Systém pro automatické určení tvaru historických artefaktů

System for automatic shape recognition of historical artifacts

Bc. Jiří Špaček

Diplomová práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jiří ŠPAČEK**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Systém pro automatické určení tvaru historických artefaktů.**

Zásady pro vypracování:

1. **Zpracujte literární rešerši na dané téma.**
2. **Vytvořte systém pro základní morfometrickou analýzu bifaciálních nástrojů.**
3. **Navrhněte vhodnou strukturu systému a grafický vzhled.**
4. **Vytvořte nástroj, který umožní automatické archivování a měření archeologických nálezů. Předmětem bude zpracování obrazu sestávající z prahování, segmentace a úpravy nalezených objektů a následná reprezentace tohoto objektu komplexním vektorem. Dále řešte problematiku měření velikosti a tvaru hrotu.**
5. **Vytvořte databázi statistických morfometrických dat vztahující se ke konkrétním archeologickým nálezům.**

Rozsah práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **Šonka, M., Hlaváč, V.: Počítačové vidění, Grada, Praha, 1992, ISBN 80-85424-67-3**
2. **Hlaváč, V., Sedláček, M.: Zpracování signálů a obrazů, Vydavatelství ČVUT, Praha, 2005, ISBN 80-01-03110-1**
3. **Kotek, Z.: Metody rozpoznávání a jejich aplikace, Academia, Praha, 1993, ISBN 80-200-0297-9**
4. **Pratt, W. K.: Digital Image Processing, Second Edition. New York: Wiley-Interscience, 1991.**
5. **Šonka, M., Hlaváč, V., Boyle, R.: Image Processing, Analysis and Machine Vision. Boston: PWS, 1998.**
6. **Sojka, E.: Digitální zpracování a analýza obrazů, VŠB-TU Ostrava, 2000.**
7. **Kerre, E., Nachtgeal, M.: Fuzzy Techniques in Image Processing, Springer 2000.**

Vedoucí diplomové práce: **Ing. Bronislav Chramcov, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **19. února 2010**

Termín odevzdání diplomové práce: **8. června 2010**

Ve Zlíně dne 19. února 2010

prof. Ing. Vladimír Vašek, CSc.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Diplomová práce se zabývá problematikou rozpoznání artefaktů neboli bifaciálních nástrojů ze zdrojových fotografií, zjištěním jejich základních rozměrů, archivaci těchto získaných dat a jejich následné prezentace pomocí uživatelského rozhraní. Při detekci artefaktu jsou postupně použity techniky zpracování obrazu, kdy je postupně aplikováno prahování a segmentace obrazu. Nalezená data jsou pak uložena do databáze.

K implementaci systému bylo použito vývojové prostředí Microsoft Visual Studio a databáze typu Microsoft SQL Server.

Klíčová slova:

Bifaciální nástroje, rozpoznání obrazu, prahování, segmentační techniky, detekce hran, rozhraní systému, MS Visual Studio, C# .NET, MSSQL.

ABSTRACT

This dissertation covers the area of artifacts recognition or bifacial tools from source photographs, detecting of original size, archiving of data gained and their subsequent presentation via user interface. When detecting an artifact, techniques of picture processing are sequentially applied. Starting with threshold method and proceeding with picture segmentation. Data gained are then saved into a database.

Microsoft Visual Studio as a development and Microsoft SQL Server as a database type were used for an implementation of the system.

Keywords:

Bifacial tools, picture processing, threshold method, segmentation techniques, edge recognition, system interface, MS Visual Studio, C# .NET, MSSQL.

Děkuji vedoucímu práce, panu Ing. Bronislavu Chramcovovi, Ph.D. za vedení a spolupráci při tvorbě této diplomové práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….

podpis diplomanta

OBSAH

ÚVOD

Pro většinu výzkumných profesí je nutností archivace nalezených či zjištěných údajů o předmětech jejich zájmu. Pro takovou archivaci dat je ideální systém, který bude dané profesi sestaven na míru a nebude třeba řešení různých kompromisů či nedostatků jiných softwarových aplikací. Proto vznikla tato diplomová práce.

Cílem zadání mé diplomové práce je realizace systému (uživatelského rozhraní), který by umožňoval přehlednou archivaci a katalogizaci historických artefaktů a to především kamenných hrotů nejrůznějších tvarů. Hlavním požadavkem na systém je schopnost z digitálních obrazů nalezených artefaktů získat požadované informace. Já, jako laik v této oblasti, jsem neměl představu, jaké jsou pro archeology v této oblasti klíčové informace. Proto došlo ke specifikaci požadavků na základě získaných informací z řad odborníků.

Obsah diplomové práce pojednává o jednotlivých postupech a metodách, které je třeba provést před tím, než dojde k samotnému rozpoznání artefaktu. Tato diplomová práce je rozdělena do několika hlavních kapitol, které se zadanou problematikou postupně zabývají. Nutno ještě předeslat, že jsem dopodrobna nepopisoval teorii návrhu informačního systému, která by se do této práce také dala zařadit. Teorie návrhu není hlavním cílem diplomové práce. V každé části návrhu zmiňuji jen minimum teoretických informací, které poslouží k popsání jejího obsahu. Pro analýzu a realizaci aplikace jsem si vystačil se svými zkušenostmi, které, jsem účelně využíval tam, kde je to obzvláště potřeba.

V první kapitole této práce popisuji problematiku zpracování obrazu, kde uvádím několik důležitých pojmů spjatých s digitálním obrazem, který bude použit jako hlavní vstup aplikace a jeho předzpracování pro další postupy rozpoznání artefaktu.

V druhé kapitole se zabývám problematikou segmentace obrazu a všech jejich nutných součástí, jako je nalezení ideálního kanálu zdrojového obrazu, následné zvolení ideální prahovací hodnoty, která má zabránit zkreslení vstupních dat při převodu na černobílý obraz a samotné segmentace či detekce jednotlivých částí zdrojového obrazu. V této kapitole uvádím řadu informací o detekčních metodách, které jsou pro segmentaci klíčové.

V prvních dvou kapitolách věnuji větší pozornost těm postupům, které budou později třeba pro samotnou realizaci systému.

V třetí kapitole přichází na řadu získání informací o detekovaném artefaktu z obrazu. Popisuji zde jednotlivé metody vedoucí k získání klíčových údajů o artefaktu, kterými jsou

jeho rozměry. Rozměry musí být přepočítány na základě detekovaní dle standardizované měrky, vrcholových úhlů artefaktu a také jeho obvodu.

V čtvrté kapitole se věnuji samotnému návrhu systému, který nesmí být opomenut, protože důkladná a kvalitní analýza systému později ulehčí práci při samotné realizaci systému. Součástí této analýzy či rozboru je seznam funkčních a nefunkčních požadavků na systém, který je produktem této diplomové práce.

V poslední páté kapitole se věnuji samotné realizaci uživatelského rozhraní systému, kde je popsáno ovládání a použití jednotlivých částí uživatelského rozhraní.

Při realizaci diplomové práce jsem dbal na to, aby uživatelské rozhraní bylo co nejvíce intuitivní a ergonomické a umožňovalo ovládání aplikace i po jednoduchém proškolení nového uživatele. Toto je jedno z hlavních kritérií nových systémů.

I. TEORETICKÁ ČÁST

1 ZPRACOVÁNÍ OBRAZU

První část diplomové práce je zaměřena na definování některých pojmů, na popsání způsobů zpracování obrazu a obeznámení se s jednotlivými metodami, které jsou s problematikou zpracování obrazu spojeny.

Při této diplomové práci je důležité pracovat s fotografiemi artefaktů, ze kterých budou získávány informace o jednotlivých artefaktech. Tyto digitální obrázky (fotografie) slouží jako datový vstup do aplikace a musí projít několika typy zpracování, tak aby byly připraveny na rozpoznání jejich obsahu. Digitální obrázky jsou ve formátu RGB. Je třeba je převést do formátu, se kterým se lépe pracuje. Tímto formátem je formát BW, čili černobílý obraz. Než však dojde k převodu na BW formát, je třeba určit práh obrázku, který je nutný pro onen převod na černobílý obrázek metodou prahování. Prahovací metodě se věnuji v podkapitole 2.3.1 Prahování.

Po dokončení prahování obrazu je na pořadí další neméně důležitá část a tou je segmentace obrazu. Existuje několik druhů segmentačních metod popsaných v kapitole 2.

1.1 Obraz

Slovo obraz obecně znamená jakékoliv grafické vyjádření. Obraz může být zachycen optickými přístroji nebo přírodními objekty. Při jeho získávání se využívá různých optických jevů. Obraz získaný těmito způsoby je dále zpracováván nervovou soustavou nebo počítačem. Při zpracování počítačem se už dále mluví o získaném obraze jako o digitálním obraze.

Obraz je dán spojitou funkcí $I : \langle 0, w \rangle \times \langle 0, h \rangle \rightarrow C$, kde *w* je šířka, *h* je výška obrazu a *C* je barevný prostor, ale pro vyjádření digitálního obrazu se využívá diskrétní obrazová funkce. [1]

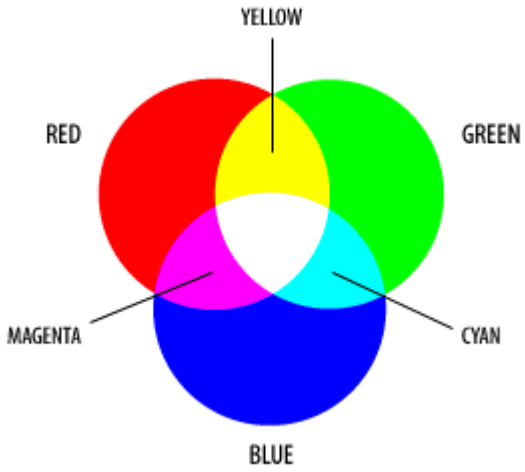
1.2 Barevný prostor

Máme jen jeden barevný prostor a ten je reprezentován různou interpretací.

Šedotónový diskrétní barevný prostor C_G, který je dán pouze jedinou dimenzí,

která je reprezentována hodnotami z množiny $C_G = \{b \in N; 0 \leq b \leq 255\}$ a obrazová funkce se pak promítá do C_G.

Model RGB popisuje možnost, jak znázornit barevný prostor. Existují ještě další metody, např. CMY, která také reprezentuje barevný prostor. Model RGB definujeme jako $C_{RGB} = C^3{}_G$. Tento popis znamená, že se barevný obraz skládá z 3 barevných složek (červené, modré a zelené), které svou intenzitou určují výslednou barvu. [9] Jak je uvedeno na *Obrázek 1-1*.



Obrázek 1-1: Model RGB. [9]

1.3 Intenzita modelu

Jednotlivé složky modelu RGB mají jiný podíl na výsledné intenzitě pixelu. Bylo dokázáno, že hlavní podíl na intenzitě barvy modelu má zelená složka (55 %), pak červená složka (30 %) a nejmenší podíl na intenzitě má složka modrá (15 %). [1]

Tyto podíly se mohou lišit dle použitého zdroje. Výpočet intenzity pixelu se využít při převodu RGB obrázku na obrázek ve stupních šedi.

1.4 Filtrace

Během získávání digitálního obrazu může dojít k možným chybám obrazu, které se označují jako šum. Digitální fotografie tyto chyby v obraze často obsahuje a můžeme při větším přiblížení zjistit, že každý pixel má jinou hodnotu. Tyto hodnoty však nejsou od sebe příliš vzdáleny. Přidání malého množství šumu do synteticky vzniklého obrazu pomáhá vytvořit jeho realistickou podobu.

V digitálním obraze, který má být dále zpracováván, může tento šum způsobit chybovost zpracovávacích metod.

Šum dělíme na dva typy:

Gaussův šum, který je označován jako závislý šum, kde je každý pixel obrazu mírně pozměněn.

Náhodný šum, který je označován jako nezávislý šum a může vzniknout vadnými CCD elementy, jako příklad se uvádí šum typu „sůl a pepř“.

1.5 Předzpracování obrazu

Příprava kvalitního a vyčištěného obrazu je podmínkou pro možnost provedení segmentace. Náročnost segmentace souvisí nejen s charakterem objektů v obraze, ale úspěšnou segmentaci nemalou měrou ovlivňují i vstupní data. Nerovnoměrné rozložení jasu nebo větší intenzita zašumění zpracovávaného obrazu mohou segmentaci úplně znemožnit. Pokud nastane takový případ, je třeba takto poškozený obraz vhodnými metodami předzpracovat, aby mohlo dojít k lepší výsledné segmentaci obrazu pomocí segmentačních algoritmů.

Jednotlivé metody, kterými se dají částečně odstranit vady z obrazu:

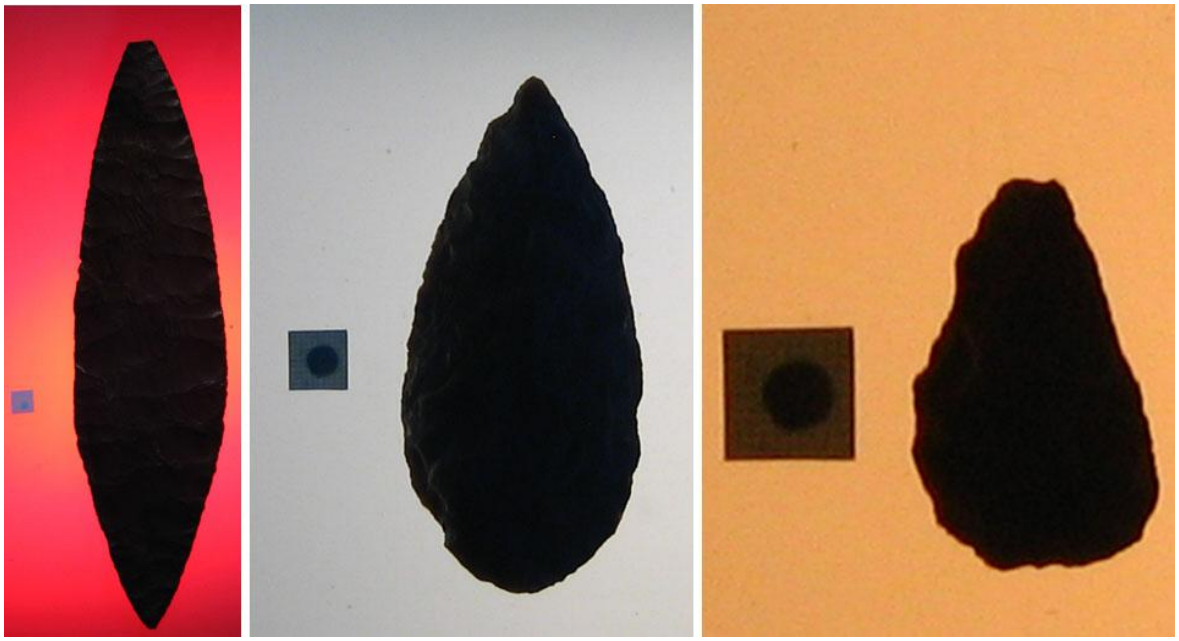
- Filtrace a odstranění šumu – používají se lineární a nelineární filtry.
    - Lineární filtry:
        - Gaussův filtr – používá konvoluci s maskou, způsobuje rozmazání obrazu.
        - Průměrování – hodnota bodu je určena průměrnou hodnotou sousedních bodů a způsobuje rozmazání obrazu.
        - Dolní propust
    - Nelineární filtry:
        - Mediánový filtr – hodnota bodu je určena mediánem z okolních bodů, je vhodný pro odstranění náhodného šumu.

- Prahování vlnkových koeficientů. Rozložení obrazu na diskrétní vlnkovou transformaci a aplikování na získané koeficienty tvrdé nebo měkké prahování.
- Konzervativní prahování - odstraňuje vadné body s vysokou či nízkou intenzitou, které nepatří do obrazu.

- Adaptivní filtrace
- Anizotropní filtry
- Potlačení vlivu osvětlení a jiných nehomogenit

1.6 Vstupní data a převod barevného obrazu na černobílý

Jako vstupní data aplikace slouží sada fotek artefaktů, kde není zaručena barva pozadí a ani barva artefaktu (viz. následující příklady vstupních dat *Obrázek 1-2*). Aby nedošlo ke ztrátě informace o tvaru artefaktu, je třeba dbát na pečlivé rozpoznání vstupních dat.



Obrázek 1-2:Typy vstupních dat.

Převod barevného obrazu na černobílý je nutností pro následné rozpoznání tvaru artefaktu ze vstupního obrazu. Tento převod na černobílý snímek se provádí pomocí metody prahování.

2 SEGMENTACE OBRAZU

Segmentace se dá rozdělit na 3 hlavní skupiny metod, s pomocí kterých je možné rozpoznat obraz.

Jsou to:

- prahování
- detekce hran
- detekce oblastí.

Všem těmto třem skupinám se věnuji podobněji v následujících podkapitolách.

Dále je třeba definovat pojmy, které jsou se segmentací úzce spojeny. Obecný popis segmentace je možné formulovat jako proces dělení obrazu do částí. Tyto rozdělené části odpovídají konkrétním objektům ve zdrojovém obraze. Segmentace se dá považovat za jednu z hlavních částí analýzy obrazu.

Získanou informaci o dělení částí obrazu do dílčích segmentů pak dále zpracovávají algoritmy v následných krocích zpracování obrazu. Tyto algoritmy mají za úkol porozumět obsahu obrazu. Typickým úkolem algoritmů může být nalezení výskytu hledaného objektu nebo nalezení a klasifikace objektů v obraze. Kvalitní segmentace obrazu je nezbytná i pro modelování objektů ve 3D formátu.

2.1 Definice segmentace

Segmentace obrazu $f(x,y)$ je dělení obrazu na podobrazy $R_1, R_2, ..., R_n$ tak, že podobrazy splňují následující požadavky:

- $\bigcup_{i=11}^{n} R_i = f(x,y)$
- $R_i \cap R_j = \varnothing, i \neq j$
- Každý nalezený podobraz splňuje alespoň jedno z následujících tvrzení:
    - Všechny body v podobraze R_i se neliší úrovní šedi více, než je předepsaná hodnota.

- Standardní odchylka úrovni šedi všech bodů v podobraze R_i je dostatečně malá.
- Všechny body v podobraze R_i mají stejnou úroveň šedi.

Definice segmentace je převzata z literatury. [7]

2.2 Způsoby řešení segmentace

Metody založené na detekci hran slouží pro detekci významných hran v obraze. Jednotlivé hrany jsou nalezeny pomocí hranových detektorů, které pracují na základě zjištění rozdílu hodnot okolních bodů. Výstupem algoritmu hranového detektoru je množina bodů nebo fragmentů. Metoda hranového detektoru bude detailně podrobněji popsána později v podkapitole 2.3.2.2.

Metody založené na detekci oblastí v obraze mají podobný průběh, jako metody hranového detektoru. Jestliže lze nalézt hrany, potom lze teoreticky říci, že měly ohraničovat oblasti. Obrysy oblastí mohou být neúplně a tyto nedostatky mohou způsobit, že nebude nalezena celá oblast. Pokud jsou obrysy objektů úplné, tak ani pak není zaručeno, že oblasti nalezené detekcí hran a detekcí oblastí budou shodné.

Metody založené na statistickém charakteru obrazu využívají jako základ segmentace statistickou analýzu obrazových dat, kde se nejčastěji využívají hodnoty jednotlivých bodů. Informace o struktuře je převážně nepotřebná.

Metody, které už nelze zařadit do předchozích metod a využívající jejich kombinace, se nazývají hybridní segmentační techniky. Do hybridních metod patří také metody založené na matematické morfologii. Tyto metody využívají matematických charakteristik obrazu pro zjištění výsledné segmentace obrazu.

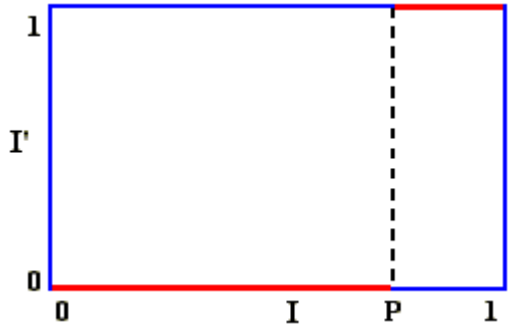
2.3 Segmentační metody

V této části se věnuji podrobnějšímu popisu některých segmentačních metod, které jsem v předchozí podkapitole 2.2 zmínil. Dále v této části hledám ideální metodu, kterou budu používat pro rozpoznání tvaru artefaktu, což je jedním z hlavních bodů mé diplomové práce.

2.3.1 Prahování

Metoda prahování se řadí to statických segmentačních metod a je nezbytná pro vytvoření černobílého obrazu, který je pak vhodným podkladem pro detekci objektu v obraze. Pro tuto operaci je třeba stanovit hodnotu threshold, která určí, které všechny části obrazu budou ve výsledném obrazu bílé a které naopak černé. Tato hodnota musí být zvolena natolik přesně, aby nedocházelo k ztrátě informací o tvaru hledaných objektů. [6] Proto existuje pro detekci hodnoty threshold více způsobů.

