ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

KATEDRA ELEKTROMECHANIKY A VÝKONOVÉ ELEKTRONIKY

BAKALÁŘSKÁ PRÁCE

Automatizovaná úprava dokumentů

Tomáš Kurc **2015**

Abstrakt

Předkládaná bakalářská práce se zabývá analýzou, návrhem a implementací *Automatizované úpravy dokumentů*. Jejím cílem je pomocí sady skriptů automatizovat převod PDF dokumentů do snímků odpovídajících svým rozlišením zobrazovacímu zařízení, umožnit vkládání snímků s náhledem webových stránek, snímky videa, webkamery či textové snímky s výzvami nebo důležitými informacemi. Součástí práce je analýza nástrojů použitelných nejen v jiných podobných aplikacích.

Klíčová slova

Automatizovaná úprava dokumentů, snímky PDF, náhledy webových stránek, zachycení snímku webkamery, náhledy video souboru, PHP skripty, ImageMagick.

Abstract

Submitted bachelor thesis deals with analysis, proposal and implementation of an A*utomated editing of documents*. Its task is to make and to use a set of scripts for automatic conversion of PDF documents into images corresponding its resolution to display device, to allow insertion a preview of web pages, video thumbnails, pictures from webcameras, text notice and other important informations. A part of analysis contains a description of tools for use not only in other similar applications.

Key words

Automated editing of documents, images from PDF, previews of a webpages, images from webcameras, video thumbnails, PHP scripts, ImageMagick.

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této bakalářské práce.

Dále prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

_____

Podpis

V Plzni dne 7.6.2015 Tomáš Kurc

Obsah

Seznam symbolů a zkratek

zkratka	anglický význam	český význam
API	Application Programming Interface	rozhraní pro programování aplikací
CSV	Comma-separated values	jednoduchý souborový formát určený pro výměnu tabulkových dat
DOM	Document Object Model	objektový model dokumentu
fps	frames per second	počet snímků za sekundu
FTP	File Transfer Protocol	protokol pro přenos souborů
HTML	HyperText Markup Language	značkovací jazyk určený k tvorbě webových stránek
HTTP	HyperText Transfer Protocol	internetový protokol určený pro výměnu HTML dokumentů
JSON	JavaScript Object Notation	JavaScriptový objektový zápis, platformě nezávislý datový formát
LCD	Liquid Crystal Display	monitor z molekul tekutých krystalů
OCR	Optical Character Recognition	optické rozpoznávání znaků, metoda k digitalizaci tištěných textů
OS	Operating System	operační systém
PDF	Portable Document Format	přenosný formát dokumentů
PHP	Hypertext Preprocessor	hypertextový preprocesor (rekurzivní zkratka)
PNG	Portable Network Graphics	grafický formát bezeztrátové komprese rastrové grafiky
PS	PostScript	programovací jazyk určený ke grafickému popisu tisknutelných dokumentů
px	pixel	obrazový bod
REST	Representational State Transfer	architektura rozhraní navržená pro distribuované prostředí
URL	Uniform Resource Locator	jednotná adresa zdroje

Seznam obrázků

Úvod

Dosavadní průběh zpracování dokumentů a jejich prezentace

V současné době se pro účely zobrazování dokumentů na informačních panelech Elektrotechnické fakulty ZČU v Plzni zpracovávají prezentace vytvořené např. v aplikacích MS Office PowerPoint či OpenOffice Impress. Z těchto prezentací je následně nutné vytvořit soubor obrázků definovaných jmen, formátů a rozlišení. Na základě pojmenování jsou snímky později v abecedním pořadí zobrazovány, aktuálně používaný formát obrázků je PNG. Rozlišení musí být stejné jako u LCD obrazovek, které je 1920x1080 pixelů. Připravené snímky jsou umístěny do složek na FTP serveru, odkud jsou synchronizovány do mikropočítače, který zajišťuje zobrazování snímků na LCD panelech.

Úskalí stávajícího systému

Jelikož PowerPoint, ani Impress dle získaných informací neumožňují vytvoření snímků v potřebném rozlišení, lépe řečeno to zvládají jen při zásahu do registrů počítače, což je z pohledu běžného uživatele nepřijatelné, je nejprve nutné prezentaci převést do souboru PDF. Z PDF jsou pomocí programu ImageMagick získávány obrázky ve formátu PNG a potřebném rozlišení 1920x1080 pixelů. Komplikace spojené s přípravou jednotlivých obrázků a jejich konverze zapříčiňují ztrátu času.

Cíl práce

Na základě komplikované přípravy snímků a robustnosti systému byl vznesen požadavek na vytvoření sady skriptů, které pro prezentaci na LCD panelech fakulty zajistí přípravu konečných snímků z PDF souborů, a zároveň systém rozšíří o možnost zpracovávat snímky z dalších zdrojů (webové stránky, videa …). Mimo výše zmíněné je cílem práce analyzovat možné nástroje, vyhodnotit jejich výhody a nevýhody, navrhnout vhodné postupy a následně je implementovat.

Požadavky

Funkční požadavky

1. Aplikace na straně serveru bude umožňovat export PDF souboru do jednotlivých obrázků.
2. Aplikace na straně serveru bude umožňovat uložení vybrané webové stránky do podoby obrázku.
3. Aplikace na straně serveru bude umožňovat export vybraného snímku videa do obrázku.
4. Aplikace na straně serveru bude umožňovat export snímku webkamery do obrázku.
5. Aplikace na straně serveru bude umožňovat tvorbu speciálních snímků s textovými upozorněními.
6. Snímky získané aplikací budou vyhovovat požadavkům zobrazovacího zařízení.
7. Aplikace na straně serveru bude umožňovat indexování snímků.
8. Aplikace na straně serveru bude umožňovat řazení snímků do fronty dle indexu v názvu snímku.
9. Aplikace na straně serveru bude umožňovat zadání priorit snímků.
10. Aplikace na straně serveru bude umožňovat pravidelné zařazování snímků s textovou výzvou.

Nefunkční požadavky

1. Aplikace na straně serveru bude upřednostňovat jazyk PHP.
2. Bude-li to možné, použijte při kódování aplikace balík ImageMagick.

Způsob řešení

V programovacím jazyku PHP bude implementována sada samostatných skriptů, z nichž každý bude svou funkčností odpovídat jednomu z prvních pěti požadavků. V maximální míře bude (dle nefunkčního požadavku č. 2) využito balíku aplikace ImageMagick, ale i dalších nástrojů (např. online REST API Page2Image či nástroje FFmpeg) vhodnějších či účinnějších pro danou aplikaci. Možné nástroje vhodné pro použití konkrétního požadavku jsou popsány v kapitole Analýza, konkrétní použitý nástroj je zmíněn v kapitole Návrh či Implementace, a to včetně odůvodnění svého použití a případného vyjmenování další funkčnosti.

Software ImageMagick

ImageMagick je platformě nezávislá volně dostupná softwarová sada umožňující zpracovávání bitmapových obrázků v nejrůznějších formátech. Mezi podporované formáty, kterých je v současnosti více jak 100, patří například i PDF, JPEG, PNG, PS, MPEG, TIFF či třeba HTML (úplný výčet všech podporovaných formátů je možné dohledat v příslušné dokumentaci na webu). Kromě samotného čtení a zápisu obrázků v široké škále podob umožňuje ImageMagick také bohaté spektrum nástrojů sloužících k editaci. Obrázky lze pomocí tohoto softwaru nejrůzněji transformovat a deformovat, měnit jejich velikost či barvy, kreslit do nich základní obrazce (linie, polygony, elipsy) i obrazce z beziérových křivek, zapisovat do nich text a mnoho dalšího. Volání ImageMagicku se provádí prostřednictvím příkazového řádku či některého z mnoha defaultních rozhraní umožňujících jeho použití v pestré paletě programovacích jazyků (C, .NET, Perl, PHP, Ruby, …). Pro jazyk PHP slouží Imagick, který je oficiálním wrapperem ImageMagicku.

1. Analýza

1.1. Rozbor požadavků

1.1.1. Export PDF souboru do obrázků

V současnosti jsou snímky prezentací určených k projekci na fakultních LCD panelech převedené nejprve do PDF souboru, z něhož jsou následně pomocí balíku ImageMagick získávány jednotlivé obrázky v požadované velikosti. Pomocí skriptu v jazyce PHP bude celý proces automatizován. K převodu PDF dokumentu lze skrze PHP využít např. již zmíněný ImageMagick, případně REST API Aspose.Pdf.

Při psaní skriptu bude nutné kalkulovat s rozličnými vstupy, kterými mohu být soubory odpovídající proporcím LCD panelů (soubory orientované na šířku v poměru 16:9), ale i soubory se zcela odlišnými, např. naprosto opačnými poměry stran (orientované na výšku). Rovněž bude nutné stanovit vhodný způsob převzorkování snímků, aby nedocházelo k příliš velkým ztrátám informace.

1.1.1.1. Nástroje k převodu PDF dokumentu

1.1.1.1.1. Software ImageMagick

Umožňuje také konverzi PDF dokumentů, a to pomocí jednoduchého příkazu skrze příkazovou řádku či přes rozšíření mnoha programovacích jazyků. Výhodou je možnost snímky v průběhu konverze libovolně transformovat, zvětšovat či zmenšovat, nebo jinak dále upravovat.

1.1.1.1.2. Rozhraní Aspose.Pdf

Platformě nezávislé REST API využitelné množstvím programovacích jazyků, např. jazykem PHP či Java. Umožňuje vytvářet, editovat a konvertovat dokumenty PDF do mnoha formátů (např. HTML, XML, XPS, …), včetně obrázků (PNG, JPEG, …).

Aspose.Pdf je součástí velké rodiny cloudových rozhraní Aspose, jež také nabízí i API pro práci s dokumenty MS Office, Open Dokumenty, čárovými kódy, OCR, e-maily či obrázky. Nevýhodou rozhraní Aspose je jejich individuální zpoplatnění přinejmenším v řádu desítek dolarů měsíčně.

1.1.1.2. Nástroje k převodu prezentací

1.1.1.2.1. Popisovač okna HWND

Popisovač okna, kterým lze zachytit snímek prezentace, např. v PHP pomocí funkce *imagegrabwindow*. Nevýhodou je existence pouze pod OS Windows a také potřeba skutečného okna, v němž je prezentace otevřena, nejedná se tedy pouze o virtuální podobu okna.

1.1.1.2.2. Rozhraní Aspose.Slides

Platformě nezávislé rozhraní, které podporuje mnoho programovacích jazyků, včetně PHP či Javy. Poskytuje nástroje pro vytváření, editaci a konverzi prezentací různých formátů vzájemně mezi sebou, případně do jiných podob (např. HTML, PDF, XPS …), včetně obrázků (PNG, JPEG …).

Aspose.Slides je (stejně jako Aspose.Pdf) součástí velké skupiny Aspose poskytující mnoho dalších cloudových REST API. Nevýhodou těchto API je jejich zpoplatnění.

1.1.1.2.3. Knihovna JODConverter

Knihovna programovacího jazyka Java slouží k převodu nejrůznějších formátů kancelářských dokumentů a také prezentací z aplikací MS Office PowerPoint či OpenOffice Impress, do mnoha výstupních formátů, např. PDF, JPG, PNG. Lze využít v Java aplikacích, případně jako webová služba volaná z aplikace psané v jiném jazyce, třeba PHP.

1.1.1.3. Příprava snímků

Optimálnímu exportu snímků z PDF souboru může uživatel pomoci při přípravě prezentace. V základu je vhodné prezentaci nastavit pro zobrazování na zařízení s poměrem 16:9 (poměr LCD obrazovek), čemuž odpovídá rozměr stránky 25,4x14,29 cm při rozlišení 960x540 px. Tím bude zaručeno převzorkování snímku na nové rozlišení, a to s roztažením snímku na celou zobrazovanou plochu při zachování poměrů stran. Nastavení stránky na rozlišení LCD obrazovek je rovněž možné, a to změnou velikosti stránky na rozměr 50,8x28,57 cm, což odpovídá rozlišení panelů, tedy 192x1080 px. Veškerá výše popsaná nastavení stránky lze v aplikaci MS Office PowerPoint provádět v dialogovém okně „Vzhled stránky“, k němuž se lze dostat skrze záložku „Návrh“.

Na rozdíl od aplikace PowerPoint umožňuje Impress nastavení rozlišení snímku při jeho exportu. I zde je nejprve nutné změnit poměr stran, jelikož PowerPoint i Impress mají iniciálně tento poměr nastaven na 4:3. V aplikaci Impress lze rozměry snímku, které lze zadávat pouze v centimetrech, nastavit v dialogovém okně „Stránka…“, jež lze otevřít z menu „Formát“.

1.1.2. Snímání webové stránky

Formát webové stránky může dosahovat nejrůznějších rozměrů, proto lze vytvoření snímku provést několika způsoby.

Prvním způsobem je vytvoření obrázku z webové stránky, kterou je nejprve nutné vykreslit. K tomu lze použít několik nástrojů, např. page2images, html2ps (html2pdf), html2cavas, wkhtmltopdf (wkhtmltoimage) či třeba popisovač okna HWND. Z vykreslené webové stránky vytvoříme snímek, jehož vlastnosti (velikost snímané oblasti, rozlišení, …) záleží na použitém nástroji. Následně můžeme snímek dále upravovat, např. ořezávat, zvětšovat/zmenšovat atd.

Další možností tvorby snímku z webové stránky je (za využití projektu Beautiful Soup) výběr určité oblasti uzavřené mezi definované HTML tagy, z nichž bychom importovali pouze jejich obsah.

1.1.2.1. Nástroje k vykreslení webové stránky

1.1.2.1.1. Projekt Page2Image

Projekt poskytující několik online nástrojů (*Mobile Emulator, Batch URL to Image Converter, Free Mobile OK Checker, Batch Mobile Checker, HTML to Image a Screenshot Generator*) a také několik rozhraní (*REST API, <IMG> API, Javascript API*) umožňujících snímání webové stránky.

A. *Online nástroje*

a. *Mobile Emulator*

Poskytuje webové rozhraní k náhledu webové stránky na mobilních zařízeních iOS, Android, Windows Mobile i BlackBerry, s volbou velikosti displeje od 3,5 do 9,7 palce.

b. *Batch URL to Image Converter*

Batch URL to Image Converte ke nástroj k dávkovému zpracování více URL adres, z nichž vytváří náhledy stránek, a to jak jejich reálně viditelných částí, i celé stránky. Vzhled webu lze simulovat pro zařízení typu iPhone, iPad, Android, Windows Phone i Desktop. Sadu obrázků lze po simulaci stáhnout v zip archivu.

c. *Free Mobile OK Checker*

Služba vyhodnocující kompatibilitu webové stránky s mobilním zařízením, která poskytuje např. zhodnocení dodržení webových standardů, grafiky a barvy či šetrnosti k objemu přenášených dat. Rovněž vykresluje podoby webu na mobilním zařízení.

d. *Batch Mobile Checker*

Nástroj k dávkovému zpracování URL adres, který vyhodnocuje kompatibilitu webu s mobilními zařízeními. Výsledky vyhodnocení vrací v podobě CSV dat, jež obsahují

informaci, zda zadaný web je či není kompatibilní s mobilním zařízením, a také odkaz na příslušný náhled webové stránky.

e. HTML to Image

Služba HTML to Image převádí zadaný HTML kód na obrázek. Simulace kódu je dostupná pro zařízení iPhone, iPad, Android, Windows Phone, BlackBerry a Desktop. Poskytuje možnost vyhodnocení chyb v HTML kódu.

f. Screenshot Generator

Nástroj, který vytváří rychlý náhled podoby webu ze zadané URL adresy. Nabízí volbu zařízení typu Desktop, Tablet či Mobile, včetně volby několika rozlišení.

B. Rozhraní

a. REST API

Poskytuje REST rozhraní, které je možné volat skrze URL adresu prohlížeče, zdrojový kód vlastní aplikace či příkaz *wget* ze systému UNIX. Podporuje obě (GET i POST) dotazovací metody HTTP protokolu, nabízí mnoho nastavení parametrů výsledného obrázku webu (celý web/viditelná oblast, rozlišení zařízení, pro nějž je vzhled stránky simulován, rozlišení získaného snímku, možnost zpoždění vytvoření náhledu, …).

b. <IMG> API

Jednoduché rozhraní, které se volá z HTML <*img*> tagu jako zdroj obrázku. Volání API probíhá skrze URL adresu se zadanými parametry (rozlišení, typ simulovaného zařízení) v podobě dotazovací metody GET, API vrací odkaz na náhled webu.

c. Javascript API

Javascript API je rozhraní volané pomocí Javascript kódu, založené na <IMG> API. Při zpracovávání náhledu webu zobrazuje animaci načítání obrázku, následně plynule přejde ve výsledný snímek stránky.

