VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Autoclicker s rozpoznáváním obsahu obrazovky

Autoclicker with Screen Content Recognition

2018 **Jan Křístek**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jan Křístek**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Autoclicker s rozpoznáváním obsahu obrazovky
Autoclicker with Screen Content Recognition

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem bakalářské práce je vytvořit aplikaci, která simuluje činnost uživatele (posouvání a klikání myší, stisk kláves) na základě obsahu obrazovky. Uživatel aplikace bude mít možnost zvolit, jaká automatizovaná akce se provede, na základě konkrétní podoby určité části obrazovky (např. objeví-li se na určitém místě obrazovky konkrétní obrázek, aplikace na něj klikne).

Bakalářská práce musí splňovat následující body:
1. Stručný přehled metod rozpoznávání obrazu.
2. Přehled aktuálně používaných řešení autoclickeru.
3. Výběr metody vhodné k řešení problému a zdůvodnění volby konkrétní metody.
4. Popis a implementace zvolené metody.
5. Vyhodnocení výkonnosti řešení.

Seznam doporučené odborné literatury:

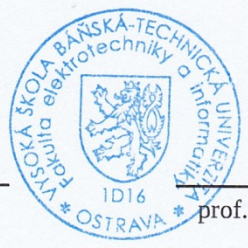
[1] Christopher M. Bishop. Pattern Recognition and Machine Learning, 2006
[2] Stuart J. Russell and Peter Norvig. Artificial Intelligence: A Modern Approach. Pearson Education, 2003.

Dále podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Dohnálek**

Datum zadání: 01.09.2017
Datum odevzdání: 13.07.2018



doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě – Porubě dne 2. 7. 2018

.....

Podpis

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce Ing. Pavlu Dohnálkovi za odbornou pomoc a konzultaci při vytváření této práce.

Abstrakt

Účelem této bakalářské práce je vytvořit aplikaci autoclicker. Autoclicker umí standardně simulovat činnost uživatele v podobě klikání na určitá místa na obrazovce a zadat text na klávesnici. Aplikace podporuje uložení pozic myši a textu do souboru. Hlavní funkce této aplikace je možnost vyhledávat na obrazovce určité vzory a po jejich nalezení spustit odpovídající skript.

Klíčová slova

klikání, obrazovka

Abstract

The purpose of this bachelor thesis is to develop software autoclicker application. Autoclicker can simulate actions done by user as clicking on certain points on the screen and typing text on keyboard. Appliaction supports saving of mouse locations and text into a file. Main function of this application is a possibility of finding certain patterns and after they are found start certain script.

Key words

clicking, screen

Obsah

Seznam použitých symbolů a zkratek

DC	obsah zařízení (device content)
GUI	grafické uživatelské rozhraní (Graphical User Interface)
CV	počítačové vidění (Computer Vision)

Seznam ilustrací a tabulek

1 Úvod

V této bakalářské práci bude přiblížen popis bakalářské aplikace, zpracování obrazu a přiblížení případných konkurentů na trhu. Popis bakalářské aplikace je rozdělen na popis uživatelského rozhraní, funkcionality a implementace. Mým cílem bylo vytvořit aplikaci, která bude splňovat následující body:

- Bude integrovat funkčnost požadovanou na základě zadání bakalářské práce, která bude moci konkurovat jiným softwarovým řešením dostupným na trhu.
- Uživatelské rozhraní bude pro uživatele co nejvíce intuitivní a přátelské.
- Aplikaci by šlo nadále rozvíjet v případě pokračování studia v navazujícím magisterském studiu.

V kapitole 2 budou představeny jednotlivé konkurenční programy. U každého je uveden krátký základní popis a jeho porovnání s bakalářskou aplikací.

V kapitole 3 jsou rozebrány různé možnosti, které máme pro zpracování obrazu, a dále knihovny, které lze použít pro vytváření bakalářské aplikace. Podrobněji zde budou rozebrány použité důležité části Open CV knihovny. Kapitola také obsahuje popis testů na testovacím projektu.

V kapitole 4 je uveden popis samotné bakalářské aplikace. Bude zde uvedena struktura programu, její funkčnost a použitá knihovna Windows Forms. Podrobně zde bude rozebráno uživatelské rozhraní a části zdrojového kódu.

Závěrečná část práce obsahuje závěrečné hodnocení.

2 Konkurenční programy

Na trhu se nacházejí podobné programy různé kvality, ať už se jedná čistě o autoclicker, nebo autoclicker s rozpoznáváním obrazu. V této části budou uvedeny jednotlivé konkurenční programy, jejich krátký popis, porovnání s bakalářskou aplikací. Konkurence v oblasti samotného autoclickeru je poměrně velká, ale v oblasti autoclickerů s rozpoznáváním obrazu je konkurence malá.

Auto Mouse Click je součástí balíčku programů, které jsou k dispozici na stránce autora murgee, kde jsou k dispozici i další užitečné programy. Auto Mouse Click je dobře zpracovaný autoclicker, který obsahuje jednoduše vytvořenou funkcionalitu rozpoznávání obrazu. Po jeho nalezení ale umí jen na nalezenou pozici kliknout. Tabulka uprostřed nabízí velmi přehledný seznam akcí, které se mají vykonat.

Auto Mouse Click je značně velký konkurent bakalářské aplikace, jeho výhodou jsou větší možnosti výběru a nastavení akcí. Jedná se o velmi komplexní program. Jeho nevýhodou oproti bakalářské aplikaci je, že v případě vyhledání programu umí na nalezený obrázek jen kliknout, neumí načíst skript s příkazy a nenabízí možnost vytvoření vyhledávaných obrázků.

AC Auto Cliker a AC Picture Clicker jsou součástí balíčku programů, jež lze nalézt na stránce autora autoclickersoft. AC Auto Clicker se specializuje na klikání na aktuální pozici myši a na vypsanou fixní pozici myši. AC Picture Clicker se specializuje na hledání obrázku na ploše a na klikání na nalezené místo. Zjednodušeně bakalářská aplikace představuje spojení těchto dvou programů do jednoho celku.

Free Mouse Auto Cliker je jednoduchý a minimalistický autoclicker, který umožňuje jen klikat na aktuální pozici myši, a nenabízí možnost rozpoznávání obrazu. Je určen pro uživatele, který chce základní jednoduše nastavit automatizované klikání.

Free Auto Clicker je dalším příkladem jednoduchého autoclickeru, který umí zajistit klikání na obrazovku, ale neumí psát virtuálně na klávesnici a rozpoznávat obraz. Stiskem klávesy mezerník se do programu postupně ukládají pozice myši, které se uloží do tabulky v pravé části programu.

AutoHotkey je skriptovací jazyk pro systém Windows, který může splnit funkčnost autoclickeru. Oproti ostatním autoclickerům je netradiční absencí uživatelského rozhraní. Uživatel vytváří v textovém souboru skripty, který může spustit, nebo se spustí po zadání nastavené kombinace stisknutých kláves. Umožňuje uživateli například automaticky nahradit text jiným, nastavit klávesové zkratky pro spouštění programů a nabízí uživateli možnost vyhledat na ploše obrázek a kliknout na něj.

3 Možnosti rozpoznávání obrazu

[1], [5] Cílem rozpoznávání obrazu neboli segmentace je nalezení objektů v obrázku, a jejich rozlišení od pozadí. Kompletní segmentace nastane v případě nalezení oblastí, které se nepřekrývají a odpovídají reálným objektům. Částečná segmentace je založená na homogenitě oblastí.

Pro práci s obrazem máme následující možnosti

- Rozpoznávání obrazu založené na neuronových sítích.
- Rozpoznávání obrazu založené na template matchingu (párování vzorů).
- Rozpoznávání obrazu založené na rozpoznávání hran.
- Rozpoznávání obrazu založené na prahování.
- Rozpoznávání obrazu založené na spojování oblastí.
- Rozpoznávání obrazu založené na dělení oblasti.

Rozpoznávání na základě neuronových sítí je založené na strojovém učení. Počítač naučíme, jak jednotlivé objekty vypadají, a postupným opakováním bude umět lépe rozpoznávat dané objekty. Počítači budou postupně prezentovány obrázky se štítky, které počítač zanalyzuje. Opakováním se bude zvyšovat přesnost.

Template matching je založen na zjištění, jestli se námi určený vzor nachází v zadaných datech, jež jsou větší než vzor. Vzor postupně přesouváme po datech a vypočítáváme chybu shody mezi vzorem a daty. Menší chyba znamená větší shodu. Výpočet záleží na zvolené porovnávací metodě.

Rozpoznávání na základě hran pracuje na principu změn jasu na hranách. Metody detekce hran jsou založeny na první a druhé derivaci. V oblasti se změnou jasu se nachází lokální maximální hodnota první derivace a druhá derivace má hodnoty blízko nule. Obraz se následně konvoluje s vhodnou maskou pro obě derivace.

Prahování je nejjednodušší formou segmentace. Na základě prahu rozlišíme, jestli daný bod náleží do pozadí, nebo do popředí. Výsledkem je dvouúrovňový obraz, ve kterém se nachází pozadí a popředí, která představují námi hledaný objekt. Správné určení prahu je klíčové pro úspěch této metody.

Spojování oblastí na začátku rozdělí obraz na menší části, mohou to být až samotné pixely. Jednotlivé oblasti se postupně spojují na základě jasu, nebo barvy. Spojování se opakuje, dokud nelze spojit další části.

Dělení oblasti má opačný postup než jejich spojování. Homogenní obraz se postupně rozkládá na menší části.

