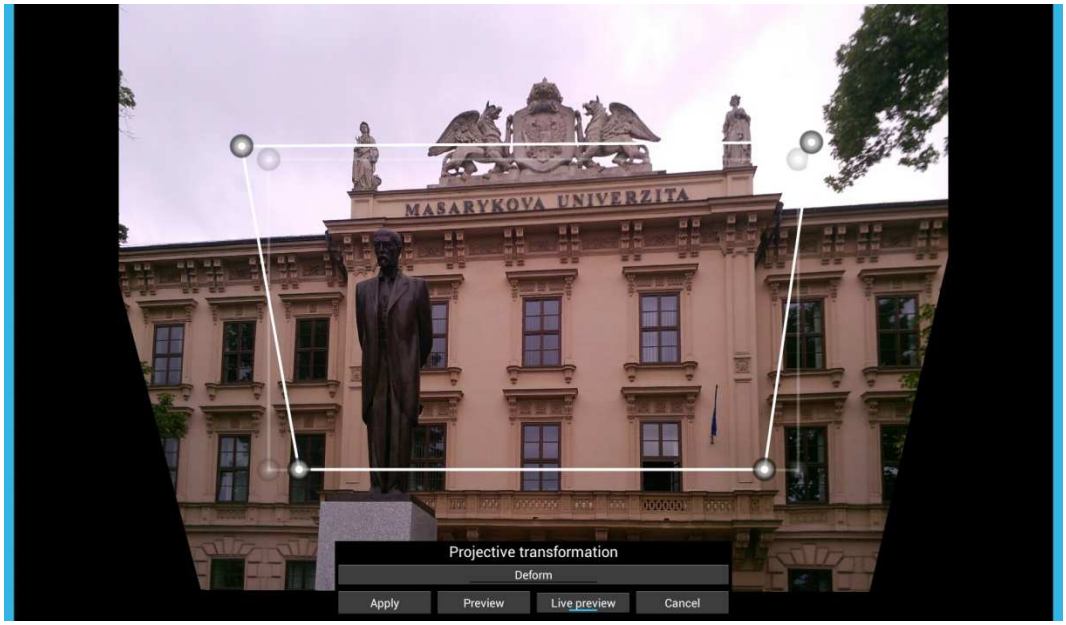
Obr. A.19 Výsledek úprav.

Obr. A.20 Původní fotografie.

A.3 Masarykova Univerzita

Cílem této ukázky je vyrovnat deformaci fotografie (obr. A.21) a dále prosvětlit sochu T. G. Masaryka, která se v obraze poněkud ztrácí.

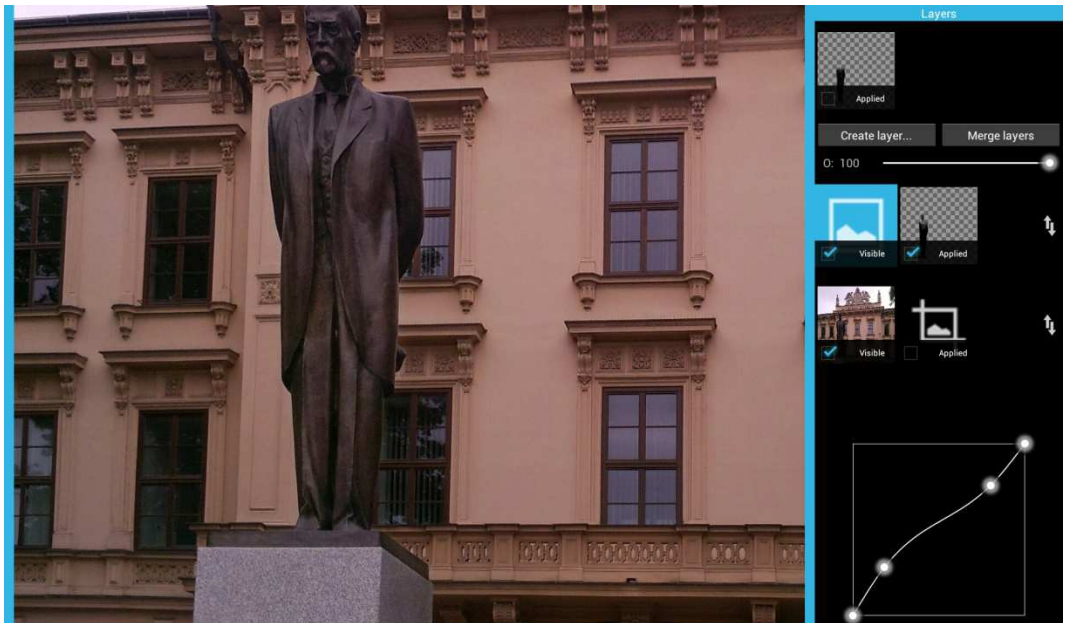
Obr. A.21 Načtená původní fotografie.



