

UNIVERZITA PARDUBICE  
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

UŽIVATELSKÁ PŘÍRUČKA

2019

Bc. Tomáš Tichý

Univerzita Pardubice  
Fakulta elektrotechniky a informatiky

Biometrické porovnávací algoritmy v procesech ověření identity

Bc. Tomáš Tichý

Uživatelská příručka

2019

## **OBSAH**

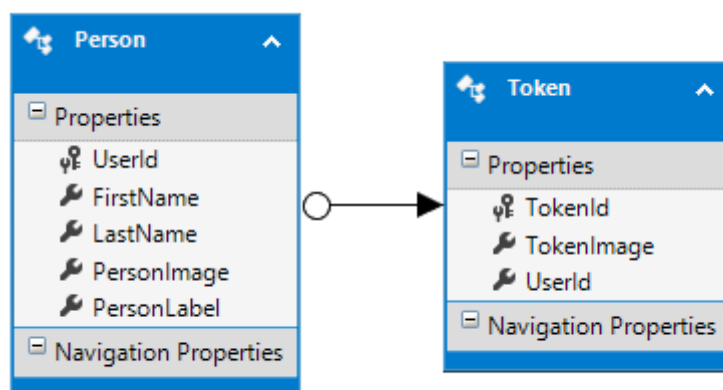
1	Obecné informace.....	8
2	Ovládání aplikace pro správu osob.....	9
3	Ovládání aplikace pro zpracování obrázku .....	13
4	Aplikace pro hledání prahových hodnot a testování algoritmů.....	18
4.1	Hledání prahových hodnot .....	18
4.2	Testování biometrických algoritmů .....	21

Obrázek 1 - Databázový model .....	8
Obrázek 2 - Rozdělení aplikace pro správu osob dle MVVM.....	9
Obrázek 3 - Hlavní okno správy osob .....	10
Obrázek 4 - Dialogové okno pro editaci/přidání osoby .....	10
Obrázek 5 - Dialogové okno pro přidání tokenového snímku.....	11
Obrázek 6 - Zvětšený obrázek s detekovanými tvářemi .....	12
Obrázek 7 - Aplikace pro detekci a rozpoznání obličejů.....	14
Obrázek 8 - Kontextové menu snímku detekovaného obličeje .....	15
Obrázek 9 - Výsledek porovnání pomocí různých algoritmů .....	16
Obrázek 10 - Nesprávné určení shody .....	17
Obrázek 11 - Průběh hledání prahových hodnot .....	19
Obrázek 12 - Výsledek hledání prahové hodnoty pro algoritmus EigenFaces .....	20
Obrázek 13 - Výsledek hledání prahové hodnoty pro algoritmus FisherFaces .....	20
Obrázek 14 - Výsledek hledání prahové hodnoty pro algoritmus LBPH .....	21
Obrázek 15 - Úspěšnosti algoritmů před nastavením prahové hodnoty .....	22
Obrázek 16 - Úspěšnosti algoritmů po nastavení prahových hodnot .....	22

# 1 Obecné informace

Tato kapitola je zaměřena na popis implementace a hlavních funkcí porovnávacího programu. Bude zde nastíněn způsob implementace, aplikace a využití technologií, uvedených v předchozí kapitole.

Z hlediska praktické části této kvalifikační práce byla prvním krokem implementace tvorba databázové struktury. Z analýzy již vyplývá, že se nebude jednat o velmi rozsáhlou databázi, nýbrž o pouhé uložení osob a jejich snímků obličeje. Výsledný model tedy vypadá následovně:



Obrázek 1 - Databázový model

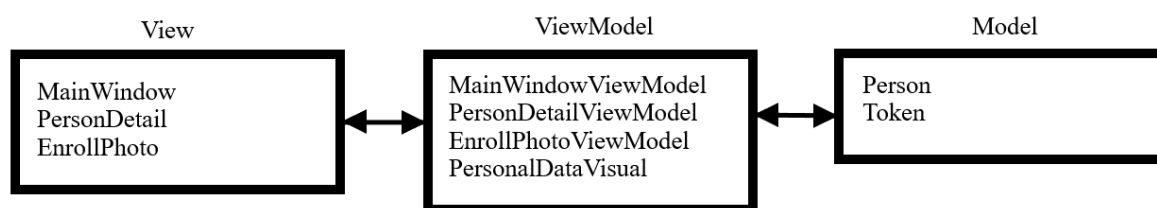
S tímto databázovým modelem komunikuje modul, založený na Entity Frameworku (v práci je použita poslední stabilní verze, tedy 6.2.0). Celé řešení je rozděleno do 3 balíčků – první (Core) obsahuje základní a společné prvky (především pro komunikaci s databází), ImageDemo je samotná porovnávací aplikace, které bude věnován prostor v oddíle 3 a PersonManager je správa osob, uložených v databázi (detailněji popsáno v oddíle 2).