Pro bakalářskou aplikaci jsem se rozhodl využít funkci založenou na template matchingu. Neuronové sítě by při své vyšší složitosti nejspíše měly horší výkon. Počítač by bylo potřeba naučit velmi malé rozdíly mezi ikonami na ploše a nemyslím si, že neuronové sítě jsou na tento úkol dostatečně přesné. Mohlo by docházet k velkému množství špatných nálezů. Template matching možná výkonově vyjde hůře, ale dokáže zajistit větší přesnost. Pro uživatele aplikace je lepší přesně určit vzor, který chce vyhledávat, než počítač učit, jak vypadá ikona MS Wordu a v čem se liší od podobné ikony PSPadu. Prahování, spojování oblasti a dělení oblasti se díky své jednoduchosti nehodí na rozpoznávání vzorů ve snímku obrazovky. Rozpoznávání vzorů by bylo velmi nepřesné.

3.1 Open CV

Open CV je volně dostupná knihovna pro práci s obrazem v jazycích C/C++, Python a Java. Podporuje zařízení s operačním systémem Windows, Linux, Mac, iOS a Android. Open CV byla vytvořena se zaměřením na real-time aplikace. Vydávána je pod licencí BSD.

Obrázek 1: logo Open CV (zdroj: https://en.wikipedia.org)

3.1.1 Zpracování obrazu s pomocí Open CV

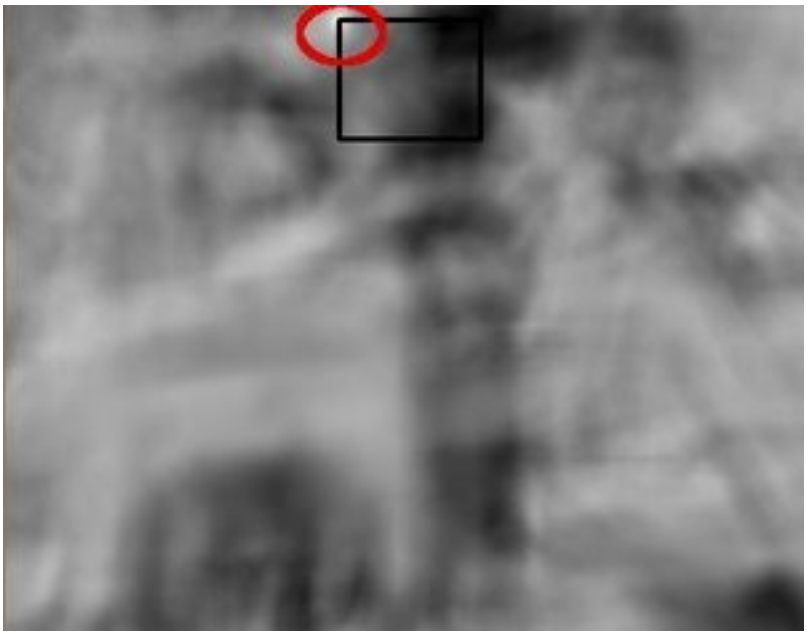
[3], [4] Knihovna Open CV pro zpracování obrazu nabízí funkci matchTemplate, která umožňuje vyhledat, jestli se v obrázku nachází vybraný vzorový obrázek. Funkce funguje na principu posouvání obrázku od levého horního rohu až k pravému dolnímu rohu (obrázek 3). Vzorový obrázek se postupně posouvá a na každém místě se vypočítá kvalita shody, která se zanese do výsledné matice. Na obrázku 4 jsou vidět výsledky porovnání. Tmavé oblasti značí místa s malou shodou a světlejší oblasti označují místa s velkou shodou. Vstupní parametry pro funkci jsou

- Zdrojový obrázek, ve kterém se bude vyhledávat.
- Vzorový obrázek, který vyhledáváme. Musí být větší než obrázek, ve kterém vyhledáváme.
- Matice, do které se uloží výsledky.
- Číslo některé z porovnávacích metod, které budou popsány dále.

Zdrojový obrázek má velikost W a H, kde W je šířka a H je výška. Vzorový obrázek má velikost w a h, kde w je šířka a h je výška. Výsledná matice má velikost (W-w+1) pro šířku a (H-h+1) pro výšku.

Obrázek 2: Princip rozpoznávání obrazu (zdroj: https://docs.opencv.org)

Obrázky 2 a 3 jsou staženy z dokumentace Open CV.



Obrázek 3: Výsledek porovnání obrazu (zdroj: https://docs.opencv.org)

Open CV nabízí celkem šest porovnávacích metod, které slouží pro porovnání vzorového a zdrojového obrázku. Výpočet se provádí po každém posunutí vzorového obrázku nad zdrojovým.

- I představuje zdrojový obrázek.
- T vzorový obrázek.
- R představuje výslednou matici.
- x´ odpovídá 0 až w-1.
- y´ odpovídá 0 až h-1.
- x, y jsou souřadnice.

Následující vzorce jsem čerpal z [4].

Square difference matching methods: Suma čtverců rozdílu mezi hodnotou pixelů ve vzoru a zdroji.

Metoda 1 TM_SQDIFF

$$R(x,y) = \sum_{x',y'} T(x',y') - I(x+x',y+y'))^2$$

Metoda 2 TM_SQDIFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'}(T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 * \sum_{x',y'} I(x+x',y+y')^2}}$$

Correlation matching methods: Suma čtverců korelace pixelů ve vzoru a zdroji

Metoda 3 TM_CCORR

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x', y+y'))$$

Metoda 4 TM_CCORR_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x', y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 * \sum_{x',y'} I(x+x', y+y')^2}}$$

Correlation coeff. matching methods: Suma čtverců korelačních koeficientů pixelů ve vzoru a zdroji

Metoda 5 TM_CCOEFF

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I'(x+x', y+y'))$$

kde

$$T'(x',y') = T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'')$$

$$I'(x+x', y+y') = I(x+x', y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x+x'', y+y'')$$

Metoda 6 TM_CCOEFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x', y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 * \sum_{x',y'} I'(x+x', y+y')^2}}$$

Výsledná matice na obrázku 4 je dělána s pomocí metody TM_CCORR_NORMED, pro kterou nejsvětlejší pozice znamená pozici s nejvyšší shodou.

Přesnější výsledky dostaneme používáním sofistikovanějších metod, a to od TM_ SQDIFF k TM_CCOEFF, která je nejsofistikovanější. Sofistikovanější nadřazují přesnost nad rychlost zpracování.

Pro další zpracování se použije funkce minMaxLoc, která vyhledá pozici s nejvyšší a nejnižší shodou. Porovnávací metody CV_SQDIFF a CV_SQDIFF_NORMED mají nejlepší shodu na nejnižších hodnotách, protože počítají rozdíl v podobnosti. Čím menší je rozdíl, tím větší je shoda. TM_CCORR, TM_CCORR _NORMED, TM_CCOEFF a TM_CCOEFF_NORMED na nejvyšších hodnotách, protože počítají podobnost. Čím větší je podobnost, tím větší je shoda. Funkce minMaxLoc má následující vstupní parametry:

- Pole hodnot, ve kterém chceme vyhledávat.
- Ukazatel na číslo typu double, do kterého se uloží nalezená minimální hodnota.
- Ukazatel na číslo typu double, do kterého se uloží nalezená maximální hodnota.
- Ukazatel na bod, do kterého se uloží nalezený bod s minimální hodnotou.
- Ukazatel na bod, do kterého se uloží nalezený bod s maximální hodnotou.
- Maska.

Pro testování template matchingu byl vytvořen speciální projekt ve vývojovém prostředí CodeBlocks. Koláž (obrázek 4), v níž budu vyhledávat vzor, jsem sestavil z náhodných obrázků nalezených přes vyhledávač Google. Vybíral jsem barevné obrázky, protože pro bakalářskou aplikaci jsem plánoval použití barevných obrázků a snažil jsem se vybírat tematicky a barevně rozmanité obrázky.

Obrázek 4: Koláž před hledáním

Na levé straně se pod sebou nachází dva obrázky, to samé platí pro pravou stranu. V koláži je nejvýraznější velký obrázek uprostřed. Zbytek pozadí byl vyplněn červenou barvou. Jako vzor byl vybrán obrázek vpravo nahoře. Po nalezení daného vzoru v koláži se okolo místa nalezení nakreslí modrý obdélník, nebo čtverec. Záleží na tvaru vyhledávaného vzoru.

Na obrázku 5 je poznat, že program našel pozici vzoru správně. Pro nalezení místa použil dvě porovnávací metody. Výsledné hodnoty po vyhledávání musely být pro minimum menší než 0.03 a pro maximum vyšší než 0.97. Úplné minimum je 0 a maximum 1. Kritéria jsou nastavená velmi vysoko, aby se co nejvíce eliminovala možnost nesprávného výsledku.

Obrázek 5: Koláž po nalezení

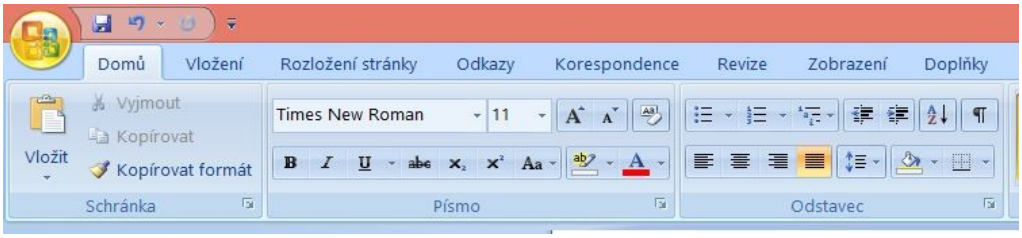
V tabulce jsou uvedeny naměřené časy pro všechny metody. Normované verze používat nemohu, protože ztrácí informace o kvalitě nalezených míst. Pro zjištění časů jsem aplikaci desetkrát spustil. Časy se pohybují ve stejných rozmezích pro všechny tři metody. První tři zapnutí byly časy v horní

hranici, dalším spouštěním se pohybovaly blíže dolní hranici. Časové rozdíly mezi jednotlivými metodami jsou zanedbatelné. Naměřená přesnost se neměnila. TM_CCORR je nejpřesnější, následuje TM_SQDIFF a poslední je TM_CCOEFF.