Obr. A.22 Nastavení transformace.

Pro korekci geometrické transformace využijeme opět nástroj „Projective transform“, pomocí kterého budeme obraz deformovat (využijeme funkce živého náhledu – obr. A.22). Pro prosvětlení sochy využijeme vrstvu křivek s aktivní maskou, (obr. A.23). Masku použijeme jen tam, kde je zesvětlení nutné (ramena sochy zesvětlit nepotřebují). Detail aplikace vrstvy viz obr. A.24.

Obr. A.23 Maska pro prosvětlení sochy.



Obr. A.24 Výsledek aplikace vrstvy křivek.

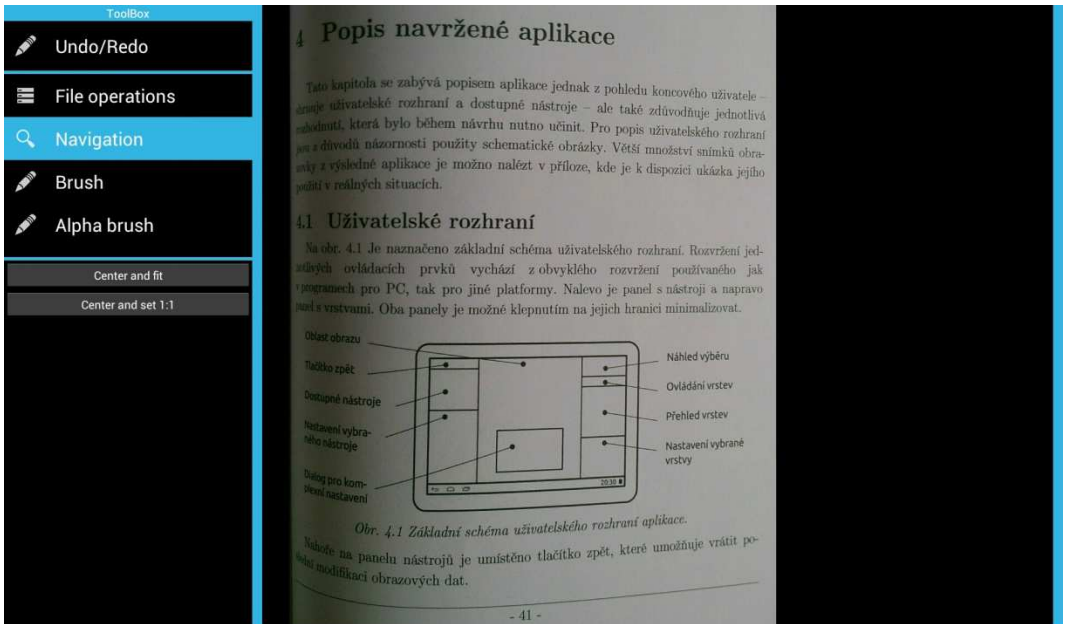
Výsledný obraz je na obr. A.25, pro porovnání je na obr. A.26 uveden i obraz původní.

Obr. A.25 Výsledek úprav.

Obr. A.26 Původní fotografie.