Celá implementace probíhala dle návrhového vzoru MVVM, přičemž všechny dílčí aplikace využívají jednoho modelu – třídy Person a Token, které jsou třídními reprezentacemi stejnojmenných databázových tabulek. Kolekce těchto tříd jsou obsaženy v třídě DataProvider, která slouží jako prostředník pro přístup do databáze. Při kterékoliv práci s daty (tedy s modelem) je využívána tato třída. Ostatní jsou tedy pouze View nebo ViewModel, které budou popsány u jednotlivých aplikací.

## 2 Ovládání aplikace pro správu osob

Jak již název napovídá, tato aplikace poskytuje kompletní možnosti pro správu osob, uložených v databázi. Aplikace sestává ze 3 dialogových oken (view) spolu s příslušícími viewmodely. Pro lepší komunikaci mezi okny a připojení k viewmodelům zde bylo využito vlastní rozhraní **IDialog**, které si uchovává základní znalosti o kontextu (DataContext), výsledku (DialogResult), vlastníkovi (Owner – okno, ze kterého nový dialog byl zavolán) a poskytuje metody pro zavření (Close) a zobrazení dialogu (ShowDialog).

Spolu s vlastním rozhraním je zde použita vlastní implementace třídy **DialogService** (jakožto implementace rozhraní **IDialogService**), jejíž hlavní zodpovědností je správa View a přiřazení specifických ViewModelů. Je zde tedy obsažen slovník, do kterého se ukládají dvojice View – ViewModel. Dále je zde obsažena metoda pro registraci nové dvojice. Proces přiřazení ViewModelů k View a registrace dvojic probíhá v metodě OnStartup v souboru App.xaml.



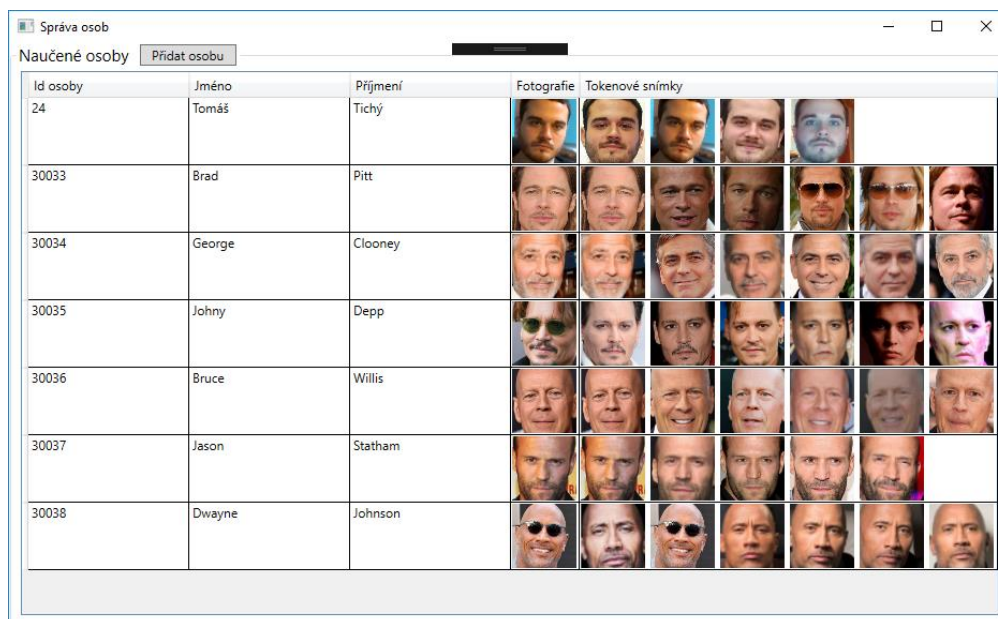
Obrázek 2 - Rozdělení aplikace pro správu osob dle MVVM

Z důvodu správného propojení s View implementují všechny ViewModely rozhraní **INotifyPropertyChanged**, které pomocí **PropertyChangedEventHandler** obstarává aktualizaci View při změně kterékoliv z vlastností, uložené ve ViewModelu a propojené s grafickým uživatelským rozhraním.