Tabulka 1: Porovnání metod v template matchingu

Porovnávací metody	Naměřené časy v sekundách	Naměřená přesnost	Doplněk do minima/maxima
TM_SQDIFF	0,39 až 0,43	0,0396004	0,0396004
TM_CCORR	0,39 až 0,43	0,980206	0,019794
TM_CCOEFF	0,39 až 0,43	0,935135	0,064865

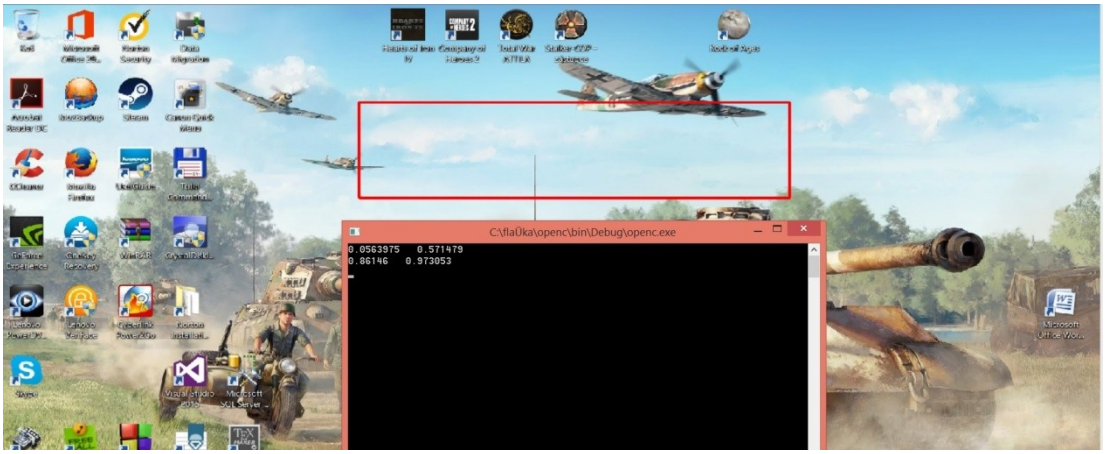
Aplikace používá porovnávací metodu TM_SQDIFF, která má v tabulce horší výsledek, než metoda TM_CCORR. Když jsem v bakalářské aplikaci testoval použití TM_CCORR samotné, dostalo se mi nesprávných nálezů. Rozhodl jsem se speciální program upravit pro otestování nesprávných nálezů. Na snímku obrazovky jsem nechal vyhledat část horní lišty z Microsoft Wordu (obrázek 6) za použití metody TM_CCORR a TM_SQDIFF.



Obrázek 6: část lišty Microsoft Wordu

Na obrázku 7 je vidět, že template matching za použití metody TM_CCORR našel část lišty na ploše Windows. Hodnota maxima je 0,973053. Bakalářská aplikace používala kritérium, aby maximum bylo vyšší než 0,95. Hodnota minima je 0,0563975. V současné době má kritérium, aby hodnota minima byla menší než 0,045. TM_SQDIFF odpovídá modrý obdelník a TM_CCORR červený.

Podle výše uvedených hodnot podmínku pro nalezení splnila jen metoda TM_CCORR. Z obrázku jde usuzovat, že došlo k nesprávnému nálezu, protože se daná část lišty na ploše nenachází. Za použití samotné metody TM_CCORR dojde k nesprávnému nálezu. za použití TM_SQDIFF k nesprávnému nálezu nedojde. Velkou roli na nesprávném nálezu lze spatřit v barevné podobnosti mezi vzorem a nalezenou oblastí. Vzor tvoří z většiny kombinace světle modré a bílé barvy. Na nalezeném místě se nachází obloha světlé modré až bílé barvy. Červená lišta se žlutou kulatou ikonou v levé části červené lišty představuje odchylku od modré a bílé většiny ve snímku, přesto nezabránila korelaci v nesprávném nálezu.

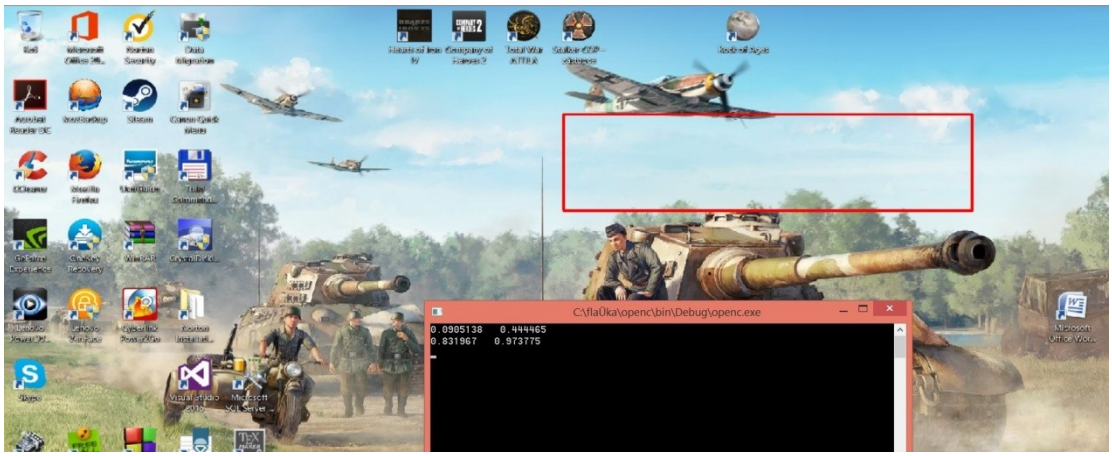


Obrázek 7: nalezení lišty na ploše Windows

Pro další pokus jsem nechal na ploše hledat část hlavičky webu firmy Tieto. Testovací program zůstal nezměněný, jen jsem jako vzor načetl obrázek 8.

Obrázek 8: Část hlavičky webu firmy Tieto

Na následujícím obrázku 9 lze vidět, že template matching za použití metody TM_CCORR našel část lišty firmy Tieto na ploše Windows. Hodnota maxima je 0,973775 a hodnota minima je 0,0905138. Jako v předcházejícím pokusu metoda TM_CCORR našla místo na ploše a metoda TM_SQDIFF nenašla. Při pohledu na obrázek lze dospět k závěru, že část hlavičky firmy Tieto se na ploše nenachází. Došlo k nesprávnému nálezu.



Obrázek 9: nalezení části hlavičky webu firmy Tieto na ploše Windows

Vyzkoušel jsem také vyhledat červenou lištu z prvního testovaného vzorového obrázku na ploše Windows s pomocí TM_SQDIFF a TM_CCORR. Korelace nalezla nesprávné místo v oblasti hlavního panelu. TM_SQDIFF nedošla k nesprávnému nálezu. Mohu usoudit, že TM_SQDIFF je stabilnější

a odolnější vůči nesprávným nálezům. Chybovost TM_CCORR bude způsobena malou závislostí na odstínech barev obrázků. TM_SQDIFF funguje na principu výpočtu sumy rozdílu mezi zdrojovým a vzorovým obrázkem. Zjednodušeně se dá popsat následujícím zdrojovým kódem:

```
1     SSD = 0;
2     for( int i=0; i<height-1; i++){
3           for( int i=0; i<width-1; i++){
4     SSD   +=   (source[i][j]   -   template[i][j])   *   (source[i][j]   -
template[i][j]);
5           }
6     }
```

3.1.2 Funkce selectROI

Do aplikace jsem chtěl přidat uživateli možnost vytvořit si vzorový obrázek pro vyhledávání v samotné aplikaci, proto jsem se rozhodl využít metodu selectROI z knihovny Open CV.

[2] Jedná se o funkci z knihovny Open CV, která není použita pro samotné rozpoznávání obrazu, ale bakalářská aplikace ji využívá pro vytváření vzorových obrázků. Překvapivé na této funkci je, že je součástí Tracking API, ale ne API pro zpracování obrazu. Po spuštění funkce se uživateli zobrazí obrázek, ve kterém tahem vybere svou obdélníkovou oblast zájmu. Následně stiskne klávesu enter, nebo mezerník pro další zpracování. Funkce má následující parametry:

- Jméno okna.
- Obrázek, který chce zobrazit a vybírat v něm oblast zájmu.
- Boolean, jestli chceme táhnout z prostřední části výběru nebo z levého horního rohu do pravého dolního.
- Boolean, jestli se po vybrání oblasti zájmu má v oblasti objevit mřížka.

Standardně lze vybrat jen jednu oblast zájmu. Funkce umí vybírat i více, ale tato možnost nemusí vždy fungovat správně, protože v implementaci knihovny se nachází chyba. Po vybrání první oblasti je potřeba stisknout klávesu Enter dvakrát a Python verze úplně nefunguje s více vybranými oblastmi.

3.1.3 Canny Edge Detection

Během vývoje aplikace jsem zkoumal možnost použití detekce hran pro úpravu snímku a vzorových obrázků a následném vyhledávání vzorů v těchto upravených snímcích. Knihovna Open CV pro vyhledávání hran používá metodu Canny() s parametry:

- Vstupní pole obrázku.
- Výstupní pole hran.
- První úroveň prahu.
- Druhá úroveň prahu.
- Velikost aparatury.