Prahování zjednoduší ve většině rozpoznávaných obrazů proces identifikace objektů. Obecně platí, jestliže jsou intenzity *I* jednotlivých složek zdrojového obrazu menší než nalezená prahová hodnota P neboli také threshold hodnota, pak se jim přiřadí hodnota 0 a v opačném případě hodnota 1. Výsledná intenzita *I'*, pak splňuje popsaná pravidla. Tyto funkce mohou pracovat i s více prahy, kdy aplikací jednotlivého prahovacího pravidla je prahována ta barevná část obrazu, která splňuje parametry pravidla. Ale taková funkce nebude pro práci s artefaktem potřeba.



Obrázek 2-1:Prahování obrazu podle hodnoty P. [9]

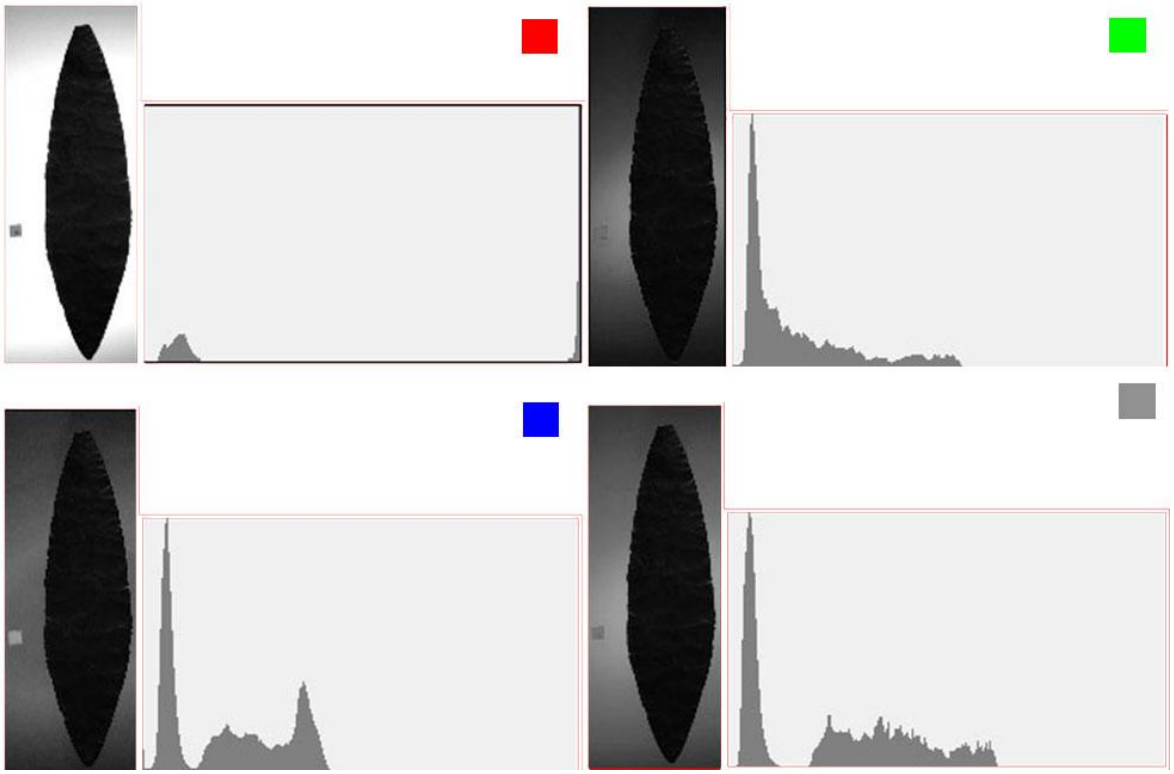
2.3.1.1 Histogram

Barevný histogram znázorňuje rozložení barevných složek v obraze. Je vyjádřen poměrným zastoupením množství bodů každého z daných barevných rozsahů. Histogram může být reprezentován pomocí dvourozměrného grafu, kdy je pro každou barevnou složku potřeba jeden graf, anebo je celý histogram zobrazen pomocí trojrozměrného grafu. [2]

Histogram obrazu reprezentuje pohled na rozložení dat v obrazu. Porovnáváním získaných hodnot histogramu u dvou obrazů a porovnáváním jejich dílčích barevných hodnot se dá také histogram využít k nalezení objektu na neznámé pozici v obraze.

Lze jej použít i při rozeznávání objektů v obraze, ale existují zde určitá omezení. Jedním z problémů je, že předem nelze předpokládat, že v našem případě budou mít všechny obrázky stejnou barvu a intenzitu pozadí. Proto je třeba s tímto ohledem zvolit i způsob určování prahu obrazu.

V případě vstupních dat je třeba rozhodnout, který ze získatelných histogramů bude pro určení prahové hodnoty obrazu ten pravý. Pro příklad uvedu histogramy všech kanálů *Obrázek 2-2*, aby bylo možno porovnat hledání maximálních hodnot histogramu a jejich ideální prahovací hodnotu, která by měla být umístěna mezi nimi.



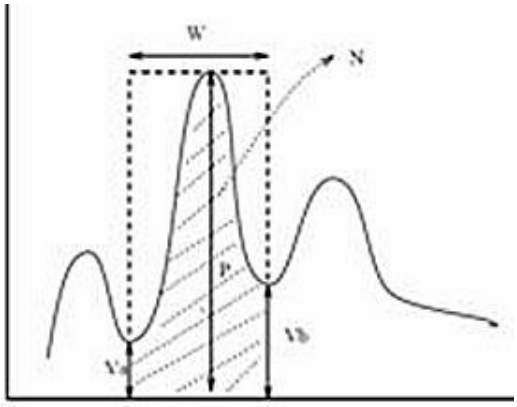
Obrázek 2-2:Histogramy barevný kanálů.

Jak je z většiny histogramů patrné, jedná se o bimodální histogramy, zejména u vyfiltrovaného červeného kanálu a u obrazu převedeného do odstínů šedi. U červeného kanálu je patrné, že došlo k výraznému vzdálení obou maxim histogramu a není problém určit prahovací hodnotu. Toto dělení kanálu však stojí výpočetní čas, a proto by bylo dostačující v tomto případě zvolit obraz převedený do šedotónového odstínu barev. Ale ne vždy se dá na toto pravidlo šedotónového obrazu spolehnout a je nutno najít vhodnější barevný kanál. Pro samotné určení nejlepší možné předlohy byla zvolena důmyslnější metoda, kterou je Otsuova metoda. Otsuovu metodu popisuji v sekci 2.3.1.2 Určování prahu.

2.3.1.2 Určování prahu

Při určování prahu je třeba získat z obrazu histogram, který udává počet bodů v obraze konkrétní barvy. Histogram může znázorňovat jednotlivé barevné složky nebo obrázek v barvách šedi. Analýzou získaných histogramů získáme práh *P*, který rozdělí obraz na podoblasti.

Nalezení hledaného prahu v histogramu není jednoduchý úkol, protože histogramy skutečných obrazů většinou neobsahují hrany a vrcholy. Tyto histogramy mohou mít i několik maxim o různé výšce a strmosti. Jako jedna z metod rozhodnutí, zda jde o hledaný vrchol, může sloužit metoda, kdy se zjišťuje poměr plochy vrcholu s poměrem plochy jeho obálky. [7]



Obrázek 2-3:Určení prahu pomocí detekce vrcholu histogramu. [7]

V předešlém obrázku znamenají popisky následující proměnné, V_a a V_b jsou minima okolí vrcholu, *N* označuje četnost bodů vrcholu, *P* je výška a *W* je šířka oblasti vrcholu.

Z těchto známých proměnných pak získáme rovnici, která určuje ostrost vrcholu.

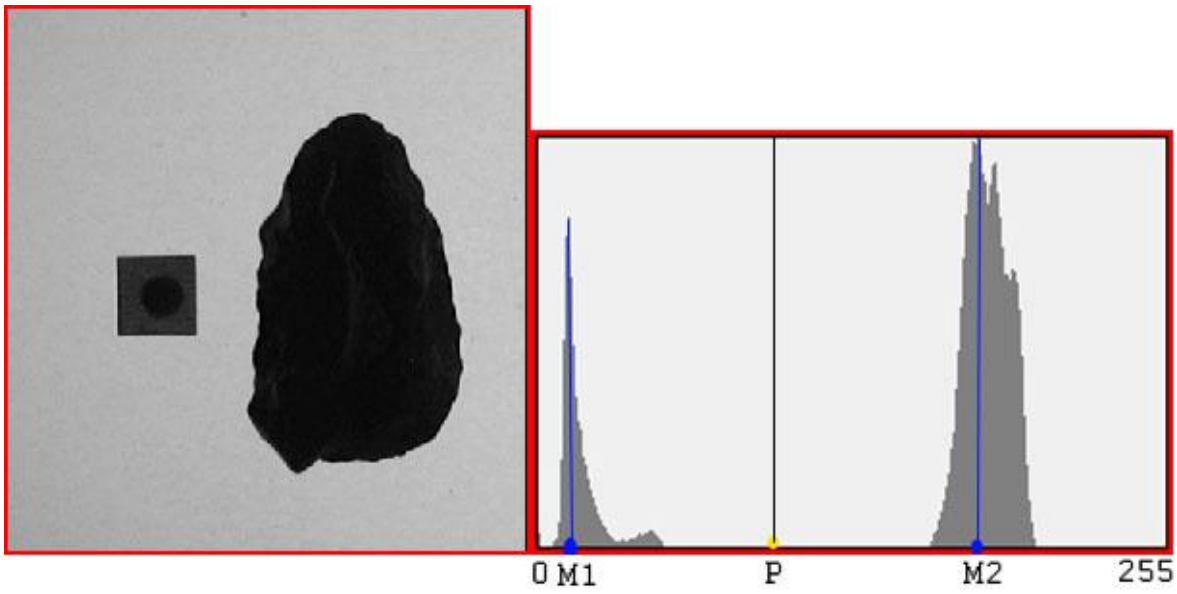
$$Q = \left(1 - \frac{V_a + V_b}{2P}\right) * \left(1 - \frac{N}{W * P}\right) \tag{2.1}$$

Menší výsledná hodnota rovnice značí lepší maximum histogramu. Tolik o obecném určování prahu histogramu.

V případě vstupních dat bude zjišťování prahu snazší, jak jsem již uváděl v předchozí podkapitole. Jelikož zdrojové histogramy jsou bimodální, není pak nalezení maxim

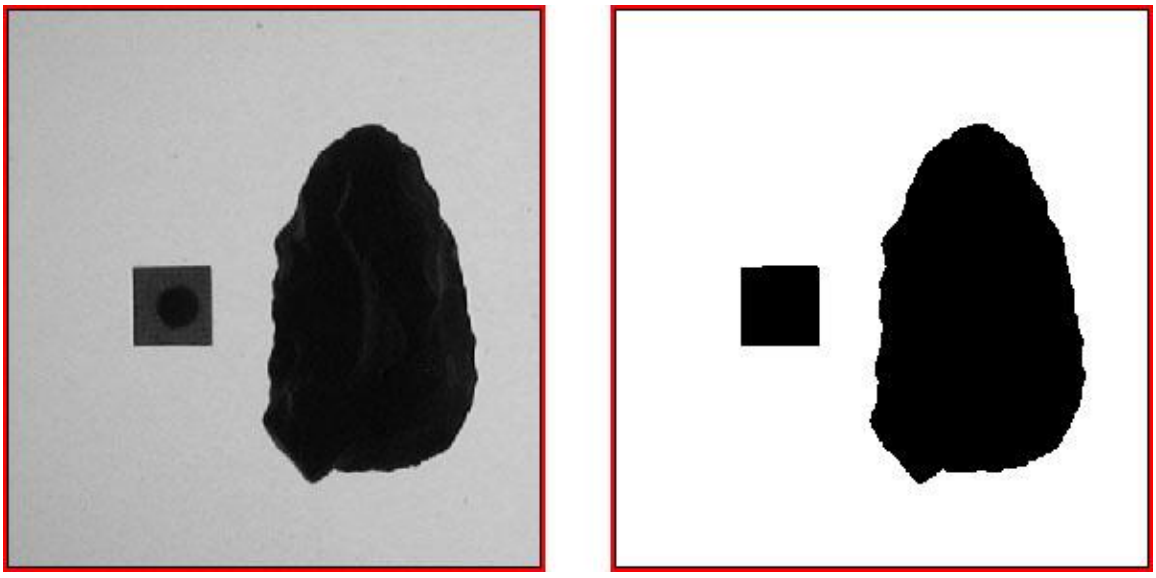
histogramu a prahovací hodnoty tak složité. Toto tvrzení je znázorněno na následujícím obrázku.

Ve zdrojovém obraze proběhne oddělení jednotlivých barevných kanálů, ze kterých jsou vytvořeny histogramy. Jelikož má histogram dvě maxima, mělo by být dalším krokem jejich nalezení, ale tento krok vynecháme a v těchto histogramech nalezneme přímo jejich práh pomocí Otsueovy metody. Jako výchozí kanál pro další zpracování je zvolen ten kanál, který má svou prahovací hodnotu co nejblíže hodnotě 128, což je polovina rozsahu histogramu pro daný barevný kanál, která je <0,255>.



Obrázek 2-4:Určení prahovací hodnoty.

V získaném histogramu na *Obrázek 2-4* ze vstupního obrázku jsou patrná maxima M_1 a M_2, mezi kterými leží hledaná prahovací hodnota *P*. Výsledkem takového prahování je černobílý *Obrázek 2-5*, důležitý pro detekci hran artefaktu.



Obrázek 2-5:Výsledek prahování.

Pokud je vstupní obraz hůře prahovatelný, pak je třeba použít metodu, která zvládá vyhodnocení prahování s lepšími výsledky. Touto metodou je Otsuova metoda.

V počítačovém vidění a zpracování obrazu slouží Otsuova metoda pro automatické určení histogramu u obrazu, který obsahuje objekt pomocí thresholdingu nebo redukcí šedotónového obrazu do binárního černobílého obrazu. Algoritmus předpokládá, že zdrojový obraz obsahuje dva typy bodů a to popředí a pozadí. Potom vypočítá optimální hranice oddělující obě tyto kategorie, tak že rozptyl těchto tříd je minimální. [4]

V Otsuově metodě je pak náročné hledání prahu, který minimalizuje vnitřní třída rozptylu, definovaná jako vážený součet rozptylů dvou tříd.

$$\sigma_{\omega}^2(t) = \omega_1(t) * \sigma_1^2(t) + \omega_2(t) * \sigma_2^2(t) \tag{2.2}$$

Váhy ω_i jsou pravděpodobností dvou tříd oddělených prahem *P* a rozptylů těchto tříd σ_i^2.

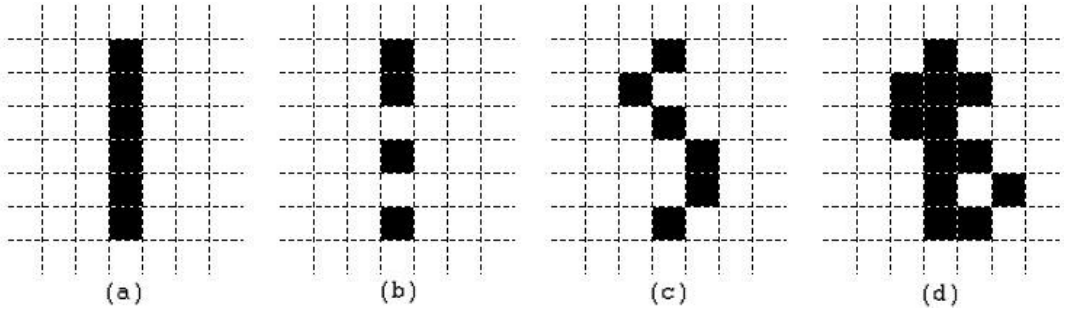
Obecně nám tato metoda slouží pro automatické zjištění hodnoty, kterou se pak řídí metoda prahování.

2.3.2 Detekce hran

Metoda detekce hran je považována za jednu z nejdůležitějších oblastí segmentace obrazu a také je nejvíce používanou metodou. Všechny tyto metody založené na detekci hrany předpokládají, že každé těleso či objekt umístěné v obraze, je obklopeno body, které tvoří celistvou hranici tohoto objektu. Cílem detekce hran je nalezení těchto hraničních bodů a

utvoření souvislé hrany objektu. Spojení nalezených bodů pak proběhne pomocí metod prokládání bodů.

Hranami objektů v obraze chápeme ty body obrazu, u kterých se hodnota jasu rychle mění. Nejzákladnější vlastnosti obrazu jsou skoky a přerušení, protože reprezentují umístění těles a objektů v rozpoznávaném obraze. V takovém obraze jsou dva typy možných jasových změn, kterými jsou lokální a globální změny. Lokální změny se označují jako jasové hrany a globální změny jsou označovány jako jasové hraniční segmenty. [11] Na následujícím *Obrázek 2-6* je znázorněno, jaké se vyskytují defekty hran v obraze.



Obrázek 2-6:Defekty hran v obraze při tloušťce 1. [11]

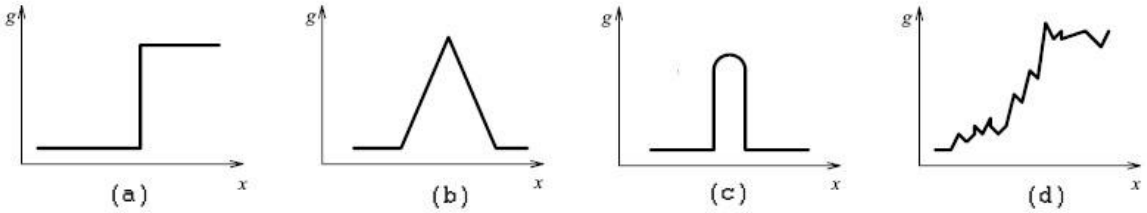
V obraze se mohou vyskytovat skutečné hrany (a), nespolehlivě detekovatelné hrany (b), špatně lokalizovatelné hrany (c) a nejednoznačně detekovatelné hrany (d). Toto znázornění hran je znázorněno na ideální skokové hraně, tudíž se jedná pouze o zjednodušený model reality.

Samotná detekce hran se skládá ze třech dílčích fází. Jsou to:

- Filtrování – nastavením vhodného filtru lze ze zdrojového obrazu odstranit některé jeho vady jako je šum nebo rozmazání.
- Diferencializace – slouží pro zvýraznění míst ve zdrojovém obraze, ve kterých je změna intenzity jasu nejvíce patrná.
- Detekce – v procesu detekce dochází k nalezení nejvýznamnějších míst s nejvíce patrnou intenzitou změny hodnoty jasu.

Všechny tyto detekční metody jsou pak závislé na vhodně zvolené segmentační metodě, která určuje kvalitu výstupních či nalezených dat.

Hrany v obraze mohou být různých tvarů. Na následujícím *Obrázek 2-7* jsou zobrazeny nejčastější typy hran.



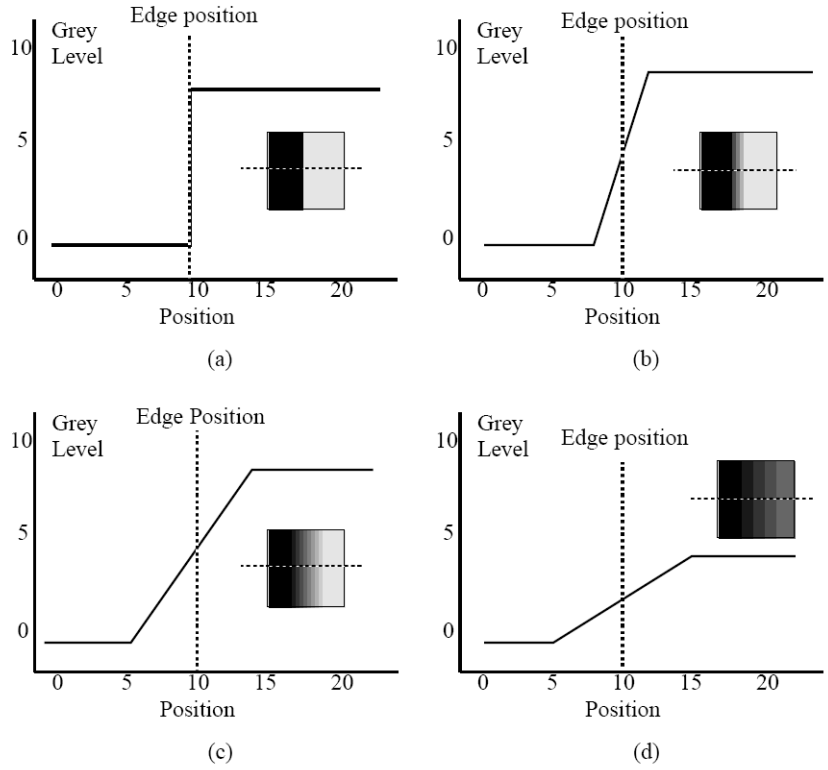
Obrázek 2-7:Typy hran v obraze. [11]

Hrana (a) označuje skokovou hranu, další dvě hrany jsou pak kombinací typů hran (a) a (d), kde hrana (b) označuje spojení dvou šikmých či zašuměných hran a tvoří tak hranu typu střecha, hrana (c) označuje spojení dvou skokových hran a tvoří tak hranu typu čára a poslední uváděný typ hrany je hrana (d), která označuje šikmou hranu nebo také zašuměnou hranu.