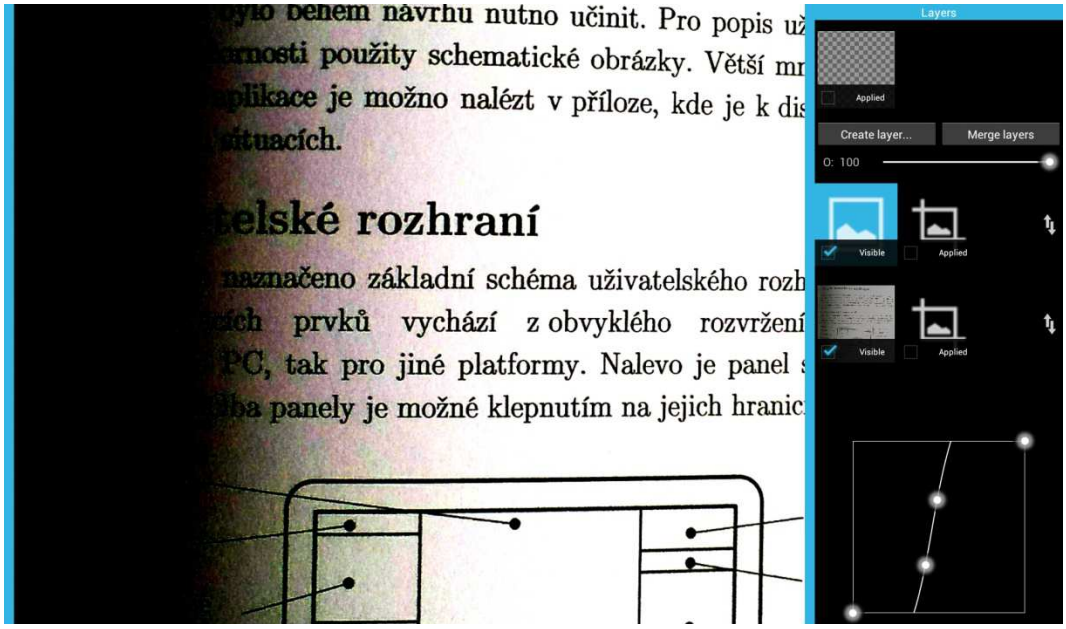
A.4 Zvýraznění textu

Tato ukázka kombinuje použití filtrů určených k zostření obrazu a vrstev křivek a výplně za účelem zvýraznění špatně nafoceného textu. Jedná se o stránku z této diplomové práce, která je jednak křivě nafocená, ale také nevhodně nasvětlená. Nejprve pomocí známých nástrojů pro deformaci a ořez stránku vyrovnáme a odstraníme nadbytečné oblasti (obr. A.27).

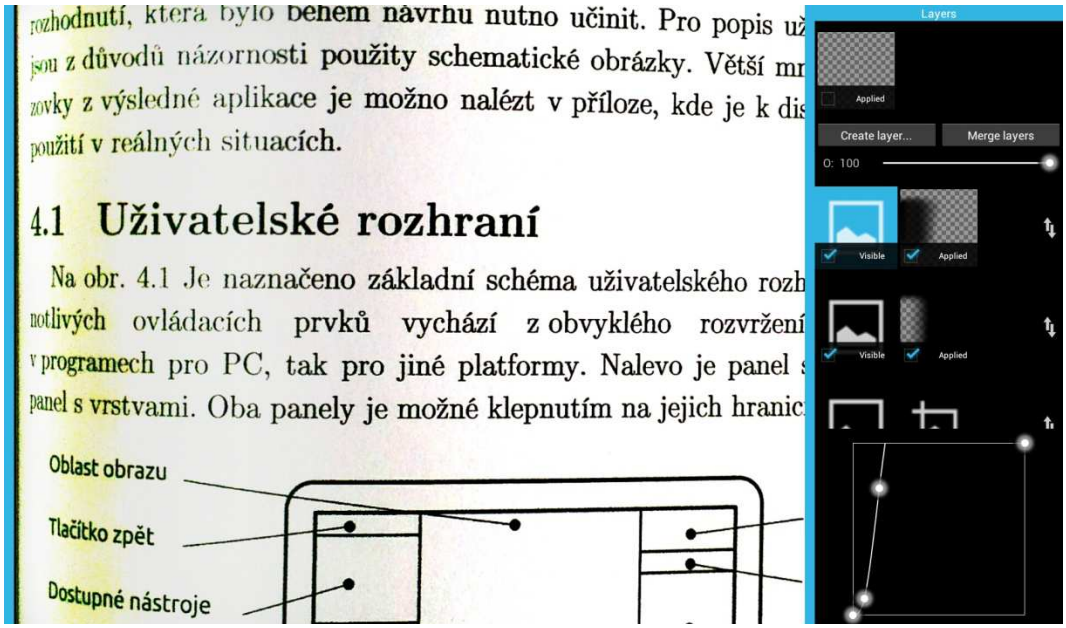


Obr. A.27 Načtená a zkorigovaná fotografie.