Nevýhodou výše popsaného projektu je omezený počet volání URL adres. V případě přístupu bez registrace je počet volání omezen na 10, u bezplatného účtu je tento počet již vyšší (100 unikátních URL za den). Při vyšších nárocích je nezbytné zpoplatnění služby v řádu několika desítek dolarů měsíčně.

1.1.2.1.2. Skript Html2ps (html2pdf)

Volně dostupný PHP skript, který umožňuje vykreslení webové stránky do souboru PDF či PS. Podporuje vnořené tabulky, rekurzivní zpracování odkazů, zvládá pokročilé CSS vlastnosti, plovoucí DHTML prvky, XHTML tagy a vlastnosti i cizojazyčné znakové sady.

Výhodou je řízení podoby výsledného PDF či PS dokumentu pomocí konfiguračních souborů. Nevýhodou je nezbytnost jiného nástroje, pomocí něhož lze výstupní soubor převést do obrazového formátu.

1.1.2.1.3. Script Html2canvas

Html2canvas je Java skript určený k tvorbě snímků webové stránky i jejích částí založený na DOM. Nevytváří skutečný náhled stránky, ale sestavuje stránku pomocí DOM objektů, a tu následně vykresluje. Je volně dostupný.

Nevýhodu je nekompatibilita se všemi vlastnostmi CSS, výsledná stránka může vypadat zcela odlišně. Neumí pracovat s Flash objekty či Java applety.

1.1.2.1.4. Wkhtmltopdf (wkhtmltoimage)

Volně dostupný nástroj v podobě příkazové řádky (instalace je možná i na server), který umožňuje vykreslení HTML do PDF či jiných obrazových formátů pomocí vykreslovacího modulu QT Webkit.

1.1.2.1.5. Popisovač okna HWND

Popisovač okna, kterým lze zachytit podobu webové stránky. Nevýhodou popisovače HWND je jeho existence pouze pod operačními systémy Windows a také potřeba reálného okna prohlížeče s otevřenou webovou stránkou.

1.1.2.2. Projekt Beautiful Soup

Beautiful Soup je Python knihovna určená k parsování HTML a XML souborů, k čemuž využívá populární pythonské *lxml* a *html5lib* parsery. Umožňuje procházet stromovou strukturu dokumentu, vyhledávat v ní, modifikovat ji a extrahovat z ní potřebné informace. Beautiful Soup zároveň dokáže rozpoznat většinu kódování a následně převést vstupní soubory do Unicode a výstupní soubory do UTF-8, netřeba se ve většině případů starat o kódování dokumentu.

Pomocí Beautiful Soup lze v HTML a XML dokumentech např. vyhledávat všechny odkazy, či odkazy s danou URL adresou, hledat tabulky s tučně napsanou hlavičkou a mnoho dalších informací, s nimiž můžeme dále pracovat.

1.1.3. Export snímku videa

Při tvorbě skriptu zajišťujícího export snímku (či snímků) z videa bude nutné vzít v potaz především formát vstupního souboru, respektive jeho rozlišení. Není vhodné snímek příliš zvětšovat z důvodu zachování jeho čitelnosti. Nabízí se několik možností využití, pokud výsledný obrázek nedosáhne rozlišení 1920x1080 px, či se mu alespoň nepřiblíží. Snímek je možné v původním rozlišení umístit do horního levého rohu a zbytek plochy využít třeba pro umístěn popisu zdrojového video souboru. Rovněž lze uživateli nabídnout možnost utvořit z videa více snímků s jejich

poskládáním do dlaždic (např. 2x2, 3x2, 3x3 …), čímž by došlo k využití více místa v promítané ploše.

Náročnost zpracování videosouboru roste s jeho délkou, tj. s celkovým počtem snímků, kterých je standardně 25 či 30 za sekundu (minimum pro vytvoření iluze plynulého pohybu je 10 fps) bude nutné vytvořit skript tak, aby zpracování videa bylo co nejjednodušší.

K exportu snímků z videa lze využít několik nástrojů. Jedním z nich je již dříve zmíněný ImageMagick, dalším pak například FFmpeg či Zencoder.

1.1.3.1. Nástroj FFmpeg

FFmpeg je kolekce knihoven a nástrojů umožňující práci s multimediálními daty, jako např. kódování a dekódování, streamování, konverzi či přehrávání. Pravděpodobně nejdůležitější knihovnou je *libavcodec* obsahující audio/video kodéry a dekodéry (kodeky). K dalším knihovnám patří např. *libavformat* poskytující multiplexy a demultiplexy pro kontejnerové audio/video formáty, či *libavfilter* obsahující filtry. Ke své činnosti využívá FFmpeg i ImageMagick.

1.1.3.1.1. Knihovna ffmpeg-php

Oficiální knihovna nástroje FFmpeg, která je určená pro jazyk PHP. Pomocí ní lze v PHP využívat stejné funkčnosti FFmpegu jako přes příkazovou řádku.

Nevýhodou je dlouhodobá neaktualizace knihovny v závislosti na nových verzích FFmpeg.

1.1.3.1.2. Nástroj PHPVideoToolkit

Sada tříd PHP, která využívá emulaci knihovny ffmpeg-php, rozšiřuje ji o další nástroje a umožňuje její kompatibilitu s novými verzemi FFmpeg. Poskytuje rozhraní pro interakci PHP s video a audio soubory.

Nevýhodou je její testování na starší verzi FFmpeg. Výhodou je fakt, že oproti ffmpeg-php se stále vyvíjí.

1.1.3.2. Rozhraní Zencoder

Zencoder je online REST API služba určená vývojářům, umožňující konverzi, kódování a další práci s mnoha formáty video souborů. Díky této službě lze video soubory převádět tak, aby byly kompatibilní např. s webovým přehráváním, mobilními telefony či jinými zařízeními. Do PHP lze integrovat pomocí knihovny *zencoder-php*.

Nevýhodou je zpoplatnění služby v řádu několika desítek až set dolarů měsíčně, bezplatně je poskytováno pouze testovací rozhraní omezené na 5 sekund délky videa.

1.1.4. Export snímku webkamery

Tvorba snímku z webkamery je v podstatě jen konkrétním případem dříve popsaného zachycení snímku webové stránky, s omezeními popsanými v předešlé kapitole 1.1.3 Export snímku videa, vztahujícími se k rozlišení webkamery. Z tohoto důvodu je snímání obrázků z webové kamery popsáno pouze stručně.

Z webové stránky získáme v daný okamžik pouze jeden snímek, lze zbylé místo ve výsledném obrázku využít např. pro umístění popisku webkamery či data a času zachycení snímku.

1.1.5. Tvorba speciálních snímků

Při psaní skriptu umožňujícího vytváření speciálních snímků využijeme opět ImageMagick, který poskytuje třídu *ImagickDraw* se sadou funkcí pro kreslení jednoduchých i složitějších objektů či psaní textů, a pro potřeby skriptu bude dostačovat. Složitost skriptu bude záležet ve větší míře pouze na možnostech, které uživateli nabídne, např. volba barvy pozadí, nastavení vlastností textu (font písma, velikost, barva, …) apod. Důležitým faktorem pro vytváření snímků bude volba velikosti písma, aby byla zachována čitelnost textu, která zároveň přinese omezení na množství textu umístěného do jednoho snímku.

1.1.6. Snímky získané aplikací

Požadavek č. 6 říká, že snímky získané aplikací mají vyhovovat požadavkům zobrazovacího zařízení. Vztahuje se ke všem požadavkům předcházejícím. Pro psaní skriptů to znamená, že obrázky v nich vytvořené musí dosahovat rozlišení LCD panelů (1920x1080 px). Konkrétní možnosti řešení jsou popsány v rámci analýzy jednotlivých požadavků. V případě, že nebude získaný obrázek dosahovat i po případném zvětšení požadovaného rozlišení, je řešením obvykle vložení obrázku do výsledného snímku, přičemž k obrázku jsou přidány např. informace o zdroji, z něhož byl pořízen, či datum a čas pořízení.

1.1.7. Indexování snímků

Index snímku je nezbytné stanovit tak, aby byl jedinečný a v souladu s požadavkem č. 8 týkajícího se řazení snímků do fronty, z níž jsou snímky následně promítány.

Z hlediska unikátnosti indexu se jako vhodné řešení jeví využití nějakého čítače. Ideální možností je časové razítko. Díky němu je zajištěna jedinečnost při volbě dostatečně krátkého časového pulzu. V PHP existují dvě funkce časového razítka, obě vracejí čas, který uplynul od 1. 1. 1970 00.00:00. Funkce *time* vrací tento čas v sekundách, *microtime* v mikrosekundách. Z hlediska rychlosti zpracování skriptu rozlišitelnost jedné sekundy nemusí postačovat, miliónkrát větší rozlišitelnost vystačí potřebám aplikace.

V otázce pozice indexu v názvu souboru se vhodně jeví umístění indexu na začátek názvu (za předpokladu, že index nebude stanoven náhodným řetězcem). Důvody jsou objasněny v následující kapitole týkající se řazení snímků podle indexu.

1.1.8. Řazení snímků podle indexu

Řazení snímků nebude zasahovat do procesu mikropočítače, který v současnosti snímky řadí na základě abecední posloupnosti názvů, je tedy vhodné umístit index na začátek názvu souboru. Pak bude zaručeno řazení snímků v posloupnosti (za předpokladu posloupnosti indexů), v níž byly snímky jednak již na začátku (případ PDF a video souborů), navíc netřeba zohledňovat samotný název souboru, který by řazení mohl ovlivnit. Zároveň zůstane do jisté míry ponechána možnost řídit pořadí zobrazování snímků, a to díky možnosti vložit stejný snímek vícekrát.

1.1.9. Priorita snímků

Bez zásahu do řídícího mechanismu mikropočítače bude jen velmi obtížné ovlivnit prioritou snímku způsob jeho zobrazení. Jedinou možností zůstává opakované vložení téhož snímku. Zde jsou dva způsoby, oba založené na opakovaném vkládání snímku do fronty. Bude-li uživatel chtít ovlivnit prioritu četností zobrazování snímku, bude nutné snímek vložit vícekrát, vždy s proložením jinými snímky. V případě požadavku na prodloužení zobrazovací doby bude nutné vícenásobné vložení snímku ihned po sobě. Ten se v rámci řízení mikropočítačem Raspberry vyobrazí na zobrazovacím zařízení dvakrát či vícekrát za sebou.

1.1.10. Periodické zařazování snímků s textovou výzvou

Požadavek týkající se speciálních snímků je zobecněním požadavku č. 8 na řazení snímků do fronty (v závislosti na indexu) a souvisí i s požadavkem č. 9. Periodické zařazování snímků lze chápat jako zobrazení snímku v pravidelném cyklu, což je splněno z hlediska charakteru promítání zajištěného mikropočítačem Raspberry. Každý snímek se stejně jako všechny ostatní, zobrazí v promítacím cyklu všech obrázků právě jednou. Bude-li chtít uživatel periodu promítání snímku zkrátit, lze to učinit opakovaným vložením téhož snímku, viz předešlá kapitola.

1.2. Filtry

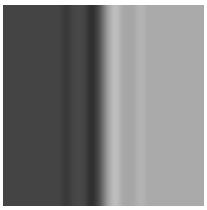
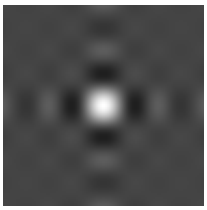
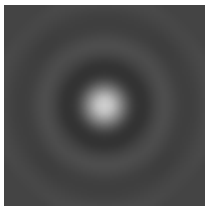
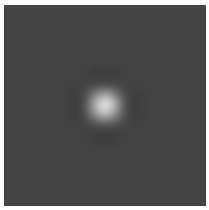
Během přípravy konečných snímků bude nevyhnutelná úprava jejich velikosti. Ať již bude nutné snímek zmenšovat či zvětšovat, bude vždy docházet k převzorkování, které přináší zkreslení obrazu v podobě artefaktů, jež je možné z části ovlivnit použitým filtrem.

1.2.1. Artefakty

Artefakty vznikající při použití filtrů, které ImageMagick nabízí, je možné rozdělit do následujících čtyř kategorií.

1.2.1.1. Prstencové artefakty (ringing)

Vznikají přílišnou změnou velikosti, chybným kompresním algoritmem, případně špatným nastavením použitého filtru. Projevují se v oblastech hran barevných přechodů.

zdrojový obrázek	zvětšení při špatně nastaveném filtru	zvětšení při dobře nastaveném filtru
		
	 	

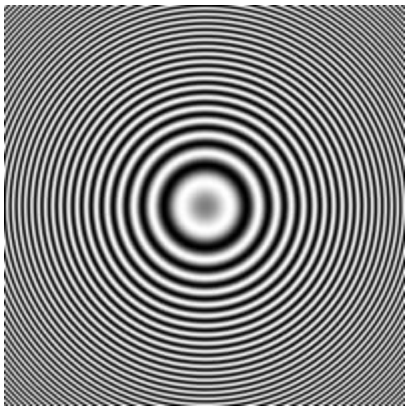
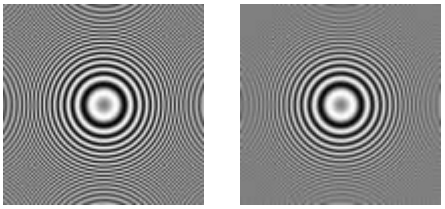
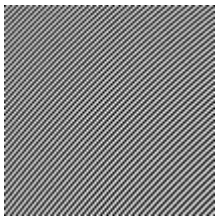
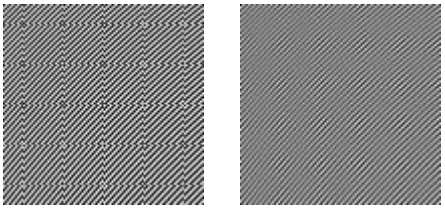
Tabulka 1-1 Prstencové artefakty

1.2.1.2. Čtvercové artefakty (blocking)

Primární příčinou těchto artefaktů, které vznikají přepočtem množství pixelů (např. při dvojnásobném zvětšení připadá na čtyři nové pixely informace jednoho původního a naopak), je špatně vyhlazený zdrojový obraz. Minimalizace čtvercových artefaktů je možná užitím obrázků vysokého rozlišení, případně vektorové grafiky.

1.2.1.3. Aliasing a moaré

Optické jevy, které souvisí s nedodržením vzorkovacího teorému, který říká, že vzorkovací frekvence musí být alespoň dvojnásobně větší, než nejvyšší frekvence obsažená ve vzorkovaném signálu. Aliasing se nejčastěji projevuje nerovnými efekty podél okrajů obrazu, zatímco moaré vnáší do obrazu shluky pixelů v podobě nepravidelných barevných pruhů.

zdrojový obrázek	**příklady vzniklého aliasingu a moaré**
	
	

Tabulka 1-2 Aliasing a moaré

1.2.1.4. Rozostření (blurring)

Projevuje se rozostřením hran a podobně jako blocking souvisí s přepočtem pixelů rastru. Hloubku rozostření lze ovlivnit použitým filtrem, efekt rozostření je využíván k zakrytí čtvercových a prstencových artefaktů a aliasingu/moaré. Dosahuje se tak pomocí něj jistého kompromisu ve vzhledu obrazu při změně jeho velikosti.

1.2.2. Činnost filtrů

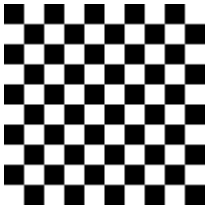
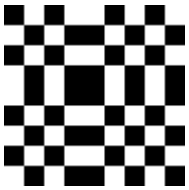
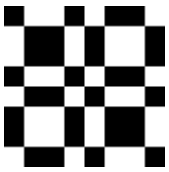
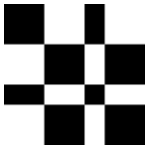
Při změně velikosti snímku dochází u každého nového pixelu k výpočtu jeho hodnoty na základě pixelů ve zdrojovém obraze. Protože si pozice pixelů neodpovídají, je nutné stanovit způsob určení této hodnoty, která je váženým průměrem okolních pixelů. Rozsah okolí a váha jednotlivých pixelů užitých k výpočtu je udávána použitým filtrem.