2.3.2.1 Popis hrany

Hrana je z pohledu matematiky definována jako derivace jasové funkce, která v oblasti hrany dosahuje lokálního maxima. Když se na hranu podíváme z pohledu počítačové grafiky či zpracování obrazu, pak hrana označuje místo, kde nastává prudká změna sousedících obrazových bodů. [3]

Ideálním případem hrany je skoková hrana, kdy dochází k prudké změně jasu, ale takové hrany se v obraze vyskytují jen zřídka a proto můžeme najít v obraze ve většině případu šikmé hrany s různým sklonem šikmosti. Obecně pak platí, že čím je širší tato hrana, tím hůře se dá tato hrana nalézt. Tento problém šikmosti hrany popisuje následující *Obrázek 2-8.*



Obrázek 2-8:Změny intenzity okolí hran. [9]

Pokud se detailně podíváme na jednotlivé obrázky, tak první obrázek označuje skokovou hranu, u které dojde k okamžité změně mezi dvěma sousedními obrazovými body. U ostatních obrázků je potom už detekce přesné pozice hrany obtížnější, protože zasahuje svou šířkou přes více sousedících bodů v obraze.

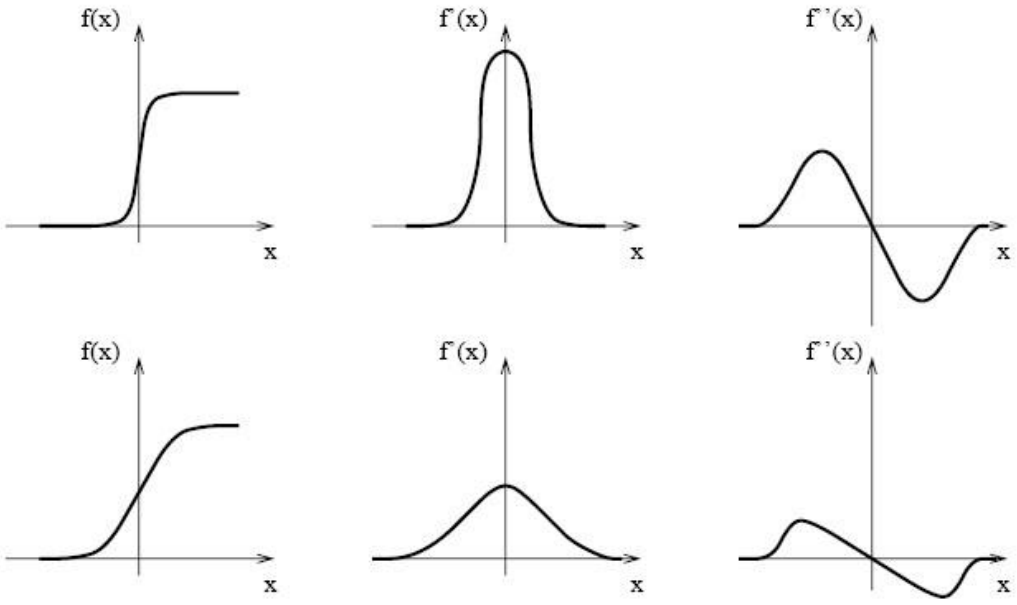
Jednou z možných důvodů vytváření těchto šikmých hran je digitalizace obrazu a šum, který vzniká při pořizování zdrojových obrazových dat.

2.3.2.2 Metody Detekce hran

Metody detekce hran lze zařadit do dvou hlavních okruhů řešení. Jsou to detekční metody, které využívají první derivaci nebo druhou derivaci obrazové funkce.

Při aplikaci prvního okruhu detekčních metod s využitím první derivace, je pak výsledný hledaný hranový gradient porovnán s prahem. Tento práh určuje, zda se jedná o hranu. Při použití druhého okruhu metod týkajících se druhé derivace, je výskyt hrany závislý na prostorové změně v polaritě druhé derivace. Pokud je tato změna polarity dostatečně významná, pak se jedná o výskyt hrany. [11]

Na následujícím *Obrázek 2-9* je zachycen průběh úrovně jasu s jeho první a druhou derivací.



Obrázek 2-9:Detekce hrany pomocí první a druhé derivace. [11]

Z tohoto obrázku je patrné, že určení hrany z běžné funkce není triviální úlohou a proto dochází k aplikaci první a druhé derivace, kde je detekce hrany o poznání jednodušší.

První derivace, která je na předchozím obrázku označena funkcí $f'(x)$, se stala základem velké většiny gradientních metod. První derivace funkce $f(x)$ označená jako $f'(x)$, má své maximum právě v bodě hrany. Toto maximum pak označuje velikost intenzity obrysu hrany, neboli se dá nazvat jako velikost hrany. [7]

Hodnotu první derivace diskrétního obrazového zdroje spočítáme jako diferenci okolních bodů, které nemusí být přímo sousední. Výpočet probíhá pro oba hlavní směry odděleně. Výpočet derivace pro řádky je realizován zleva doprava a výpočet derivace pro sloupce je realizován shora dolů. Hodnota hledaného gradientu je spočtena ze vzorce:

$$G(x,y)=\sqrt{G_R(x,y)^2+G_C(x,y)^2} \tag{2.3}$$

kde G_R je řádkový gradient a G_C je sloupcový gradient. Pokud je tento výpočet početně náročný, pak lze použít vzorec v aproximovaném tvaru, ale je zde riziko, že dojde k poměrnému zhoršení výsledného gradientu.

$$G(x,y) = |G_R(x,y)| + |G_C(x,y)| \tag{2.4}$$

Výslednou orientaci gradientu vzhledem k řádkům zdrojového obrazu určím ze vzorce:

$$\partial(x,y) = acrtn\frac{G_C(x,y)}{G_R(x,y)} \tag{2.5}$$

K zjištění gradientu lze použít metody konvolučního filtrování obrazu. Jedná se o hranové detektory, které mají odlišné jádro konvolučního filtru znázorněné na *Obrázek 2-10*. Význam jednotlivých jader filtrů značí, které body se zahrnou do výpočtu gradientu a do jaké míry ho ovlivní. Výsledný gradientní obraz pak vznikne konvolucí jádra filtru a původního obrazu. Některé filtry mohou mít velikost jádra i větší, lépe pak reagují na případný šum.

Operator	Row gradient	Column gradient
Pixel difference	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
Separated pixel difference	$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$
Roberts	$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
Prewitt	$\frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$	$\frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
Sobel	$\frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$	$\frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
Frei–Chen	$\frac{1}{2+\sqrt{2}}\begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix}$	$\frac{1}{2+\sqrt{2}}\begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}$

Obrázek 2-10:Konvoluční jádra hranových detektorů. [7]

Další možnou alternativou, která je už výkonnější, jsou detektory s váhovací funkcí Gausovského tvaru, které dokáží určit směr úhlu hrany.

Druhá derivace funkce $f(x)$, která je na předchozím obrázku označena jako $f''(x)$,umožňuje ještě lepší detekci hrany, což je patrno z obrázku. Druhá derivace funkce

prochází nulou v bodě hrany a tudíž je nalezení hrany jednoduší, než hledání maxima funkce, jak tomu bylo v případě první derivace.

Ale i v případě druhé derivace je nutno provést ověření, zda se jedná o hranu ve více směrech. Pro toto zjištění je třeba alespoň dvou různých směrů. Jako nejběžnějších směrů se využije směru os souřadnicového systému. Pro výpočet se v tomto případě používá Laplacián neboli také Laplaceův operátor. Laplacián je založen na druhé derivaci a tím pádem je invariantní vzhledem k rotaci. [3]

Laplacián je určen následujícím vztahem:

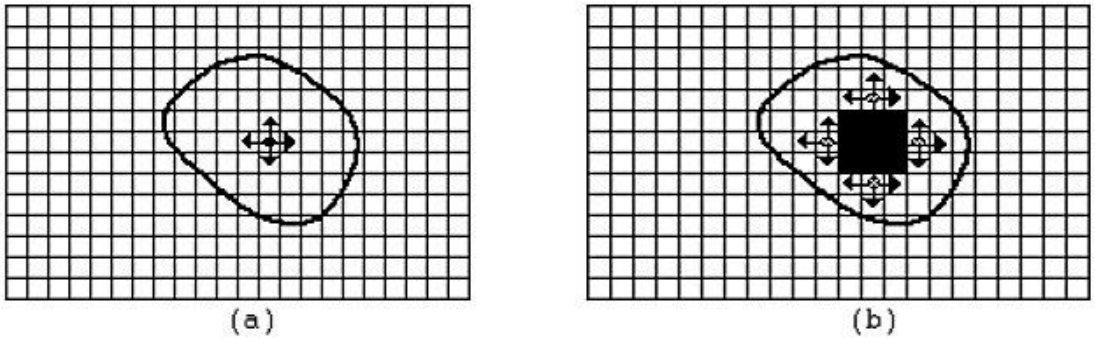
$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \tag{2.6}$$

Nevýhodou Laplaciánu je velká citlivost na šum a vytváření falešných hran anebo také může vznikat dvojitá odezva na některé hrany v obraze.

2.3.3 Detekce oblastí

Detekce oblastí se od detekce hran liší, jak už tomu napovídá název těchto dvou skupin metod. Zatímco se detekce hran hledá jen hranice objektů v obraze pomocí změny jasu, detekce oblastí oproti tomu hledá oblasti, které mají podobné vlastnosti. Těmito podobnými vlastnostmi mohou být jas, barva, obrysy, které jsou hledány ve zdrojovém obraze. Obecně lze říct, že hlavním faktorem při hledání jednotlivých oblastí je jejich homogenita. Další výhodou metod na detekci oblastí oproti metodám detekce hran jsou možnosti rozpoznat i zašumělý obraz, na který byly metody detekce hran „krátké“.

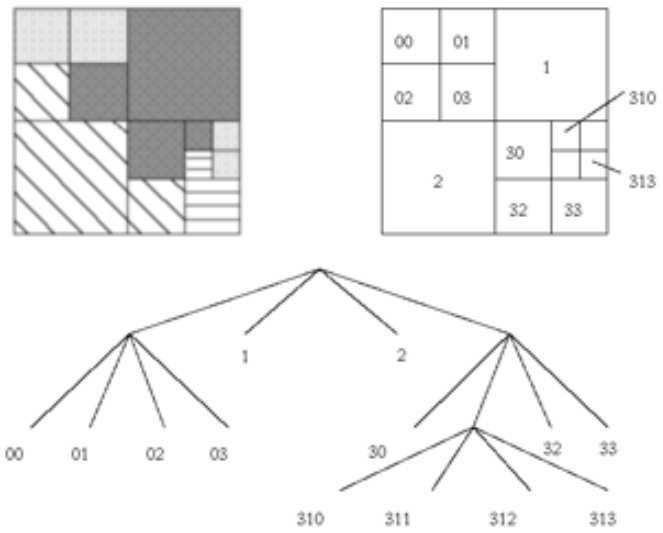
Z metod pro detekci oblastí lze zmínit **metodu šíření oblastí**, která je považována za nejsnadnější pro implementaci. Postup této metody spočívá v seskupování sousedících obrazových bodů s obdobnou amplitudou, ze kterých se vytváří či detekuje segmentová oblast. [5] Toto je pouze teoretický postup, který je v praxi složitější. Stěžejní část se především týká pravidel řídící seskupování. V počátku detekce oblastí je zdrojový obraz rozdělen na atomické oblasti, které mohou mít velikost pouze jednoho bodu, a postupně jsou tyto podoblasti spojovány do větších oblastí s okolními oblastmi, které mají podobné vlastnosti. Detekce se zastaví ve fázi, kdy už není možno spojovat další podoblasti. Tato metoda může mít nastaveno, že nebere v úvahu přerušované hranice, neboli je nastaveno jen to, jak silných hranic si má všímat, a podle toho určuje, zda se už jedná o další objekt.



Obrázek 2-11:Vyhledávání pomocí metody Šíření oblastí. [9]

Na obrázku nahoře je naznačen způsob vyhledávání oblasti v obraze, kde na obrázku (a) je zvolena podoblast, která postupně k sobě připojuje další oblasti patrné na obrázku (b), dokud nenarazí na vyznačenou hranu.

Další metodou je **Metoda dělení a spojování**. V podstatě se jedná o opačnou metodu k výše popisované metodě šíření oblastí. Funkce této metody spočívá v dělení obrazu na kvadranty a subkvadranty, dokud není obraz dostatečně rozdělen tak, že jednotlivé skupiny jsou homogenní. Jako hledisko homogenity lze požít histogram subkvadrantů nebo také pouhý rozdíl mezi maximální a minimální hodnotou obrazového bodu daného subkvadrantu. Pokud jsou některé sousední subkvadranty navzájem homogenní, dojde k jejich spojení do jedné oblasti. [5]



Obrázek 2-12:Vyhledávaní pomocí metody Dělení a spojování. [7]

Na *Obrázek 2-12* je patrné jednotlivé dělení na subkvadranty či suboblasti, které jsou pak dodatečně pospojovány podle stejných vlastností.

II. PRAKTICKÁ ČÁST

3 NALEZENÍ INFORMACÍ O ARTEFAKTU

V této kapitole se budu věnovat praktickému nalezení artefaktu a dalších informací, jako jsou výška, šířka, natočení, koeficienty a další požadované hledané údaje. Tyto dílčí úkony budou rozepsány do jednotlivých podkapitol.

Před samotným rozpoznáním artefaktu je třeba říci, že objekt ve vstupních datech není vždy svisle umístěn a většinou se zde objevuje i natočení objektu, které je třeba odstranit.

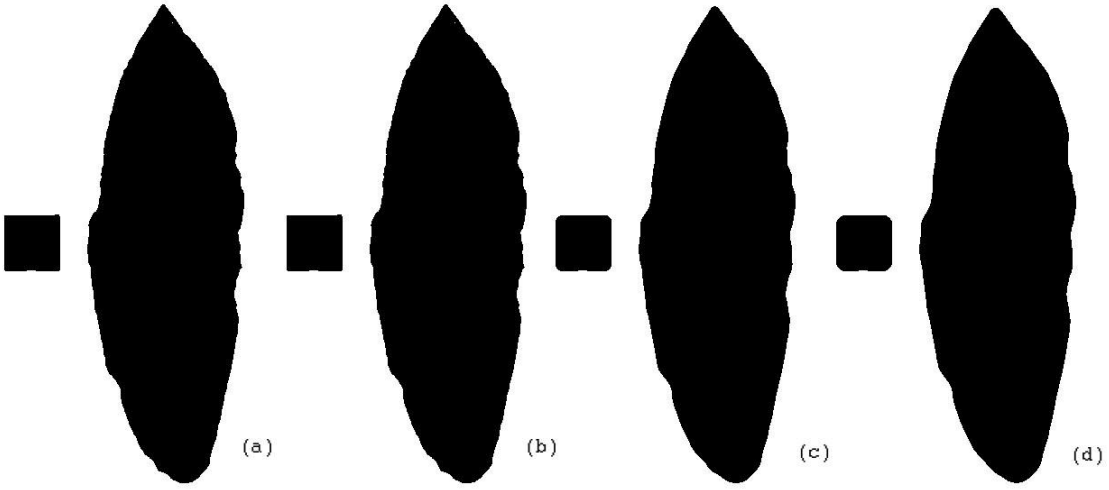
Stěžejní částí je rozpoznání měrky a určení její velikosti, která je součástí každého zpracovávaného obrazu. Jinak bychom neměli možnost zjistit skutečnou velikost artefaktu.

Do aplikace se bude také importovat čelní a boční pohled artefaktu, takže u artefaktu bude známa i jeho hloubka a bude možno udělat kontrolní ověření velikosti artefaktu.

3.1 Úpravy obrazu

Před samotnou segmentací obrazu či nalezení artefaktu je třeba upravit obraz, který bude rozpoznáván. Touto úpravou je myšleno vyhlazení hran artefaktu, protože při převodu na černobílý obrázek by mohlo dojít s ohledem na prahování ke ztrátě některých okrajových obrazových bodů hledaného artefaktu. Proto v této časti popíši metody, které se zabývají vyhlazením hran obrazu. Hrana artefaktu bude po tomto kroku méně nerovná a bude se více podobat původnímu snímku. Ale v podkapitole 3.4 zdůvodním, že tato úprava nepřináší výsledek, který by byl uspokojivý a také má za následek delší dobu zpracování vstupního obrazu do získání hledaného objektu artefaktu s jeho dalšími údaji.

Pro ukázku uvedu testovanou metodu, od které jsem očekával zkvalitnění obrazu před jeho segmentací. Jako vyhlazovací metodu jsem si vybral základní vyhlazovací funkci Medián, která má jako vstupní parametr obrázek a velikost matice, která je použita pro vyhlazení. Funkce Medián funguje na principu zjištění hodnot v okolí upravovaného bodu, které jsou seřazeny do řady a je z nich vybrána hodnota, která dělí tuto řadu na dvě poloviny. Tato hodnota je uložena na místo bodu, který je takto upravován. [9] Matice mediánu má většinou velikost 3x3 nebo 5x5. Čím je však tato matice větší, tím dochází k větší výpočetní náročnosti.



Obrázek 3-1:Vyhlazení obrazu pomocí Mediánového filtru.

Na *Obrázek 3-1* je patrné postupné vyhlazení artefaktu, kdy obrázek (a) je originální obrázek převedený prahováním do černobílé barvy. Na tento obrázek jsou pak aplikovány matice o velikosti 3x3 (obrázek b), 5x5 (obrázek c) a 7x7 (obrázek d).

Jelikož dochází k dávkovému zpracování většího množství vstupních dat ve formě obrázků, je třeba rychlosti, kterou nám tyto vyhlazovací metody nijak nezajistí. Proto je nezařazuji do aplikace vyhlazovací metody a problémem členitosti hran řeším způsobem. kdy dochází k vypouštění některých hranových bodů nalezeného artefaktu.

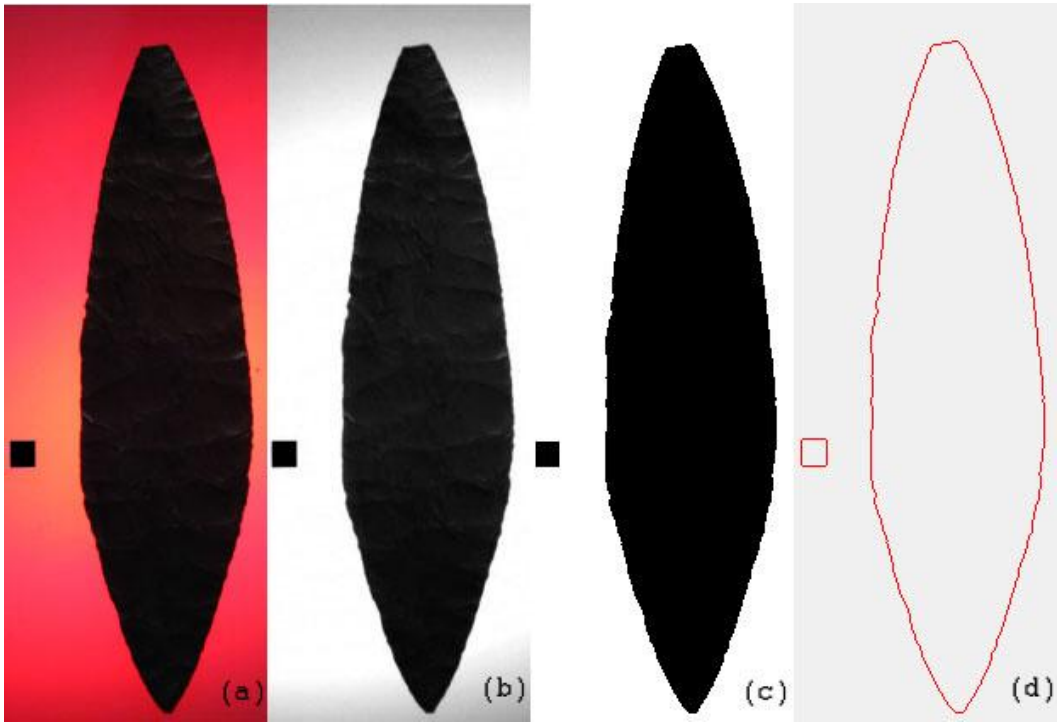
3.2 Segmentace obrazu pomocí knihovny OpenCVDotNet

V této diplomové práci používám z knihovny *OpenCVDotNet*, popsané v podkapitole 5.3.3, několik funkcí, které především zjednodušují získávání hranových bodů artefaktu z předpřipravených dat.

V předešlých kapitolách bylo použito několik metod, které teď popíši s ohledem na jednotlivé kroky přípravy dat. Všechny z uvedených metod náleží do třídy *CVImage*, která v knihovně *OpenCVDotNet* reprezentuje obrázek a dá se považovat za jednu z nejdůležitějších tříd této knihovny. Z doposud využitých metod to jsou metody *Split, ToBitmap, ToGrayscale, GetConture* a *TrasholdImage*. Později tento seznam rozšířím o některé další, které budou třeba pro pozdější úpravy zdrojových dat aplikace. Tolik k popisu knihovny sloužící k získávání objektů z obrazu.

Dále bude následovat popis chodu aplikace při detekci objektů a ukázky výstupů z rozpoznávací části systému v jednotlivých důležitých krocích. Na následujícím *Obrázek*

3-2 jsou znázorněny jednotlivé kroky a jejich průběžné výstupy, které je potřeba postupně udělat pro získání objektu od načtení obrazu až po rozpoznání objektů v obraze.