[6] Jedná se o detektor vyvinutý Johnem F. Cannym v roce 1986. Zaměřuje se na splnění tří kritérií:

- Nalezení hran s nejmenším počtem chyb.
- Nalezení přesných míst hran.
- Hrana bude nalezena jednou, ne vícekrát.

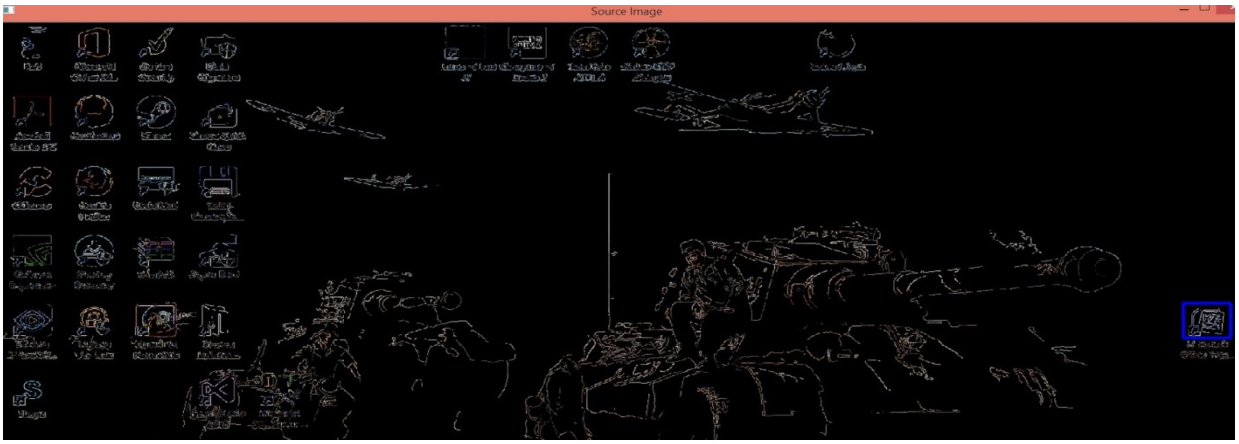
Canny Edge Detection se používá pro získání strukturálních dat v obrázku a minimalizaci množství zpracovávaných dat. Canny Edge Detection je jedním z nejvíce striktních metod, která zajišťuje spolehlivou detekci a přesnost, proto se jedná o jeden z nejpopulárnějších algoritmů pro detekci hran.

Canny Edge Detection se nejvíce uplatní pro zpracování obrazu, u kterého neznáme přesný vzhled vzoru. Například program na rozpoznávání listů bude zpracovávat obrázky listů různého tvaru, velikosti, barvy. Na obrazovce se nachází ikony, které jsou neměnné. Uživatel zná obsah své obrazovky a bude vyhledávat prvky, které zná. Použití metody rozpoznávání obrazu na základě hran naopak může zvýšit riziko nesprávných nálezů, zvláště v následném hledání obrysů. Detekci hran jsem se rozhodl využít v testovacím projektu.

Speciální projekt, který byl použit pro vyhledání vzoru v koláži, byl využit na porovnání efektivity template matchingu pro normální snímek obrazovky a snímek obrazovky upravený přes Canny Edge Detection.

Obrázek 10: Normální snímek

Jedná se o shodné snímky. Jediný rozdíl je, že druhý snímek byl upraven Canny Edge detektorem.



Obrázek 11: Upravený snímek

Tabulka 2: Naměřené hodnoty neupraveného a upraveného snímku

Výsledné hodnoty po porovnání	Normální snímek	Upravený snímek
Minimum	0,00324002	0,00238307
Maximum	0,998414	0,998813

Jak můžeme po porovnání výsledků vidět, template matching v kombinaci s detekcí hran je o trochu přesnější, než samotný template matching. Rozdíl je velký u metody SQDIFF. U CCORR je rozdíl malý. Z hlediska časové náročnosti není mezi těmito možnostmi rozdíl.

3.2 Emgu CV

Emgu CV je multiplatformní knihovna pro zpracování obrazu. Je určena primárně pro .NET podporované jazyky C#, VB, VC++, Python a další. Pracuje primárně s vývojovým prostředím Visual Studio a podporuje operační systémy Windows, Mac OS, iOS a Android.

Pro porovnání obrázků nabízí Emgu CV stejnou metodu MatchTemplate jako knihovna Open CV se stejnými vstupními parametry:

- Zdrojový obrázek, ve kterém se bude vyhledávat.
- Vzorový obrázek, který vyhledáváme. Musí být větší než obrázek, ve kterém vyhledáváme.
- Matice, do které se uloží výsledky.
- Číslo některé z porovnávacích metod, které budou popsány dále.

Emgu CV také obsahuje stejnou funkci MinMaxLoc pro nalezení minimální a maximální hodnoty ve výsledném poli se stejnými vstupními parametry:

- Pole hodnot, ve kterém chceme vyhledávat.
- Ukazatel na číslo typu double, do kterého se uloží nalezená minimální hodnota.
- Ukazatel na číslo typu double, do kterého se uloží nalezená maximální hodnota.
- Ukazatel na bod, do kterého se uloží nalezený bod s minimální hodnotou.
- Ukazatel na bod, do kterého se uloží nalezený bod s maximální hodnotou.
- Maska.

3.3 TensorFlow

TensorFlow je open-source knihovna primárně určená pro numerické výpočty za použití grafů. TensorFlow byla vytvořena skupinou pracovníků, kteří pracovali v organizaci Googlu Machine Intelligence Research za účelem vedení strojového učení a hluboké neuronové sítě.

Open CV jsem si vybral, protože má funkci zpracování obrazu založenou na template matchingu a lze tuto knihovnu propojit s vývojovým prostředím CodeBlocks, ve kterém byla tato bakalářská aplikace v počáteční fázi vyvíjena. Na internetových stránkách Open CV jsou k nalezení dobře popsané funkce této knihovny a dobře vypracované tutoriály. Open CV nabízí také další možnosti práce s obrazem, které by se mohly použít v případě rozšíření funkčnosti aplikace. Oprávněně se jedná o oblíbenou knihovnu pro práci s obrázky.

4 Popis aplikace

Po představení různých metod rozpoznávání obrazu a možností Open CV se můžeme zaměřit na popis samotné bakalářské aplikace. Nejdříve popíšu uživatelské rozhraní, následně funkčnost a implementaci.

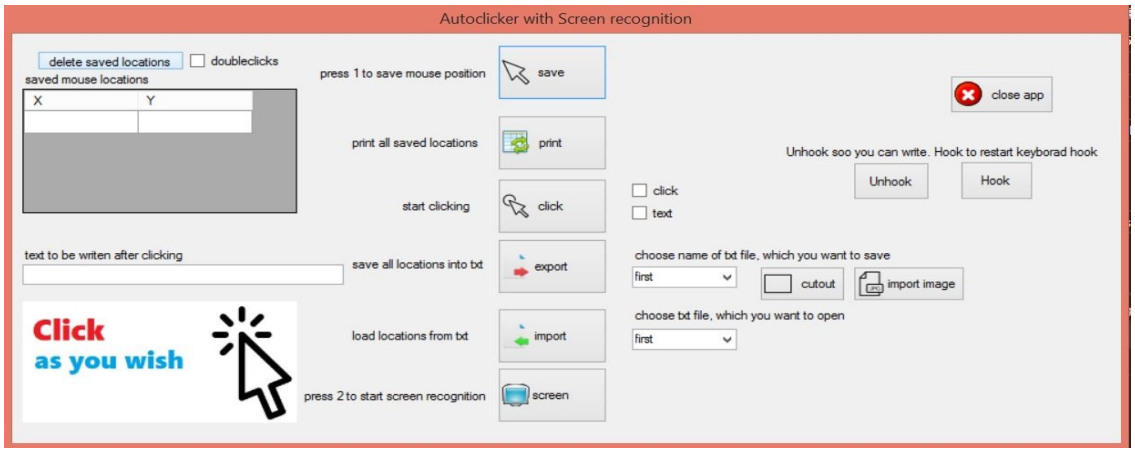
Vývoj aplikace probíhal inkrementálně postupným plněním zadaných úkolů:

- Vytvoření základního autoclickeru, který bude klikat na uložené pozice myši, bude umět uložit zachycené pozice myši do souboru a načíst je zpátky do aplikace. Po startu se aplikace shodí na plochu.
- Přidání Windows Hooku (vysvětlení na konci seznamu), aby uživatel mohl ovládat aplikaci, i pokud je okno minimalizované. Uživatel tak může zachytit pozice myši a spustit snímání obrazu. Dále bylo přidáno celé rozpoznávání obrazu s pomocí knihovny Open CV a pořízení snímku obrazovky s pomocí Windows API.
- Přidání možnosti na smazání uložených pozic myši a virtuální psaní kláves na klávesnici s pomocí Windows API.
- Přidána možnost vytvoření výřezu obrazovky pomocí Open CV a možnost kliknout na nalezené místo umístění vzoru.
- Přidání výběrového pole pro větší uživatelský komfort a nahrání vzorových obrázků do aplikace.

Windows Hook představuje mechanizmus, který umožní aplikaci zachytávat události a reagovat na ně. V aplikaci běží funkce, která zachytává události z klávesnice. Pokud uživatel stiskne předem definované klávesy, zajistí se funkčnost aplikace. Tu si může uživatel sám vypnout a zapnout. Vypnout ji musí, pokud chce psát do textového pole aplikace, jinak funkce pohltí události z klávesnice a stisknuté znaky neprojdou do textového pole. Aplikace používá nízkoúrovňový Hook WH_KEYBOARD_LL. Windows Hook umí zpracovat i jiné události než jen vstup z klávesnice.

První konzolová verze bakalářské aplikace byla vytvářena ve vývojovém prostředí CodeBlocks. V průběhu vývoje jsem práci převedl do formulářového projektu ve Visual Studiu.