1.2.3. Interpolační filtry

Při změně velikosti snímku využívají k výpočtu logické hodnoty barvy pixelu jen minimální okolí pixelu zdrojového obrazu, což činí tyto filtry velmi jednoduché a rychlé. Neschopnost sloučit dohromady větší počet pixelů snižuje jejich kvalitu, neboť dochází k silnému aliasingu a moaré. Vzhledem k výše uvedenému se nehodí pro změnu velikosti obrazu, využívají se však např. při otáčení či jiných menších deformacích obrazu.

1.2.3.1. Filtr Point

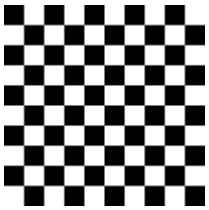
Využívá pouze barvu pixelu ve zdrojovém snímku, nikoliv i v jeho okolí. Při změně velikosti snímku dochází k nerealným výsledkům.

zdrojový obrázek	90%	80%	70%	60%	50%
					

Tabulka 1-3 Aplikace filtru Point při zmenšování obrazu (10x zvětšeno)

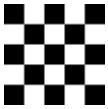
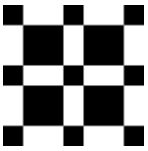
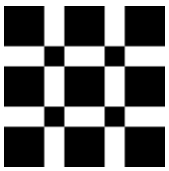
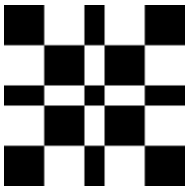
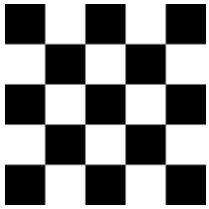
1.2.3.2. Filtr Box

K výpočtu hodnoty nového pixelu se počítá s velmi blízkým okolím zdroje (půl pixelu, viz obr. 1 Graf filtru Box). Při zvětšování ovlivňuje barvu výsledného pixelu okolí pixelu zdrojového, při zmenšování je více pixelů komprimováno do jednoho.

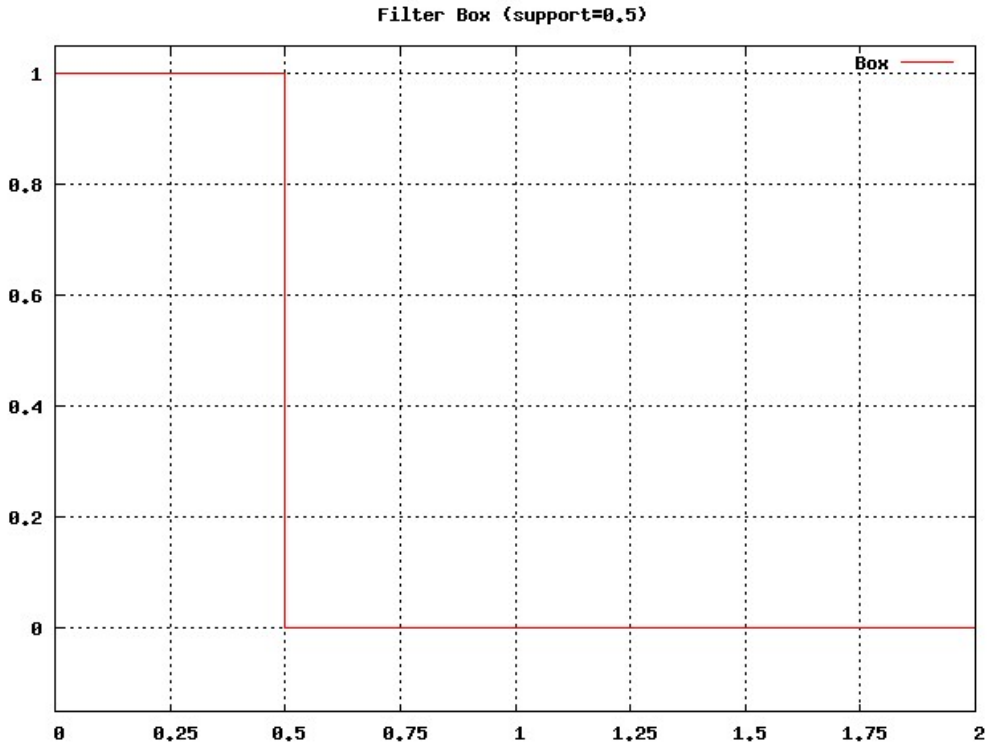
zdrojový obrázek	90%	80%	70%	60%	50%
					

Tabulka 1-4 Aplikace filtru Box při zmenšování obrazu (10x zvětšeno)

Příklad zvětšování obrazu uvedený v Tabulka 1-5 Aplikace filtru Box (Point) při zvětšování obrazu (10x zvětšeno) je v tomto případě shodný i pro filtr Point.

zdrojový obrázek	120%	140%	160%	180%	200%
					

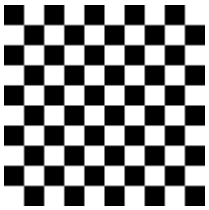
Tabulka 1-5 Aplikace filtru Box (Point) při zvětšování obrazu (10x zvětšeno)



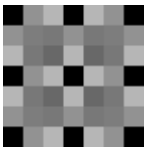
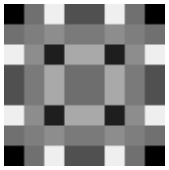
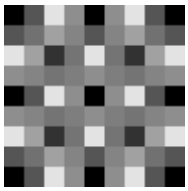
obr. 1 Graf filtru Box – váha hodnoty pixelu v závislosti na vzdálenosti

1.2.3.3. Filtr Triangle

Podobně jako filtr Box počítá s velmi blízkým okolím zdroje, ale dvojnásobným (jeden pixel). Na rozdíl od předešlého filtru je váha okolních pixelů určena i dle jejich vzdálenosti od zdroje (viz obr. 2 Graf filtru Triangle).

zdrojový obrázek	90%	80%	70%	60%	50%
					

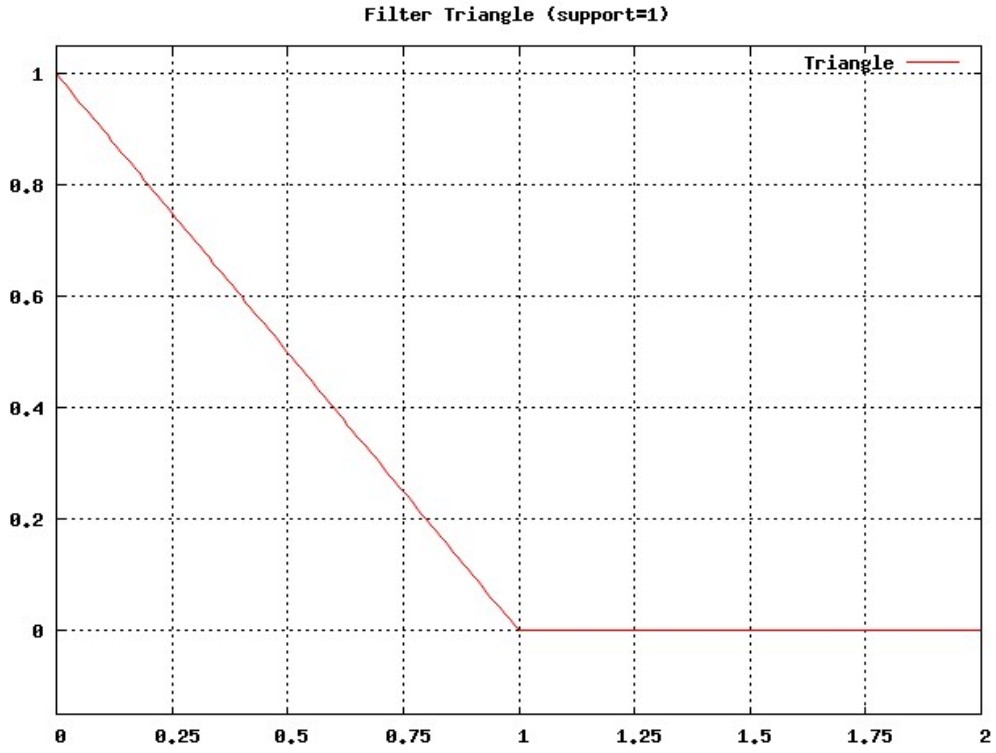
Tabulka 1-6 Aplikace filtru Triangle při zmenšování obrazu (10x zvětšeno)

zdrojový obrázek	120%	140%	160%	180%	200%
					

Tabulka 1-7 Aplikace filtru Triangle při zvětšování obrazu (10x zvětšeno)

zdrojový obrázek (20x zvětšeno)	300 násobné zvětšení s aplikací filtru triangle
	

Tabulka 1-8 Aplikace filtru Triangle při zvětšování obrazu



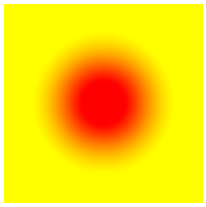
obr. 2 Graf filtru Triangle – váha hodnoty pixelu v závislosti na vzdálenosti

1.2.4. „Gaussovy“ filtry

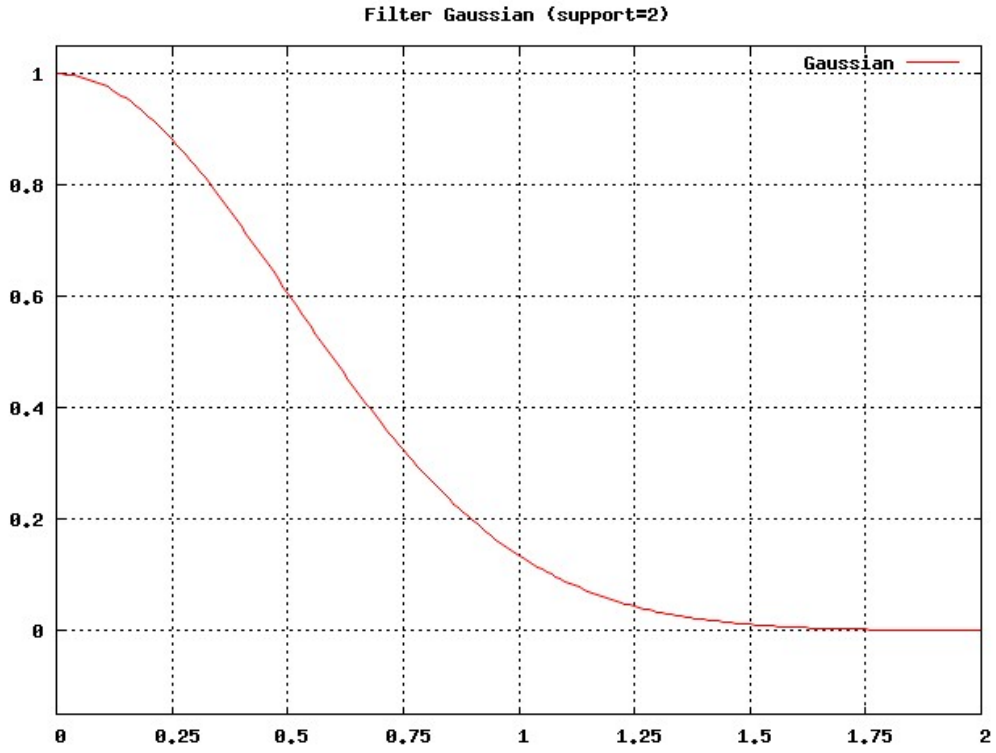
Charakterem jsou tyto filtry předurčeny ke zpracování snímků reálného světa, při jejichž převzorkování dochází ke vzniku vysokofrekvenčního šumu, projevujícího se shluky aliasingu a moaré. Okolí zdrojového pixelu zasahuje 1,5 až 2 pixely daleko, čímž dochází k odstranění většiny artefaktů, avšak okraje barevných přechodů nejsou ostré, ale rozmazané.

1.2.4.1. Gaussův filtr

Je základním filtrem Gaussových filtrů, z něhož jsou ostatní filtry této rodiny odvozeny. Vhodnou aplikací dochází k odstranění aliasingu, čtvrcových i prstencových artefaktů, vše na úkor velkého rozostření.

zdrojový obrázek (10x zvětšeno)	**33násobné zvětšení s aplikací Gaussova filtru**
	

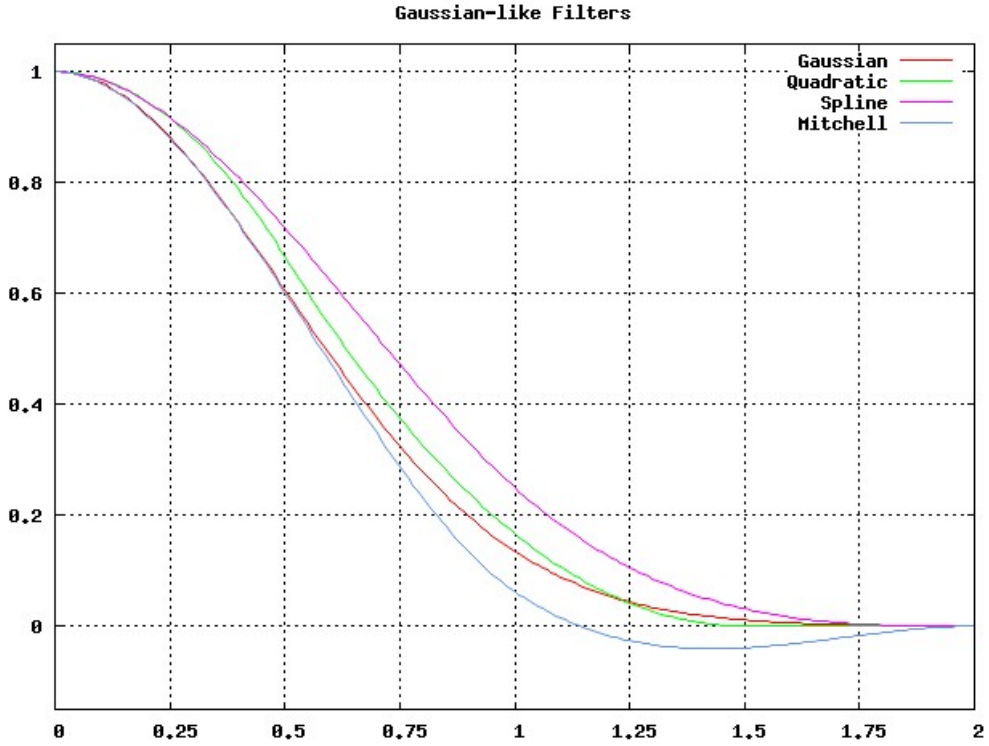
Tabulka 1-9 Aplikace Gaussova filtru při zvětšování obrazu



obr. 3 Graf Gaussova filtru – váha hodnoty pixelu v závislosti na vzdálenosti

1.2.4.2. Ostatní „Gaussovy“ filtry

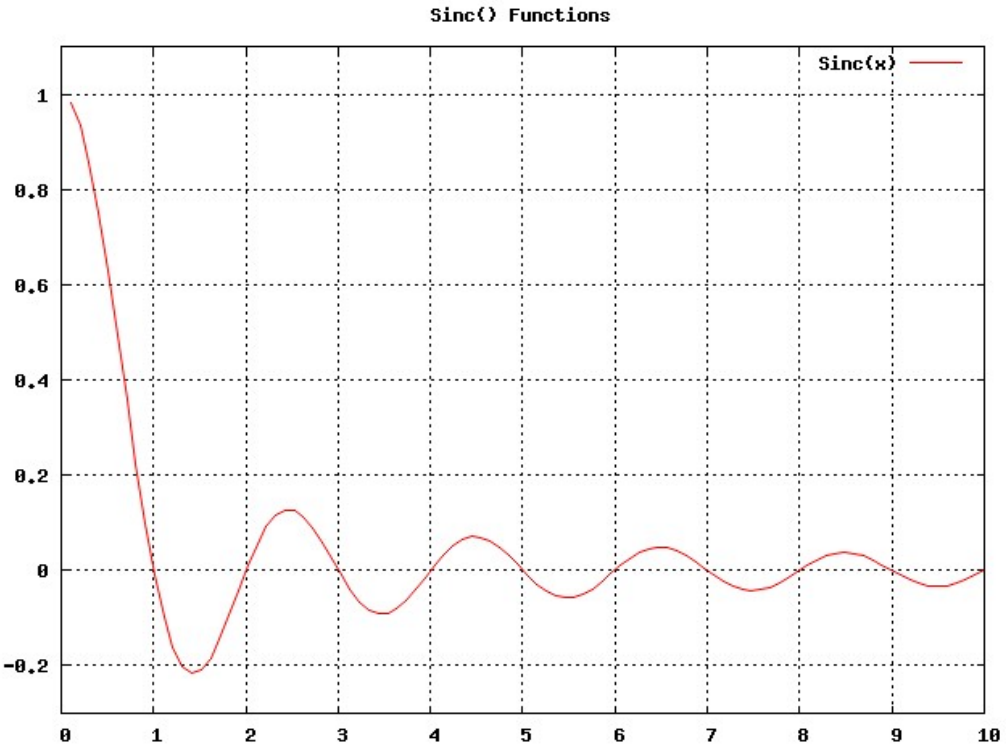
Křivky filtrů Quadratic, Spline a Mitchell, patřících do této skupiny, sledují křivku Gaussova filtru. Jedná se o křivky polynomické funkce, proto jsou tyto filtry výpočetně mnohem rychlejší, což bylo hlavním důvodem jejich vzniku. Zatímco filtr Quadratic se projevuje při aplikaci u snímků jen o trochu více rozostřeně, nežli Gaussův filtr, filtr Spline rozostřuje nový snímek o poznání více. To vše je dáno hodnotou křivky ve vzdálenosti 1 (vodorovná osa). Naproti tomu filtr Mitchell rozostřuje výsledný obrázek méně, nežli Gaussův filtr, zachovává však v blízkosti ostrých hran prstencové artefakty, což je zapříčiněno zápornou hodnotu váhy pixelu (svislá osa) ve vzdálenosti cca 1,15 až 2 pixely.