Hlavnímu oknu aplikace dominuje tabulka, zobrazující uložená data o všech osobách z databáze. Pro každou osobu je zde údaj o identifikačním čísle, jménu, příjmení a hlavní profilové fotografii (data z databázové tabulky **Person**). Poslední sloupec obsahuje všechny tokenové snímky<sup>1</sup> osoby, které jsou pro danou osobu v databázi uloženy (tabulka **Token**).

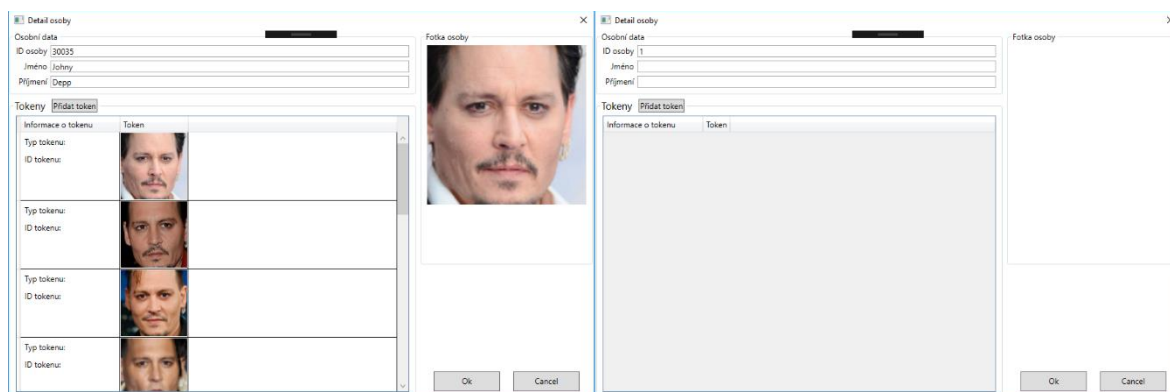
---

<sup>1</sup> Tokenový snímek: oříznutá část vstupního obrázku, která obsahuje pouze detekovaný obličej (vyříznutá ohraničená oblast získána z procesu detekce obličeje). Tyto snímky budou použity jako trénovací sada rozpoznávacího algoritmu.