4.1 Uživatelské rozhraní

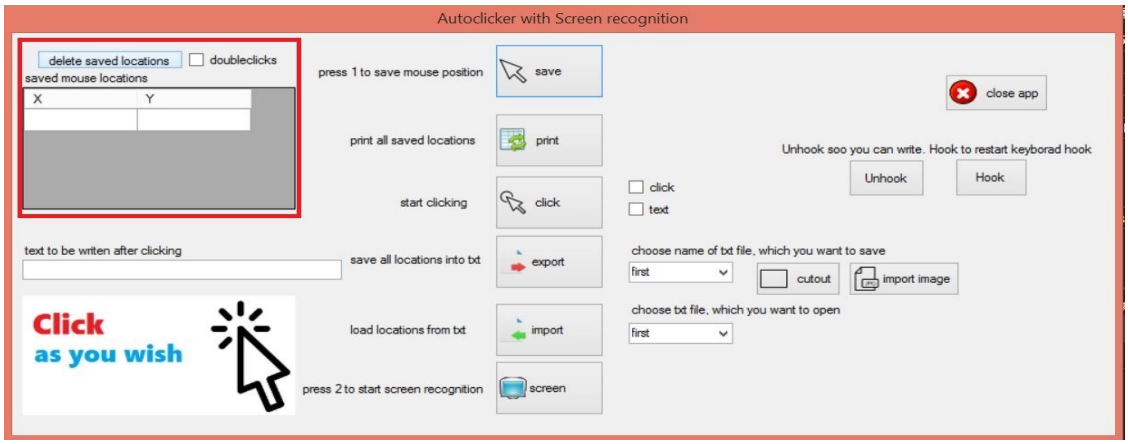


Obrázek 12: Uživatelské rozhraní

Pro vytvoření grafického rozhraní byl použit standardní formulář a ovládací prvky ve Visual Studiu pro Windows Forms. Cílem bylo vytvořit jednoduché, uživatelsky co možná nejpřívětivější rozhraní. Pro vytvoření tohoto uživatelského rozhraní byly použity ovládací prvky:

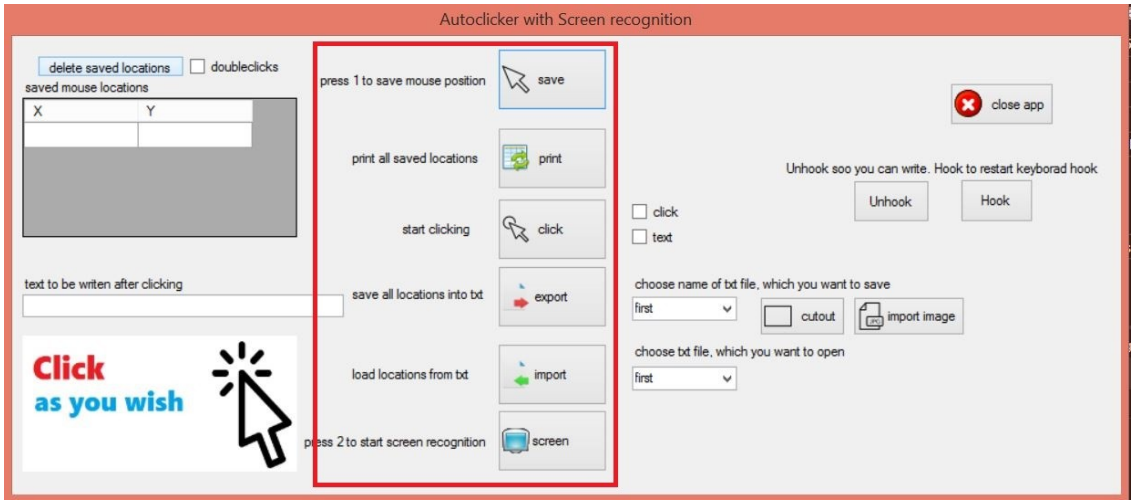
- Button – tlačítko.
- Label – štítek.
- TextBox – textové pole.
- CheckBox – zaškrtávací pole.
- DataGridView – tabulka.
- ComboBox – výběrové pole.
- PictureBox – obrázkové pole.

Každé textové pole, výběrové pole, tabulka a většina tlačítek jsou doplněny štítkem, aby si uživatel mohl přečíst jejich význam. Na dalších řádcích je uživatelské rozhraní popsáno podle funkčních bloků.



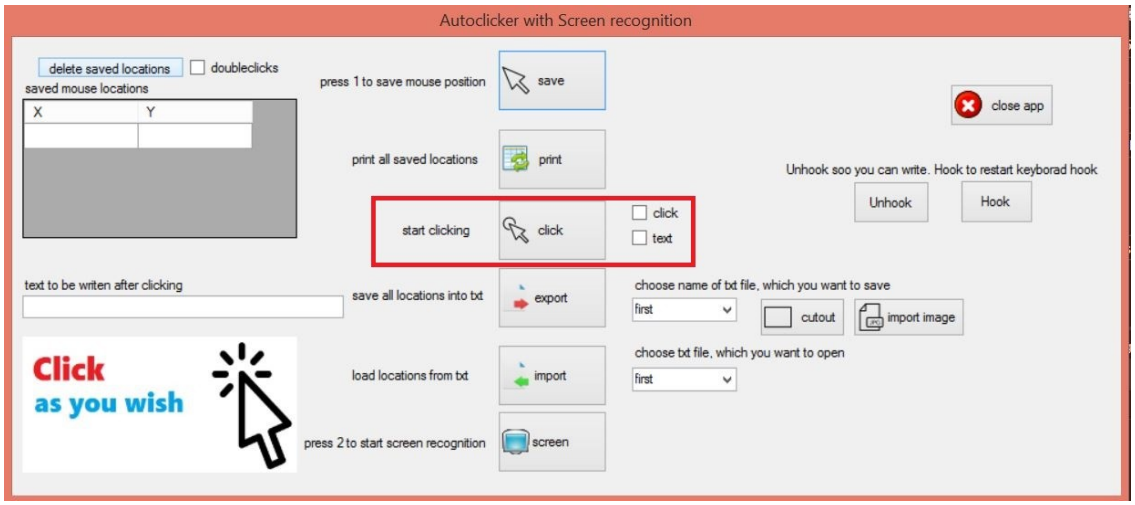
Obrázek 13: Levá část uživatelského rozhraní

Na obrázku 13 se v levé horní části se nachází tabulka uložených pozic, kde pro každou z nich se vypíše souřadnice X a Y. V tabulce může být uložena více než jedna souřadnice. Dále se zde nachází tlačítko pro smazání pozic myši, které si uživatel uložil, a zaškrtávací políčko, které slouží pro zapnutí a vypnutí dvojitého poklikání. Jestli uživatel zaškrtne toto políčko, program na všechny uložené pozice pokliká dvakrát za sebou, v opačném jen jednou. Tato funkce se hodí například pro klikání na ikony na ploše.



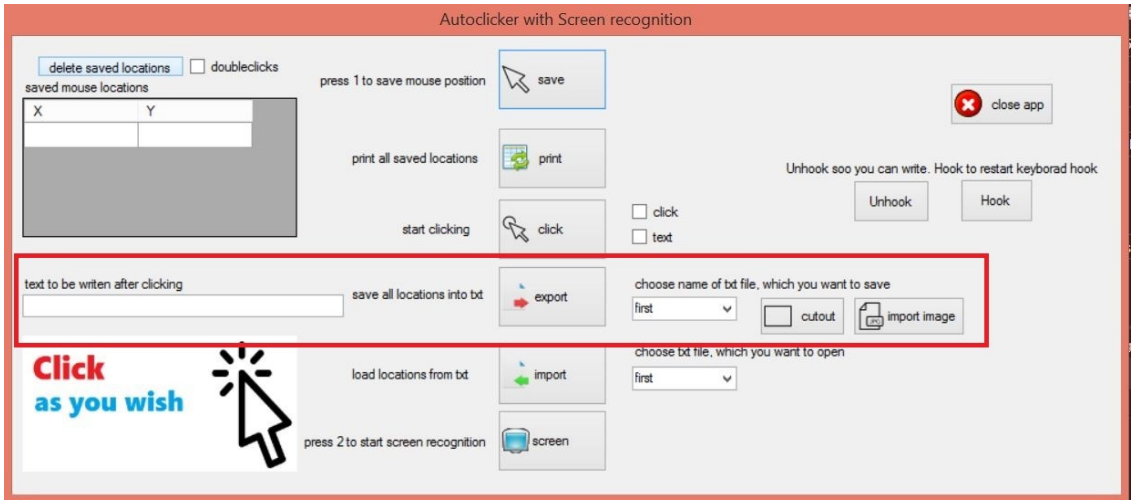
Obrázek 14: Prostřední část uživatelského rozhraní

V prostřední části se nachází vertikálně seřazená řada tlačítek (obrázek 14), která zajišťují nejdůležitější funkčnost aplikace. Na levé straně od tlačítek se nachází štítky s popiskem odpovídajících tlačítek.