obr. 4 Grafy filtrů Quadratic, Spline a Mitchell ve srovnání s Gaussovým filtrem

1.2.5. Sinc filtry

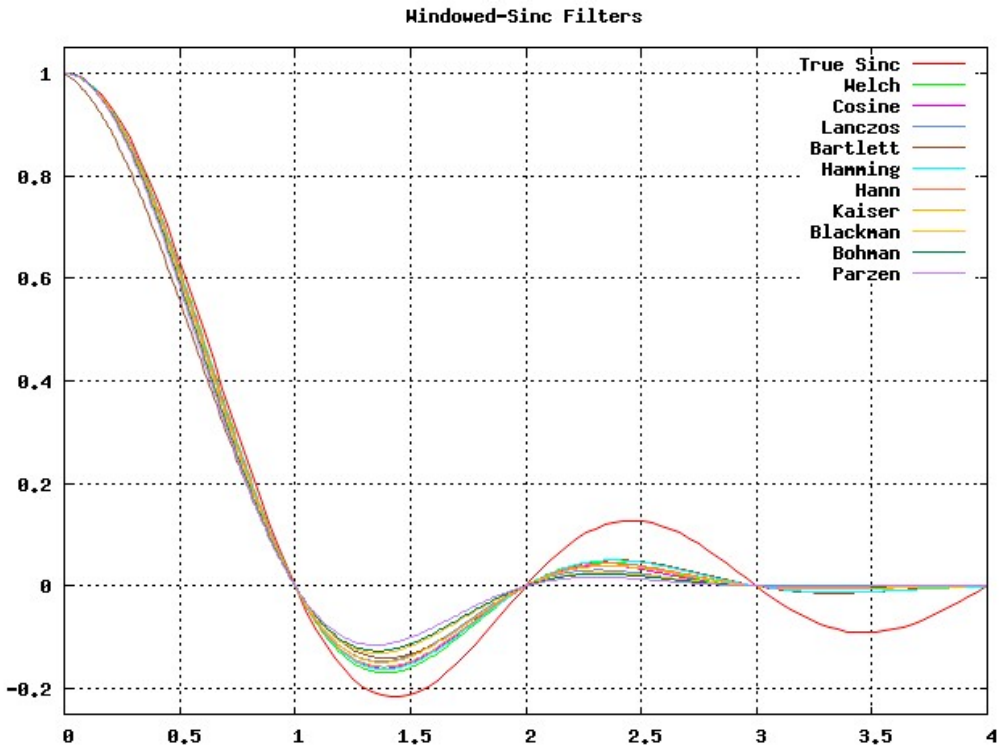
Jedná se o matematicky dokonalé filtry, jejichž křivka prochází v každé celé vzdálenosti pixelu nulovou váhou. To v důsledku znamená, že výsledný snímek není po aplikaci sinc filtru rozostřen více, než je nutné. Snímek, u něhož nedochází ke změně velikosti, není rozostřen na rozdíl od Gaussova filtru. Vlivem více pozitivních a negativních vln křivek, které zasahují do značně rozsáhlého okolí, dochází (ve srovnání s předešlými filtry) ke zvýraznění hran kontrastních přechodů, podél nichž vznikají prstencové artefakty. Zároveň jsou tyto filtry výpočetně velice pomalé.



obr. 5 Grafy filtru Sinc

1.2.5.1. Windowed-Sinc filtry

Byly vyvinuty pomocí modulace Sinc filtru za účelem zjednodušení a zrychlení, přičemž došlo k omezení využívaného okolí převzorkovávaných pixelů do vzdálenosti 3 (např. filtr Lanczos) až 4 (např. filtr Blackman).

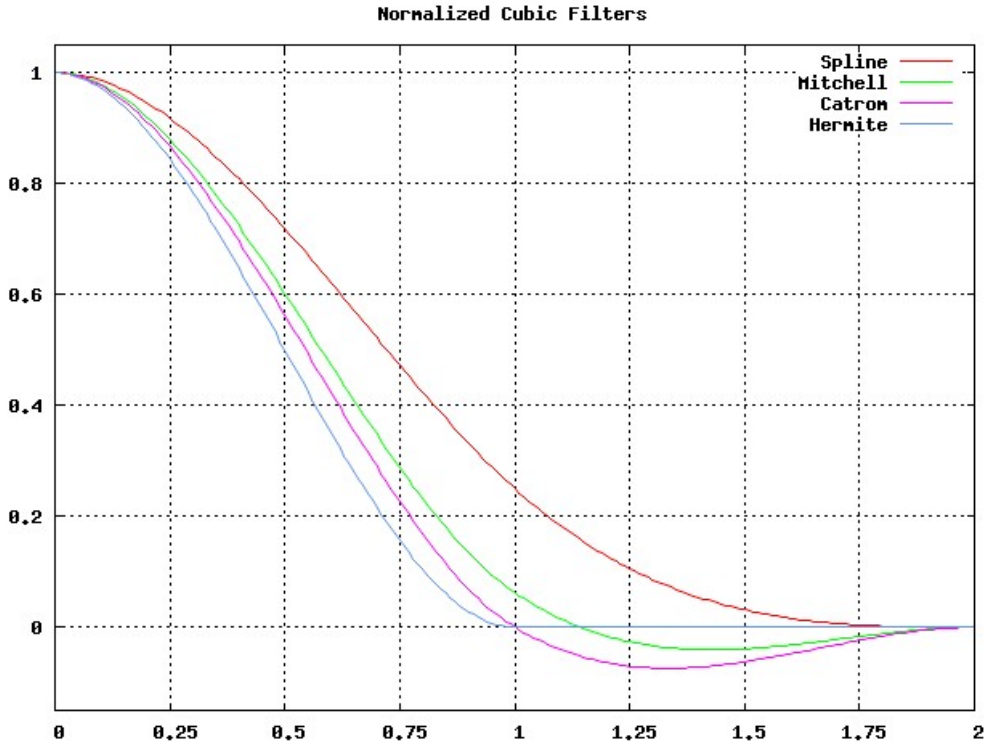


obr. 6 Grafy modulovaného filtru Sinc

Všechny windowed-sinc filtry jsou tlumenou formou filtru Sinc. Až na velmi malé změny v množství prstencových artefaktů jsou od sebe téměř nerozeznatelné. Pravděpodobně nejčastěji využívaným filtrem z této skupiny je filtr Lanczos, který leží mezi ostatními windowed-sinc filtry.

1.2.6. Kubické filtry

Vznikly na základě vývoje rychlých filtrů určených primárně pro změnu velikosti obrazu, kdy efekt jejich využití je kompromisem mezi rozostřením, čtvercovými a prstencovými artefakty. Nejlépe vyváženého kompromisu mezi zmíněnými artefakty dosahuje filtr Catrom. Do této skupiny filtrů můžeme rovněž zařadit filtr Sinc, na základě něhož byly kubické filtry vyvinuty, a také Mitchell.



obr. 7 Grafy kubických filtrů

2. Návrh

V této kapitole jsou popsána konkrétní řešení jednotlivých požadavků, či návrhy na řešení požadavků, které jsou mimo rozsah práce, a to v návaznosti na možné varianty vyjmenované v kapitole *Analýza*. Návrh je rovněž odůvodněn, v závislosti na jeho kladech či záporech.

2.1. Export PDF souboru do obrázků

Z možností vyjmenovaných v analýze tohoto požadavku bude (z důvodu jednoduchosti, dostačující funkčnosti a především bezplatné dostupnosti) využito nástroje ImageMagick, který využijeme i ke zpracování získaných obrázků, a to nejen při realizaci tohoto požadavku. S jeho pomocí a přes rozhraní Imagick, zpřístupňující použití ImageMagicku jazyku PHP, bude vytvořena ze zadaného PDF souboru sada snímků odpovídající jednotlivým stránkám souboru. V případě potřeby bude velikost stránek upravena na rozlišení odpovídající zobrazovacímu zařízení. Bude-li stránka PDF souboru i po zvětšení příliš malá, případně nevyplní celou plochu snímku o rozlišení 1920x1080 pixelů, bude doplněna černým pozadím, případně názvem dokumentu, z něhož pochází a také datem a časem provedené konverze.

Dojde-li při zpracování stránek PDF souboru k nutné úpravě velikosti, přijde na řadu převzorkování. Při této operaci standardně dochází ke zkreslení obrazu, případně ke ztrátě informací, proto je nutné tomuto kroku věnovat dostatečnou pozornost. Lze očekávat, že obsah PDF bude smíšeného charakteru, ponese tedy textovou i grafickou informaci. Z tohoto vyjde i volba filtru použitého při převzorkování obrazu. Z charakteru filtrů se jako vhodná varianta jeví některý ze Sinc filtrů. Z důvodu většinového zpracování vícestránkového dokumentu je s ohledem na rychlost zvolen kubický filtr Catrom.

Vzhledem k možným orientacím a velikostem získané stránky PDF souboru byla navržena rozvržení výsledného snímku do celkem tří layoutů. Rozložení dvou layoutů bude možné v případě potřeby změnit jednoduchým zásahem do zdrojového kódu skriptu.

2.1.1. Rozložení snímku

2.1.1.1. Samostatný obrázek stránky

Layout bude použit v případě, že získaný obrázek bude dosahovat požadovaného rozlišení. Dále se použije v případě, že bude možné obrázek na požadované rozlišení zvětšit/zmenšit aniž by byla zásadněji ovlivněna jeho čitelnost (doporučené zvětšení je maximálně dvojnásobné), případně z větší části vyplní plochu výsledného snímku.

2.1.1.2. Layout se dvěma sloupci (poměr 2:1)

Snímek je rozvržen do dvou sloupců, přičemž v levém (širším) sloupci je umístěn obrázek stránky (obrázek s přesahující výškou, případně čtvercový obrázek), doplněný o název souboru, datum a čas pořízení snímku umístěné v pravém (užším) sloupci.

2.1.1.3. Layout se dvěma řádky (poměr 3:1)

Snímek je rozvržen do dvou řádků. V horním (vyšším) řádku je umístěn obrázek stránky (s převažující šířkou), v dolním (nižším) informace o názvu souboru, datum a čas pořízení snímku.

2.2. Snímání webové stránky

K vytvoření snímku webové stránky bude (mimo funkčnosti ImageMagicku) využito Page2Images REST API, které se postará o vykreslení a uložení webové stránky do obrázku PNG. Rozhraní bylo zvoleno především kvůli jednoduchosti použití, kvalitě výstupu a možnosti nastavení jeho parametrů, a také proto, že nevyžaduje instalaci přídavných knihoven či jiných aplikací. Cenou za poskytované služby jsou omezení, která bezplatný účet skýtá - nepodporuje volání zabezpečených protokolů, a zároveň umožňuje pouze zpracování 100 jedinečných URL adres za den (3000 za měsíc), vzhledem k využití skriptu lze ale předpokládat, že tyto limity nebudou překážkou.

Konečnou úpravu snímku webu, u něhož se uživateli nabízí možnost v průběhu skriptu definovat ořez, případně přidat krátký popis snímku, zhotoví ImageMagick. Dle zadaných parametrů snímek ořízne, v případě potřeby zmenší či zvětší. Pokud snímek nedosáhne potřebného rozlišení, přidá černé pozadí, a zbyde-li dostatek prostoru, vloží informaci, z které URL adresy a kdy byl snímek pořízen, případně přidá uživatelem zadaný popis.

Při změně velikosti snímku získaného z rozhraní Page2Images bude opět docházet k převzorkování obrazu. I zde lze předpokládat smíšený charakter webu. Zpracovávaný snímek bude v daném okamžiku pouze jeden, nebude tedy zohledňována rychlost filtru, která se v případě převzorkování jednoho snímku tolik neprojeví, ale bude použit filtr pro danou aplikaci nejvhodnější. Z vlastností filtrů se jedná o filtr typu Sinc. Z této skupiny bude zvolen nejčastěji používaný filtr Lanczos.

Podobně jako u realizace skriptu pro konverzi PDF souborů lze předpokládat rozličnou orientaci výsledných snímků webu. Zachovánu zůstane i rozložení layoutů výsledného snímku, viz výše.

2.3. Export snímku videa

K vytvoření snímku videa bude využit nástroj FFmpeg, který bude s PHP propojen sadou tříd nástroje PHPVideoToolkit. V případě obou nástrojů se jedná o nástroje bezplatné, platformě nezávislé, stále aktualizované a navíc rychlé. Pomocí zmíněných nástrojů bude získán z uživatelem definovaného video souboru jeden či více snímků (v závislosti na volbě uživatele), které budou dále zpracovány, stejně jako v předešlých skriptech, funkcemi poskytovanými ImageMagickem.

Při změně velikosti získaného obrázku dojde k převzorkování, které může do snímku vnést artefakty, jejichž vznik a vliv na výsledný snímek bude minimalizován volbou vhodného filtru. Za předpokladu, že snímky videa jsou obrazového charakteru, bude použit filtr ze skupiny „Gaussových" filtrů, které jsou pro převzorkování obrazů reálného světa vzhledem ke svému charakteru nevíce využívané. Z důvodu vyšší rychlosti bude použit filtr Quadratic, který se nejvíce blíží skutečnému Gaussovu filtru.

Mimo volbu video souboru nabídne skript uživateli možnost zvolit počet snímků videa, a to výběrem rozložení ve výsledném snímku. K dispozici budou uživateli čtyři layouty, přičemž u každého zůstane ve spodní části vyhrazena malá oblast (1920x80 pixelů) pro vložení informace o názvu video souboru, datu a času pořízení snímku.

2.3.1. Rozložení snímku

2.3.1.1. Samostatný obrázek videa

Rozložení bude použito v případě volby jednoho obrázku. Vzhledem k vyhrazené části zbyde na pořízený snímek videa prostor 1920x950 pixelů.

2.3.1.2. Layout se dvěma sloupci a dvěma řádky (2x2)

Layout bude použit v případě volby čtyř obrázků. Po odečtení vyhrazené oblasti a rozpočtení zbylého prostoru dostáváme pro jeden snímek plochu 960x475 pixelů.

2.3.1.3. Layout se třemi sloupci a dvěma řádky (3x2)

Bude použit v případě volby šesti obrázků. Odečteme-li vyhrazenou oblast, dostáváme se po rozpočtení zbývající plochy k ploše 640x475 pixelů příslušející každému obrázku.

2.3.1.4. Layout se třemi sloupci a třemi řádky (3x3)

Rozložení, které bude použito při devíti získaných obrázcích, přičemž každému bude náležet plocha 640x316 pixelů.