Obrázek 3-2:Postupné kroky od zdrojového obrazu k detekovanému artefaktu.

Popis jednotlivých kroků:

Pro načtení vstupních dat slouží inicializační metoda třídy *CVImage*, která načte zvolená vstupní data ze souboru. Vykreslená načtená zdrojová data označuje *Obrázek 3-2* (a).

```
CVImage sImage = new CVImage(fPath);
```

V dalším kroku je obraz rozložen na jednotlivé barevné kanály a u každého je hledána pomocí Otsuovy metody(2.3.1.2) prahovací hodnota. Pro další zpracování je vybrán barevný kanál, který bude mít nejlepší nalezenou prahovací hodnotu. Ta by měla být optimálně okolo 128, což je polovina rozsahu histogramu každého barevného kanálu. Rozdělení zdrojového obrazu na barevné kanály se provede metodou *Split*. Výstupem této metody je seznam jednotlivých kanálů. Barevný kanál s nejlepší hodnotou histogramu je zobrazen na *Obrázek 3-2* (b).

```
CVImage[] cImages = sImage.Split();
```

Dalším krokem je převedení obrazu do černobílé barvy pomocí prahovací metody TrasholdImage. Vstupním parametrem této metody je hodnota Threshold, která je zjištěna Otsuovou metodou z vybraného barevného kanálu. Výstup této metody je zobrazen na *Obrázek 3-2* (c).

```
CVImage thImage = cImage.TrasholdImage(Threshold);
```

Posledním krokem v této podkapitole je aplikace metody *GetConture*. Tato metoda se spustí pro provedení všech předešlých kroků, při kterých se připravovala obrazová data, volil se optimální barevný kanál pro detekci artefaktu vzhledem k jeho histogramu a hledala se Threshold hodnota, která je zároveň i vstupním parametrem této metody a obraz se převádí na černobílý s použitím Threshold hodnoty. Návratový typ metody *GetConture* je dvou dimenzionální pole bodů typu *PointF[][],* kde první dimenze pole obsahuje jednotlivé nalezené objekty v obraze a druhá dimenze pole určuje body nalezeného objektu. Výsledek rozpoznání objektů je zobrazen na *Obrázek 3-2* (d).

Toto je příklad zdrojového kódu, který slouží k získání detekovaných objektů ve zdrojovém obraze:

```
PointF[][] shapes = prepareImage.GetConture(ThresholdVal);
```

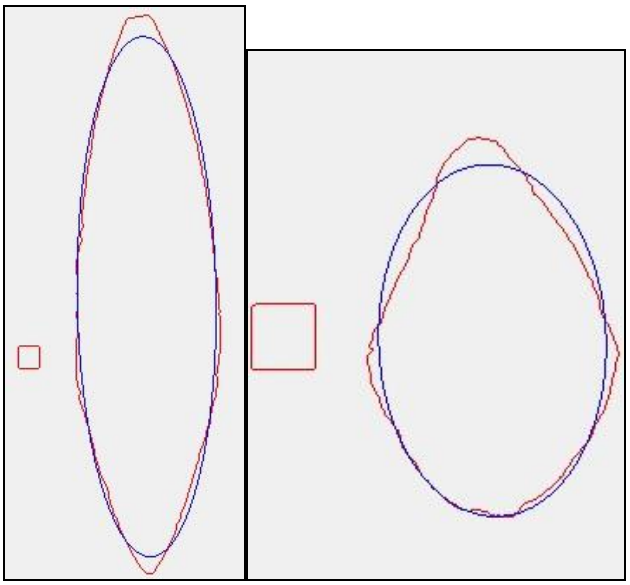
3.3 Nalezení artefaktu

I když už máme nalezené objekty ve zdrojovém obraze, stále ještě nevíme, jak dané objekty identifikovat. Lidskému oku, které tuto skutečnost vyhodnotí okamžitě, se počítačový software nevyrovná. Proto je třeba vytvořit postup, kterým rozpoznávací nástroj systému pozná, o jaký typ objektu se jedná. Pro rozpoznání artefaktu lze použít více postupů, které jsou schopny hledaný artefakt seznamu objektů určit.

Jednou ze dvou testovaných metod určení artefaktu je postup, kdy se snažíme k objektu přichytit elipsu. O daném artefaktu je možno říct, že je podobného tvaru jako elipsa a tudíž detekovatelný právě elipsou. Pro tuto detekci nám poslouží další třída z knihovny *OpenCVDotNet*, která se nazývá *CVUtils.Ellipse* a obsahuje metodu *FitEllipse*. Vstupní proměnnou této metody je pole bodů nalezeného objektu. Výstupem této metody je elipsa, která má přibližné rozměry artefaktu. Porovnání rozměrů elipsy a vstupního obrazu, pak slouží jako ukazatel toho, zda byl opravdu nalezen artefakt.

```
CVUtils.Ellipse el = CVUtils.FitEllipse(points[i]);
```

Jak je na následujícím *Obrázek 3-3* znázorněno, došlo k nalezení artefaktu v obou případech. Toto nalezení artefaktu je zvýrazněno modrou elipsou.



Obrázek 3-3:Detekované artefakty ve zdrojovém obraze.

Druhým testovaným způsobem je hledání tělesa s největším obvodem či největším počtem bodů, které tento objekt reprezentují. Jelikož by na obraze nemělo být nic jiného než pouze artefakt a měrka, tak i tento způsob určení artefaktu je možný, ale není vyloučen výskyt šumu, který by svým počtem hraničních bodů mohl chybně určit hledaný artefakt.

Pro toto rozpoznání jsem nakonec zvolil metodu pomocí elipsy, protože ihned po označení objektu známe přibližné rozměry objektu. Když bude poměr velikosti elipsy a zdrojového obrázku v nastavených mezích, potom si můžeme být jisti nalezením artefaktu.

3.4 Rozpoznání natočení artefaktu

V této kapitole zjišťuji natočení artefaktu vhledem k svislé ose obrazu. Stěžejním problémem je nalezení vrcholových bodů artefaktu, zjištění odchylky osy artefaktu od svislice a natočení artefaktu zpět do svislé polohy. Tímto otočením docílím toho, že s objektem bude možné lépe později pracovat při měření velikosti, při porovnání s dalšími artefakty a při aplikování dalších vyhodnocovacích metod.

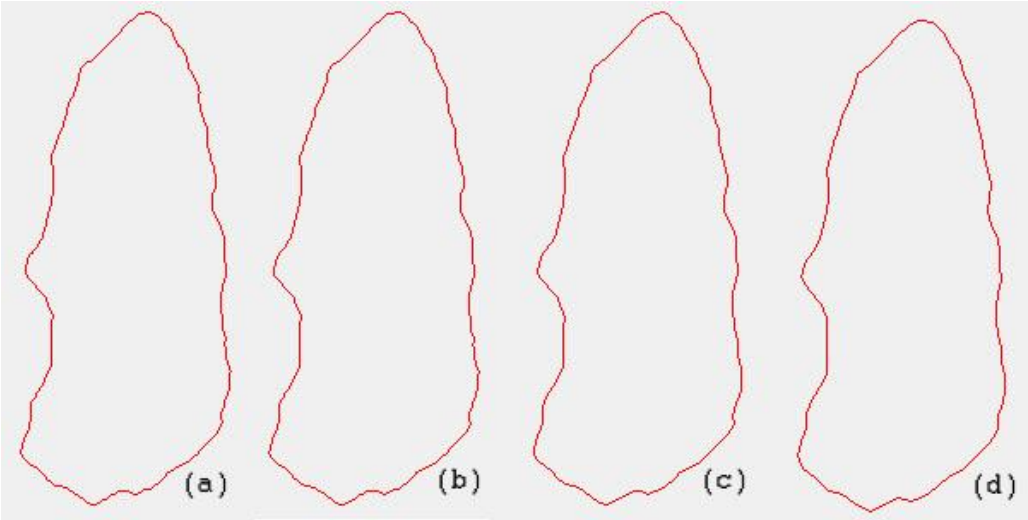
Výstupem detekční metody *GetConture* je sada či pole jednotlivých hraničních bodů nalezeného artefaktu. Tyto body reprezentují jednotlivé body obrazu a mají počátek v levém horním rohu obrazu, odkud se všechny body vykreslují. Toto je pro další práci s polem hodnot nepoužitelný způsob reprezentace dat, protože by docházelo k častému přepočtu hodnot, což by opět prodloužilo čas výpočtu. Proto je třeba tyto body vhodně uspořádat a zlepšit tak jejich organizaci. Způsoby přichycení bodů k zvolenému souřadnému systému řeším v podkapitole 3.5.

Jako možné způsoby uložení bodů se nám nabízí několik možností, jak reprezentovat tato data. Jednou z možností je vektor komplexních čísel, se kterým se optimálně pracuje vzhledem k 2D prostoru a využití vlastností komplexních čísel, vzniká řada výhod.

Druhou možností je ponechat body ložené ve formátu bodu, který obsahuje souřadnice bodu reprezentované hodnotami *X* a *Y*. Tyto body jsou pak uloženy do pole. Způsob uložení bodů do pole nám bude později trochu komplikovat práci, ale jak si ukážeme, tak poměrně jednoduchou úpravou tento problém vyřešíme. Problém vzniká při prohledávaní pole v cyklech, kdy je potřeba přeskočit z počátku na konec pole a opačně, aby bylo zajištěno „kruhové pole“. Pro lepší práci se sadou bodů by bylo vhodné vytvořit datovou strukturu, která by tento problém řeší automaticky. Prozatím si vystačíme s běžným datovým typem pole.

Před tím, než se dostaneme k samotnému hledání úrovně odklonění artefaktu, je třeba provést ještě jednu úpravu, o které jsem se zmiňoval v podkapitole 3.1, kde jsem uvedl, že je vhodné použit nějaký způsob pro vyhlazení obrazu. Pro vyhlazení je zvolena metoda částečné ztráty dat, kdy je dle volitelného parametru vypouštěn každý i-tý bod pole. V našem případě je parametr nastaven na hodnotu 2, z čehož vyplývá, že ztratíme některé zbytečné body, které způsobují větší nerovnost povrchu pole. Uchovávání těchto nerovností je zbytečné a mnohdy i chybné, protože vzniká při procesu prahování, jak jsem psal dříve. Ale zároveň nesmí být vynecháno větší množství bodů, protože by mohlo dojít ke ztrátě skutečného tvaru artefaktu. Zároveň je také málo pravděpodobné, abychom vynechali právě vrcholový bod. Tato redukce bodů bude mít tedy za následek vyhlazení tvaru artefaktu, ale také snížení celkového počtu bodů artefaktu, což je pro rozpoznávací mechanismy výhodné a znamená to úsporu počítačového času. Na následujícím *Obrázek 3-4* je patrné vyhlazení artefaktu. Obrázek (a) označuje původní pole bodů artefaktu, obrázek (b) znázorňuje počet bodů redukovaných na polovinu, obrázek (c) znázorňuje

počet bodů redukovaných na čtvrtinu a obrázek (d) znázorňuje počet bodů redukovaných na osminu.



Obrázek 3-4:Úrovně vyhlazení hrany rozpoznaného artefaktu.

Pro nalezení odklonění od svislé osy je nutno nalézt ve výše zmíněném poli bodů vrcholové body artefaktu, které určí, do jaké míry je artefakt na zdrojovém obraze natočen. Tyto body jsou nalezeny způsobem, kdy aplikace v cyklu porovnává jednotlivé vzdálenosti jednotlivých bodů. Toto porovnání si můžeme dovolit, protože je nám známo, že artefakt je vždy vyšší než širší a tudíž nedojde k jeho chybnému natočení. Pro nalezení vzdálenosti dvou bodů nám poslouží Pythagorova věta o pravoúhlém trojúhelníku.

$$r = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{3.1}$$

V tomto známém vzorci značí proměnné *X* a *Y* body a hodnota *r* hledanou maximální vzdálenost bodů. V případě použití komplexních čísel by stačilo vypočítat pouze jejich rozdíl a znali bychom vzdálenost počítaných bodů. Ale postupně si dokážeme, že výhody pole oproti vektoru složeného z komplexních čísel , jsou pro další práci více užitečné. Proto budeme i nadále respektovat návratový datový typ metody *GetConture*.

Po nalezení dvou nejvzdálenějších bodů artefaktu následuje určení úhlu odklonu od svislé osy zdrojového obrazu. Současně s tímto krokem si stanovíme budoucí nulový bod souřadného systému. Jako výchozí nulový určíme bod, který leží na nalezené odkloněné ose artefaktu a reprezentuje jeho nespodnější bod. Tento bod bude později přesunut do polohy [0,0]. Druhý bod této osy bude sloužit jako centrální bod rotace celého artefaktu.

Mohly být zvoleny i jiné body jako je například těžiště či spodní vrchol artefaktu, ale vhodnější je první uvedená varianta.

Nyní přejdu k samotné realizaci natočení artefaktu. Jako první je nutno zjistit úhel osy. Zjištění úhlu vypočítám pomocí vztahu goniometrických funkcí a komplexních čísel využitím metody převodu goniometrického tvaru komplexního čísla na algebraický tvar.

Goniometrický tvar komplexního čísla je:

$$z = r(\cos\varphi + i\sin\varphi), \text{ kde } r = |z|,\ \varphi = \arg z. \tag{3.2}$$

Převod n algebraický tvar je dán rovnicemi:

$$x = r * \cos\varphi \text{ a } y = r * \sin\varphi. \tag{3.3}$$

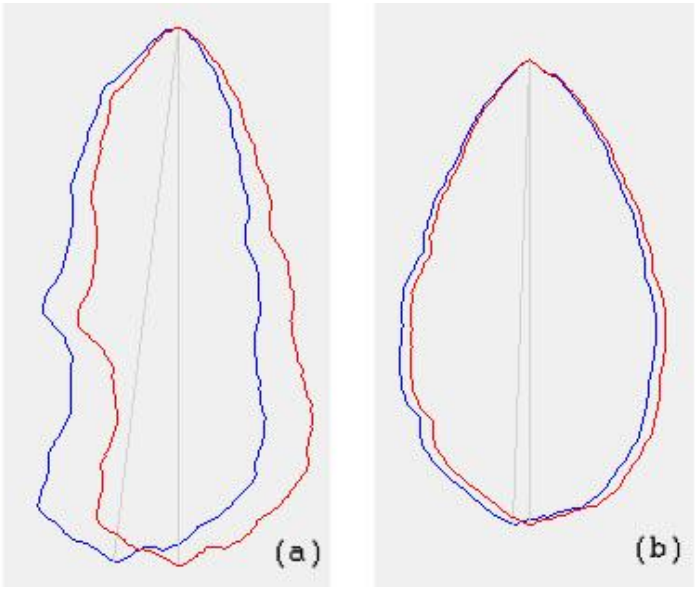
Dále platí vztahy:

$$\cos\varphi = \frac{x}{r}, \sin\varphi = \frac{y}{r}, \tan\varphi = \frac{y}{x}. \tag{3.4}$$

Odtud už dokážeme zjistit hodnotu úhlu φ dosazením do vzorce $\tan\varphi = \frac{\Delta y}{\Delta x}$, kde Δx a Δy jsou rozdíly hodnot *x* a *y* u nalezených nejvzdálenějších bodů artefaktu.

Po úspěšném nalezení úhlu osy řešíme následující krok, kdy postupně pootočíme všemi body okolo zvoleného centrálního bodu o zjištěný úhel. Pro toto posunutí v 2D prostoru budeme opět využívat předešlých vztahů. Při pootočení bodu je třeba zjistit jeho aktuální úhel oproti svislici souřadného systému ze současné polohy bodu a vzdálenost bodu od centrálního bodu. Nyní odečteme od zjištěného úhlu celkový úhel natočení artefaktu, zpětně dopočítáme novou polohu bodu z posunutého úhlu a vzdálenosti bodu od centrálního bodu. Tato vzdálenost je v rámci celé operace neměnná, protože body se posouvají po kružnici vzhledem centrálnímu bodu.

Tato operace se provede pro každý bod a nově dopočítané hodnoty při uloží nahradí hodnoty původní. Na následujícím *Obrázek 3-5* je zřejmé natočení objektů (a) a (b) do svislé polohy. Je jednoznačně viditelné, že vytvořený nástroj dokáže odhalit i minimální natočení. Toto natočení dokládá obrázek (b).



Obrázek 3-5:Přichycení detekovaného artefaktu ke svislé ose.

Modrá hrana artefaktu znázorňuje původně natočenou pozici artefaktu ve zdrojovém obraze a červená hrana artefaktu znázorňuje artefakt přichycený ke svislé ose souřadného systému. Šedé čáry pak už jen zvýrazňují propojení nalezených nejvzdálenějších vrcholů.

3.5 Přichycení artefaktu do souřadnicového systému

Tuto problematiku jsem částečně už zmínil v předchozím bodě, kde jsem uvedl, jakým způsobem bude nalezen a zvolen onen nulový bod, od kterého se budou počítat ostatní hodnoty. Jako nulový bod bude zvolen nejspodnější bod artefaktu ležící na jeho svislé ose.

Postup přichycení artefaktu k zvolenému souřadnému systému či pomyslnému nulovému bodu bude opět řešen cyklem. V tomto cyklu budou postupně upraveny všechny body, od kterých bude odečtena hodnota centrálního bodu.

Při pozdějším vykreslování hrany artefaktu bude třeba každý bod opět přepočítat vzhledem k velikosti kreslícího plátna v uživatelském rozhraní systému, kde se vykreslují jednotlivé body, a vzhledem k výchozímu počátečnímu bodu kreslícího plátna. Přepočet polohy bodů je nezbytný, protože každý detekovaný artefakt má jinou velikost a také pozici, na které se nachází ve zdrojovém obraze. Pokud by nedošlo k přichycení hranových bodů artefaktu k souřadnicovému systému, pak by nebylo možné souběžně zobrazit více artefaktů a zároveň je vztáhnout k jednomu centrálnímu bodu za účelem jejich přehlednějšího vizuálního porovnání.

3.6 Zjištění rozměrů artefaktu

Nalezení velikosti artefaktu už částečně proběhlo v předešlých bodech, kdy jsme hledali dva body na hraně artefaktu, které mají od sebe největší vzdálenost. Hodnoty indexů nejvyššího a nejnižšího bodu v poli uložil rozpoznávací nástroj v předchozích krocích do atributů objektu třídy *cArtefact*. Indexování bodů v poli je výhodnějším způsobem uložení dat než vektor, který indexování neumožňuje. Při použití datového typu vektor by musela aplikace vždy nalézt tyto krajní body. Třída *cArtefact* slouží pro uchovávání informací o aktuálně načteném či načítaném artefaktu. Jedná se o mnou vytvořenou třídu, kterou budu podrobněji popisovat v podkapitole 5.2.

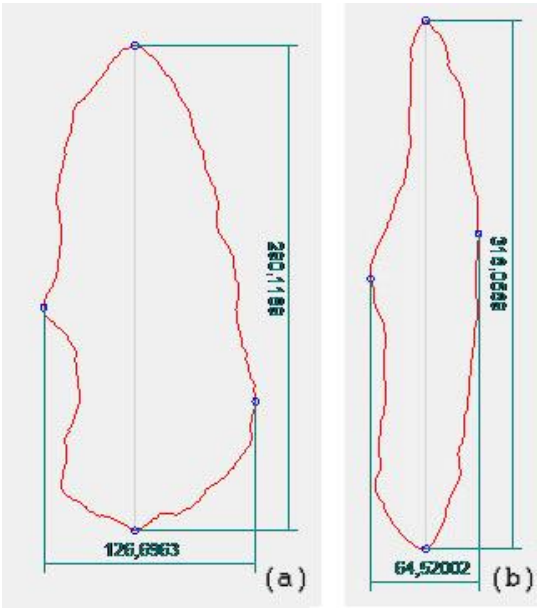
Při znalosti nejvyššího a nejnižšího bodu už jednoduchým způsobem zjistíme výšku artefaktu na zdrojovém obraze. A jelikož jsme nejnižší bod umístili do pozice [0,0], tak nám stačí pouze hodnota souřadnice *Y* nejvyššího bodu. Tuto nalezenou hodnotu uložíme opět do objektu třídy cArtefact. Většina artefaktů je vyfocena i z boční strany, tudíž při rozpoznání těchto dat budeme mít v tomto kroku další údaj o výšce artefaktu. Tyto dvě nalezené výšky by měly být shodné. Není však vyloučena nějaká minimální odchylka, proto bude do údaje o výšce vložen průměr těchto hodnot.

Tímto jsme určili výšku artefaktu. Nyní ještě musíme najít další dva hledané rozměry zkoumaného artefaktu. Jsou to šířka a hloubka. Obě tyto hodnoty budeme hledat stejně, protože jde vždy o vodorovný rozměr artefaktu ze zdrojového obrázku. Pokud je rozpoznáván čelní pohled, pak rozměr artefaktu vztažený k vodorovné ose je šířka. Pokud se jedná o rozpoznání bočního pohledu artefaktu, pak výsledkem bude hloubka artefaktu.

Vodorovný rozměr artefaktu zjistíme poměrně jednoduchým způsobem, kdy se postupně procházejí v cyklu všechny body a hledá se nejmenší a zároveň největší hodnota souřadnice *X*. Po dokončení vyhledávání vypočítáme rozdíl nalezených hodnot. Aby nedošlo k zápornému výsledku, převedeme výsledek do absolutní hodnoty.