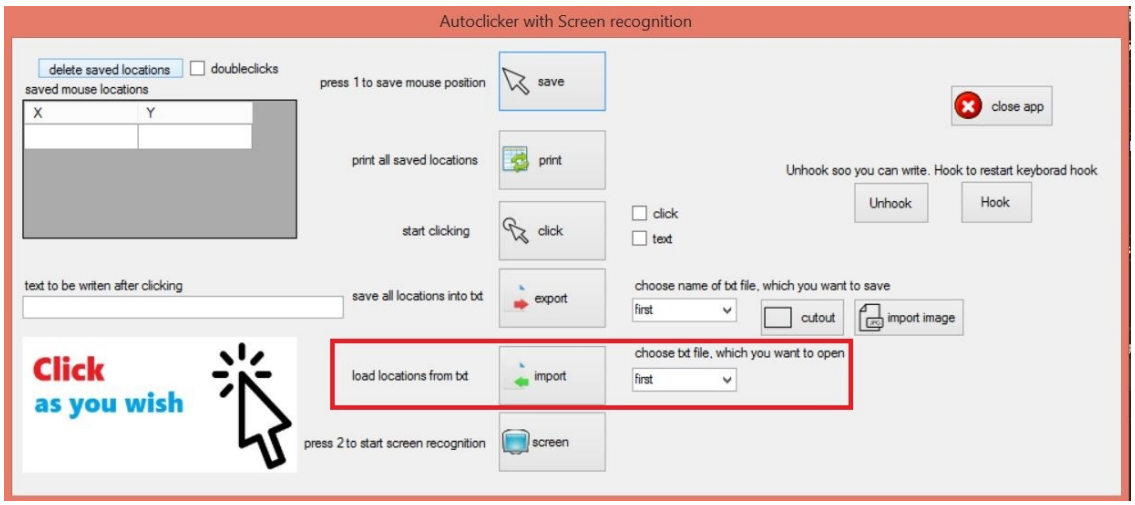
Obrázek 15: Klikací část

Na obrázku 15 si můžeme všimnout, že na pravé straně od tlačítka **„**click**“** se nalézají dvě zaškrtávací pole, přes něž si uživatel určí, jestli chce, aby aplikace prováděla klikání, psaní virtuálně na klávesnici, oboje zároveň nebo ani jednu z těchto funkcí.



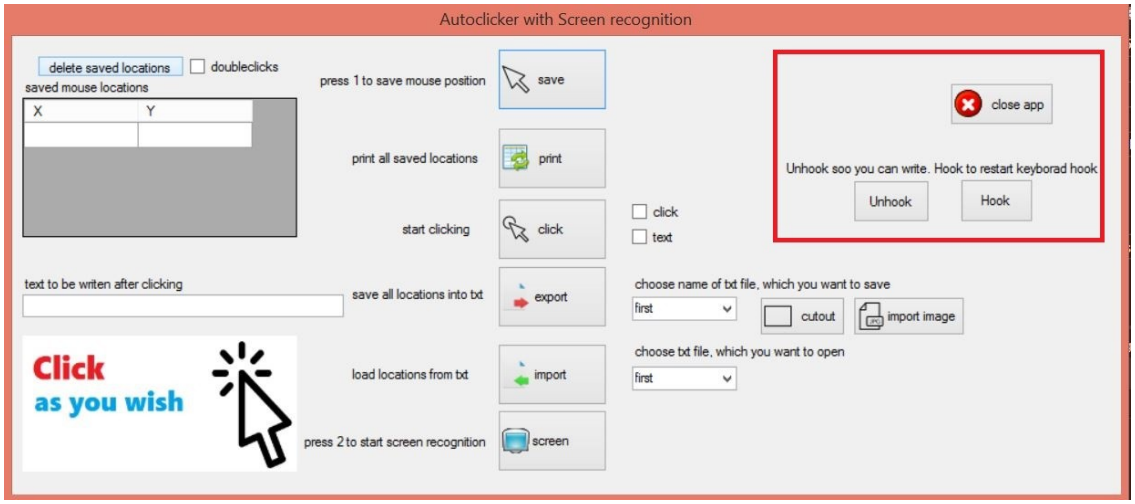
Obrázek 16: Exportovací část

Na levé straně od popisku tlačítka **„**export**“** se nachází textové pole s popiskem, ve kterém je umístěn text, jenž bude program virtuálně psát na klávesnici (obrázek 16). Na pravé straně od tlačítka **„**export**“** se nachází výběrové pole, ve kterém si uživatel vybere název textového dokumentu, do něhož si přeje uložit text, který se bude virtuálně psát na klávesnici, a uložené pozice myši z tabulky. Napravo od výběrového pole se nachází tlačítko **„**cutout**“** pro vytvoření snímku obrazovky a vytvoření výřezu. Poslední tlačítko **„**import image**“** umožní vložení .jpg obrázků do aplikace.



Obrázek 17: Importovací část

Na pravé straně od tlačítka **„**import**“** se nachází výběrové pole, ve kterém uživatel vybere jméno textového, které chce načíst do aplikace (obrázek 17).



Obrázek 18: Pravá část uživatelského rozhraní

Obrázek 18 nám ukazuje, jak vypadá poslední část aplikace. Nejvíce vpravo se nachází tlačítko pro vypnutí aplikace a dvě další tlačítka, jedno pro vypnutí, druhé pro zapnutí, pro snímání stisknutých kláves.

4.2 Funkcionalita

K uživatelskému rozhraní se váže funkčnost, jednotlivé prvky mají nějaký účel svého použití v aplikaci. V této části budou vypsané jednotlivé funkce, které bakalářská aplikace umí, a způsob, jak požadované funkce dosáhnout.

- Uložení pozice myši: najeďte myší na místo, které chcete uložit. Jestliže je Hook zapnutý, stiskněte klávesu 1 nebo 2, v opačném případě Hook zapněte a poté stiskněte klávesu 1. Uložit pozici můžete také pomocí tlačítka **„**save**“**, ale tlačítko se musí nacházet na vámi chtěném místě. Po stisku dojde k uložení pozice myši do pole.
- Zobrazení uložených pozic: pro vypsání uložených pozic do tabulky stiskněte tlačítko **„**print**“.** Tabulka s uloženými pozicemi myši se obnoví.
- Smazání uložených pozic: stiskněte tlačítko **„**delete saved locations**“.** Pole uložených pozic myši se vyprázdní.
- Zapnutí/vypnutí dvojitého klikání: nastavte odpovídající hodnotu zaškrtávacího pole **„**doubleclicks**“**. Prázdné zajistí klikání jedenkrát, zaškrtnuté dvakrát. Vnitřní proměnná se nastaví pro dvojité klikání se nastaví podle stavu tohoto zaškrtávacího pole.
- Napsaní textu pro virtuální klikání: napište text do textového pole na levé straně pod tabulkou.
- Zapnutí klikání: na pravé straně od tlačítka **„**click**“** zaškrtněte zaškrtávací pole podle vašich požadavků, a klikněte na tlačítko **„**click**“**. Ověřte si, že máte alespoň jednu uloženou pozici myši a zaškrtnuté políčko **„**click**“**. Pokud uživatel nastaví spuštění klikání, tak se sekvenčně provede klikání na pozice myší uložené v poli. Jestliže uživatel nastavil psaní textu, tak se text z textového pole po znacích nakliká na klávesnici a následně po dokončení aplikace pípne.
- Uložení do souboru: Z výběrového pole napravo od tlačítka **„**export**“** si vyberte, do kterého souboru chcete ukládat a klikněte na tlačítko **„**export“. V reakci na stisknutí se sekvenčně

projde pole uložených pozic a uloží se do textového souboru, následně text z textového pole a na závěr se uloží hodnota pro dvojité klikání. Po dokončení aplikace pípne.

- Načtení souboru: Z výběrového pole napravo od tlačítka „import“ si vyberte jméno souboru, ze kterého chcete načítat, a klikněte na tlačítko „import“. V tabulce se zobrazí pozice, vypíše se text pro virtuální psaní a zaškrtávací pole pro dvojité klikání se nastaví podle hodnot uložených v textovém souboru. Po dokončení aplikace pípne.
- Snímání obrazovky: na pravé straně od tlačítka „click“ zaškrtněte zaškrtávací pole podle vašich požadavků a klikněte na tlačítko „screen“. Aplikace začne po omezenou dobu snímat obrazovky a vyhledávat vzorové obrázky. Aplikace vypne Windows Hook a desetkrát spustí metodu třídy screen_matching pro rozpoznávání obrazu. Po dokončení aplikace pípne.
- Vytvoření vzorového obrázku: Z výběrového pole napravo od tlačítka „export“ si vyberte, do kterého souboru chcete ukládat, a klikněte na tlačítko „cutout“. Aplikace spustí metodu třídy screen_matching pro vytvoření výřezu obrazovky. Zobrazí se snímek obrazovky. Vyberte si místo, které chcete uložit, a zmáčkněte mezerník nebo enter. Výřez se zobrazí a uloží.
- Načtení vzorového obrázku: Z výběrového pole napravo od tlačítka „export“ si vyberte, do kterého souboru chcete ukládat, a klikněte na tlačítko „import image“. Spustí se OpenFileDialog a správce souborů, přes který si vyberete .jpg obrázek, který si přejete uložit do aplikace. Po potvrzení se obrázek zobrazí v obrázkovém poli v levém dolním okraji a uloží se.
- Ukončení aplikace: stiskněte tlačítko „close“. Vypne se Windows Hook, dealokuje se objekt třídy screen_matching a program se vypne.
- Zapnutí/vypnutí Hooku: pro zapnutí snímání stisknutých kláves stiskněte tlačítko „Hook“, pro vypnutí tlačítko „Unhook“.

Pokud si přejete vyhledávat ikonu na obrazovce, u které neznáte její přesnou polohu, soubor s pozicemi, který odpovídá dané ikoně, nechte prázdný. V opačném případě se bude klikat na pozice uložené v souboru. Pro zajištění co nejvyšší přesnosti doporučuji vytvářet vzorové obrázky malých rozměrů.

4.3 Implementace

Funkcionalita aplikace je nejdůležitější částí aplikace. Vzhled aplikace je také důležitý z hlediska marketingu, ale jestli aplikace nebude funkční, tak si ji potencionální zákazníci nekoupí díky recenzím nespokojených uživatelů. O funkcionalitu se stará zdrojový kód, který bude dále popsán.