2.4. Export snímku webkamery

K vytvoření snímku z webkamery bude použit skript velmi podobný snímání webové stránky. Uživatel bude chtít snímat jen požadovanou část webové stránky, do níž je výstup webkamery implementován, jelikož ne vždy má webkamera svoji samostatnou stránku, z níž by bylo možné sejmout právě jeden obrázek (ať již z obrázku webkamery nebo ze streamu), zůstane otevřená možnost vybrat konkrétní oblast, v níž se obraz webkamery nachází. Uživatel zadá URL adresu webkamery a ze získaného snímku bude moci vytvořit ořez umístěný do výsledného snímku. K tomu všemu bude opět využito REST rozhraní Page2Images a funkcí ImageMagicku.

V otázce převzorkování obrázku bude skript inspirován změnou velikosti snímků videa a zrovna tak bude používat filtr Quadratic z rodiny „Gaussových" filtrů, který se nejvíce hodí pro zvětšování a zmenšování obrazů reálného světa, které u webkamery předpokládáme.

V neposlední řadě zůstane uživateli zachována možnost přidat komentář pořízeného snímku. K němu zaznamenáme i adresu webkamery a datum a čas pořízení snímku.

2.5. Tvorba speciálních snímků

Oproti předešlým snímkům nebude při psaní skriptu pro tvorbu uživatelských textových snímků potřeba žádného dalšího nástroje mimo ImageMagicku, jehož součástí je třída ImagickDraw. Ta poskytuje mnoho funkcí, s nimiž si skript vystačí. Rovněž nebude nutné řešit převzorkování snímku, který budeme tvořit přímo pro rozlišení zobrazovacího zařízení. Důležitým faktorem zůstává volba velikosti písma pro zachování čitelnosti textu. Pro titulek bude zvolen font s větší velikostí a zůstane mu vyhrazen prostor v horní části snímku. Dolní část snímku (1920x50 px) bude vyhrazena pro umístění data a času pořízení. Zbývající část náleží informačnímu textu střední velikosti. Layout snímku je jasně definovaný, zbývá určit, kolik prostoru bude přiděleno nadpisu a kolik uživatelskému textu. Rozložení prostoru bylo zvoleno s ohledem na očekávané množství a velikost textu v poměru 1:5. Zásahem do zdrojového kódu jej lze snadno změnit.

Uživatel bude moci zvolit barvu nadpisu, textu i pozadí. Napíše vlastní text a hotový snímek dle potřeby zvýrazní rámečkem zvolené barvy a tloušťky.

2.6. Indexování snímků a řazení do fronty

Indexování snímků bylo, (vzhledem k naplnění požadavku č. 8) zvoleno pomocí časového razítka získaného v PHP funkcí *microtime*, která vrací čas s přesností na mikrosekundy, čímž bude zaručena jedinečnost indexu. Index bude umístěn do začátku názvu snímků, čímž bude naplněn i požadavek č. 8. Uživateli zůstane ponechána možnost ovlivnit řazení snímků do fronty, podle níž jsou snímky pod řízením mikropočítače Raspberry zobrazovány na LCD panelech, kde zobrazování probíhá dle abecedního uspořádání názvu snímků.

Vzhledem k nastavení mikropočítače Raspberry, kdy nelze zcela řídit frontu zobrazovaných snímků pomocí indexu (později vkládaný snímek nelze umístit před snímek dříve vložený) a jiné řízení by bylo na straně serveru velmi komplikované (v krajním případě by vedlo k zpětným zásahům do indexů), lze doporučit změnu konfigurace mikropočítače tak, aby byl schopen reagovat na řazení snímků definované uživatelem. To by v budoucnu mohlo probíhat například předáním jednoduchého textového souboru ze serveru do mikropočítače (např. jako součást synchronizace snímků), který by udával pořadí snímků, jež by bylo možné jednoduše měnit pouhým uspořádáním názvů snímků v tomto souboru, nikoliv zásahem do jejich názvu.

2.7. Priorita snímků

Možnosti ovlivnění priority zobrazovaných snímků na straně serveru jsou dosti minimalistické. To je dáno, podobně jako ovlivňování uspořádání fronty, konfigurací mikropočítače Raspberry a funkčností na něm využívaného prohlížeče. Přesto je v kapitole 1.1.9 Priorita snímků vyjmenováno několik možných způsobů řízení priority při současném zachování konfigurace mikropočítače.

Vzhledem k výše zmíněné robustnosti řízení priority nezbývá než doporučit změnu konfigurace mikropočítače Raspberry a na něm používaného prohlížeče tak, aby bylo umožněno mikropočítači a prohlížeči pružně reagovat na prioritu snímků předanou například ve formě jednoduchého souboru se seznamem snímků a uvedením jejich priorit.

3. Implementace

3.1. Export PDF souboru do obrázků

PHP skript *pdf2image* je rozdělen do dvou částí, v první je umístěn formulář určený k volbě PDF souboru včetně kontroly, zda byl soubor skutečně zadán a jedná-li se opravdu o PDF soubor. Druhá část zastává hlavní funkčnost skriptu, zajišťující konverzi souboru a konečné zpracování snímků.

Formulář, který předává parametry uživatelem zadaného souboru metodou POST přes proměnnou *$_FILES*, zasílá dále i skrytou proměnou *$_POST[‚form‘]*. S těmito parametry skript volá sám sebe, je-li nastavena skrytá proměnná, proběhne kontrola zadání PDF souboru typu application/pdf a v případě splnění obou podmínek přejde skript do druhé fáze. Není-li zadán žádný soubor, či je jeho typ nesprávný, vypíše se chybové hlášení a dojde k opětovnému zobrazení formuláře. Kontrolu zastává následující část zdrojového kódu:

```
if(!isset($_POST['form']) or strcmp($_FILES['file']['type'],
  'application/pdf') != 0 ? $err = "Zvolený soubor neodpovídá souboru
  PDF! Zvolte, prosím, jiný soubor." : $err = "")
```

Druhá část skriptu načítá pomocí funkce třídy *Imagick* předané PDF, které je v průběhu načítání rozděleno na snímky odpovídající jednotlivým stránkám souboru. Následný cyklus *foreach* zajišťuje pro každý snímek úpravu do konečné podoby, o což se z větší části stará funkce *image*. Ta je společná pro více skriptů a jejímu bližšímu popisu je níže věnována samostatná kapitola. Funkci *image* jsou předávány celkem 3 parametry – získaný snímek stránky PDF, text, který má být doplněn do konečného snímku, a filtr, který má být k převzorkování snímku použit (pro PDF soubory byl zvolen filtr Catrom). Předávaný text není uživatelským vstupem, je ve skriptu pevně definován a sestává z názvu zpracovaného souboru (proměnná $*filename*), který je při délce větší než 50 znaků zkrácen (jinak by se nevešel do layoutu rozdělujícího výsledný snímek na dva sloupce) a také informace o datu a času zpracování konkrétního snímku. V závěru je změněno kódování textu na UTF-8, a to kvůli třídě ImageDraw, jejíž funkce zajišťují vykreslení snímku do textu.

```
if(strlen($filename > 50)) $filename = substr($filename, 0, 50).'...';
$text = "Snímek z PDF souboru:\n".$filename."\nzískán dne ".date("j.
  n. Y, G:i:s",time());
$text = iconv("Windows-1250", "UTF-8", $text);
```

V samém závěru skriptu je snímku vypočítán index (výpočet indexu popsán v samostatné kapitole) a snímek je uložen do složky definované v proměnné *$PATH* zadané na samém začátku skriptu, aby ji bylo možné snadno měnit.

3.2. Snímání webové stránky

Skript *webpage2image.php* je rozdělen do tří částí, v první je umístěn formulář k zadání URL adresy, v druhé je URL adresa odeslána API Page2Image, a v případě, že byl získán snímek webu, je zobrazen formulář k oříznutí pořízeného snímku (uživatel má možnost si jej v této fázi prohlédnut) a přidání případného krátkého komentáře. Poslední část sestává z hlavní funkčnosti skriptu, zajišťující oříznutí snímku a jeho zpracování do konečné podoby.

První formulář posílá metodou POST kromě zadané URL adresy i skrytou proměnou *$_POST[‚form‘]*, určující jakou fází již skript prošel.

V druhé fázi je vytvořeno připojení k adrese již zmíněného API, kterému je mimo adresu webové stránky prostřednictvím proměnné *$parameters* předáno (opět metodou POST) v poli i několik dalších parametrů (povinné jsou pouze *p2i_url* a *p2i_key*). Konkrétní konfigurace parametrů skriptu je následující:

```
$parameters = array(
    "p2i_url" => $_POST['url'],      // zadaná URL adresa
    "p2i_key" => "…",                // osobní klíč k REST API
    "p2i_device" => $device,         // typ zařízení, pro nějž má být snímek vytvořen
    "p2i_size" => "1920x0",          // rozlišení získaného snímku
    "p2i_screen" => "1920x1080",     // rozlišení simulovaného zařízení
    "p2i_fullpage" => 1,             // 0 = viditelná oblast / 1 = celý web
    "p2i_imageformat" => "PNG",      // formát snímku (JPG / PNG / PDF)
    "p2i_quality" => "85"            // kvalita snímku (PNG: 70 – 85 /JPG: 80 – 95)
);
```

Proměnná *$device* může nabývat hodnot 0 – 8, přičemž každá udává jiné zařízení (0 - iPhone4, 1 -iPhone5, 2 - Android, 3 - WinPhone, 4 - iPad, 5 - Android Pad, 6 - Desktop, 7 - iPhone6, 8 - iPhone6+), v našem případě je *$device* rovna 6. Nula zapsaná v parametru *p2i_size* udává maximální možnou velikost snímku.

Dle odpovědi na připojení k API je následně určen další průchod skriptem. Za méně zajímavými částmi skriptu zajišťujícími zobrazení chybového hlášení v případě, že se nepodaří připojení k rozhraní Page2Image vytvořit, případně dojde k chybě API, následuje část, která zpracovává zaslanou odpověď, jež je předávána JSON objektem. Status JSON dat může nabývat následujících hodnot a významů:

processing	zaslaný požadavek dosud nebyl vyřízen
error	při zpracování požadavku došlo k chybě (pravděpodobně zadání nesprávné URL adresy)
finished	požadavek byl vyřízen, je vrácena URL adresa získaného snímku

V případě odpovědi „processing“ skript čeká na jinou z dalších dvou možných odpovědí, a to do doby, než vyprší lhůta stanovená jako maximální doba pro zpracování požadavku (2 minuty). Je-li vrácena odpověď „error“, je vypsáno chybové hlášení a běh skriptu je ukončen. Při odpovědi „finished“ je snímek webu uložen do objektu typu Imagick a následně do předem definované složky:

```
$img->readImageBlob(file_get_contents($json_data->image_url));
$img->writeImage($PATH.(string)$id."_webpage.png");
```

Proměnná *$PATH*, definovaná v samém začátku skriptu, skrývá cestu k ukládání získaných snímků, *$id* pak zastupuje index snímku. Zajímavý je způsob načítání obrázku do třídy *Imagick*. K přečtení obrázku nabízí *Imagick* tři funkce – *readImage*, *readImageFile* a *readImageBlob*. Zatímco *readImage* slouží k načtení obrázku z diskového prostoru, *readImegeFile* čte obrázek z ukazatele obrázku, tedy například z URL adresy, a *readImageBlob* načítá vyjádření obrázku v podobě řetězce. Načítání obrázku pomocí *readImageFile* se bohužel nepodařilo realizovat, proto je obrázek převeden nejprve pomocí funkce *file_get_contents* do textové podoby a následně funkcí *readImageBlob* načten do skriptu.

Ještě v průběhu prostřední části skriptu následuje vypsání druhého formuláře, do nějž uživatel zadá parametry ořezu snímku v podobě hodnot X, Y, šířka, výška a přidá krátký komentář. Rovněž je k dispozici i volba „Přizpůsobit ořez snímku/snímek poměru 16:9“, která obrázkem webu maximálně vyplní plochu výsledného snímku. Tato volba je defaultně zapnuta.

Odesláním formuláře přechází skript do závěrečné části, na jejímž začátku proběhne kontrola zadaných hodnot, která částečně probíhá již ve formuláři samotném (není možné zadat hodnoty menší než 0 a větší než rozměr obrázku). Při této kontrole se ověří, zda lze zadané rozměry obrázku vzhledem k souřadnicím bodu [X, Y] splnit, a pokud ne, jsou přepočteny na maximální hodnoty, což je učiněno i v případě, jsou-li zadané hodnoty nulové. Pokud zůstala zvolena volba zachování poměru, přepočteme navíc šířku/výšku tak, aby výsledný ořez snímku tomuto poměru odpovídal. Přitom je nutno změnit pouze větší parametr, který nesplňuje podmínku:

```
if(($crop['width']/16) > ($crop['height']/9))
    $crop['width'] = $crop['height']/9*16;
if(($crop['width']/16) < ($crop['height']/9))
    $crop['height'] = $crop['width']/16*9;
```

Následně je obrázek oříznut funkcí cropImage, která je součástí třídy Imagick. Nyní zbývá připravit text obsahující URL adresu stránky (ta je v případě délky větší než 50 znaků zkrácena), datum a čas zachycení snímku a také případný komentář uživatele, který je rozdělen do řádků po přibližně 50 znacích (dělení realizuje funkce *wordwrap*, která řádky odděluje po celých slovech).

V závěru zavoláme funkci *image* (bližší popis je k dispozici v samostatné kapitole), která zajistí konečnou úpravu snímku, a snímek uložíme:

```
$img = image($img, $text, Imagick::FILTER_LANCZOS);
$img->setImageFormat('png'); // nastavení formátu snímku
$img->writeImage($PATH.$_POST['id'].'_webpage.png');
```

V závěru popisu implementace je nezbytné dodat, že část kódu byla převzata z vzorových kódů, které jsou k dispozici ke stažení na webových stránkách API Page2Image. Konkrétně se jedná o části kódu zajišťující připojení k API a definující smyčku pro zpracování odpovědi v podobě JSON dat.

3.3. Export snímku videa

Skript určený k tvorbě snímků videa je nejvíce členitým skriptem. To je dáno především konfiguračními skripty použitého nástroje PHPVideoToolkit – *autoloader.php* a *config.php*. Tyto skripty jsou součástí balíku PHPVideoToolkit, slouží ke konfiguraci jmenného prostoru a cest k třídám balíku a dalším potřebným nástrojům a nejsou zde více popsány. Skripty, které jsou součástí této práce, se jmenují *video2images.php* a *thumbnails.php*. První skript zajišťuje kontakt s uživatelem, kterému zobrazí formulář pro zadání parametrů snímku a dále zpracovává konečný snímek z obrázků vytvořených druhým skriptem.

Skript *video2images* je rozdělen na dvě hlavní části. V úvodu je umístěno nastavení důležitých proměnných, aby bylo možné snadno konfigurovat základní nastavení skriptu. Následuje krátký kód pro vypsání případných chybových hlášení. Ta jsou v tomto případě předávána nikoliv pomocí proměnné *$_POST*, ale přes *$_SESSION*, stejně jako další hodnoty, které vrací skript *thumbnails*. Jediné volání metodou POST zprostředkovává formulář, do něhož uživatel zadává cestu k video souboru a volí počet obrázků, které budou zobrazeny na konečném snímku. Předání video souboru do skriptu *thumbnails* je zajištěno proměnnou *$_FILES*. Spolu s ní odesílá formulář prostřednictvím proměnné *$*_POST rozvržení layoutu a také cestu k nastavené složce pro dočasné soubory (*$TMP*).

Následuje přechod do skriptu *thumbnail*, který je součástí jmenného prostoru PHPVideoToolkit. V samém úvodu jsou pro skript připraveny některé předané proměnné. Skript pokračuje testováním typu předaného souboru. Není-li soubor formátu AVI, MPEG, MOV, MP4 či WMV, dojde k volání prvního skriptu spolu s předáním chybového hlášení, které se zobrazí nad formulářem. V případě, že byl předán podporovaný formát videa, začne jeho zpracování.

Nejprve je pomocí třídy *MediaParser* (součástí PHPVideoToolkit) získána informace o době trvání videa. Tu třída vrací v podobě hodin, minut a vteřin, pro potřeby skriptu je použita proměnná *$video_duration_seconds*, která je rovna přepočítanému časovému údaji na sekundy.