Poté se pokusíme pomocí vrstvy křivek vytáhnout text stránky. Jak je vidět, vzhledem k nerovnoměrnému nasvícení toto není možné (obr. A.28). Musíme proto použít dvě vrstvy křivek s maskami, které se částečně překrývají. Pro snadnou práci využijeme možnost kopírovat a invertovat masky pomocí kontextové nabídky. Pomocí velmi agresivního nastavení vrstvy křivek pro tmavou oblast jsme dostali relativně přijatelné výsledky (obr. A.29), ovšem v obraze je nyní velké množství žluté barvy.

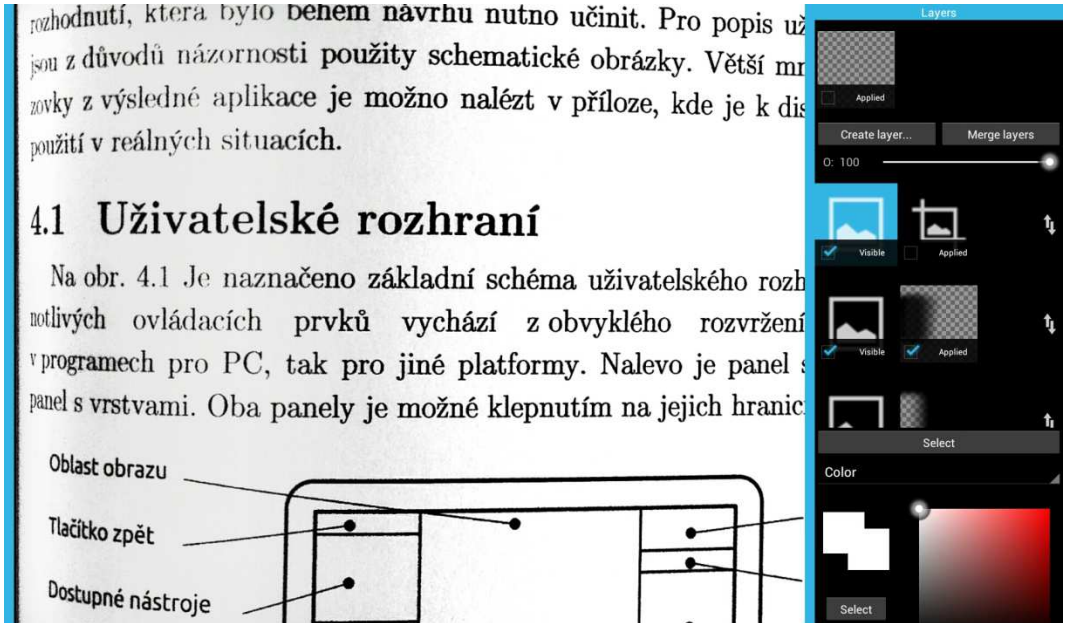


Obr. A.28 Aplikace vrstvy křivek na celý obraz.



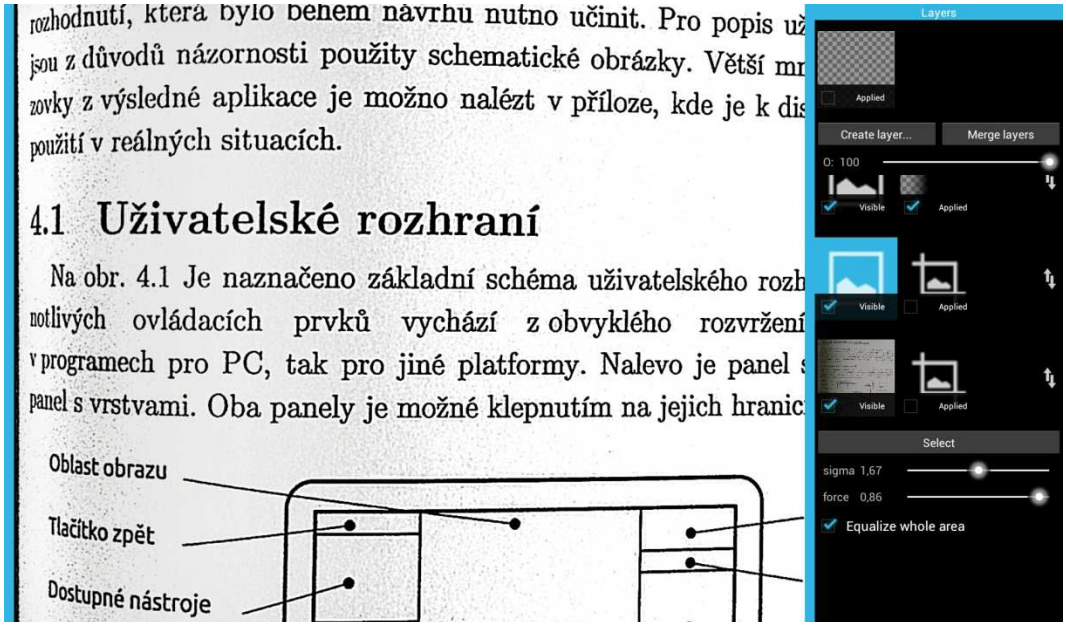
Obr. A.29 Výsledek použití dvou vrstev křivek s příslušnými maskami.