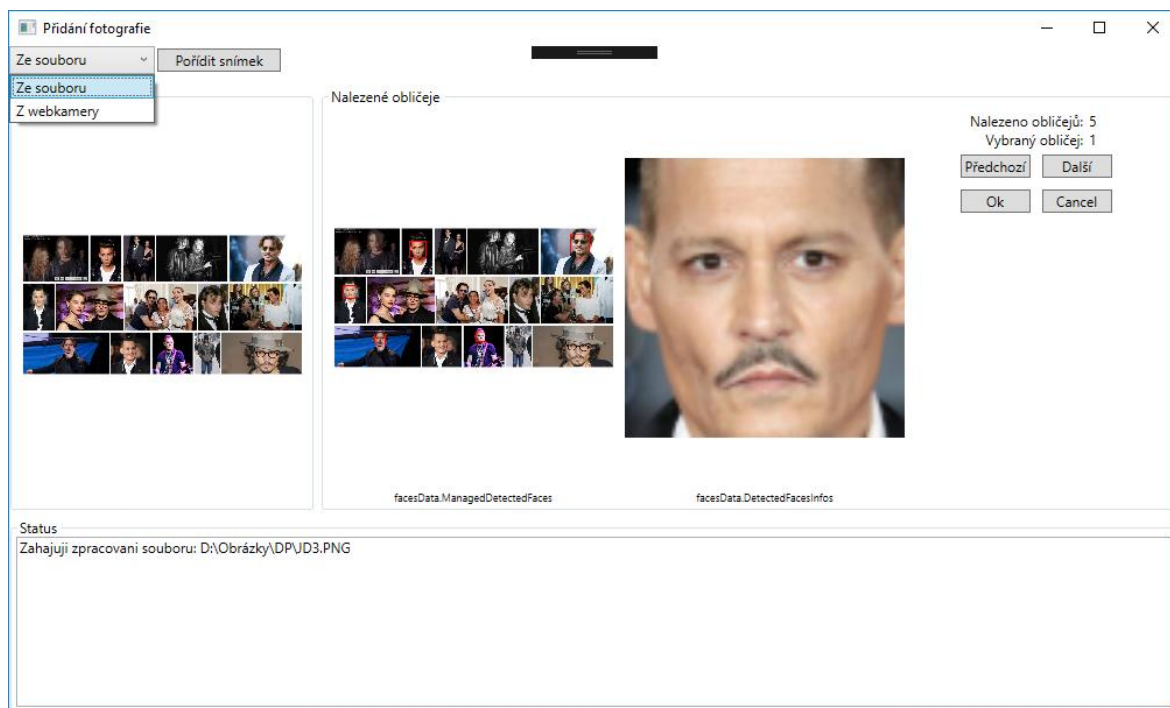
**Obrázek 3 - Hlavní okno správy osob**

V levé horní části aplikace se nachází tlačítko pro přidání nové osoby. Úprava a mazání existující osoby jsou možné pomocí kontextové nabídky po pravém kliknutí na řádek tabulky (záznam reprezentující konkrétní osobu). Pro přidání nové osoby i editaci stávající slouží jednotné dialogové okno, přičemž rozdíl je, zda jsou tomuto dialogu poslána data o konkrétní osobě, nebo zda je dialog prázdný – připravený pro vytvoření nového záznamu.



**Obrázek 4 - Dialogové okno pro editaci/přidání osoby**

Nejdůležitější částí tohoto dialogového okna je správa tokenových obrázků (v levé spodní části). Zde jsou zobrazeny existující, uložené snímky, přičemž po kliknutí pravým tlačítkem je možné daný snímek odstranit nebo jej nastavit jako profilovou fotku uživatele (větší fotografie v pravé horní části dialogu). Kromě těchto možností a změny jména a příjmení je zde tlačítko pro přidání tokenových snímků, které vyvolá nové dialogové okno.



**Obrázek 5 - Dialogové okno pro přidání tokenového snímku**

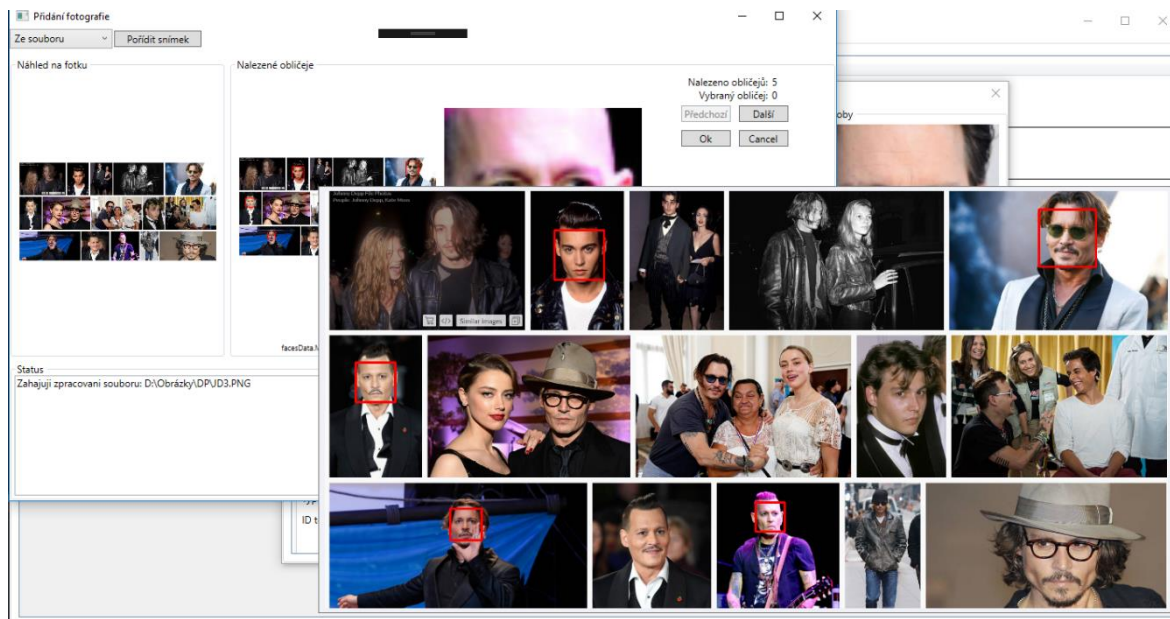
V tomto dialogovém okně již probíhá několik operací. Nejprve je třeba zvolit zdroj dat (jak je znázorněno na Obrázek 5 – rozbalený combo box v levém horním rohu). Možnosti jsou ze souboru, tedy nahrání uloženého obrázku nebo z webkamery. Při druhé volbě dojde k zahájení přenosu snímků z kamery a jejich následnému zpracování. Při obou volbách se nejprve načte vstupní obrázek do levé části dialogu (při volbě vstupu z webkamery se zde začne přehrávat vstupní proud dat).