Dále je v závislosti na počtu uživatelem zvolených obrázků rozhodnuto o metodice tvorby náhledů videa:

```
if($count == 1) {
    $video  = new Video($_FILES['file']['tmp_name']);
    $process = $video->extractFrames(new
      Timecode($video_duration_seconds / 2))
      ->save($tmp.'frame1.png', null, Media::OVERWRITE_EXISTING);
} else {
    for($i = 0; $i < $count; ) {
        $video  = new Video($_FILES['file']['tmp_name']);
        $process = $video->extractFrames(new
          Timecode(round($video_duration_seconds / ($count - 1) * $i)))
          ->save($tmp.'frame'.++$i.'.png', null,
          Media::OVERWRITE_EXISTING);
    }
}
```

Byl-li zvolen jeden obrázek z videa, je náhled vytvořen uprostřed video souboru, v případě jiné volby jsou obrázky získávány v pravidelných časových intervalech od začátku videa do konce. Pořízené obrázky jsou ukládány do složky pro dočasné soubory. Proběhne-li vše bezchybně, je připraveno několik proměnných k předání *video2image*, který je na konci skriptu zavolán.

Ke skriptu *thumbnails* je nezbytné poznamenat, že získávání snímků z video souboru je v současné době naprogramováno poněkud těžkopádně. Lze si povšimnout, že při získávání obrázků z videa je vždy otevírán nový video soubor, což není nezbytné. Za normálních okolností by postačilo soubor otevřít pouze jednou a z něho by následně bylo možné získat libovolný počet obrázků. V průběhu vytváření skriptu však byla odhalena chyba v balíku PHPVideoToolkit, související se špatným indexováním získaných obrázků, která znemožnila vytvoření více obrázků z jednoho souboru. O této chybě byl autor balíku PHPVideoToolkit informován a přislíbil její nápravu, jakmile to jen bude možné. Po odstranění výše popsané chyby bude vhodné skript opravit.

Po návratu do *video2image* přejde skript (v případě úspěšného pořízení obrázků) do druhé části. V té je nejprve otevřena složka s dočasnými soubory a jsou z ní načteny všechny obrázky formátu PNG (složka by měla obsahovat pouze tyto obrázky, které byly získány skriptem *thumbnails*). Při načítání obrázků zároveň dochází k přepočtu jejich velikosti v závislosti na proměnné *$MAX_RESIZE* a na zvoleném layoutu konečného snímku:

```
if(($img->getImageWidth()*$MAX_RESIZE > ($SCREEN_RESOLUTION_WIDTH /
    $layout[0])) or ($img->getImageHeight()*$MAX_RESIZE >
    (($SCREEN_RESOLUTION_HEIGHT - 80) / $layout[1]))) {
    $resize["width"] = $SCREEN_RESOLUTION_WIDTH / $layout[0];
    $resize["height"] = ($SCREEN_RESOLUTION_HEIGHT - 80) / $layout[1];
} else {
    $resize["width"] = $img->getImageWidth()*$MAX_RESIZE;
    $resize["height"] = $img->getImageHeight()*$MAX_RESIZE;
}
```

Následuje samotná změna velikosti obrázku za použití filtru *Quadratic*. Jakmile je načítání obrázků dokončeno, je vytvořen černý snímek o rozlišení 1920x1080 pixelů. Do něho je poté podle zvoleného layoutu umístěn příslušný počet snímků:

```
for($i = 0; $i < $layout[1]; $i++) {
    for($j = 0; $j < $layout[0]; $j++) {
        $composite->compositeImage($img, Imagick::COMPOSITE_DEFAULT,
          ($j * $composite->getImageWidth() / $layout[0] +
          ($composite->getImageWidth() / $layout[0] -
          $img->getImageWidth()) / 2), ($i *
          ($composite->getImageHeight() - 80) / $layout[1] +
          (($composite->getImageHeight() - 80) / $layout[1] -
          $img->getImageHeight()) / 2));
        if($img->hasNextImage()) $img->nextImage();
        else break;
    }
}
```

Do snímku je v závěru přidán ještě text s informacemi o zdrojovém souboru a o datu a času pořízení, snímku je vypočten index a snímek je uložen. Na konci skriptu jsou smazány všechny soubory umístěné v dočasném adresáři.

3.4. Export snímku webkamery

V souvislosti s tím s obsahem webcam2image, který je totožný se skriptem pro tvorbu snímku webové stránky, je zde uvedena pouze odlišnost tohoto skriptu, nikoliv celý jeho popis.

Jedinou, o to však důležitější odlišností skriptu je filtr použitý k převzorkování získaného obrázku. Místo filtru Lanczos je zde, stejně jako v případě převzorkování snímků videa, použit filtr Quadratick, jež se svou definicí pro dané účely hodí více.

3.5. Tvorba speciálních snímků

Skript text2image je nejméně komplikovaným skriptem, který se (vyjma ImageMagicku) obejde bez dalších nástrojů. Je rozdělen do dvou částí (podobně jako pdf2image), přičemž první slouží k zobrazení formuláře pro zadání uživatelského vstupu, druhá zastává hlavní funkčnost skriptu, během níž je ze zadaného textu a dalších parametrů vytvořen výsledný snímek.

V první fázi zobrazí skript uživateli formulář sloužící ke sběru dat. Mimo nadpis snímku a text samotný může uživatel do snímku přidat datum a čas vytvoření, případně podpořit jeho zvýraznění barevným rámečkem, a to tloušťky až 25 pixelů. Všechny parametry jsou doplněny možností zvolit barvu dané položky skrze zobrazenou barevnou paletu, jejíž vykreslení zajišťuje <*input*> tag typu *color*. Ten se však nezobrazí zcela vždy, neboť daný HTML prvek není podporován prohlížeči Internet Explorer a Safari. V takovém případě se místo barevné palety zobrazí textové pole s možností

zadat barvu pomocí jejího hexadecimálního kódu. Z tohoto důvodu je doporučeno skript používat v jiných prohlížečích, u nichž je volba barvy mnohem snazší a především intuitivní.

Po odeslání formuláře odešle skript metodou POST parametry sám na sebe. Dle skryté proměnné rozpozná skript přechod do druhé fáze a začne vykonávat instrukce k sestavení konečného snímku. Na samém začátku proběhne výpočet velikosti snímku, kde je od rozlišení zobrazovacího zařízení odečtena, v případě volby orámování snímku, dvojnásobná tloušťka rámečku, který rozměry snímku navyšuje a vytvoří se prázdný snímek s obarvením dle zadané barvy snímku. Následuje vytvoření objektu třídy ImagickDraw, do něhož probíhá vykreslení textu, případně i dalších objektů. Poté skript pokračuje nastavením parametrů textu, které jsou společné pro nadpis, i samotný text (font, antialiasing, kódování). Vzápětí jsou nastaveny parametry nadpisu (velikost, barva, tloušťka písma) a zadaný text je vykreslen do horní šestiny snímku (rozložení snímku je dáno nastavením layoutu, který byl pro tento skript zvolen v poměru 1:5). Pokračuje se nastavením parametrů textu a jeho vypsáním do prostřední části snímku (prostřední proto, protože pod textem je vyhrazena menší část pro vykreslení případného data a času). Než dojde k vypsání textu, projde úpravou, kdy je pomocí funkce *wordwrap* nejprve rozdělen do řádků (po přibližně 150 znacích), jejichž počet je poté omezen na maximálně 25, a to z důvodu omezeného prostoru. Nastavení parametrů textu a jeho úprava jsou zobrazeny v kódu níže.

```
$draw->setFontSize(25);
$draw->setFillColor(new ImagickPixel($_POST['textcolor']));
$draw->setTextAlignment(Imagick::ALIGN_CENTER);
$draw->setFontStyle(Imagick::STYLE_ITALIC);
$draw->setFontWeight(100);

$text = wordwrap($_POST['text'], 150, "\n");
$text_edit = "";
$length = strlen($text);
for($i = 0, $lines = 1; $i < $length && $lines <= 25; $i++) {
    if($text[$i]=="\n") $lines++;
    $text_edit .= $text[$i];
}
```

Bylo-li zvoleno zapsání data a času pořízení snímku, jež je zapnuto, je nutné doplnit i tyto údaje. Pokud byl zadán rámeček snímku, je přidán i ten, čímž případně dojde k navýšení rozměrů snímku na konečných 1920x1080 pixelů. V závěru skriptu je dopočítán index a snímek je uložen do adresáře zadaného v proměnné *$PATH*, která se nachází spolu s dalším nastavením (layout, rozlišení zobrazovacího zařízení) na začátku skriptu. Po uložení je uživatel informován o úspěšném vytvoření snímku a nabídneme mu možnost si jej prohlédnout.

3.6. Funkce „image“

Funkce *image* je používána celkem ve 2 skriptech (pdf2image, webpage2image), je jí proto věnována samostatná kapitola a rovněž i zvláštní skript. Na samém začátku skriptu, ještě před definicí funkce, nalezneme několik globálních proměnných:

```
$SCREEN_RESOLUTION_WIDTH = 1920;      // rozlišení zobrazovacího zařízení - šířka
$SCREEN_RESOLUTION_HEIGHT = 1080;     // rozlišení zobrazovacího zařízení - výška
$MAX_RESIZE = 2;                      // konstanta maximálního zvětšení
$LAYOUT_WIDTH = "2:1";                // rozložení snímku (sloupce)
$LAYOUT_HEIGHT = "3:1";               // rozložení snímku (řádky)
```

Tím, že se jedná o proměnné udávající vlastnosti a podobu výsledného snímku, je hlavním důvodem jejich umístění ve skriptu docílení snadného přístupu k nim a jejich případná změna při potřebě změnit nastavení skriptu. První dvě hodnoty udávají rozlišení zobrazovacího zařízení (a tedy i snímku), na němž bude výsledný snímek prezentován. Následuje konstanta maximálního zvětšení, která definuje, kolikrát maximálně bude zvětšen obrázek vkládaný do konečného snímku. Hodnota, která je jí nastavena, byla zvolena s ohledem na zachování dostatečné kvality obrázku, u něhož se artefakty projeví v přípustné míře. Poslední proměnné specifikují rozdělení konečného snímku při aplikaci jednoho z layoutů. U rozložení snímku do sloupců byl zvolen poměr 2:1, kdy v levém sloupci (plochy 1280x1080 px) je umístěn získaný obrázek a v pravém (640x1080 px) text. Poměr rozložení snímku do řádků je definován jako 3:1, obrázek je umístěn nahoře (plocha 1920x810 px), text dole (1920x270 px). Rozložení snímků je možno snadno změnit, je však třeba dát pozor na to, aby se do sloupce určeného pro text tento text skutečně vešel.

Funkce *image ($img, $text, $filter)* přijímá na vstupu 3 parametry, vrací objekt třídy Imagick a zajišťuje zvětšení získaného obrázku a jeho zakomponování do výsledného snímku. Proměnná *$img* odkazuje na zpracovávaný obrázek, *$text* obsahuje informace, které mají být do snímku vepsány, konkrétně zdroj obrázku, datum a čas jeho pořízení a případně také text zadaný uživatelem, proměnná *$filter* funkci říká, který filtr má být při převzorkování snímku použit. Na začátku funkce probíhá analýza předloženého obrázku:

```
if(($img->getImageWidth()*$MAX_RESIZE > $SCREEN_RESOLUTION_WIDTH) or
($img->getImageHeight()*$MAX_RESIZE > $SCREEN_RESOLUTION_HEIGHT)) {
    $resize["width"] = $SCREEN_RESOLUTION_WIDTH;
    $resize["height"] = $SCREEN_RESOLUTION_HEIGHT;
} else {
    $resize["width"] = $img->getImageWidth()*$MAX_RESIZE;
    $resize["height"] = $img->getImageHeight()*$MAX_RESIZE;
}
```

Je-li obrázek tak rozměrný, že by po dvojnásobném zvětšení nabyl rozlišení většího, než je rozlišení zobrazovacího zařízení, je třeba změnit jeho velikost na rozlišení 1920x1080 pixelů. V opačném případě jeho rozměry vynásobíme konstantou *$MAX_RESIZE*. Má-li obrázek rozlišení

požadované zobrazovacím zařízením, je funkce u konce, nic dalšího neprovádí, pouze vrátí obrázek zpět jako konečný snímek (text není do snímku vložen, abychom případně nepřekryli jeho důležitou část). Ten tak zcela odpovídá vstupnímu obrázku. Je-li však obrázek o trochu menší, je třeba jej zvětšit, připravit pod něj černé pozadí, nastavit parametry vkládaného textu a dle jeho současné, tedy již zvětšené velikosti rozhodnout o použití layoutu. Pokud u obrázku převažuje výška nad šířkou, či je výška šířce rovna, lze použít layout sloupcový, do levé části černého pozadí vycentrovat obrázek, do pravé zapsat bílý text. V případě, že je větší šířka, použije se layout řádkový, obrázek je na černé pozadí vložen do horní části, bílý text do dolní. Oba layouty pochopitelně předpokládají, že se obrázek do části jemu určené vejde. Při nedostatku prostoru pro obrázek bude výsledný snímek obsahovat pouze obrázek vystředěný na černém pozadí, bez textu, který by se do snímku nevešel. Funkce *image* končí návratem výsledného snímku splňujícího všechny potřebné parametry.

3.7. Indexování snímků

Index snímku je získáván funkcí *microtime* a má podobu časového razítka. Funkce *microtime*, libovolně s žádným nebo jedním parametrem v podobě logické hodnoty *bool*, vrací čas uběhlý od 1. 1. 1970, 0.00:00, a to, v závislosti na nastavené hodnotě, v podobě „microsec sec“ (defaultně, logická hodnota *false*), respektive „sec.microsec“ (hodnota *true*). Ve skriptech je použito defaultní volání funkce *microtime*. Index je upraven do podoby, kdy jsou za sekundami připojeny první čtyři číslice z mikrosekund:

```
$id = explode(" ", microtime(false));
$id = $id[1].substr(str_replace("0.","",$id[0]),0,4);
```

Časové razítko, které zajišťuje zobrazování snímků v pořadí, ve kterém byly zpracovány (v závislosti na nastavení mikropočítače Raspberry), je vloženo na začátek názvu každého snímku, který vzniká v libovolném výše popsaném skriptu a vypadá například takto:

14330079787306(odpovídá datu 30. 5. 2015, 19.46:18,7306)

3.8. Konfigurace systému

K funkci skriptů je nezbytná podpora jazyka PHP a podpora volání Imagicku. Ke správné funkci skriptu *video2image* jsou rovněž nezbytné správné konfigurace nástroje ffmpeg a cesty k balíku PHPVideoToolkit. Bez této podpory nemusí některé skripty fungovat.

4. Testování

V průběhu jednotlivých fází vývoje i na samém závěru byly skripty podrobeny několikerému testování. Některé testy lze nazvat experimentálními, jiné funkčními.

4.1. Experimentální testy

Za experimentální testy považujme takové testy, které byly použity k určení vhodného nastavení, např. v otázce rozložení snímku (layoutu), velikosti fontu z hlediska čitelnosti i objemu, polohy obrázku apod. Můžeme říci, že tyto testy souvisely s grafickým nastavením snímků. V průběhu testování byly ve větší míře potvrzeny původní předpoklady, ovšem v několika případech bylo prokázáno chybné rozložení snímků či objektů v nich použitých. Závěrem těchto ukazatelů byla změna konfigurace grafického uspořádání.

U skriptu *webpage2image* bylo odhaleno, že při zvoleném fontu nelze URL adresu umístit do pravého sloupce layoutu, který je pro delší adresy příliš úzký. Z tohoto důvodu bylo přistoupeno ke zkracování adresy webové stránky při překročení délky 50 znaků, i k menší změně layoutu, která neměla žádný dopad na vkládaný obrázek. Zmíněný nedostatek a jeho odstranění se promítlo i do skriptů *webcam2image* a *pdf2image*, neboť změna skriptu proběhla ve funkci *image*, která je pro skripty společná.

Rovněž umisťování textu do uživatelského snímku vedlo ke změně rozložení snímku (zmenšení prostoru pro nadpis, zvětšení textové oblasti), dále k úpravě velikosti písma použitého pro nadpis (zvětšení) a také omezení množství vloženého textu na 25 řádků (zmenšení).

Ostatní nastavení parametrů skriptů bylo při testování potvrzeno jako vhodné. Například v otázce konstanty maximálního zvětšení byl potvrzen původní návrh stanovit pro tuto konstantu hodnotu 2. Při větší hodnotě již docházelo k příliš viditelným artefaktům, které jsou i při této konstantě na hraničních hodnotách.

4.2. Funkční testy

Funkční testy byly určeny k ověření činnosti skriptů a bylo při nich odhaleno několik nedostatků, které byly ve většině odstraněny.