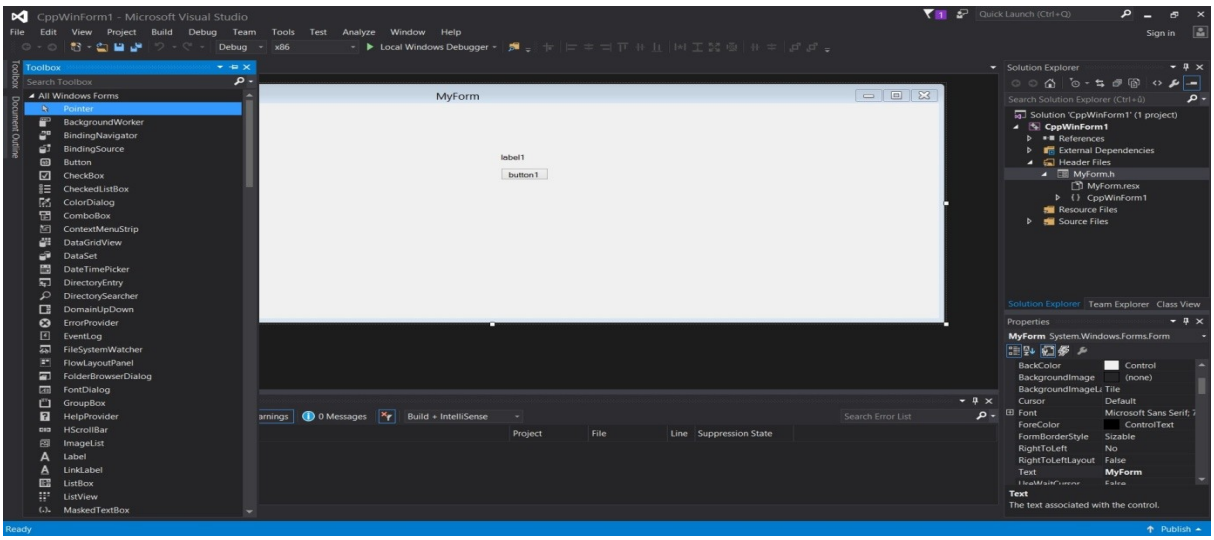
Před popisem samotné aplikace bych rád přiblížil knihovnu pro vytváření GUI aplikací, se kterou byla aplikace vytvářena. Jedná se o knihovnu tříd pro tvorbu formulářových aplikací. Skládá se ze tří základních částí

- **Designer** – V Designeru vidíme vzhled formuláře.
- **Properties** – V okně Properties vidíme a můžeme upravovat vlastnosti jednotlivých prvků ve formuláři.
- **Toolbox** – je vyskakovací okno, ze kterého si vybíráme jednotlivé prvky a vkládáme je do formuláře.

Každý formulář funguje jako objekt, který obsahuje grafický vzhled a zdrojový kód. Mezi režimy si vývojář může přepínat.

Grafické uživatelské rozhraní umožňuje uživateli ovládat program s pomocí grafických ovládacích prvků. Prvky dělíme na vstupní a výstupní. Vstupní se starají o příjem informací od uživatele a výstupní se starají o zobrazení dat uživateli.

Windows Forms jsou standardní součástí Visual Studia. Ve verzi Visual Studia, ve kterém byla tato práce vytvářena (2015), je možné stáhnout online šablonu pro C++ verzi Windows Forms, nebo si přes prázdný projekt vytvořit vlastní. Stažení je jednoduché a intuitivní.

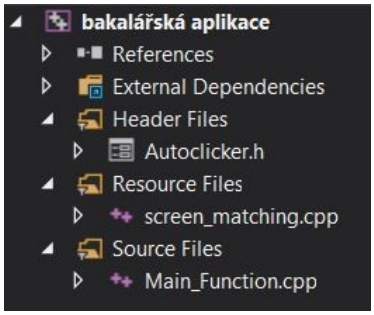


Obrázek 19: Ukázka Windows Forms

Windows Forms jakožto součást produktu Visual Studio pro mě byly ideální volbou pro vytváření bakalářské aplikace s GUI, protože pracovat s Windows Forms jsem se učil během studia v předmětu Programovací jazyky II v programovacím jazyku C#. Princip Windows Forms pro C++ je stejný jako v případě C#. Jelikož práce byla prvotně vytvářena ve vývojovém prostředí CodeBlocks, dala by se do tohoto prostředí stáhnout a integrovat knihovna pro vytváření GUI aplikací WxWidget, ale s instalací a vytvářením nového projektu jsem měl problémy.

4.3.1 Struktura aplikace

Struktura aplikace je založena na defaultním C++ projektu Windows Forms (obrázek 20). Ve verzi Visual Studio 2015 bylo potřeba tuto knihovnu prvně stáhnout, nebyla k dispozici defaultně. Projekt obsahuje jednotlivé reference, externí závislosti, hlavičkový soubor MyForm.h, ve kterém se nachází GUI a na něj napojený zdrojový kód, soubor screen_matchng.cpp kde se nachází třída pro vytváření výřezů z obrazovky a rozpoznávání obrazu a nakonec zdrojový soubor MyForm.cpp, ve kterém se nachází metoda Main. Po spuštění aplikace se vytvoří okno uživatelského rozhraní.



Obrázek 20: Struktura projektu

Bakalářská práce je kompilovaná pro platformu x64, protože používá knihovnu Open CV ve verzi pro platformu x64. Knihovna je připojena přes nastavení properties.

4.3.2 Main_Function.cpp

V následující části je popsán obsah souboru Main_function.cpp. Jedná se o soubor, který je kompilován a spuštěn.

```
1     #include "Autoclicker.h"
2     using namespace System;
3     using namespace System::Windows::Forms;
4     [STAThread]
5     void main(){
6           Application::EnableVisualStyles();
7           Application::SetCompatibleTextRenderingDefault(false);
8           ShowWindow(GetConsoleWindow(), SW_HIDE);
9           cppwin::Autoclicker form;
10          Application::Run(%form);
11
12          MSG msg;
13          while (!GetMessage(&msg, NULL, 0, 0)) {
14          TranslateMessage(&msg);
15          DispatchMessage(&msg);
```

Funkce ShowWindow() zajistí, že se okno pro konzolovou část aplikace shodí na plochu, aby uživatel viděl jen hlavní okno aplikace s GUI. V části od MSG msg se nachází část zdrojového kódu se smyčkou zpráv. Vlákno, které Hook založilo, musí obsahovat smyčku zpráv.

4.3.3 Autoclicker.h

Autoclicker.h obsahuje příkazy pro vložení použitých knihoven, seznam použitých jmenných prostorů, deklarace metod a některých vytvořených proměnných. Poté se zde nachází konstruktor, destruktor, automaticky generovaný seznam jednotlivých komponentů uživatelského rozhraní, kontejner

komponentů. V další části je automaticky generovaný zdrojový kód, který se upravuje na základě změn uživatelského rozhraní, stará se o vytvoření konkrétních prvků uživatelského rozhraní a nastavení jejich správných vlastností.

Následující část se stará o provedení kódu na základě stisknutí tlačítek uživatelského rozhraní. První je tlačítko **„save“**. Po stisknutí tlačítka se uloží aktuální pozice myši pomocí GetCursorPos(&current) do bodu current, souřadnice x a y se uloží do pole a iterátor se zvýší o jedničku. V iterátoru je uložena informace o počtu prvků v poli. Další je tlačítko **„print“** a **„deleteGrid“** pro aktualizaci pole a vyčištěni pole uložených pozic myši.

Následující je tlačítko **„click“**. V následující části je popsána implementace tohoto tlačítka.

```
1     clicklabel->Text = "started";
2     if (klikej == true){
3           for (int i = 0; i <= iterator - 1; i++) {
4                 Sleep(2000);
5                 SetCursorPos(xx[i], yy[i]);
6                 mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0);
7                 mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0);
8                 if (double_click == true) {
9                 mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0, 0);
10                mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0);
11                }
12          }
13    }
14    if (pistext == true && textBoxKlikaciText->Text->Length != 0)    {
15          if (hookused == 1) {
16                UnhookWindowsHookEx(keyboardHook);
17                hookused = 0;
18          }
19          INPUT ip;
20          for (int i = 0; i != textBoxKlikaciText->Text->Length; i++){
21          ip.type = INPUT_KEYBOARD;
22          ip.ki.time = 0;
23          ip.ki.dwFlags = KEYEVENTF_UNICODE;
24          ip.ki.wScan = textBoxKlikaciText->Text[i];
25          ip.ki.wVk = 0;
26          ip.ki.dwExtraInfo = 0;
27          SendInput(1, &ip, sizeof(INPUT));
28          ip.ki.dwFlags = KEYEVENTF_KEYUP;
```

```
29          SendInput(1, &ip, sizeof(INPUT));
30          }
31       keyboardHook=SetWindowsHookEx(WH_KEYBOARD_LL, KeyboardProc, 0, 0);
32          hookused = 1;
33    }
34    clicklabel->Text = "ended";
35          Beep(600, 400);
```

Po stisknutí tlačítka štítek vedle tlačítka oznámí, že klikání začalo. Jestliže si uživatel zaškrtl zaškrtávací pole pro klikání, bude se procházet pole uložených pozic myši a postupně klikat. Prodleva mezi kliky jsou dvě sekundy. Jestliže je zaškrtnutá možnost dvojitého klikání, provede se dvojité klikání. Když uživatel zaškrtl zaškrtávací pole pro vypsání textu a v textovém poli je nějaký text na klikání, vypne se Hook. Zapnutý Hook by psaný text odchytil do sebe a do naší požadované aplikace, např. internetového prohlížeče, by se text nedostal. Postupně se budou procházet jednotlivé znaky v textovém poli a každý znak se virtuálně vyťuká. Po vyťukání se zapne Hook. Na závěr se do štítku oznámí ukončení klikání a aplikace pípne.