Prvním krokem zpracování obrázku je převod na formát Image<Bgr, Byte> a pro detekci následně na Image<Gray, Byte>, tedy do obrázku v odstínech šedé barvy. Převedený obrázek je následně podroben procesu detekce pomocí kaskádových klasifikátorů.

Všechny nalezené oblasti, ve kterých se nacházejí obličeje jsou vloženy do kolekce obdélníků. Pro každou z obdélníkových oblastí jsou do původního obrázku zakresleny okraje této oblasti a výsledek je zobrazen vpravo od původního stavu vstupního obrázku.

Přičemž metoda SwitchFace ve výše uvedeném zdrojovém kódu slouží k přepínání mezi detekovanými tvářemi, které jsou vyobrazeny v pravé části okna aplikace. Po najetí myši na libovolný element obrázku v dialogu se obrázek zvětší.





**Obrázek 6 - Zvětšený obrázek s detekovanými tvářemi**

Na Obrázek 6 je patrné, že tento princip detekce obličejů není zcela dokonalý a vyžaduje určité podmínky pro úspěšné detekování. Všechny tváře jsou dále z původního obrázku vyříznuty do podoby tokenových snímků a přidány do kolekce pro zobrazení (Obrázek 5 – obrázek obličejů vpravo). Mezi jednotlivými obličejí v kolekci je možné přecházet pomocí tlačítek „Předchozí“ a „Další“ a po nalezení požadovaného lze potvrdit volbu kliknutím na tlačítko „Ok“. Proces přidávání snímku je zároveň možné zrušit pomocí tlačítka „Cancel“.

Obdobný proces probíhá při získávání snímku z živého přenosu z webové kamery. Pro každý přichodící snímek je provedena stejná sada operací. V druhé sekci zprava je tedy vstupní snímek s označenými oblastmi, ve kterých se nachází obličej a v pravé sekci jsou vyříznuté obličeje, mezi kterými je opět možné listovat pomocí tlačítek.

V této aplikaci tedy dochází pouze k detekování obličejů, nikoliv k samotnému porovnání. Účelem užití této aplikace je totiž vytvoření databáze osob a k nim přiřazených tokenových obrázků, které bude moci další aplikace využít k naučení porovnávacích algoritmů pro další užití.

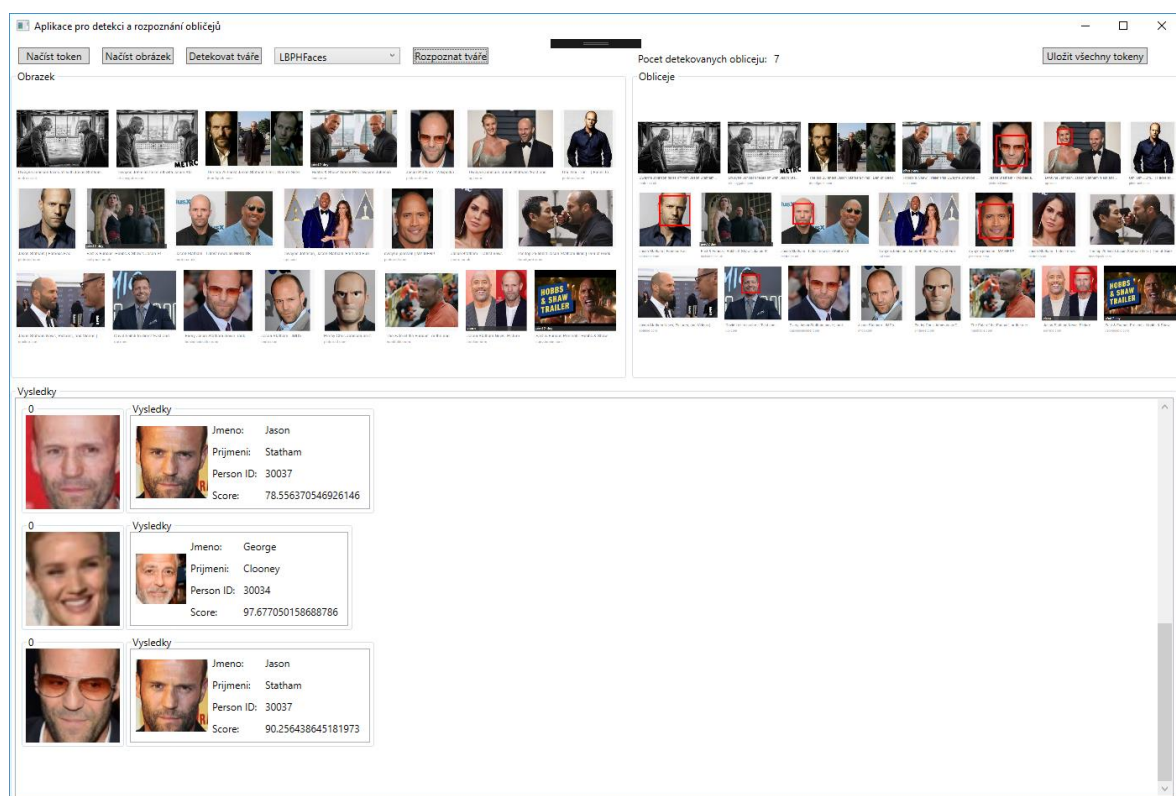
### 3 Ovládání aplikace pro zpracování obrázku