Na následujícím *Obrázek 3-6*, kde je zobrazen čelní pohled artefaktu (obrázek (a)) a boční pohled artefaktu (obrázek (b)), je zřejmé, že nedošlo k vypočítání shodné výšky artefaktu. Příčina této neshody je v různých velikostech obrázků, které slouží jako vstupní data. Proto na každém vstupním obrázku je umístěna měrka o přesně dohodnuté velikosti. Sloužící jako jednotné měřítko, podle kterého je nutno zjistit přesné rozměry artefaktu. Aktuální

hodnoty na *Obrázek 3-6* jsou vyjádřeny jako rozměry v pixelech, které jsou potřeby přesného měření nepoužitelné.



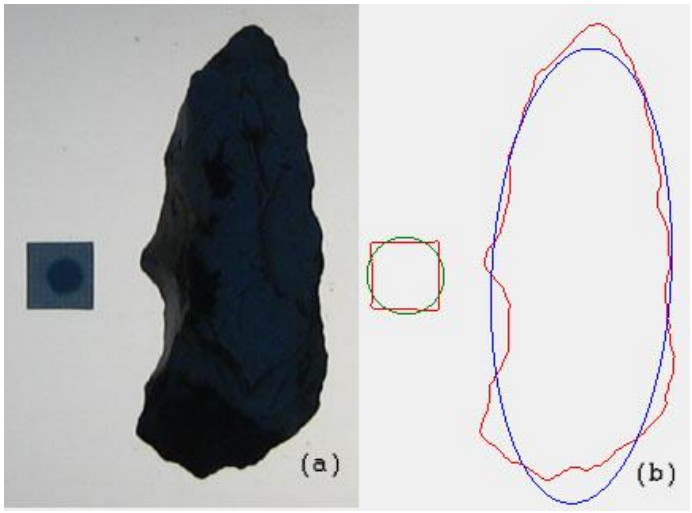
Obrázek 3-6:Zjištění relativním rozměrů artefaktu.

3.7 Nalezení měrky a zjištění reálné velikosti artefaktu

Měrka je objekt, který je na každém zdrojovém obraze. Pomocí měrky zjišťujeme, jak je artefakt na tomto zdrojovém obraze ve skutečnosti veliký. Porovnání více artefaktů by bylo zavádějící a chybné. Na zdrojových datech, která jsem měl k dispozici, byly použity dva typy měrek. První druh byl vhodný pro prvotní manuální změření velikosti, protože se jednalo o „archeologické pravítko“, na kterém byly použity kontrastní barvy pro lepší změření velikosti artefaktu. Tento typ měřítka ovšem není vhodný pro automatické rozpoznání dat, protože měřítko je rozloženo do několika dílčích objektů. U nich už nelze se spolehlivostí říct, že známe přesnou velikost jednotlivých částí. Proto bylo použito druhého typu měrky: Tu reprezentuje čtverec o délce hrany 1cm a s jednotnou, většinou černou barvou povrchu, což je pro automatické rozpoznání obrazu ideální. Na následujícím *Obrázek 3-7* jsou uvedeny jednotlivé měrky. Členitá měrka nevhodná pro automatické rozpoznání je na obrázku (a) a jednoduchá měrka vhodná pro automatické rozpoznání je v blízkosti artefaktu na obrázku (b).

Obrázek 3-7:Typy použitých měrek.

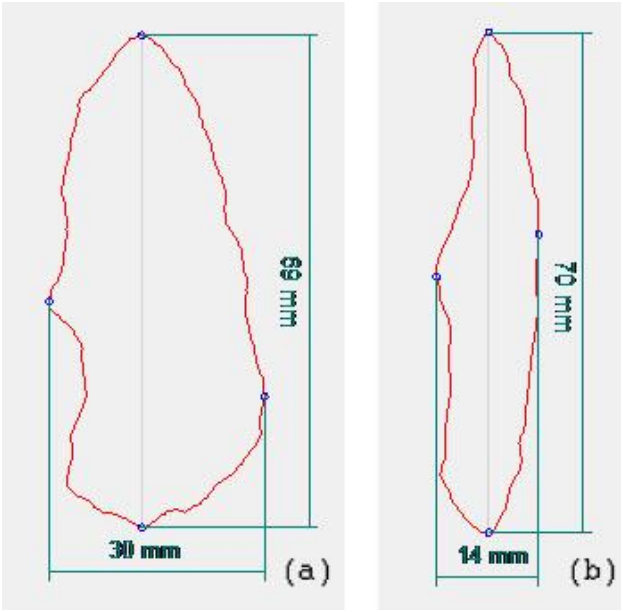
Nalezení měrky v obraze proběhne podobnou metodou jako nalezení artefaktu a to pomocí metody, kdy se aplikace snaží „obalit“ analyzovaný objekt elipsou. Pokud dojde ve zdrojovém obraze k nalezení dalšího takového objektu mimo již nalezeného artefaktu, potom zkoumáme, zda se nejedná o nalezenou měrku. Při jejím hledání se řídíme znalostmi o měrce. Zjišťujeme, zda nalezený objekt je čtverec. K tomuto zjištění nám opět pomůže detekující elipsa, u které porovnáme její výsku a šířku. Pokud si jsou tyto dvě hodnoty rovny (je uvažována malá tolerance), tak je zřejmé, že došlo k nalezení měrky. V dole uvedeném obrázku (a) jde vidět, jak měrka ve skutečnosti vypadá. Na obrázku (b) je již zvýrazněna detekovaná měrka zelenou „elipsou“.



Obrázek 3-8:Nalezení měrky v obraze.

V dalším kroku zjišťujeme velikost nalezené měrky. Nabízí se dvě možnosti řešení. První možnost spočívá v nalezení šířky i výšky měrky pomocí metody zjišťování šířky artefaktu. Aby bylo dosaženo přesnější hodnoty, dojde po získání těchto hodnot rovněž k jejich zprůměrování,

Další možností je nalezení dvou nejvzdálenějších bodů měrky, což budou její protilehlé rohy, ze kterých podobným způsobem, jak bylo popisováno u první metody, získáme její rozměry. Po nalezení rozměrů měrky je nutno upravit nalezené body hrany artefaktu. Při tomto kroku dojde k vynásobení jednotlivých hodnot získaným koeficientem založeným na poměru velikosti nalezené měrky ve zdrojovém obraze a měrky ve skutečnosti. K úpravě hodnot všech bodů artefaktu dojde opět v cyklu. Na následujícím *Obrázek 3-9* je zobrazeno, že velikosti čelního a bočního pohledu artefaktu se téměř rovnají a tudíž je dosaženo očekávaného cíle této podkapitoly. Rozdíl velikosti artefaktu na čelním a bočním snímku je pouze 1 milimetr. To v důsledku znamená rozdíl několika pixelů, což může být důsledek ztráty některých okrajových bodů artefaktu nebo měrky a to především v části prahování, kde se těmto nepatrným ztrátám neubráníme.



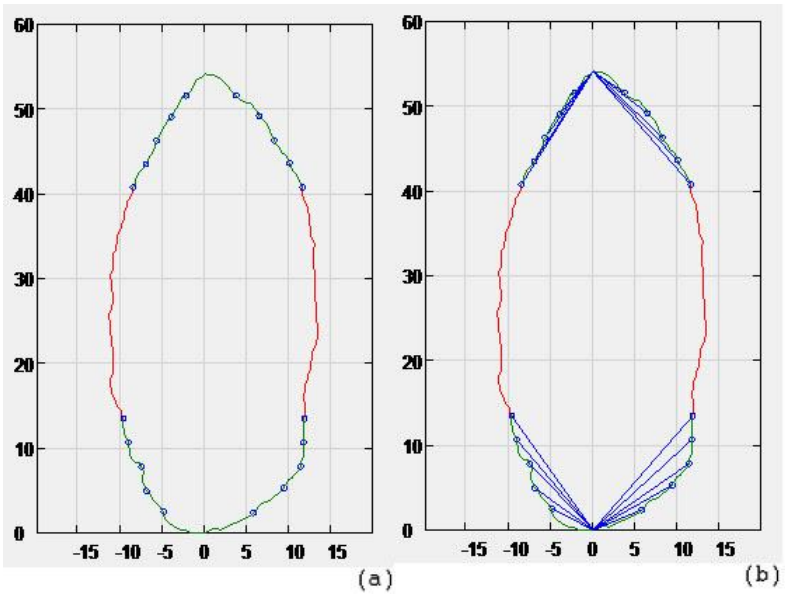
Obrázek 3-9:Skutečné rozměry detekovaného artefaktu.

Zhodnocení měření a porovnání s realitou bude provedeno v poslední části této kapitoly.

3.8 Nalezení dalších parametrů artefaktu

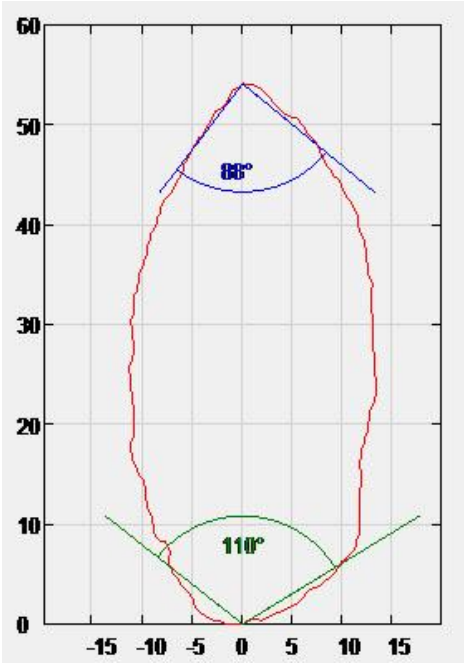
V této podkapitole se zaměřím na získávání horního a spodního úhlu artefaktu. Tyto nalezené úhly, jsou také jednou z požadovaných hodnot u rozpoznávaného artefaktu. Jednak slouží pro určení jejich původního účelu a také pro rozdělení do skupin podle ostrosti či tuposti jednotlivých úhlů.

Pro proces zjišťování vrcholových úhlů artefaktu jsem zvolil postup, kdy si v okolí vrcholu určím několik bodů, kterými proložím pomyslnou čáru a počítám postupně jejich úhly vztažené k vodorovné souřadné ose. Z těchto úhlů potom určím jejich průměrem hledaný úhel, respektive půlku úhlu, protože každý z vrcholových úhlů je třeba počítat pro jednotlivé poloviny artefaktu zvlášť. Pomocné body, které leží na hraně artefaktu, jsou zvoleny podle vstupních parametrů funkce. Těmito parametry jsou počet pomocných bodů (úhlů) a poměrná výška artefaktu, do které se budou tyto body hledat. Na dole obrázku (a) jsou znázorněny jednotlivé pomocné body, kterými jsou na obrázku (b) proloženy spojnice s vrcholem, u kterého je úhel zjišťován.



Obrázek 3-10: Výpočet vrcholových úhlů artefaktu.

Po vypočítání všech dílčích úhlů známe přibližný úhel vrcholu artefaktu. Jelikož tvar vrcholu není ve většině případů pravidelný, pak se musíme akceptovat tento vypočítaný údaj. Na *Obrázek 3-11* je výstup z aplikace, který obsahuje hledané úhly artefaktu.



Obrázek 3-11: Vrcholové úhly artefaktu.

Jak je z obrázku patrné, nalezené vrcholové úhly ne vždy zcela přesně kopírují hranu objektu v místě vrcholu. Pokud je to možné, kopíruje nalezený úhel alespoň přibližně obrys vrcholu artefaktu. Tímto krokem máme nalezen jeden z posledních parametrů hrotu.

Oproti všem předešlým obrázkům v minulých kapitolách, je na obrázku nahoře zobrazena i stupnice, protože už známe skutečnou velikost artefaktu a lze tedy na ní reálné hodnoty vynést.

Posledním zjišťovaným parametrem objektu je obvod artefaktu. Nejedná se už o složitý výpočet, protože v této fázi už máme pole hranových bodů artefaktu převedeno do skutečné velikosti a můžeme jednoduše vypočítat jeho celkový obvod.

Výpočet obvodu artefaktu proběhne v cyklu, kdy jsou spočítány a sečteny vzdálenosti mezi jednotlivými hraničními body artefaktu.

3.9 Výsledky měření

Celá tato kapitola pojednává o zjištění možných informací o artefaktu ze zdrojových dat. Ze získatelných údajů je to výška, šířka a hloubka artefaktu. Pro zjištění hloubky artefaktu je potřeba ještě jeden zdrojový boční obraz artefaktu. Tento obraz nám poslouží i pro ověření výšky artefaktu, která by měla být po rozpoznání obou pohledů shodná. V následující Tabulka 3.1 je uvedeno několik výsledků pro ilustraci přesnosti rozpoznání.

Tabulka 3.1:Porovnání naměřených hodnot artefaktu.

Označení artefaktu	Výška čelní (mm)	Výška boční (mm)	Rozdíl (mm)
L8462	45,5	46,22	0,72
L57742	86,281	87,22	0,939
L8483	30,67	30,59	0,08

Z tabulky je patrné, že výsledky měření jsou dostatečně přesné a pro určování parametrů artefaktu a jeho zařazení dostačující.

4 NÁVRH ŘEŠENÍ SYSTÉMU

V této kapitole se věnuji popisu požadavků uživatelského rozhraní a ostatních jeho částí, které budou potřeba pro jeho následnou realizaci.

Hlavním požadavkem zadání bylo vytvořit uživatelsky přívětivé prostření, které bude umožňovat správu artefaktů. Pro realizaci prostředí byla zvolena „oknová“ aplikace vytvořená v programovacím jazyce *C# .NET*, za pomoci programovacího prostředí Microsoft Visual Studio 2008.

Před začátkem vývoje uživatelského systému je třeba si vytvořit návrh všech jeho částí, aby bylo později jasno, jak bude aplikace fungovat a aby nedocházelo k zbytečným úpravám již vytvořeného rozhraní. To může vést k eventuelním chybám a prodloužení vývoje systému. Proto je důležité si pečlivě rozmyslet, jak bude celý systém realizován, jak bude ukládat data, jakým způsobem budou organizovány třídy aplikace a zda bude nějak řešeno zabezpečení dat. Proto tato kapitola obsahuje tři hlavní podkapitoly, ve kterých jsou řešeny následující problémy:

- stanovení základních požadavků na systém a vymezení jeho očekávané funkčnosti,

- stanovení celkové architektury systému, způsobu komunikace aplikace s databází a uvedení diagram tříd, který má tuto komunikaci na starosti.

4.1 Analýza a návrh systému

Na počátku každého systému je nezbytnou částí jeho analýza, která pomůže ve specifikaci požadavků na nový systém. První obrysy systému vzniknou na základě požadavků zákazníka, které jsou mnohdy obecné, protože zákazník nemá přesnou představu, co požaduje. Proto je nutno zákazníka, na základě našich analytických zkušeností dovést k reálnému řešení zadání. V následující části je nutno získané obecné požadavky převést do přesnější formy, ze které je vytvořeno zadání projektu. O toto zadání se bude potom vývoj aplikace opírat.

Ze získaného zadání je vytvořen diagram případů užití, který vymezí funkčnost aplikace a možně operace v rámci aplikace. Součástí tohoto vymezení je třeba také uvést funkční a nefunkční požadavky systému.

4.1.1 Získání prvotních informací

V této části probíhá dialog mezi zadavatelem aplikace (zákazníkem) a jejím realizátorem (analytikem, vývojářem). Pro získání informací o budoucí aplikaci existuje několik metod, kterou jsou například dotazníky sloužící pro snazší rozhodování většinou nerozhodných zákazníků. Výsledek těchto dotazníků je pak potřebný pro vznik zadání. Nutno dodat, že výsledek dotazníků je tak kvalitní, jak byl jejich obsah, a proto je nutné psát tyto dotazníky stručně a srozumitelně.

V případě tohoto systému byly požadavky na vyvíjený systém stanoveny po konzultaci s budoucími uživateli v rámci jednoho setkání.

Hlavní požadavky na systém:

- Detekce artefaktu a zjištění jeho tvaru a rozměrů ze zdrojové fotografie.
- U artefaktu zjišťovat přesný rozměr a vrcholové úhly.
- Kategorizace nalezeného artefaktu.
- Uchovávání získaných dat.
- Dávkové zpracování zdrojových fotografií.
- Přehledné uživatelské rozhraní.
- Exporty naměřených dat

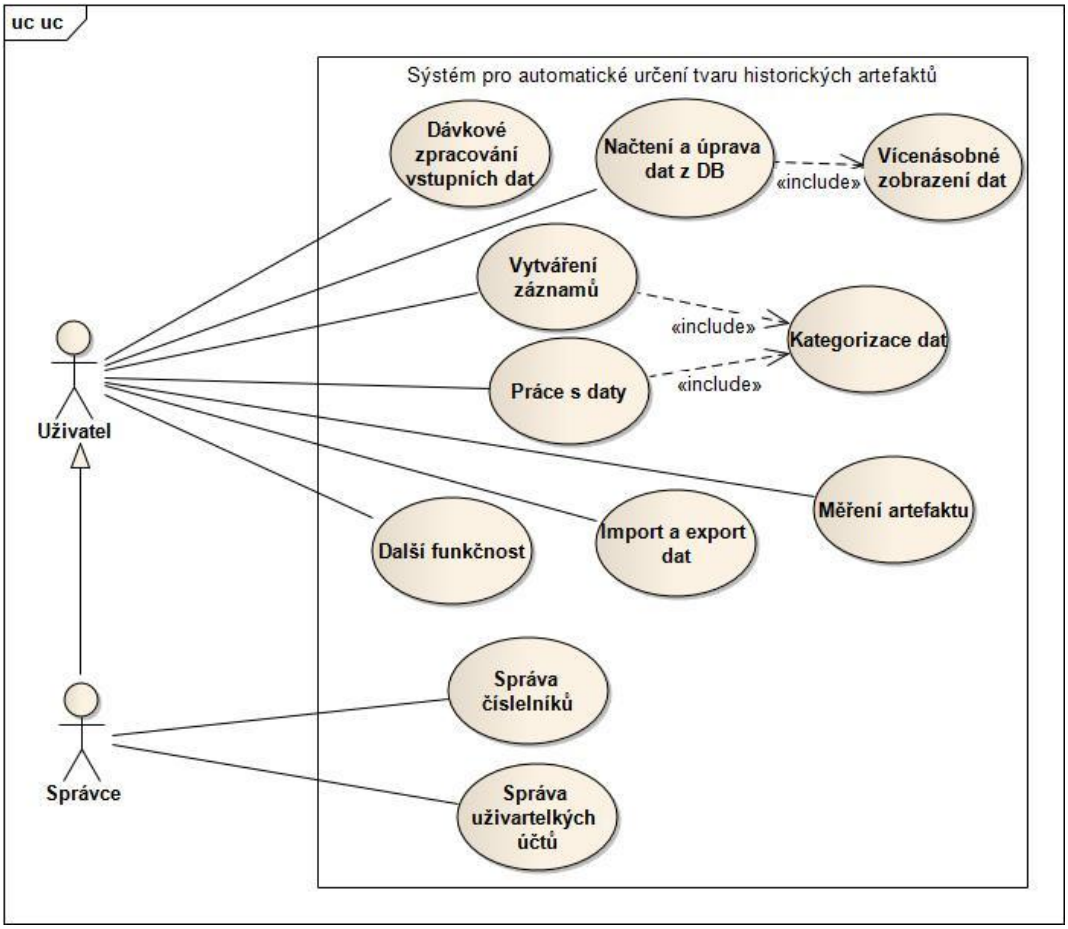
4.1.2 Příklady užití

Diagram případů užití znázorňuje pohled na budoucí systém z vnějšího pohledu, čímž slouží k nalezení omezení systému a je následně podkladem při odhadu jeho rozsahu. V diagramu případů užití jde o návaznost souvisejících akcí mezi aktérem a navrhovaným systémem v průběhu jejich interakce. Očekávaným cílem je nalezení aktérů, kteří se systém komunikují, a také vztahů mezi službami a jejich poskytovateli. Tato komunikace se v diagramu zachycuje textovou i vizuální podobou, která je pochopitelná, jak ze strany budoucích uživatelů systému, tak i vývojářům. [9]

Na základě získaných údajů a informací z předešlé podkapitoly, byl sestaven digram případu užití, který znázorňuje hlavní funkčnost aplikace doplněnou o nezbytné části, které jsou nutné pro správný a bezchybný chod aplikace.

Následující diagram popisuje funkce, které budou moci uživatelé realizovaného systému používat. Některé z uváděných funkcí jsou nutně závislé na roli uživatele.

Na následujícím *Obrázek 4-1* lze vidět dva typy elementů diagramu, kterými jsou aktéři a případy užití. Příklady užití značí posloupnost akcí ve vztahu s aktéry a obsahují také vazby (include, extend, generalization). Aktér označuje externí objekt, kterému náleží určité procesy a obsahuje také vazby (generalization).



Obrázek 4-1:Diagram Případů užití.

Aktéři se dají v rámci aplikace rozdělit na typy (role), kterými jsou uživatelé a správce systému.

- **Uživatel** je běžná osoba, která bude mít možnost ovládat všechny standardní funkce programu.
- **Správce** systému bude mít stejné pravomoci jako uživatel a bude navíc oprávněn upravovat číselníky aplikace a spravovat uživatelské účty.