Následuje tlačítko **„**export**“**. Z výběrového okna napravo od tlačítka se zjistí jméno souboru, do kterého chceme ukládat. Do textového souboru se postupně zapíše značka pro pozice, uložené pozice myši, značka pro text, text z textového pole pro klikání, značka pro dvojklik, hodnota ze zaškrtávacího pole pro dvojité klikání. Po dokončení aplikace pípne.

Následujícím tlačítkem je tlačítko „import“. Z výběrového okna napravo od tlačítka se zjistí jméno souboru, ze kterého se bude načítat. Otevře se soubor, v opačném případě se zobrazí okénko s informací, že soubor nelze otevřít. Jednotlivé části textového souboru se zpracují a uloží na odpovídající místa. Po dokončení aplikace pípne.

V linii tlačítek pokračuje tlačítko „screen“. Tlačítko pro správné fungování potřebuje, aby uživatel zaškrtl alespoň pole pro klikání nebo psaní textu, v opačném případě se přeskočí na závěrečné pípnutí. Při správném zaškrtnutí spustí metoda třídy Screen_matching pro vytvoření snímku obrazovky a zpracování obrazu.

```
1     if (hookused == 1) {
2                UnhookWindowsHookEx(keyboardHook);
3                hookused = 0;
4           }
5           if (klikej == true || pistext == true) {
6                for (int i = 0; i < 10; i++) {
7                      klikac->Screen(pistext, double_click, klikej);
8                }
9           }Beep(600, 400);
```

Za tlačítkem „screen“ najdeme tlačítko pro ukončení aplikace, které vypne Hook a ukončí program. Dále se vedle zdrojového kódu nachází tlačítka pro vypnutí a zapnutí Hooku a pro vytvoření vzorového obrázku výřezem nebo nahráním.

Ve zdrojovém kódu se nachází také metody, které se starají o zpracování hodnoty ze zaškrtávacích polí, když dojde ke změně jejich hodnoty akcí uživatele.

Jako poslední se ve zdrojovém kódu nachází metoda KeyboardProc(), která je hlavní částí Hookingu. Pracuje ve smyčce a volá po každém stisknutí klávesy na klávesnici. Zjišťování okna v popředí je důležité, aby vstup z klávesnice zůstal zároveň v okně, ve kterém uživatel píše.

4.3.4 Screen_matching.cpp

Funkce Matching(boolean pistext, boolean double_click, boolean klikej) která se stará o pořízení snímku obrazovky a jejího uložení, využívá metody z knihovny Windows.h. Vytvoříme matice, zajistíme úchyt k obrazovce pomocí GetdesktopWindow(), následně DC obrazovky a DC v paměti. Vytvoříme matici Open CV, bitmapu z DC a bitmapu DC nakopírujeme do matice Open CV, kterou následně převedeme do barevného formátu shodného s funkci imread(), uvolníme paměť a spustíme rozpoznávání obrazu.

```
1       for (int i = 0; i < 3; i++) {
2                       switch (i){
3                       case 0:
4                               template_image = imread("files/prvni.jpg",
cv::IMREAD_COLOR);
5                               jmeno_souboru = "files/prvni.txt";
6                               break;
7                       case 1:
8                               template_image = imread("files/druhy.jpg",
cv::IMREAD_COLOR);
9                               jmeno_souboru = "files/druhy.txt";
10                              break;
11                      case 2:
12                              template_image = imread("files/treti.jpg",
cv::IMREAD_COLOR);
13                              jmeno_souboru = "files/treti.txt";
14                              break;
15                      }
16
17                      matchTemplate(mat, template_image, result, 1);
18                      minMaxLoc(result, &minVal, &maxVal, &minLoc,
&maxLoc, cv::Mat());
19
```

```
20                      if (minVal<0.05) {
21                            Beep(800, 400); Beep(600, 400);
22
23                            std::ifstream myfile(jmeno_souboru);
24                            if (myfile.is_open()){
```

Začnou se postupně se budou načítat vzorové obrázky a aplikace provede jejich vyhledání v snímku obrazovky pomocí porovnávací metody CV_TM_SQDIFF, které odpovídá číslo 1. Následně se s pomocí metody minMaxLoc() naleznou nejvyšší a nejmenší místa shody. První minimální hodnota je důležitá pro zjištění shody. Jestliže je hodnota dostatečně dobrá, aplikace dvakrát zapípá. Požadavek na shodu je nastaven vysoko, aby se snížila pravděpodobnost chyby porovnávací metody a nalezení nesprávného místa. Otevře se textový dokument, který odpovídá danému vzorovému obrázku. Podle uživatelova výběru klikání a psaní textu se provedou odpovídající operace podobně jako po stisku tlačítka „click“. Pokud se v souboru nenalézá jakákoli uložená pozice myši, bude se klikat na konkrétní nalezené místo. V případě mé aplikace se ukázal problém s chováním systému Windows. Normálně má aktivní zvětšení textu, aby byl lépe čitelný. Toto zvětšení textu ale zmenšuje rozlišení obrazovky. Full HD rozlišení 1920x1080 se v mém případě zmenší o 80 % na 1536x864. Snímek obrazovky je v rozlišení Full HD. Když se v něm najde vzor, souřadnice jsou ve Full HD, ale obrazovka má kvůli přiblížení menší rozlišení, což způsobí problém. Výpočet percent_width a percent_height zajistí eliminaci této chyby. Výpočet je založen na trojčlence. Obdobně se trojčlenka provede v SetCursorPosition pro souřadnice X a Y nalezeného umístění vzoru a přičte se polovina rozměrů vzoru. Standardní nalezené místo je levý horní roh vzoru. Ikona bývá uprostřed vzoru, takže přičtením poloviny rozměrů vzoru si zajistím, že program pokliká na ikonu.

Tabulka 3: Měření délky vykonávání metody Matching

Zavolání metody Matching	Časová délka v sekundách
1.	2,02633
2.	1,43178
3.	1,42426
4.	1,42148
5.	1,41831
6.	1,41372
7.	1,42587
8.	1,43044
9.	1,42369
10.	1,43025
průměr	1,484613

Metoda Matching je nastavena tak, aby se spustila desetkrát v případě kliknutí uživatele na tlačítko „screen“, rozhodl jsem se změřit časovou náročnost této metody a výsledky jsou uvedeny v tabulce 3. Během každého volání se pořídil snímek obrazovky a provedlo se vyhledání tří vzorových obrázků. V těchto naměřených časech není započítáno zpracování skriptu v případě nálezu vzoru.

Funkce Screen_a_vyrez(string jmeno) se liší svým zakončením, kde místo spuštění zpracování obrazu se vypne Hook, zavolá se Open CV metoda pro vybrání oblasti pro výřez, daná oblast se ze snímku obrazovky vyřeže, zobrazí, uloží do souboru, zapne Hook a uvolní paměť.

5 Závěr

V práci jsem popsal konkurenty bakalářské aplikace, teoretické přiblížení použitých technologií a možností pro zpracování obrazu a v poslední části implementaci samotné aplikace. Podařilo se vytvořit autoclicker s rozpoznáváním obrazovky, který uživateli umožní ukládat pozice myši, spustit klikání a psaní textu virtuálně na klávesnici podle jeho nastavení, možnost zapnout dvojité klikání, uložení do souboru, načtení ze souboru a zapnutí rozpoznávání obrazu, které v současně verzi umí vyhledat tři vzorové obrázky. Na úkor výkonu by to bylo možné rozšířit. Kromě výkonu by se dal přepracovat vzhled uživatelského rozhraní, které se v současné době nachází v rané fázi vývoje. Rád bych pokračoval vytvořením lepšího a modernějšího vzhledu přes knihovnu Qt.

Literatura

[1] FAKULTA BIOMEDICÍNSKÉHO INŽENÝRSTVÍ ČVUT. Metody rozpoznání objektů v obrazu. *Fbmi.cvut.cz* [online]. [cit. 2018-03-25]. Dostupné z: http://www.fbmi.cvut.cz/files/predmety/3528/public/Metody%20rozpozn%C3%A1n%C3%AD%20objekt%C5%AF%20v%20obrazu.pdf

[2] MALLICK, Satya. Learn Open CV: How to select a bounding box (ROI) in OpenCV (C++/Python)? *Learnopencv.com* [online]. [cit. 2018-03-25]. Dostupné z: https://www.learnopencv.com/how-to-select-a-bounding-box-roi-in-opencv-cpp-python/

[3] OPENCV. Template matching. *Opencv.org* [online]. [cit. 2018-03-25]. Dostupné z: https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html

[4] OPENCV. Object Detection. *Opencv.org* [online]. [cit. 2018-04-10]. Dostupné z: https://docs.opencv.org/trunk/df/dfb/group__imgproc__object.html#gga3a7850640f1fe1f58fe91a2d7583695dab65c042ed62c9e9e095a1e7e41fe2773

[5] ADAMEC, Václav. Bakalářská práce. *Zpracování a rozpoznávání obrazu* [online]. Univerzita Palackého v Olomouci, 2011 [cit. 2018-04-10]. Dostupné z: https://theses.cz/id/520eu6/downloadPraceContentForThesesCZ_adipIdno_139025

[6] OPENCV. Canny Edge Detection. *Opencv.org* [online]. [cit. 2018-04-10]. Dostupné z: https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html

Adresářová struktura přiloženého disku

/práce	bakalářská práce ve formátu .pdf
/aplikace	zdrojové kódy bakalářské aplikace