Tato aplikace reprezentuje hlavní realizaci praktické části této práce. Předchozí oddíl byl zaměřen na přípravnou aplikaci pro vytvoření datových základů pro rozpoznávání obličejů. Databáze, vytvořená pomocí aplikace pro správu osob je využita v této aplikaci, kde dochází k porovnávání nových vstupních obrázků s naučenými daty z databáze.

Při spuštění aplikace dojde k načtení všech dat z databáze, za účelem trénování všech 3 porovnávacích algoritmů. Dojde tedy k inicializaci EigenFaceRecognizer, FisherFaceRecognizer i LBPHFaceRecognizer a u každého je zavolána metoda Train, jejíž parametry jsou pole tokenových obrázků a pole uživatelských identifikátorů, přičemž na určitém indexu těchto polí jsou související data (tokenu na indexu  $i$  v poli tokenů odpovídá identifikátor na indexu  $i$  v poli identifikátorů).

Stav takto naučeného *rekognizéru* je uložen do souboru, ze kterého může být následně načten. Důvodem je možnost volby porovnávacího algoritmu před zahájením procesu rozpoznání (pomocí combo boxu v horní části aplikace, bude popsáno dále). K naučení všech najednou dochází z důvodu ušetření času v rámci dalších procesů a z důvodu znovu-použitelnosti (jednou naučený a uložený *rekognizér* může být opakovaně využit k rozpoznání různých tváří).

Stavy jsou ukládány do souborů formátu YAML (YAML Ain't Markup Language), což je formát pro serializaci dat textových souborů. Tyto soubory jsou čitelné strojem i člověkem, strukturované pomocí odsazení (pomocí mezer, nikoli tabulátorů) a není zde omezení úrovně vnořování. K ukládání jsou využity knihovní funkce *Write* pro každý *rekognizér* a pro opětovné načtení funkce *Read*.



**Obrázek 7 - Aplikace pro detekci a rozpoznání obličejů**

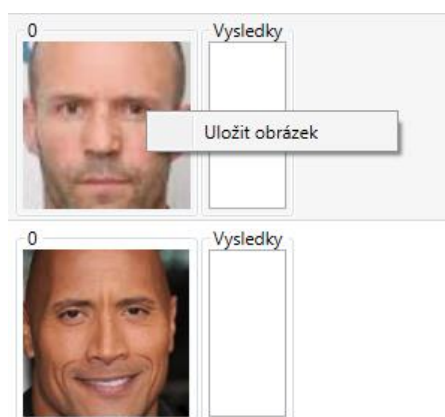
Prevažnou část aplikace pro detekci a rozpoznání obličejů zabírají zobrazovací prvky pro vykreslení obrázků v různých částech procesu. V horní části okna se nachází ovládací prvky, pomocí kterých je možné se v procesu detekce a rozpoznání pohybovat. Opět se zde vyskytuje pojem token (dříve použit jako tokenový snímek). I zde se jedná o označení předem připravené a částečně zpracované fotografie, kde již došlo k oříznutí na základě detekce (v této aplikaci pomocí zmíněných kaskádových klasifikátorů).