Příkladů užití je mnoho a proto zde uvedu jen ty klíčové, aby bylo možno lépe chápat hlavní funkčnost budoucího systému.

- **Rozpoznání artefaktu –** pomocí definovaných metod získat tvar artefaktu ze zdrojového obrazu.
- **Dávkové zpracování vstupních dat** – slouží pro zpracování většího množství zdrojových obrazů, kdy je každý obrázek jednotlivě zpracován, je na něm nalezen hledaný artefakt a získané údaje jsou uloženy do databáze.
- **Načtení a editace dat z DB** – slouží pro opětovné otevření záznamu o již rozpoznaném artefaktu, kdy je možno editovat jeho dílčí údaje.
- **Měření artefaktu** – pomocí dostupných nástrojů bude umožněno dále zkoumat nalezený artefakt a zobrazovat tyto hodnoty podle různých zobrazovacích funkcí.
- **Kategorizace dat** – členění nalezených artefaktů do kategorií pro lepší orientaci a práci s daty.
- **Export dat** – umožnit uživatelům exportovat do požadovaných formátů dříve rozpoznaná a uložená data.

4.1.3 Funkční a nefunkční požadavky

Před tím, než začnu popisovat a rozdělovat jednotlivé požadavky, je třeba si říct, co je myšleno pojmy funkční a nefunkční požadavky. Pojem Funkční požadavky definuje skupinu vlastností budoucího systému, které budou uživatelé přímo využívat nebo s nimi pracovat jiným způsobem.

Skupina nefunkčních požadavků definuje vlastnost systému jako celku a jeho omezení. Tyto nefunkční požadavky se týkají produktu i procesu vývoje systému. Jako možná metrika mohou sloužit vlastnosti, jako jsou rychlost, velikost, použitelnost, spolehlivost, robustnost a přenositelnost systému.

Funkční požadavky

- Detekce a rozpoznání artefaktů ze zdrojových dat.
- Stromová struktura kategorizovaných artefaktů.
- Uživatel bude moci vkládat, editovat a mazat artefakty.

- Dávkové zpracování vstupních dat.
- Export dat do požadovaných formátů
- Správa číselníků a seznamu uživatelů
- Rozdělení uživatelů do rolí.
- A další…

Nefunkční požadavky

- Nová aplikace je typu WinForm.
- Aplikace je naprogramována pomocí jazyka C# .NET.
- Aplikace pro ukládání dat využívá databázi MSSQL.
- Rychlost a přesnost detekce artefaktu.
- Rozšiřitelnost systému.

4.2 Návrh architektury systému

Architektura systému je založena na trojvrstvé architektuře, která umožňuje lepší přehlednost zdrojového kódu aplikace a jeho následné úpravy. Trojvrstvá architektura se skládá z datové, aplikační a prezentační vrstvy.

Datová vrstva slouží pro přístup k informacím uložených na datovém úložišti.

Aplikační vrstva je prostředníkem mezi datovou a prezentační vrstvou. V této vrstvě dochází k transformaci dat mezi datovou vrstvou a vstupně/výstupními požadavky.

Prezentační vrstva slouží pro zobrazování dat z aplikační vrstvy a právě s touto vrstvou přímo pracuje uživatel.

Výhoda této architektury spočívá v jednoduchém nahrazení kterékoliv vrstvy jinou, která komunikuje pomocí stejných vstupů a výstupů a celkový chod systému zůstane zachován. Při realizaci systému došlo k částečnému propojení datové a aplikační vrstvy. K tomuto propojení vrstev došlo při vytváření některých hlavních tříd, kdy třída obsahuje metody z obou vrstev.

Pro potřeby ukládání dat je zvolena databáze, která je v dnešní době už samozřejmostí systémů pracujících s uloženými daty. Požadavkem na databázi je nutnost, aby databáze

zvládala vícenásobný souběžný uživatelský přístup a velké množství uložených dat v rámci jedné tabulky.

4.2.1 E-R diagram

Entity-relationship diagram je grafickým vyjádřením entit a vztahů mezi nimi. Entitou je každý objekt, nebo událost, o které chceme evidovat informace. Entita je fyzický objekt (nebo abstraktní pojem), který má nějaké vlastnosti nazývané atributy. Fyzickou reprezentaci existence dané entity v databázi bude relace, respektive relační tabulka. Mezi vlastnostmi dané entity by mela byt minimálně jedna taková, která entitu jednoznačně identifikuje a rozlišuje ji od ostatních entit téhož typu. Tuto vlastnost nazýváme primární klíč. [10]

Nejčastější vyjádřením E-R diagramu je schéma či graf. Vrcholy grafu jsou jednotlivé entity a hrany grafu vztahy mezi nimi. Parametrem jednotlivých vrcholu (entit) jsou jejich atributy, popřípadě datové typy (obory hodnot) těchto atributu. Parametrem vztahu je jejich typ (1:1, 1:m, m:n), který udává vztahy mezi jednotlivými entitami.

Následující schéma databáze, uvedené na *Obrázek 4-2,* znázorňuje databázový model aplikace, který pokrývá svým rozdělením na jednotlivé tabulky, požadované nároky na ukládaná data a ostatní funkcionalitu systému. Pro lepší čitelnost uvádím diagram databáze v *příloze PI.*

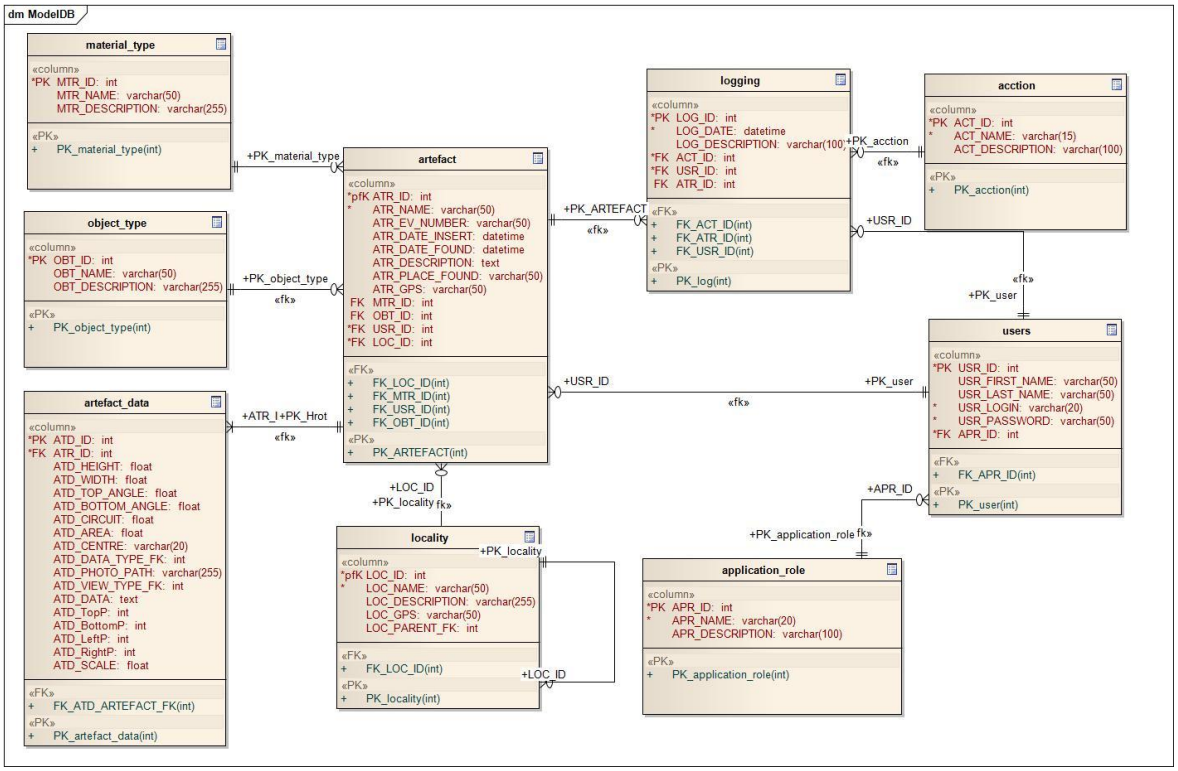
Ze všech tabulek obsažených E-R diagramu, popíši jen některé, které jsou klíčové pro hlavní funkcionalitu systému.

Tabulka **Artefact** – slouží pro ukládání základních informací o nalezených artefaktech, jako je název, lokalita, materiál, popis a další dílčí informace.

Tabulka **Artefact_data** – slouží pro ukládání samotných dat, jako jsou hraniční body artefaktu, jeho rozměry a vrcholové úhly. Tato tabulka byla nutností, protože jak jsem psal už výše, tak zdrojové fotografie každého artefaktu jsou dvě (čelní a boční) a docházelo by pak k redundantnosti dat, pokud by nedošlo k tomuto rozdělení.

Tabulka **Locality** – slouží pro kategorizaci artefaktů do hierarchie, která napomáhá lepší organizaci nalezených artefaktů.

Tabulka **Users** – slouží pro uložení přehledu uživatelů využívající systém.



Obrázek 4-2:E-R Diagram popisující strukturu databáze systému.

4.2.2 Diagram tříd

Diagram tříd znázorňuje statický pohled na systém, zejména třídy jako typy objektů, obsah tříd a statické vztahy, které mezi nimi existují. Diagram tříd patří do skupiny diagramů struktur. Tento diagram může obsahovat elementy chování (např. operace), ale jejich dynamika je vyjádřena jinými diagramy, kterými jsou diagram stavového stroje a diagram komunikací. [11] Diagram tříd částečné kopíruje E-R diagram.

Hlavní třídou celého diagramu tříd systému je třída *cArtefact*, která obsahuje téměř všechny proměnné, náležící do tabulek *Arteface* a *Artefact_data*. Spojení bylo nutno udělat, aby v rámci aplikace byla všechna data o artefaktu načtena přehledně do jednoho objektu třídy *cArtefact* a bylo možno s ním rychle pracovat a zobrazovat.

Jelikož se v E-R diagramu nachází několik tabulek, které slouží, jako číselníky, bude počet výsledných tříd oproti předchozímu E-R diagramu redukován.

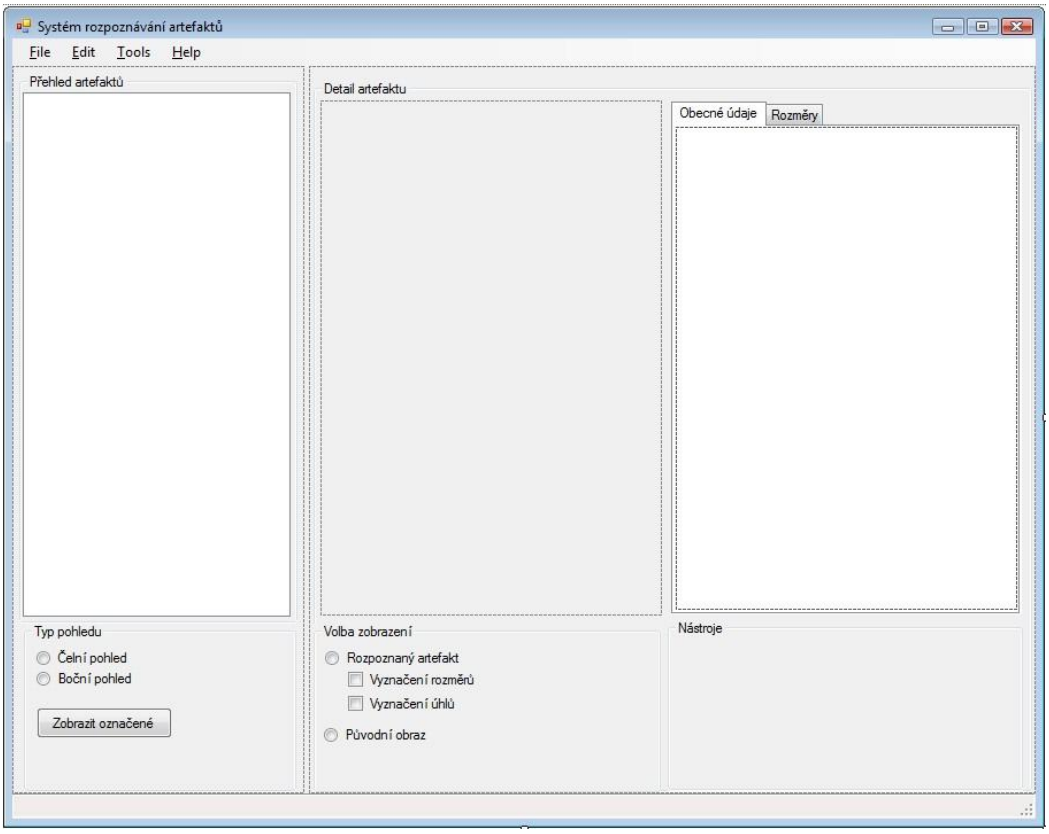
4.3 Návrh uživatelského rozhraní systému

Grafický návrh uživatelského rozhraní je více než vhodné vytvořit tak, aby byl kompromisem mezi ergonomií ovládání a poutavým a přehledným grafickým vzhledem.

Při vytváření rozhraní, by se měl jeho tvůrce vžit do role budoucího uživatele rozhraní či aplikace a navrhnout rozložení jednotlivých ovládacích prvků tak, aby se uživatel v něm mohl již od počátku intuitivně orientovat a ovládat ho.

Každé uživatelské rozhraní se dá rozdělit na více částí. První částí rozhraní, je jeho MENU, které je dostatečně a přehledně odděleno od ostatních částí uživatelského rozhraní. Z menu je dostupná veškerá funkčnost aplikace, která může být souběžně umístěna i v jiných částech rozhraní. Nepsaným pravidlem uživatelského rozhraní je dostupnost Menu při jakémkoliv stavu aplikace.

V druhé části rozhraní je umístěna prezentační část aplikace, které zobrazuje veškeré možné informace o artefaktech a za pomoci ostatních doplňkových formulářů umožňuje ostatní funkčnost rozhraní, jako je správa číselníků, správa uživatelů a nahlížení do logu aplikace.



Obrázek 4-3:Návrh uživatelského rozhraní systému.

4.3.1 Požadované prvky rozhraní

Jak jsem už psal výše, uživatelské rozhraní systému má být přehledné a zároveň má splňovat požadovanou funkčnost. Proto je rozhraní rozděleno do několika formulářů, které se dělí na hlavní formulář a vedlejší pomocné formuláře.

Na hlavním formuláři je možno zobrazit již dříve rozpoznaná data o afektech, která jsou načítána z databáze. Data jsou na formuláři zobrazena jak sumárně formou stromu (*TreeView*), který je hierarchicky řazen podle lokality nálezu artefaktu. Z tohoto stromu je možno vybrat, jak jednotlivý artefakt, tak i více artefaktů současně.

Nutnou součástí hlavního rozhraní je také formulář pro dávkové načtení dat z většího množství zdrojových fotografií, který je parametrizovatelný a umožňuje nastavení jednotlivých vlastností importu, jako je stupeň vyhlazení, upřednostňovaný barevný kanál, poměrná výška artefaktu, ze které se získal úhel jeho vrcholů a lokalita nálezu pro jeho lepší zařazení.

Ostatní formuláře slouží pro přehled a správu číselníků aplikace, jako jsou typ materiálu, lokalita nálezu a typ objektu. Mezi tyto formuláře spadá i formulář pro správu uživatelů systému a formulář pro nahlížení do logu aplikace.

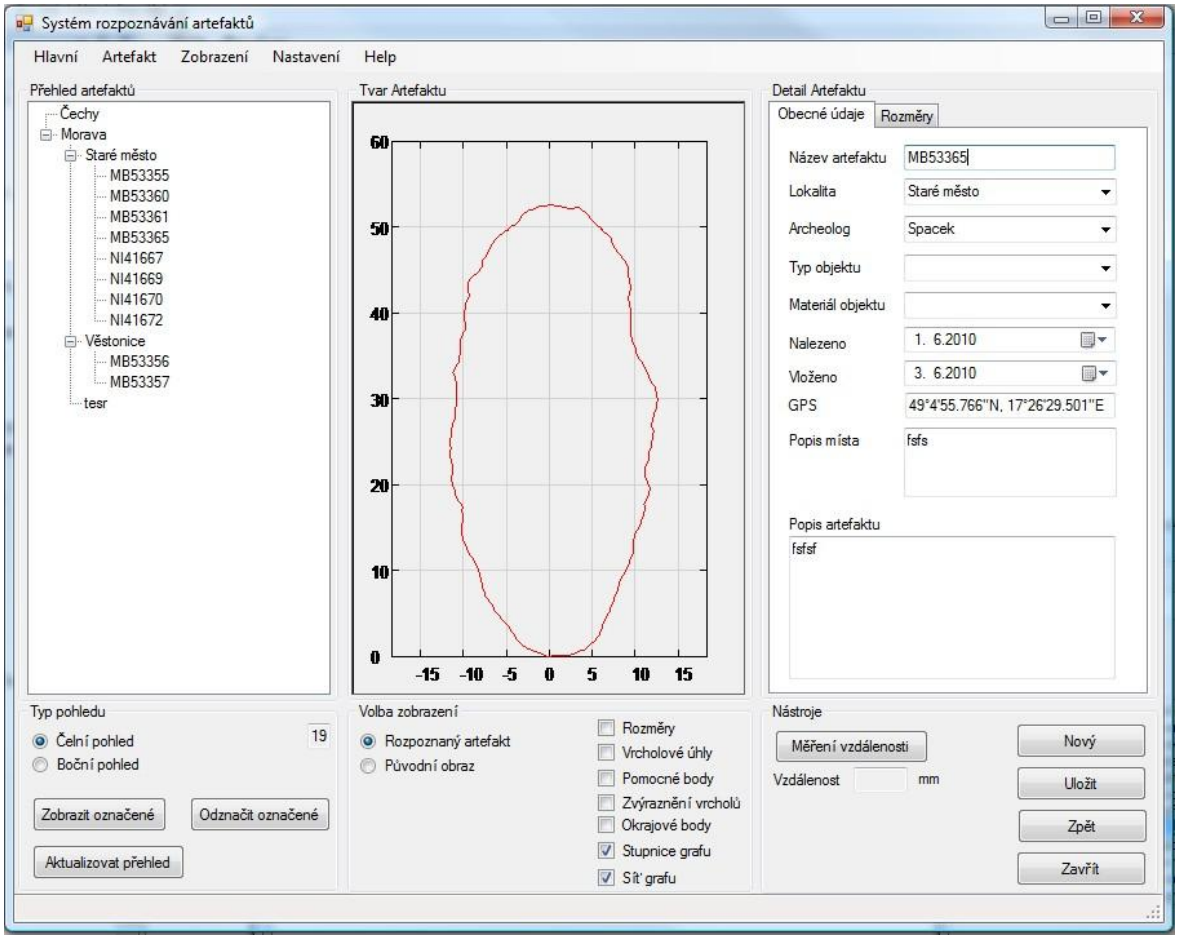
5 IMPLEMENTACE SYSTÉMU

V této kapitole jsou popsány všechny podstatné části, které bylo nezbytné vytvořit pro správně fungující požadovaný systém. Pro pokrytí požadované funkcionality bylo nutno vytvořit vlastní třídy, metody, funkce, uživatelské rozhraní a další časti jiné systému.

5.1 Rozhraní systému

Jak jsem se už zmínil, je třeba mít přehledné uživatelské rozhraní a to znamená, že nelze veškerou funkčnost pokrýt jedním formulářem, proto podle návrhu bylo vytvořeno v rámci implementace systému několik formulářů.

Hlavní rozhraní systému nebylo nikterak zásadně upravováno oproti původnímu návrhu. Došlo k doplnění nové funkcionality do formuláře a k nepatrnému přeskupení některých jeho prvků a datových polí.



Obrázek 5-1:Konečný vzhled uživatelského rozhraní systému.

Při vytváření hlavního formuláře a formulářů obecně je třeba počítat s možnými změnami velikosti formulářů či velikosti rozlišení obrazovky. Tato problematika se řeší pomocí kotvení jednotlivých prvků formuláře k některé z jeho čtyř stran anebo ke kotvení k nadřazenému objektu, který možnost kotvení také umožňuje. Ukotvení objektu ke straně znamená pro objekt, že se jeho pozice vzhledem k této straně nebude nikterak měnit, i když dojde ke změně její velikosti.

Jak je patrné na předešlém náhledu, hlavní okno systému obsahuje všechny důležité prvky, které pokrývají hlavní funkčnost systému. Hlavní okno se dá rozdělit na dvě části, kde první část obsahuje hierarchicky seřazený seznam artefaktů a druhá část obsahuje detailní informace o artefaktu s jeho vykresleným obrysem.

5.1.1 Menu

Nedílnou součástí hlavního formuláře je Menu. Menu slouží k přehledné orientaci ve funkcích, možnostech zobrazení a nastavení systému. Jednotlivé položky menu pak obsahují veškerou funkcionalitu, která nemusí být v rámci formulářů dostupná.