Při testování skriptu *webpage2image* byla odhalena nesprávná kontrola zadaných parametrů ořezu, čímž docházelo k neočekávaným výsledkům při manipulaci se sejmutým obrázkem webu. Rovněž bylo zjištěno, že zadaný časový limit (neboli timeout) pro získání snímku webu není zcela vždy dostačující, a to v takovém procentu případů (cca 10%), které bylo považováno za příliš vysoké. Kontrola hodnot ořezu byla upravena, v současnosti dochází k prvotní kontrole již ve formuláři, do něhož se parametry ořezu zadávají, druhá kontrola probíhá pomocí příkazů skriptu, které chybně zadané hodnoty upraví. Druhá fáze kontroly je zařazena z důvodu neúplné kontroly formuláře, k níž

by bylo zapotřebí Javascriptu, který by kladl další nároky na systém, na němž skripty poběží. Timeout skriptu byl z původních 2 minut navýšen na 3 minuty.

Během testování *webcam2image* nebyly odhaleny žádné funkční nedostatky, což je přičítáno vývoji z již testovaného a odladěného skriptu *webpage2image*.

Testování *pdf2image* odhalilo nedostatek ve funkci *image*, který se nepodařilo objevit při testování skriptu pro pořizování snímků webu. Tento fakt, týkající se nesprávného výpočtu poměrů obrázku při volbě layoutu je přičítán parametrům získaných snímků webu z API Page2Image. Výpočty byly opraveny, aby nadále nedocházelo k překvapivému použití neočekávaného layoutu v některých případech. Podobně jako u skriptu určeného ke snímání webových stránek byl v několika situacích zjištěn nedostačující timeout (definovaný nastavením PHP, které může být v závislosti na instalaci proměnné), který byl nově stanoven na 3 minuty.

V průběhu testů skriptu *text2image* byla objevena chyba ve výpočtu velikosti snímku. Tato chyba, projevující se v případě použití volby orámování snímku nárůstem jeho rozlišení o dvojnásobnou hodnotu šířky rámečku, byla dána nesprávným použitím funkce *borderImage* třídy *Imagick*, která, jak se ukázalo, nevkládá rámeček do obrázku, ale k obrázku jej přidává. V souvislosti s tím byl do začátku skriptu implementován přepočet velikosti snímku v závislosti na zvolené síle rámečku.

Testování *video2image* odhalilo chybu v nástroji PHPVideoToolkit (více o této chybě v kapitole Implementace). Na základě této chyby bylo hledáno alternativní řešení, které bylo implementováno jako dočasné. Z testů rovněž vyplynula chybná implementace skriptu *thumbnails* při výpočtu časové konstanty získávaného snímku, kdy u krátkých video souborů docházelo k výpočtům hodnot mimo čas videa. Tato chyba byla opravena, stejně jako chyba v části skriptu, který zajišťuje mazání souborů z dočasné složky. Zde bylo problém s oprávněním k přístupu k souboru, což bylo vyřešeno jednoduchým příkazem k úpravě oprávnění. Jiné závady nebyly v průběhu testování odhaleny.

Testování bylo provedeno pod systémy Windows i UNIX.

4.3. Závěrečné testování - snímky

Při závěrečných testech skriptů bylo pořízeno několik obrázků, na nichž jsou zaznamenány možné podoby výstupních snímků určených k prezentaci na LCD panelech o rozlišení 1920x1080 pixelů. Snímky, které jsou umístěny v příloze A, jsou doplněny popsaným nastavením, aby bylo patrné, jak bylo snímků dosaženo.

Závěr

Naplnění většiny požadavků probíhalo bez zásadních obtíží, některé požadavky přinesly větší či menší komplikace. Nejnáročnější částí realizace požadavků byla analýza, jejíž součástí je i popis nástrojů, které je možné ke splnění konkrétních požadavků využít. Jedná se o nástroje volně dostupné i placené. Tato část analýzy přináší velkou přidanou hodnotu a zmíněné **nástroje lze použít i pro mnohé jiné aplikace** v oblasti vývoje webů (i pro mobilní zařízení), zpracování obrázků, videa, ale i audia. Rovněž **informace o artefaktech a filtrech** (jsou součástí nástroje ImageMagick) mohou být **nápomocné při vývoji jiných aplikací** pracujících s obrázky, animacemi a videem.

Implementace požadavků, při níž byla vyvinuta sada skriptů, vycházela z návrhu postaveného na analýze. Ve velké míře se **podařilo návrh** do skriptů **implementovat.** Řešení požadavku č. 3 se v průběhu vývoje (vlivem chyby v knihovně nástroje PHPVideoToolkit) ukázalo jako dosti problematické. Přesto, že chyba byla komunikována s autorem knihoven, nepodařilo se ji autorovi PHPVideoToolkitu do doby zakončení této práce odstranit. Experimentálně **byla nalezena možná cesta** k realizaci požadavku, která byla jako dočasná **alternativa implementována**. Jakmile však bude chyba v balíku PHPVideoToolkit odstraněna, je doporučeno skript *thumbnails* , který získává z videa obrázky, upravit, a to dle doporučené funkce pro získávání náhledů videa.

Další komplikace, které vyplynuly již z analýzy, nastaly při realizaci požadavků 8 a 9. U požadavku 8 sice dochází vlivem konfigurace skriptů k řazení snímků do fronty, a to dle abecední (či v tomto případě spíše numerické) posloupnosti, pořadí ve frontě ale nelze ovlivnit jinak, než hierarchickým vkládáním snímků. Rovněž požadavek 9 (vyžadující umožnění zadání priority snímku) je splněn jen částečně. V tomto případě se jedná pouze o návrh, jak lze prioritu snímků ovlivnit za současného nastavení. Obě omezení při naplnění těchto požadavků jsou zapříčiněna konfigurací mikropočítače Raspberry a prohlížečem na něm použitým k zobrazování snímků na LCD panelech. Je proto **doporučeno realizovat** tyto **požadavky do budoucna změnou konfigurace mikropočítače a** především **jiným**, ideálně na míru implementovaným **prohlížečem.**

Přes zmíněné nesnáze se nakonec **podařilo realizovat všechny požadavky na tvorbu snímků.** Z ostatních požadavků zůstal neimplementován pouze **požadavek č. 9**, který **nebylo možné na straně serveru realizovat.**

Seznam literatury a použitých zdrojů

[1.] GUTMANS, Andi, Stig Sæther BAKKEN a Derick RETHANS. *Mistrovství v PHP 5*. Vyd. 1. Brno: CP Books, 2005, 655 s. ISBN 80-251-0799-X.

[2.] IMAGEMAGICK STUDIO LLC. *ImageMagick* [online]. 1999 [cit. 2015-06-06]. Dostupné z: http://www.imagemagick.org/script/index.php

[3.] RICHARDSON, Leonard. CRUMMY. *Beautiful Soup* [online]. 1996, 27. 5. 2014 [cit. 2015-06-06]. Dostupné z: http://www.crummy.com/software/BeautifulSoup/

[4.] BELLARD, Fabrice. *FFmpeg* [online]. 2007, 16. 3. 2015 [cit. 2015-06-06]. Dostupné z: https://www.ffmpeg.org/

[5.] PHP FFmpeg. *GitHub* [online]. 2012 [cit. 2015-06-07]. Dostupné z: https://github.com/PHP-FFMpeg/PHP-FFMpeg

[6.] LILLIE, Oliver. PHPVideoToolkit V2. *GitHub* [online]. 2013 [cit. 2015-06-07]. Dostupné z: https://github.com/buggedcom/phpvideotoolkit-v2

[7.] BRIGHTCOVE INC. *Zencoder: Cloud Video Encoding* [online]. 2013 [cit. 2015-06-07]. Dostupné z: https://zencoder.com/en/

[8.] NASATO, Mirko. JODConverter. NASATO, Mirko. *Artofsolving* [online]. 2010 [cit. 2015-06-07]. Dostupné z: http://www.artofsolving.com/opensource/jodconverter.html

[9.] ASPOSE. *Aspose* [online]. 2002 [cit. 2015-06-07]. Dostupné z: http://www.aspose.com/

[10.] TRUELSEN, Jakob a KULKARNI. *Wkhtmltopdf* [online]. [2014] [cit. 2015-06-07]. Dostupné z: http://wkhtmltopdf.org/index.html

[11.] VON HERTZEN, Niklas. *Html2canvas* [online]. [2013] [cit. 2015-06-07]. Dostupné z: http://html2canvas.hertzen.com/

[12.] KÄRRMAN, Jan. UPPSALA UNIVERSITY. *Html2canvas* [online]. [2012], 16. 3. 2013 [cit. 2015-06-07]. Dostupné z: http://user.it.uu.se/~jan/html2ps.html

[13.] Html2ps and html2pdf. *Darren's Script Archive* [online]. 2005, 6. 9. 2010 [cit. 2015-06-07]. Dostupné z: http://www.tufat.com/s_html2ps_html2pdf.htm

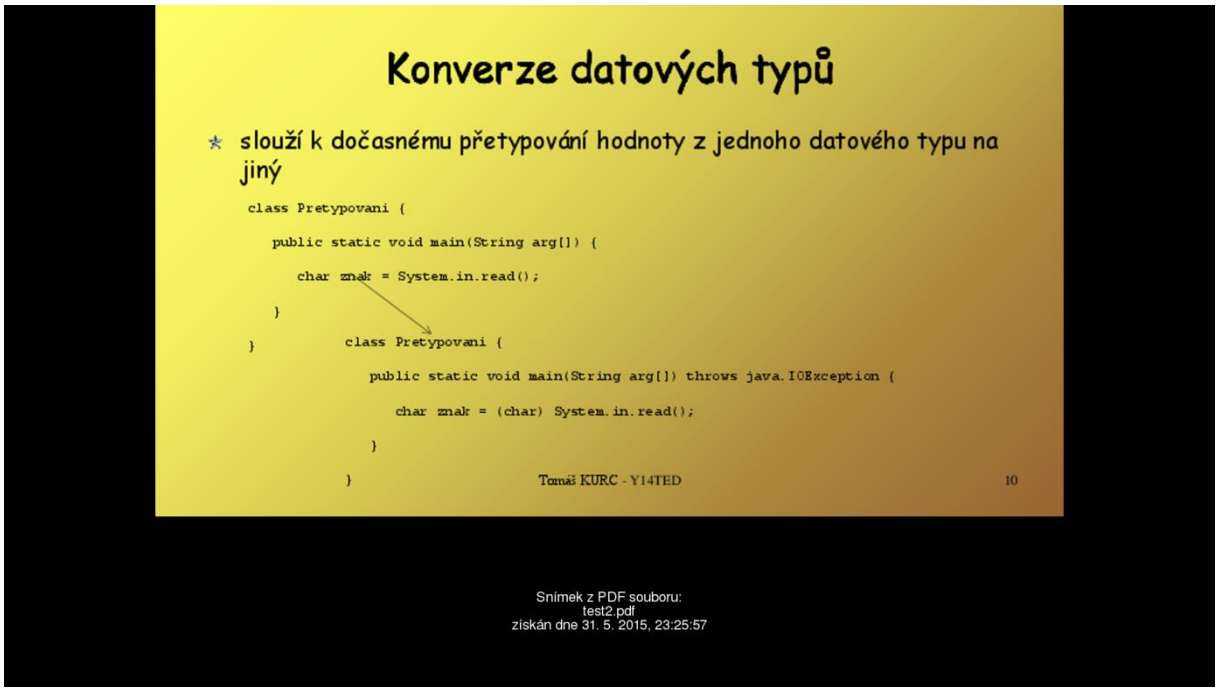
[14.] SUZHOU KADA INTERNET TECHNOLOGY CO., LTD. *Page2Images* [online]. 2013 [cit. 2015-06-07]. Dostupné z: http://www.page2images.com/

[15.] PHP GROUP. *PHP: Hypertext Preprocessor* [online]. 2201 [cit. 2015-06-07]. Dostupné z: http://uk3.php.net

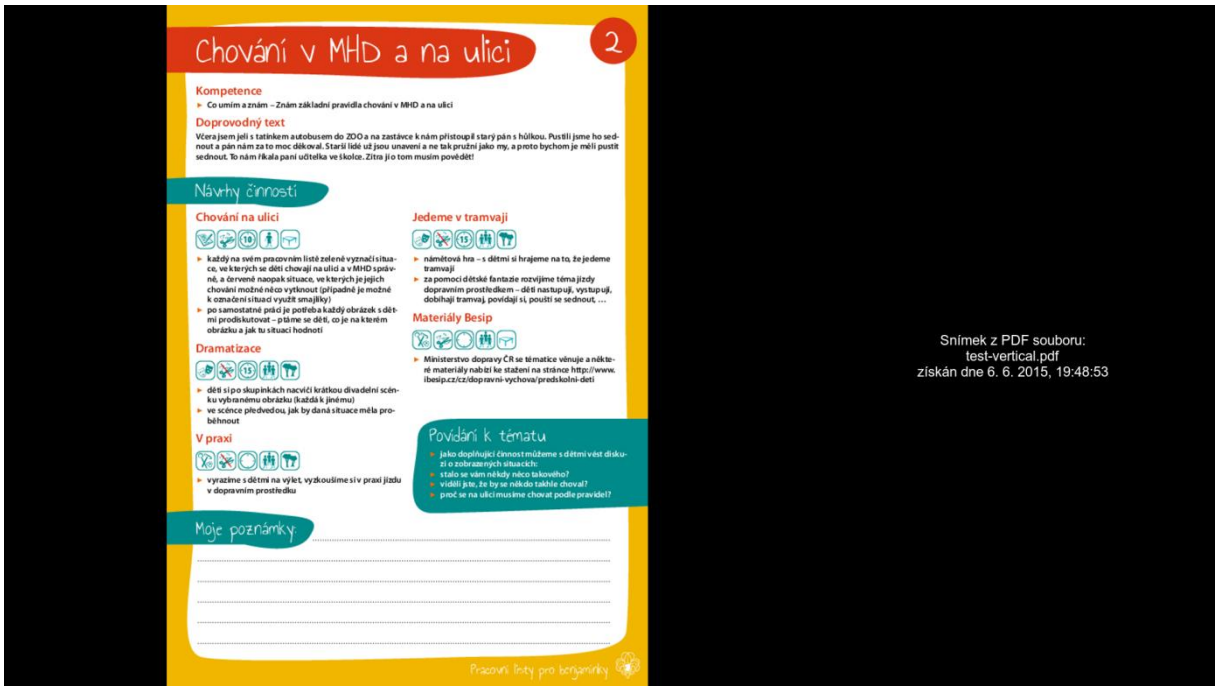
Přílohy

Příloha A – Snímky pořízené při testování

Skript pdf2image

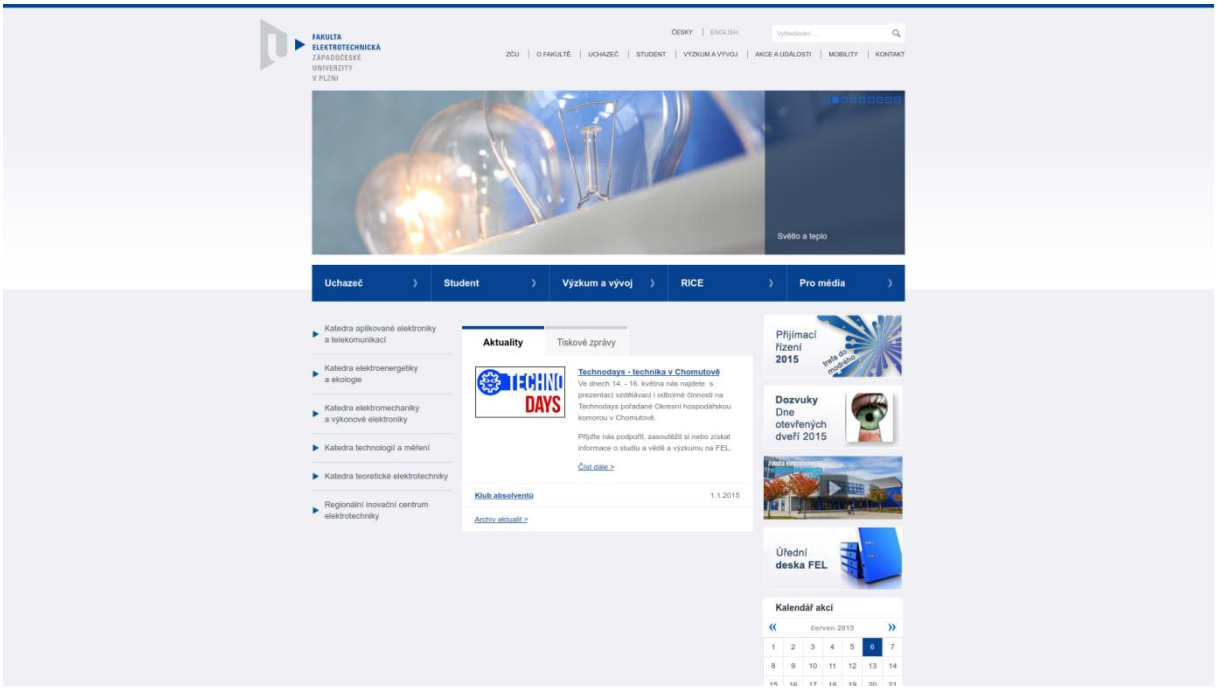


obr. A1 Snímek z PDF souboru test-horizontal.pdf (na disku CD)