Tlačítko „Načíst token“ tedy slouží k načtení předpřipraveného obrázku, na kterém se nachází pouze 1 obličej, který zabírá dostatečnou poměrnou část obrázku. V tomto procesu je snímek načten rovnou jako obličej, je tedy přeskočena samostatná část detekce a přidání do seznamu detekovaných (jak bude popsáno níže, pro klasické obrázky). Dále tedy stačí vybrat porovnávací algoritmus a kliknout na tlačítko „Rozpoznat tváře“.

Na rozdíl od prvního tlačítka, kde dochází k přeskočení jedné fáze procesu, pomocí tlačítka „Načíst obrázek“ dojde pouze k načtení a zobrazení obrázku v levé horní části. K rozfázování zde došlo za účelem dodržení analýzy, konkrétně jako realizace všech případů užití a jejich scénářů. Po úspěšném načtení obrázku je teprve možné kliknout na tlačítko „Detekovat obličeje“. Výsledkem je naplnění seznamu detekovaných tváří, které se zobrazí ve spodní části okna (Výsledky). Zároveň dojde ke zpracování vstupního obrázku podobným způsobem jako u

první aplikace, tedy k orámování detekovaných obličejů. Takto zpracovaný obrázek je zobrazen v pravé horní části. Stejně jako v předešlé aplikaci, i zde jsou pro detekci využity kaskádové klasifikátory.

V libovolné části procesu, kdy seznam detekovaných tváří není prázdný, je možné tyto tokenové snímky uložit. Jednou možností je postupné ukládání pomocí kontextového menu jednotlivých snímků, pokud je účelem uložení celého seznamu, je možné použít tlačítko v pravé horní části aplikace „Uložit všechny tokeny“.

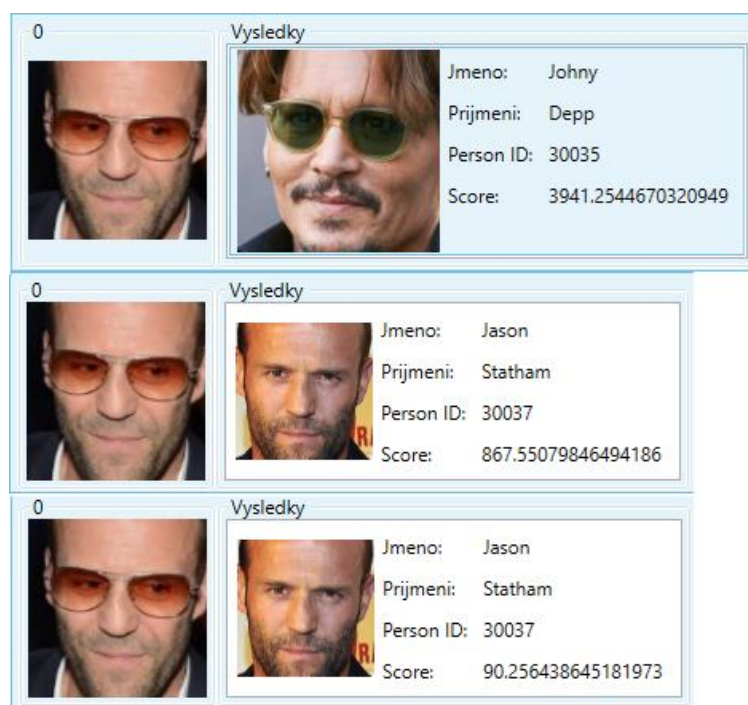


**Obrázek 8 - Kontextové menu snímku detekovaného obličeje**

Po provedení detekce je dle některých scénářů možné ukončit proces využití aplikace. Pokud je účelem užití například jen detekovat na snímku všechny tváře a uložit jejich snímky pro pozdější zpracování, je zbytečné provádět porovnání již zde. Pokud je však potřeba provést rozpoznání tváří na vstupním obrázku, je třeba zvolit porovnávací algoritmus k dalšímu zpracování.

To je možné pomocí combo boxu vedle tlačítka „Detekovat tváře“. Zde jsou k dispozici všechny 3 biometrické algoritmy, které nabízí open-source knihovna OpenCV, tedy EigenFaces, FisherFaces a LBPH.