Menu uživatelského rozhraní systému je rozděleno na následující hlavní oblasti:

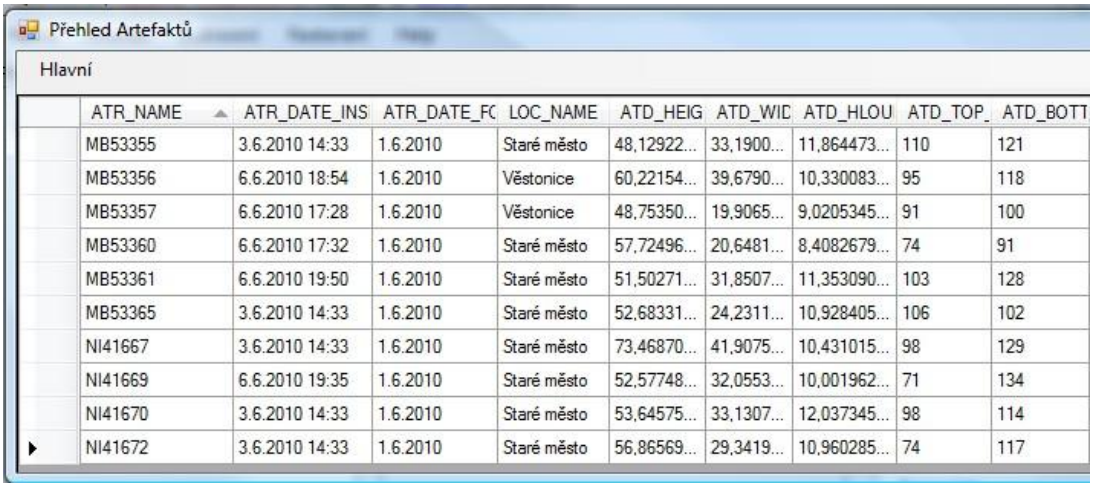
- **Hlavní** – v této části menu se nacházejí funkce, které jsou určeny pro základní ovládání uživatelského rozhraní, dávkového načtení artefaktů, export dat v požadovaných formátech a tisku výstupů.
- **Artefakt** – tato část menu obsahuje skupinu funkcí, která slouží pro správu, vytváření, editaci a případného odstranění artefaktu.
- **Zobrazení** – skupina funkcí a parametrů ovlivňující způsob vykreslení artefaktu na formuláři, při tisku a při uložení do souboru.
- **Nastavení** – ve skupině nastavení je nutno se nejprve pomocí přihlašovacího formuláře přihlásit jako administrátor a potom se zpřístupní možnosti správy jednotlivých číselníků aplikace.
- **Nápověda** – jedná se o informační část aplikace, kde je uvedena stručná nápověda a informace o uživatelském prostředí.

5.1.2 Seznam formulářů

V této části je uveden seznam všech formulářů, které vznikly v rámci zadání, aby pokryly požadovanou funkčnost zadání. Formuláře, neuvedené v textu mojí diplomové práce uvádím v *Příloze II*.

Seznam formulářů uživatelského rozhraní:

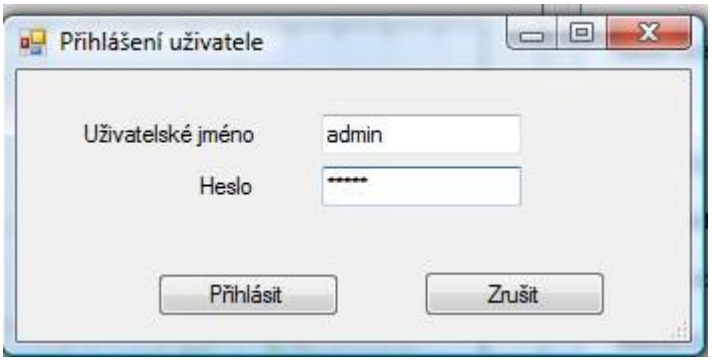
- **Hlavní formulář** uživatelského rozhraní, uvedený na *Obrázek 5-1*, slouží pro zobrazení informací o artefaktech a aplikování některých nástrojů a nastavení, které ovlivní samotné vykreslení artefaktu.
- Formulář **Hromadného přehledu artefaktů**, uvedený na *Obrázek 5-2*, slouží pro přehled artefaktů, který lze třídit podle různých kritérií.



ATR_NAME	ATR_DATE_INS	ATR_DATE_FC	LOC_NAME	ATD_HEIG	ATD_WID	ATD_HLOU	ATD_TOP_	ATD_BOTT
MB53355	3.6.2010 14:33	1.6.2010	Staré město	48,12922...	33,1900...	11,864473...	110	121
MB53356	6.6.2010 18:54	1.6.2010	Věstonice	60,22154...	39,6790...	10,330083...	95	118
MB53357	6.6.2010 17:28	1.6.2010	Věstonice	48,75350...	19,9065...	9,0205345...	91	100
MB53360	6.6.2010 17:32	1.6.2010	Staré město	57,72496...	20,6481...	8,4082679...	74	91
MB53361	6.6.2010 19:50	1.6.2010	Staré město	51,50271...	31,8507...	11,353090...	103	128
MB53365	3.6.2010 14:33	1.6.2010	Staré město	52,68331...	24,2311...	10,928405...	106	102
NI41667	3.6.2010 14:33	1.6.2010	Staré město	73,46870...	41,9075...	10,431015...	98	129
NI41669	6.6.2010 19:35	1.6.2010	Staré město	52,57748...	32,0553...	10,001962...	71	134
NI41670	3.6.2010 14:33	1.6.2010	Staré město	53,64575...	33,1307...	12,037345...	98	114
NI41672	3.6.2010 14:33	1.6.2010	Staré město	56,86569...	29,3419...	10,960285...	74	117

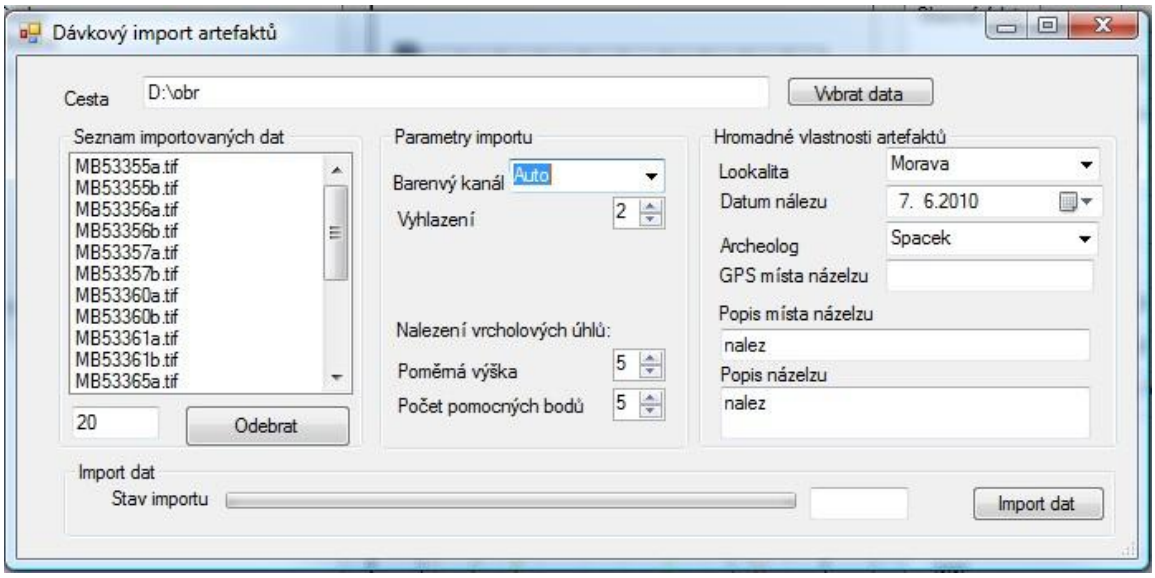
Obrázek 5-2:Hromadný přehled artefaktů.

- Formulář **Přihlášení uživatele** do systému, uvedený na *Obrázek 5-3*, slouží k ověření potřebných práv uživatele.



Obrázek 5-3: Přihlášení uživatele do systému.

- Formulář **Zpracování vstupních dat**, uvedený na *Obrázek 5-4,* je určen pro jednotlivé nebo hromadné načtení a rozpoznání artefaktů ze vstupních dat. Formulář obsahuje nastavitelné parametry importu, které ovlivňují proces rozpoznávání artefaktů.



Obrázek 5-4:Dávkový import zdrojových obrazových dat

- Formuláře **Správa typu materiálů** a **Správa typu objektu** slouží pro jednoduchou správu těchto dvou číselníků, které je třeba také upravovat.
- Formulář **Správa oblastí** je určen pro hierarchickou organizaci oblastí, což obnáší náročnější práci s daty formuláře. Formulář je navržen tak, aby práce s ním nebyla nikterak složitá.
- Formulář **Správa uživatelů** slouží pro celkovou organizaci uživatelů, kteří mohou používat tento systém a upravovat jim uživatelská práva, které definují jejich oprávnění v rámci systému.

5.1.3 Získávání dat

K získání dat o artefaktu slouží zdrojové obrázky s artefakty. Způsob získávání či vkládáni dat do databáze systému je dvojího druhu. Prvním způsobem je ruční vložení dat a druhý způsob je dávkové vložení dat.

Při každém zakládání artefaktu je do databáze vložen nový záznam, který uloží údaje do příslušných tabulek. Pokud dojde k opětovnému nahrání dat, tak jsou původní informace o

artefaktech přepsány. Původní záznamy jsou nalezeny díky názvu artefaktu, který je uložen v názvu souboru a v databázi je už neměnný. Jednotlivé hranové body artefaktu jsou v databázi uloženy v milimetrech, což je výhodnější pro pozdější prezentaci exportovaných dat.

5.1.4 Znázornění hranových bodů

Než začnu s popisem způsobu zobrazení nalezených hranových bodů, musím předeslat, že jsem neobjevil vhodný nástroj či knihovnu, který by svou funkcionalitou splňoval požadavky, které jsem od tohoto nástroje očekával. Proto jsem došel k závěru, že bude vhodnější si tuto vykreslovací součást systému vytvořit osobně.

Při vykreslování hranových bodů jsem použil kreslící plátno, které je svou osovou orientací vhodné pro znázornění 2D grafických dat ve formě souřadnic bodů o hodnotách [x, y]. Při jakékoliv práci je nutno mít na paměti, že bod [0,0] na kreslícím plátně leží v levém horním rohu a ne, jak je člověk zvyklý, v levém spodním rohu. Proto vzniká nutnost přepočítávání každého bodu. Celá záležitost se správným zobrazováním bodů je dále komplikována tím, že artefakt obsahuje bod [0,0] ve svém nejnižším bodě, který je ve výsledném zobrazení umístěn ve spodní části svislé osy plátna a proto je nutno k výsledné poloze připočíst další posunutí zobrazovaného bodu. Poslední stěžejní částí vykreslovaného bodu je použití vhodného měřítka, které určí, jak bude obrys hrany artefaktu zvětšen či zmenšen vzhledem ke kreslícímu plátnu, což má za následek další upravení výsledné pozice bodu. Spolu s tímto zvětšením se také přepočítává stupnice značící skutečnou velikost artefaktu.

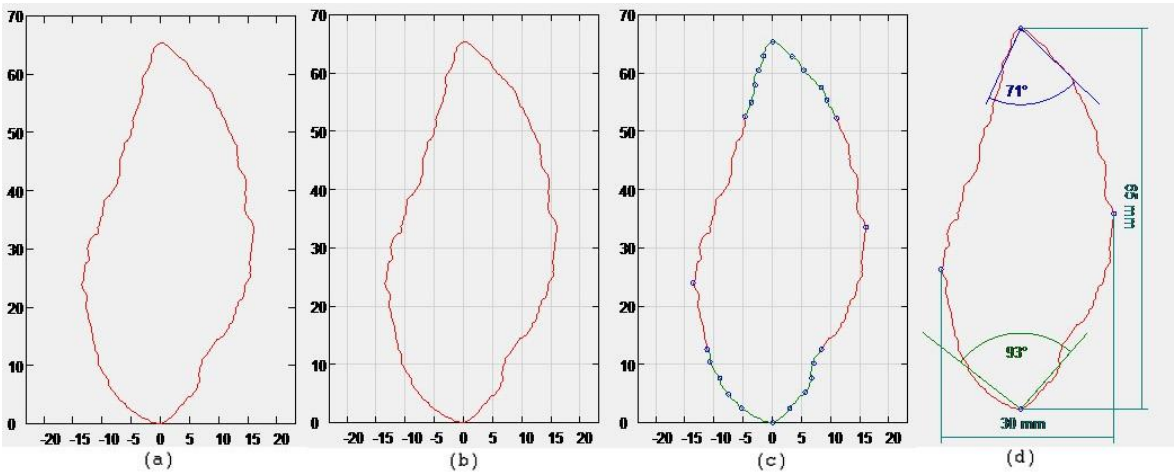
Zobrazení artefaktu také umožňuje vykreslení více vybraných artefaktů současně. Při této vícenásobné volbě artefaktů bude výsledné měřítko zvoleno podle největšího z vybraných artefaktů a ostatní artefakty budou přepočítány podle něj.

Vykreslení artefaktu má několik možností, kdy je umožněn uživateli výběr různých režimů zobrazení. Tyto možnosti jsou znázorněny na následujícím *Obrázek 5-5*.

Možnosti doplňkových údajů, které lze u artefaktu zobrazit jsou:

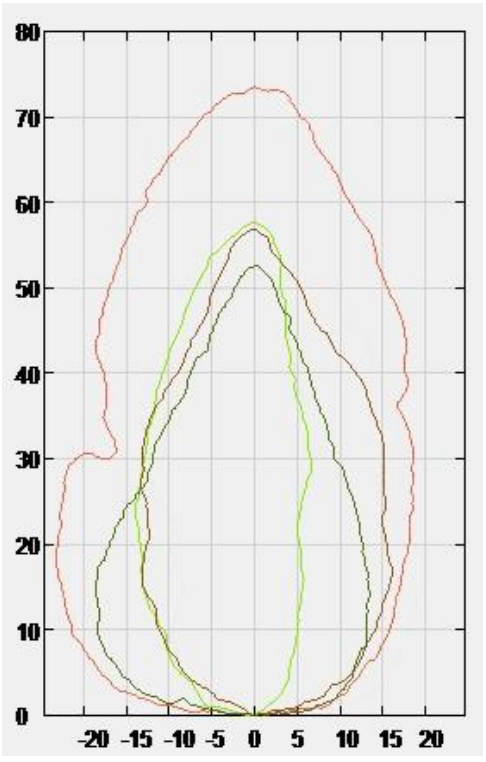
- Souřadnicový systém grafu, obrázek (a).
- Detailní síť grafu obrázek (b).
- Okótování zobrazeného artefaktu, obrázek (d).

- Zobrazení vrcholových úhlů artefaktu, obrázek (d).
- Zobrazení vrcholových bodů artefaktu, obrázek (c).
- Zobrazení pomocných bodů pro zjištění vrcholových bodů, obrázek (c).



Obrázek 5-5:Možnosti zobrazení artefaktu.

Rozhraní dále umožňuje zobrazení více vybraných artefaktů, jak je znázorněno na *Obrázek 5-6:Vícenásobný výběr.* obrázku. Z obrázku je patrná rozdílná velikost artefaktů a také jejich tvarů. Detailnější náhled je pak uveden v *Příloze PII-2.*



Obrázek 5-6:Vícenásobný výběr.

5.1.5 Export dat

Export dat ze systému může být dvojího typu. První typ exportu dat je určen pro hromadný export všech základních informacích o artefaktech, které se exportují do tabulky ve formátu aplikace MS Excel. Druhý typ exportu slouží pro vygenerování tabulky se souřadnicemi jednotlivých hranových bodů artefaktů, kde jsou vyznačeny důležité body, kterými jsou vrcholové body a body udávající šířku objektu. Na následujícím obrázku jsou vzorky dat obou typů exportů.

Tabulka 5.1: Hromadný export artefaktů.

Název	Nalezeno	Výška	Šířka	Hloubka	Vrcholový	Patový	Obvod
MB53360	1.6.2010	57,72	20,65	8,41	74	91	132,7043
MB53361	1.6.2010	51,5	31,85	11,35	103	128	144,8158
MB53365	1.6.2010	52,68	24,23	10,93	106	102	129,2348

Tabulka 5.2: Export hranových bodů artefaktu.

Název artefaktu.	MB53357		
Seznam bodu			
ID	**X**	**Y**	**Info**
0	-0,97	48,57	
3	0	48,75	Top
77	11,25	26,26	Right
173	0	0	Bottom
268	-8,66	29,7	Left

5.1.6 Ovládaní uživatelského rozhraní

Po spuštění systému na rozpoznání artefaktů se otevře hlavní okno aplikace, které je naplněno daty o artefaktech, načtenými z databáze. Tato data jsou vložena do hierarchicky seřazeného stromu podle lokalit nálezů jednotlivých artefaktů. Z tohoto stromu je pak

uživateli umožněno vybrat konkrétní artefakt, který bude pak načten do hlavního formuláře.

Jako příklady použití uživatelského rozhraní systému uvedu několik hlavních postupů (scénářů) pro ovládání rozhraní. Prezentovanou funkcionalitu uživatelského rozhraní systému lze realizovat z hlavního menu podle následujících postupů.

Stručné seznámení s jednotlivými postupy:

Vytvářeni artefaktu je proces, kdy se ručně zakládají nové artefakty do databáze. Tato funkce se spustí z menu Artefakt, kde se zvolí položka Nový artefakt. Při tomto procesu se k novému artefaktu vloží dílčí informace a z vybraného zdrojového obrázku se k tomuto artefaktu uloží nalezené hranové body. Tímto způsobem lze vložit ručně jednotlivé artefakty.

Dávkový import dat je určen pro postupný import většího množství dat z vybraného adresáře. Funkci dávkového importu lze spustit z menu Hlavní. Po zvolení této volby se otevře formulář, kde si uživatel zvolí adresář s daty, nastaví parametry rozpoznání a zvolí volbu importovat data.

Export dat slouží pro přenesení zvolených dat do požadovaného formátu. Funkce exportu dat uživatel spustí opět z menu Hlavní, kde je vybrán jeden ze dvou druhů exportů a do zvoleného adresáře bude uložen soubor s exportovanými daty.

Přehled artefaktů je formulář sloužící pro zobrazení informací o všech artefaktech a uživatel tento přehled otevře z menu Artefakt.

Ukládaní obrázku artefaktu slouží pro uložení aktuálně zobrazeného obrázku artefaktu na hlavním formuláři do souboru. Tuto funkci uživatel spustí z menu Artefakt

Přihlášení, odhlášení administrátora pokud uživatel potřebuje editovat číselníky aplikace nebo spravovat uživatele systému. Je nutno, aby se do aplikace přihlásil jako administrátor přes menu Nastavení/Přihlásit jako administrátor.

Správa číselníků systému se obsluhuje přes menu Nastavení, kdy po ověření role administrátora aktuálního uživatele, je umožněno editovat jednotlivé číselníky aplikace, spravovat oblasti a uživatele systému.

5.2 Datová a funkční část systému

Pro práci s daty v rámci uživatelského rozhraní bylo nutno vytvořit třídy, které reprezentují hlavní tabulky systému a umožňují veškeré operace s artefaktem. Třídy nad tabulkami jsou obecně určeny pro načtení dat z příslušné tabulky do jejich struktury a spouštět jim náležící metody. Po načtení dat do objektu třídy tabulky je dále možno s těmito daty dále v rámci aplikace pracovat. Pokud dojde ke změně dat v rámci formuláře, jsou změny nejprve zaneseny do objektu a až potom jsou změny uloženy do databáze.

Všechny tyto operace jsou řešeny v rámci metod, které také náleží jednotlivým objektům daných tříd. Tento způsob částečně splňuje vlastnosti trojvrstvé architektury, jak jsem se už zmiňoval v podkapitole 4.2.

Mimo nezbytných tříd tabulek, které řeší komunikaci mezi formulářem a databází, bylo nutno vytvořit celou řadu dalších funkcí, které pokrývají další funkčnost aplikace.

Třídy, které byly založeny s ohledem na databázové tabulky systému, jsou: *cArtefact*, *cUser* a *cLogging*.

Pro účely pomocných grafických funkcí byla založena statická třída *GraphicFunctions*, která obsahuje funkce, které jsou dále využívány hlavně při rozpoznávání a práci s artefakty. Pro zjištění prahovací hodnoty zdrojových obrázků byla vytvořena třída Otsu, která řeší nalezení této hodnoty pomocí Otsuovy metody.

Pro ukládání dat o artefaktech jsem zvolil databázi založenou na Microsoft SQL Serveru. Uživatelský systém je napojen na databázi pomocí komponent SqlConnector a SqlDataAdapter, které jsou součástí vývojového prostředí Microsoft Visual Studio, použitého pro vývoj systému. Komponenty *SqlConnector* a *SqlDataAdapter* usnadňují napojení tabulky do systému, kde jsou jednotlivé sloupce tabulek parametricky spravovány výše zmíněnými komponentami.

5.3 Použité nástroje

Pro realizaci zadaného systému jsem použil nástroje firmy Microsoft a konkrétně aplikací MS Visual Studio 2008 a MS SQL Server 2005. Pro vývoj aplikace jsem zvolil produkty od jedné firmy, kterou je firma Microsoft. K tomuto rozhodnutí mne vedlo obecné pravidlo, které říká, že produkty jednoho výrobce spolu lépe spolupracují než produkty

různých firem, i když se v tomto případě jedná o standardizovanou komunikaci pomocí jazyka SQL.