Abychom odstranili žlutý pruh, vytvoříme přes celý obraz vrstvu výplně. Této vrstvě přiřadíme bílou barvu a mód prolnutí Color (Barva). Výsledek aplikace vrstvy je na obr. A.30.



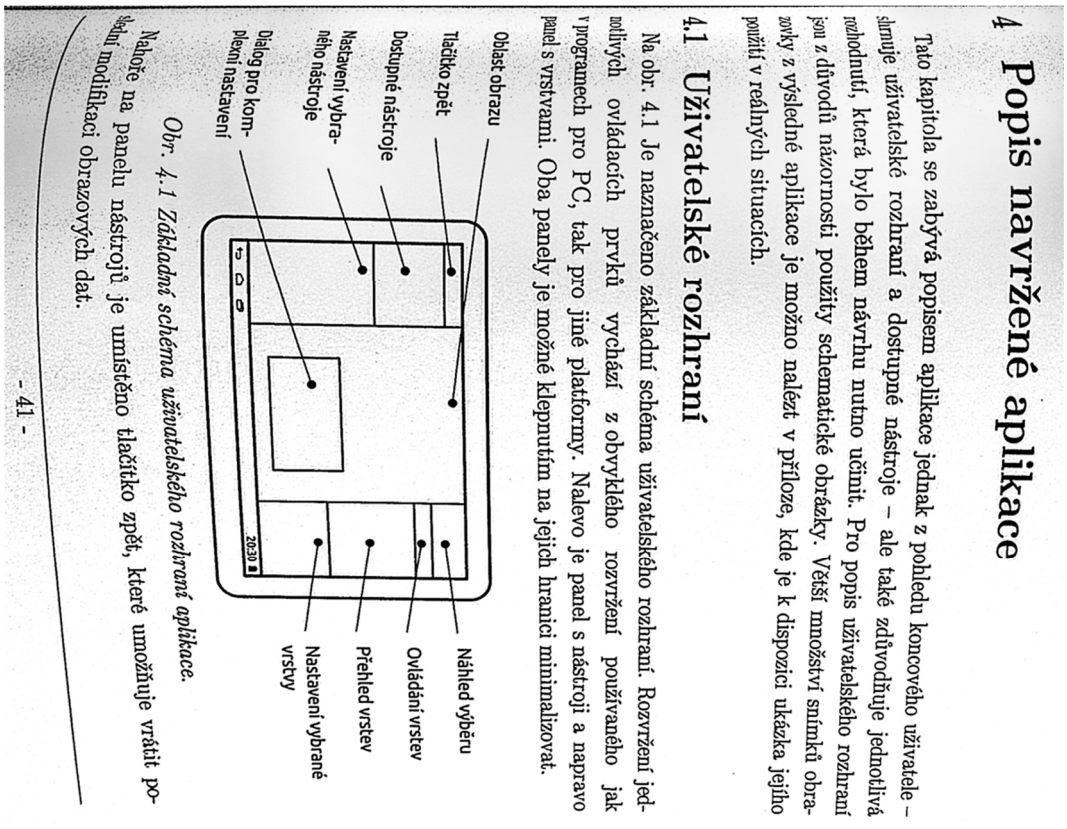
Obr. A.30 Výsledek s použitím vrstvy výplně.