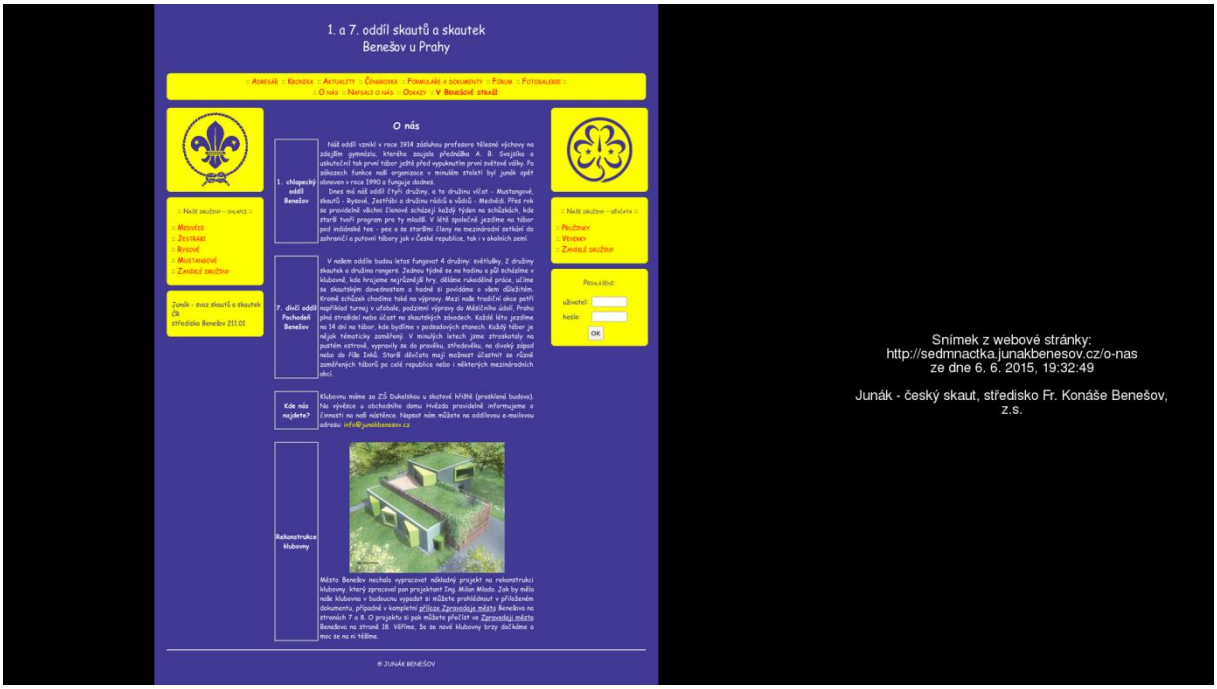


obr. A2 Snímek z PDF souboru test-vertical.pdf (na disku CD)

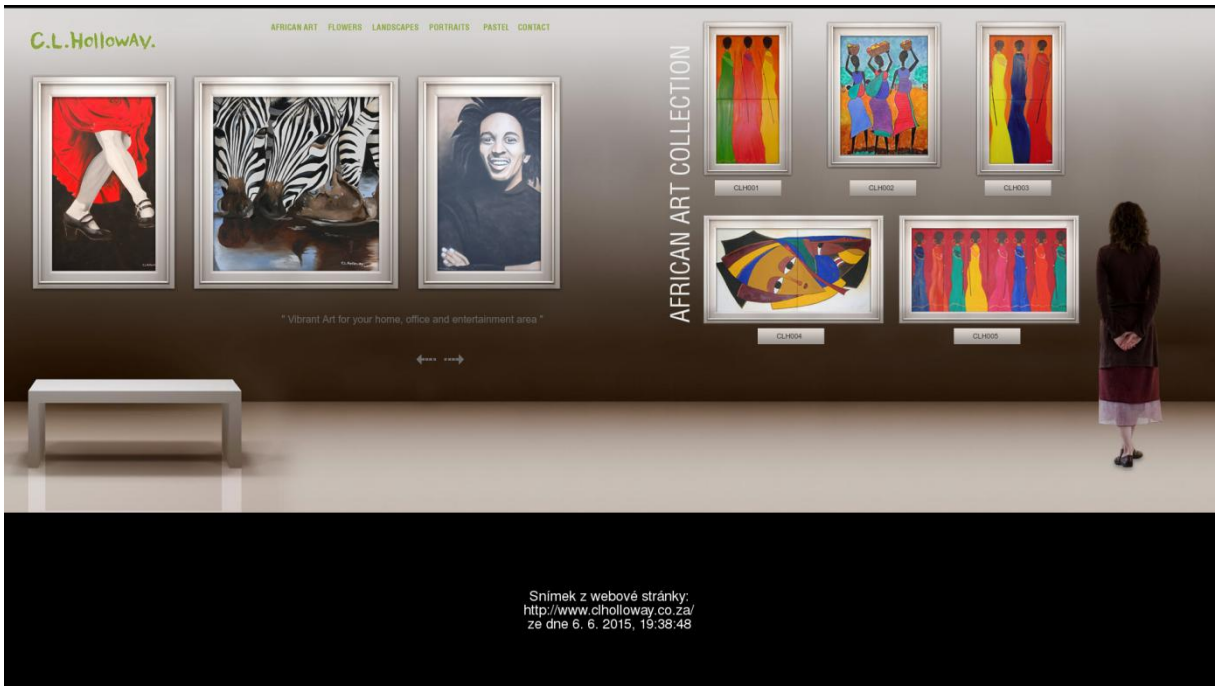
Skript webpage2image



obr. A3 Snímek webu www.fel.zcu.cz s ořezem obrázku na poměr stran 16:9.

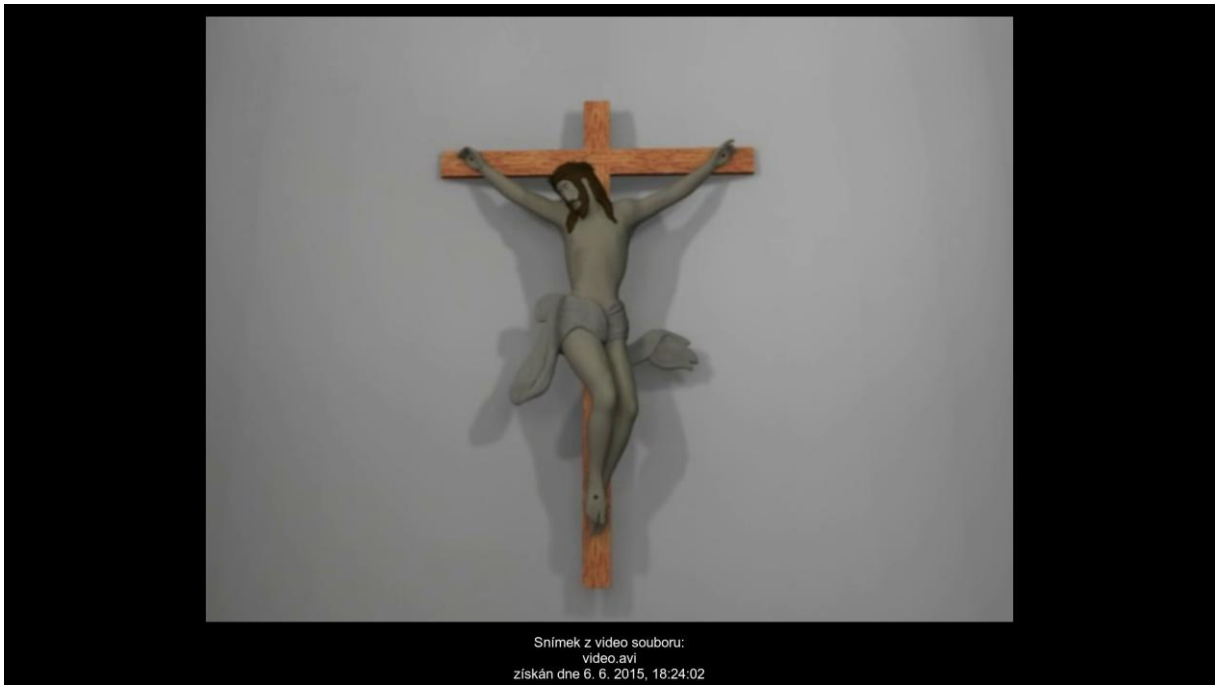


obr. A4 Snímek webu http://sedmnactka.junakbenesov.cz/o-nas s ořezem X 450, Y 0, šířka 1080, výška 0. Do komentáře přidán název organizace.



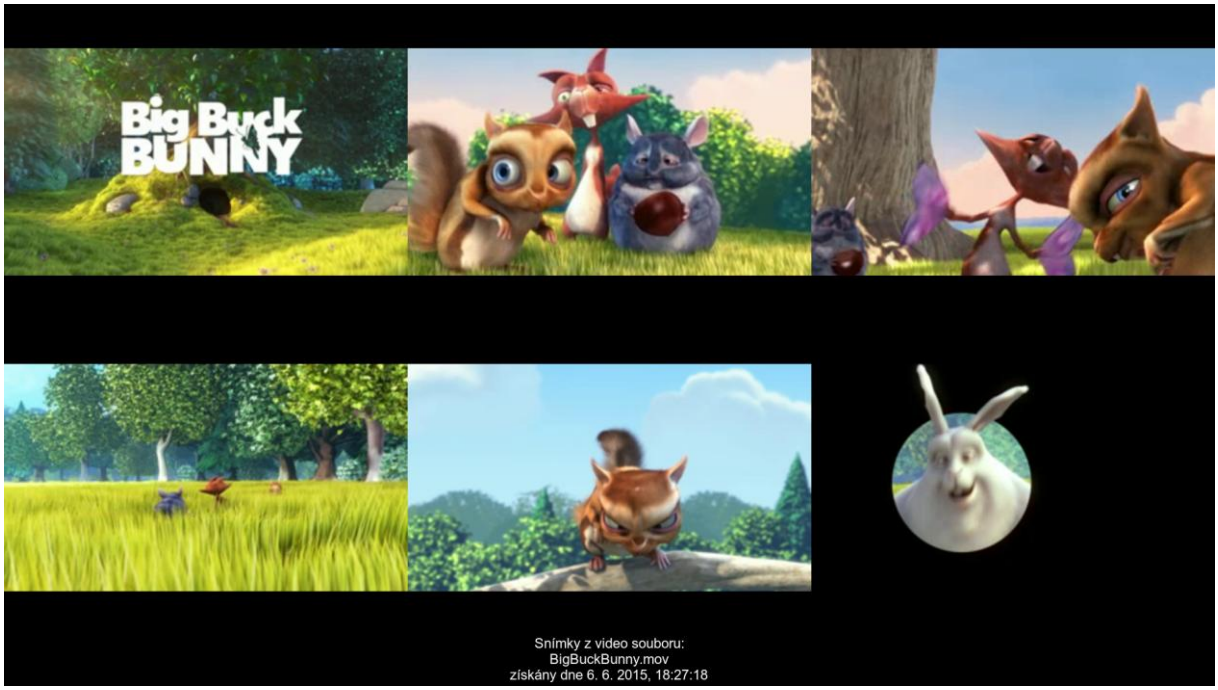
obr. A5 Snímek webu http://www.clholloway.co.za s ořezem X 0, Y 0, šířka 0, výška 800.

Skript video2image



obr. A6 Snímek z video souboru video.avi získaného z webu http://is.muni.cz/th/256007/fi_b/video.avi?info (závěrečná bakalářská práce studenta Tomáše Slouky). Layout s jedním obrázkem.

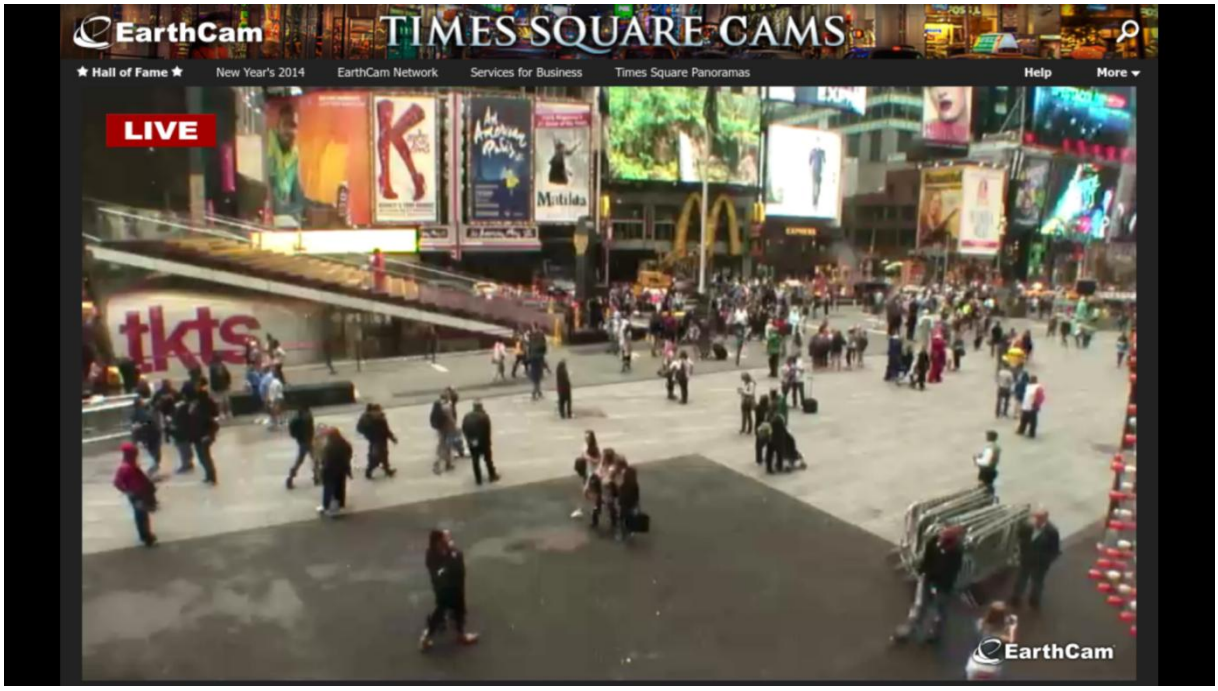
obr. A7 Snímek z video souboru BigBuckBunny (součástí balíku PHPVideoToolkit) zkráceného o úvodní a závěrečnou část a překonvertovaného z MP4 do MPEG II. Layout se čtyřmi obrázky.



obr. A8 Snímek z video souboru BigBuckBunny (součástí balíku PHPVideoToolkit) zkráceného o úvodní a závěrečnou část a překonvertovaného z MP4 do MOV. Layout se šesti obrázky.

obr. A9 Snímek z video souboru BigBuckBunny (součástí balíku PHPVideoToolkit) zkráceného o úvodní a závěrečnou část. Layout s devíti obrázky.

Skript webpage2image



obr. A10 Snímek z webu http://www.earthcam.com/usa/newyork/timessquare/?cam=tsstreet s live kamerou (stream) umístěnou na Times Square, New York. Ořez obrázku X 460, Y 0, šířka 1000, výška 620.

Skript text2image

obr. A11 Snímek s textem ze stránky http://ffmpeg.zeranoe.com/faq. Barva pozadí snímku defaultní, nadpis světle modrý, text tmavě modrý, červený rámeček tloušťky 10 pixelů.

Příloha B – Obsah CD

Na přiloženém CD je k dispozici:

- Text práce ve formátu PDF,
- sada skriptů naplňující požadavky této práce,
- snímky získané při závěrečném testování skriptů (v plném rozlišení),
- testovací soubory.