5.3.1 Vývojové prostředí

Vývojovým prostředím realizovaného uživatelného systému byl zvolen program MS Visual Studio, který přináší celou řadu komponent usnadňujících samotné vytváření rozhraní systému. Systém je napsán pomocí jazyka C# .NET, obsahující velké množství metod, které vedou k zjednodušení práce, jak jsem už předesílal.

5.3.2 Databáze

Pro databázi typu MSSQL jsem se rozhodl na základě dřívější volby vývojového prostředí. Databáze založena na Microsoft SQL Serveru také umožňuje širší škálu možností, než jiné menší databázové projekty, které jsou při větším množství uložených dat pomalejší a hůře snášejí souběžné vícenásobné přístupy k datům databáze.

5.3.3 Použité knihovny

Pro snazší práci se vstupními daty byla použita knihovna *OpenCVDotNet* library, která mi velkou mírou pomohla s funkcemi, jež nejsou standardně obsaženy ve vývojovém prostředí MS Visual Studio a částečně mi tak ulehčila práci, jak je patrno z kapitoly Nalezení informací o artefaktu.

Knihovnu jsem využil při výběru ideálního barevného kanálu, kdy pomocí jednoduché metody lze rozdělit zdrojový obraz na jednotlivé barevné kanály a z nich si vybrat, ten správný. Ale hlavním důvodem použití této metody byla možnost rozpoznávání objektů ve správně předpřipraveném obraze. Zde opět použitím jedné metody dojde k detekci všech objektů obsažených v obraze a na mnou vytvořeném systému už potom „jenom“ zbývá určit, který z objektů v obraze je hledaný artefakt a který je hledaná měrka.

Knihovna *OpenCVDotNet* slouží jako wraper na knihovny v jazyce *C#* z knihoven *OpenCV*, které jsou napsány v jazyce *C++*. Jedná se o volně dostupnou grafickou knihovnu. Knihovna obsahuje širokou škálu funkcí, které jsou ve většině případů určeny pro zpracování a úpravu obrazu. Podporuje rovněž různá externí snímací zařízení, jako například webkamera.

5.4 Instalace systému

Instalace systémů se skládá ze založení databáze pomocí přiloženého SQL skriptu v souboru *ArtefactDB.sql*, který provede k založení nové databáze a vloží do ní nezbytná data pro první spuštění systému. Pro spuštění SQL skriptu je ideální použít nástroj MS SQL Management Studio, který je součástí instalace MS SQL Serveru.

Před samotným spuštěním rozhraní systému je třeba upravit soubor *ConnectionString.txt*, který definuje umístění databázového MSSQL Serveru. Soubor *ConnectionString.txt* se rovněž nachází na přiloženém CD.

Po provedení těchto kroků je možno spustit samotné uživatelské rozhraní systému souborem *ArtefactSYS.exe* z přiloženého CD.

Následně se uživatel řídí základními kroky uvedenými v podkapitole Ovládání uživatelského rozhraní.

ZÁVĚR

Při realizaci systému, který byl cílem této diplomové práce, jsem se přesvědčil, že věnování dostatečného času dobrému návrhu systému je více než vhodné, protože delší čas strávený analýzou se později projeví v rychlejším vývoji systému. Součástí návrhu systému je komunikace se zadavateli, kteří musí odsouhlasit poslední verzi návrhu systému dříve, než se započne s prací na vývoji systému. Pro návrh aplikace jsem zvolil nástroje jazyka UML, kterými jsou diagramy užití, diagram tříd a Entity-relationship diagram. Z mé profese mám s těmito diagramy zkušenosti, proto mi odpadlo studování této jinak důležité látky.

Je nutno konstatovat, že kvalitně zpracovaný návrh ušetří pozdější, mnohdy i četnější změny a úpravy výsledného systému. I v případě tohoto systému došlo na pár drobných změn. Jejich zapracování do aplikace však nebylo už náročné. Při vývoji systému jsem si rozšířil svou znalost jazyka C# a technologie .NET o nové poznatky, týkající se komponent, metod a způsobu programování. Při vývoji jsem se snažil čerpat znalosti a postupy o tomto programovacím jazyku z dostupné odborné literatury. [8] Dospěl jsem k závěru, že bylo efektivnější studovat postupy a návody sdílené vývojáři na internetu.

Při zjednodušení celého procesu rozpoznání artefaktu lze říct, že ze vstupní fotografie po jejím zpracování je na výstupu procesu rozpoznání přeměněna na záznam v databázi, který obsahuje veškerá možná zjistitelná data o tomto artefaktu. Při vývoji fáze rozpoznání jsem navrhnul několik možných postupů, ze kterých jsem postupně vybral ty, které vracely nejlepší výsledky. Na konci procesu rozpoznání není výstupem vektor komplexních čísel, jak bylo navrženo v zadání, ale zvolil jsem pole souřadnic, které mělo pro systém lepší využití než zmíněný vektor. Výsledky měření mají vyhovující přesnost, což bylo prezentováno v podkapitole 3.9. Odchylky měření mezi čelním a bočním pohledem nepřesahují 1 milimetr. V rozpoznávacím procesu jsem použil knihovnu *OpenCVDotNet*, která obsahovala velmi užitečné metody ke zpracování obrazu, což mi částečně usnadnilo práci.

Výsledkem této diplomové práce je systém na rozpoznání a uchování artefaktů, ve kterém jsem dodržel rozsah zadání a v některých případech jsem řešení problematiky ještě rozšířil.

Z budoucích možných rozšíření uvedu požadavek, který byl se zadavatelem diskutován a byl prozatím odsunut na další možná rozšíření systému. Jedná se o možnost systému

zpracovávat 3D objekty, reprezentující historické artefakty. V průběhu vývoje aplikace jsem měl tento požadavek na paměti a snažil jsem se aplikaci na možnost tohoto rozšíření připravit tak, aby nemuselo později dojít k rozsáhlejším úpravám systému.

Úplným závěrem této práce chci říci, že věřím v to, že vzniklý systém bude uživateli plně využíván a bude svým uživatelským prostředím zefektivňovat jejich práci.

ZÁVĚR V ANGLIČTINĚ

Implementation of the system, which was the subject of this dissertation, solidified my opinion, that paying adequate attention to analysis and system projection pays itself by considerable time savings during the very development of the system. Communication with the client is a part of the system projection, because only the client can approve the final version the system draft, before the development itself starts. For the application design, I chose instruments of the UML language such as use case diagram, class diagram and Entity-ralationship diagram. Thanks to my personal experience with these diagrams, gained during my professional development, no addition study of the relevant area was needed.

It is necessary to be mentioned that precisely elaborated project saves significant changes and adjustments in the final system version. It appeared inevitable to apply minor adjustments to the final version of the system, but their execution was not followed by any serious complication. During the system development, I have the opportunity to improve my knowledge of C# language as well as .NET technology especially in the area of components, methods and techniques of programming. I aimed to gain relevant knowledge from literature available [8], but I lately realized, that it is way more efficient to study the latest instructions and suggestions of developers online.

In a simplified way of the system explanation, the input photography after being processed is converted into the database record on the output which contains all the accessible information about the artifact itself. During the recognition part of the system, I developed various procedures, but only those best-working were finally applied. At the end of the recognition process there is no vector of complex numbers, as it was required in the assignment, but array of coordinates was chosen instead, as it appeared better utilized for the purpose. Measuring results were more than satisfactory, as it was presented in the subchapter 3.9, where differences occurred between frontal and side view, do not exceed one millimeter. During the recognition process I used the *OpenCVDotNet* library, which is equipped by very useful methods for picture processing, which made my work easier.

As a result of this dissertation there is a system for recognition and archiving of artifacts, which not only followed up the set instructions, but went also even deeper in the solving of partial problems.

From the future possible extensions, I would like to mention the client's requirement of 3D objects recognition, representing historical artifacts. This requirement was discussed with the client and was evaluated provisionally as a future possible extension. Though I had the requirement on my mind during the system development and I aimed to make the system ready for future extension of this kind, so it would easier implement it with any serious reconstruction of the system.

At the very end of this dissertation I would like to claim, that I truly believe the system will be fully utilized by users and by the sophisticated user interface will make their work more efficient.

SEZNAM POUŽITÉ LITERATURY

[1] Šonka, M., Hlaváč, V.: Počítačové vidění, Grada, Praha, 1992, ISBN 80-85424-67-3

[2] Hlaváč, V., Sedláček, M.: Zpracování signálů a obrazů, Vydavatelství ČVUT, Praha, 2005, ISBN 80-01-03110-1

[3] Kotek, Z.: Metody rozpoznávání a jejich aplikace, Academia, Praha, 1993, ISBN 80-200-0297-9

[4] Pratt, W. K.: Digital Image Processing, Second Edition. New York: Wiley-Interscience, 1991.

[5] Šonka, M., Hlaváč, V., Boyle, R.: Image Processing, Analysis and Machine Vision. Boston: PWS, 1998.

[6] Sojka, E.: Digitální zpracování a analýza obrazů, VŠB-TU Ostrava, 2000.

[7] Španěl, M., Beran, V.:Obrazovné segmentační techniky, WWW: <http://www.fit.vutbr.cz/~spanel/segmentace>

[8] SHARP, J. Microsoft Visual C# 2008: krok za krokem. Vyd. 1. Brno: Computer Press, 2008, ISBN 978-80-251-2027-9.

[9] Kolektiv autorů: Wikipedie, otevřená encyklopedie, WWW: <http://cs.wikipedia.org>

[10] JONES, M. P. Základy objektově orientovaného návrhu v UML. Vyd. 1. Praha: Grada, 2001, ISBN 80-247-0210-X.

[11] Hlaváč, V., Digitální zpracování obrazu, ČVUT, WWW: http://cmp.felk.cvut.cz/~hlavac/HlavacTeachPresentCz.htm

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

RGB Barevný model skládající se z 3 barevných kanálů.

BW Černo bílý obraz.

SQL Jednoduchý dotazovací jazyk.

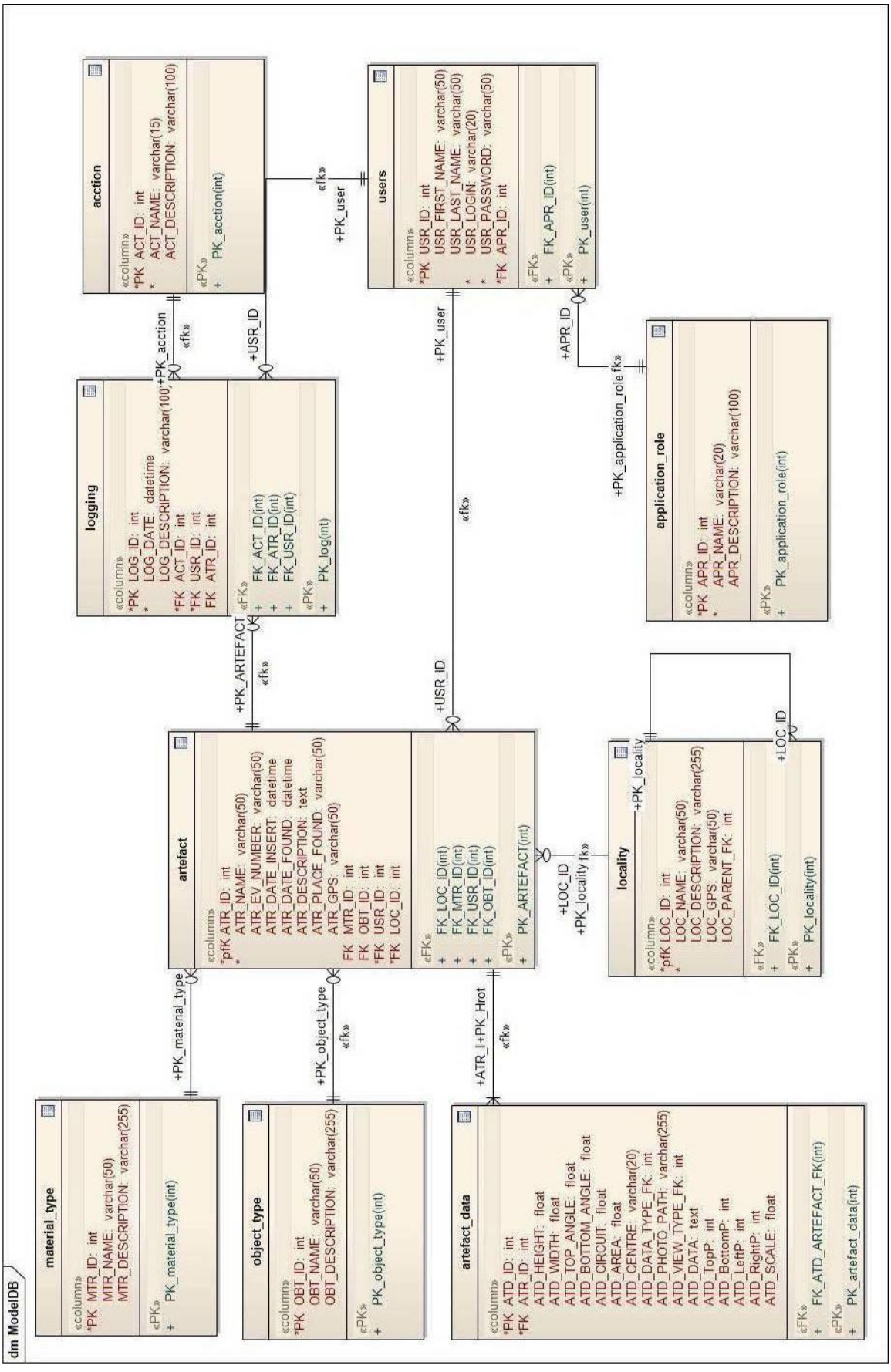
UML Unifikovaný značkovací jazyk.

SEZNAM OBRÁZKŮ

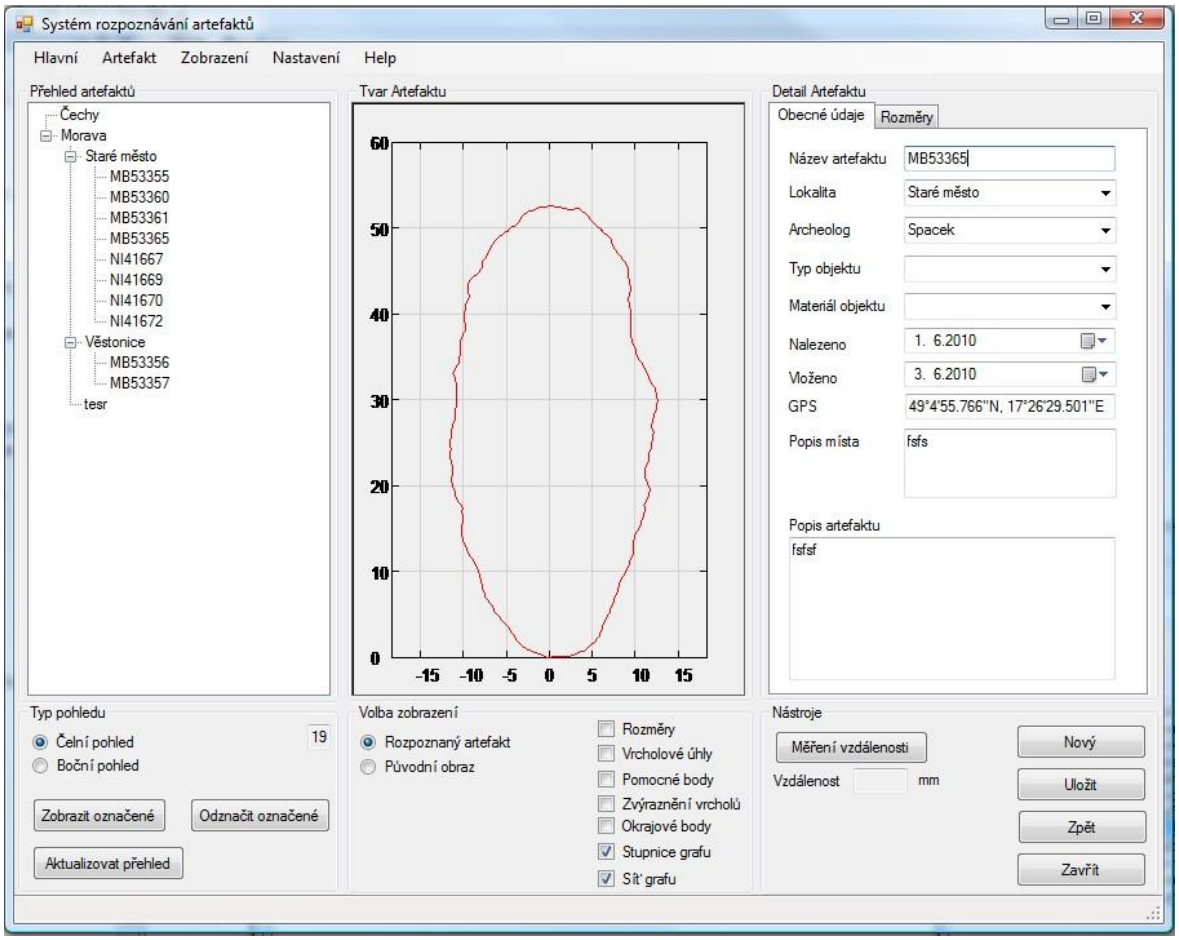
SEZNAM TABULEK

SEZNAM PŘÍLOH

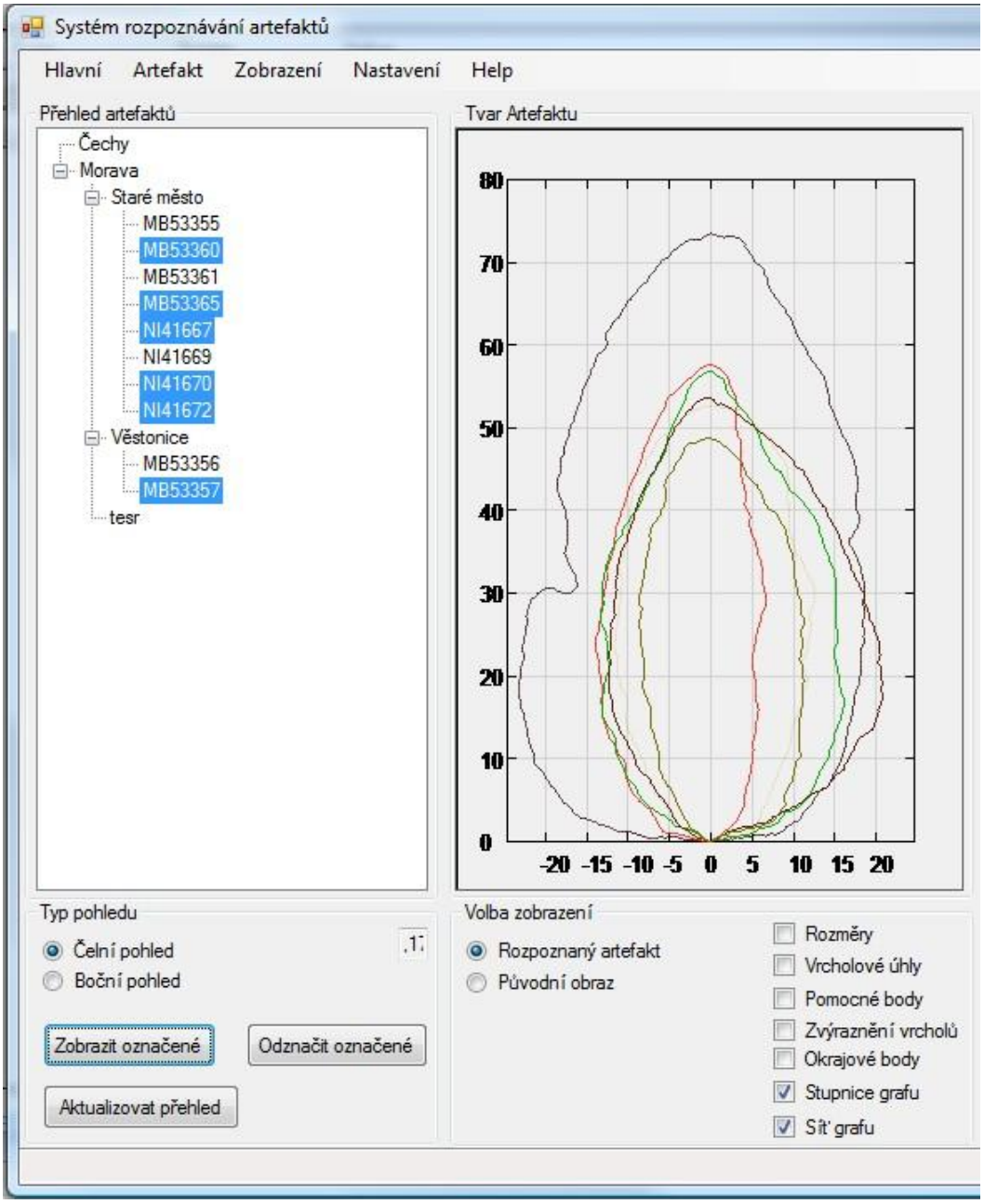
PŘÍLOHA P I: E-R DIAGRAM



PŘÍLOHA P II: NÁHLEDY UŽIVATELSKÉHO ROZHRANÍ



Obrázek PII-1: Hlavní rozhraní aplikace.



Obrázek PII-2:Více násobný výběr artefaktů.