Stále však výsledky nejsou dokonalé. V levém horním rohu a v místě ohybu listu je písmo velmi slabé. Toto již nezlepší žádné další nastavení stávajících vrstev. Můžeme tedy buď přidat další vrstvu křivek pro danou oblast (nutné vytvářet masku a upravovat masky stávající), nebo přidat vrstvu s filtrem zostření. Obr. A.31 ukazuje přidání vrstvy pro filtr LoG s vhodně nastavenými parametry (mezi původní obraz a vrstvy křivek), která danou situaci řeší bez nutnosti použití masky.



Obr. A.31 Výsledek aplikace filtru LoG pod vrstvami křivek.

Výsledný obraz je na obr. A.32, pro porovnání je na obr. A.33 uveden i obraz původní, včetně deformace.



4 Popis navržené aplikace

Tato kapitola se zabývá popisem aplikace jednak z pohledu koncového uživatele – shrnuje uživatelské rozhraní a dostupné nástroje – ale také zdůvodňuje jednotlivá rozhodnutí, která bylo během návrhu nutno učinit. Pro popis uživatelského rozhraní jsou z důvodů názornosti použity schematické obrázky. Větší množství snímků obrazovky z výsledné aplikace je možno nalézt v příloze, kde je k dispozici ukázka jejího použití v reálných situacích.

4.1 Uživatelské rozhraní

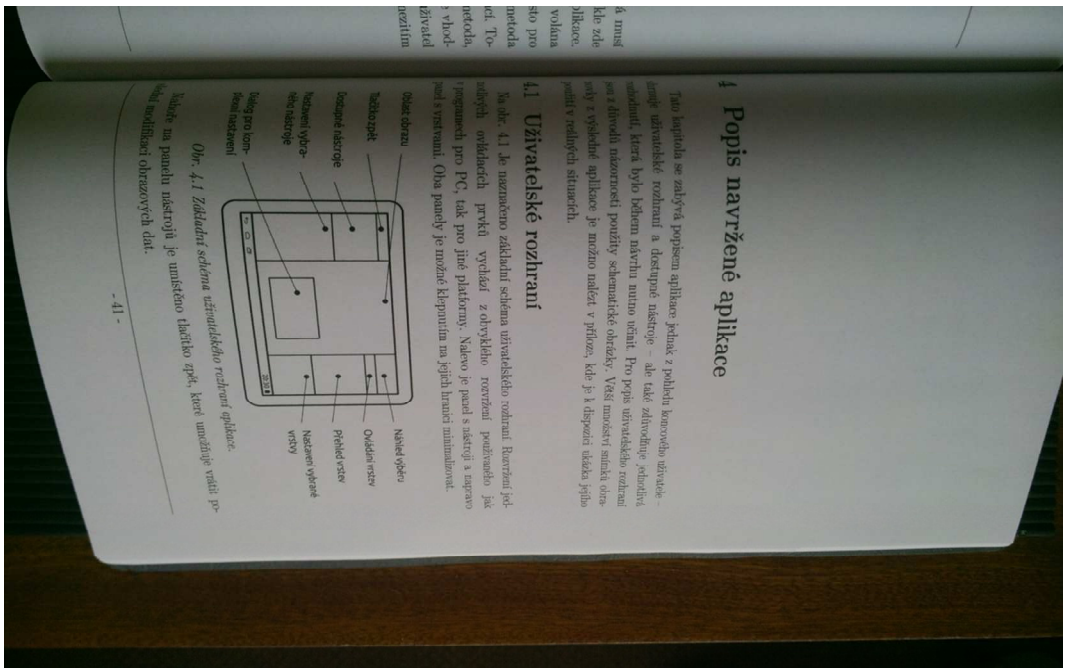
Na obr. 4.1 Je naznačeno základní schéma uživatelského rozhraní. Rozvržení jednotlivých ovládacích prvků vychází z obvyklého rozvržení používaného jak v programech pro PC, tak pro jiné platformy. Nalevo je panel s nástroji a napravo panel s vrstvami. Oba panely je možné klepnutím na jejich hranici minimalizovat.

Oblast obrazu
Tlačítko zpět
Dostupné nástroje
Nastavení vybraného nástroje
Dialog pro komplexní nastavení
Náhled výběru
Ovládání vrstev
Přehled vrstev
Nastavení vybrané vrstvy

Obr. 4.1 Základní schéma uživatelského rozhraní aplikace.

Nahoře na panelu nástrojů je umístěno tlačítko zpět, které umožňuje vrátit poslední modifikaci obrazových dat.

Obr. A.32 Výsledek úprav.



Obr. A.33 Původní fotografie.