Volba porovnávacího algoritmu je velmi důležitá, neboť má velký vliv na výsledek porovnání. Ani jeden ze 3 algoritmů není zcela dokonalý, a ne vždy tyto algoritmy poskytují spolehlivé a věrohodné výsledky. Jeden obličej může pomocí různých algoritmů být vyhodnocen různými způsoby (viz Obrázek 9, kde byly použity algoritmy postupně shora EigenFaces, FisherFaces a LBPH).

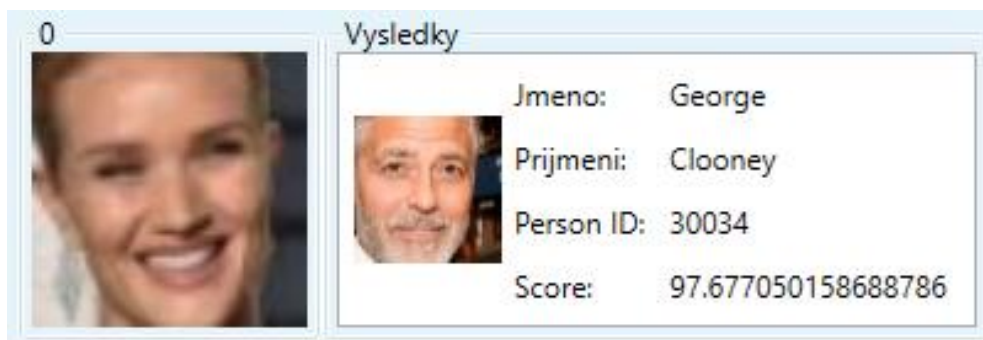


**Obrázek 9 - Výsledek porovnání pomocí různých algoritmů**

Na Obrázek 9 je tedy možné vidět, jakým způsobem dochází k výpisu výsledků porovnání. K tomuto výpisu je využita třída *DetectedFace*, která obsahuje pro každou detekovanou tvář základní údaje spolu se seznamem výsledků porovnání. Do tohoto seznamu jsou ukládány instance třídy *PersonMatchResult*, která obsahuje především profilovou fotografii, unikátní identifikátor, jméno a příjmení uživatele. Tato data jsou získána z databáze pomocí zmiňovaného *DataProvideru*.

Kromě těchto dat je zde uložena i informace o skóre. Tato hodnota udává výsledek porovnání, tedy jakousi věrohodnost nalezené shody. Obrázky, respektive jejich reprezentující vektory, jsou porovnány pomocí měření (Euklidovské) vzdálenosti mezi nimi. Ideální hodnota skóre, tedy vzdálenost mezi obrázky, které se naprosto shodují, je 0. Není zde tedy přímá úměra věrohodnosti (čím vyšší skóre, tím přesnější a věrohodnější shoda), nýbrž přesně naopak.

Často je v této souvislosti skloňována prahová hodnota. U biometrických algoritmů je toto označení používáno pro jakousi hraniční hodnotu přípustnosti. Pokud je tedy tato hodnota překročena u nejbližší shody, výsledkem je nenalezení dostatečné shody, nikoliv zobrazení nejbližší možné shody (nesprávné určení nejbližší shody – Obrázek 10).



**Obrázek 10 - Nesprávné určení shody**

Na Obrázek 9 je patrné, že každý z uvedených porovnávacích algoritmů má skóre shody (tedy vzdálenost mezi popisovými vektory) v jiných řádech. Je tedy potřeba nalézt optimální hraniční hodnotu pro každý algoritmus a tuto nastavit jako práh. Vzhledem k tomu, že názory na hodnotu tohoto prahu se rozcházejí a k důležitosti volby této hodnoty je součástí této práce rovněž aplikace pro hledání této prahové hodnoty za pomoci lidského faktoru.

## 4 Aplikace pro hledání prahových hodnot a testování algoritmů

Tato kapitola bude zaměřena na popis aplikace, která bude sloužit jednak k nalezení prahových hodnot a jednak k otestování úspěšnosti určení shody pomocí jednotlivých algoritmů. Jedná se tedy o aplikaci, která má jedno uživatelské rozhraní a dvě funkcionality. První režim, hledání prahových hodnot, vyžaduje zásah operátora – lidské obsluhy, pro stanovení správnosti určení shody, zatímco druhá část je plně automatizovaná.

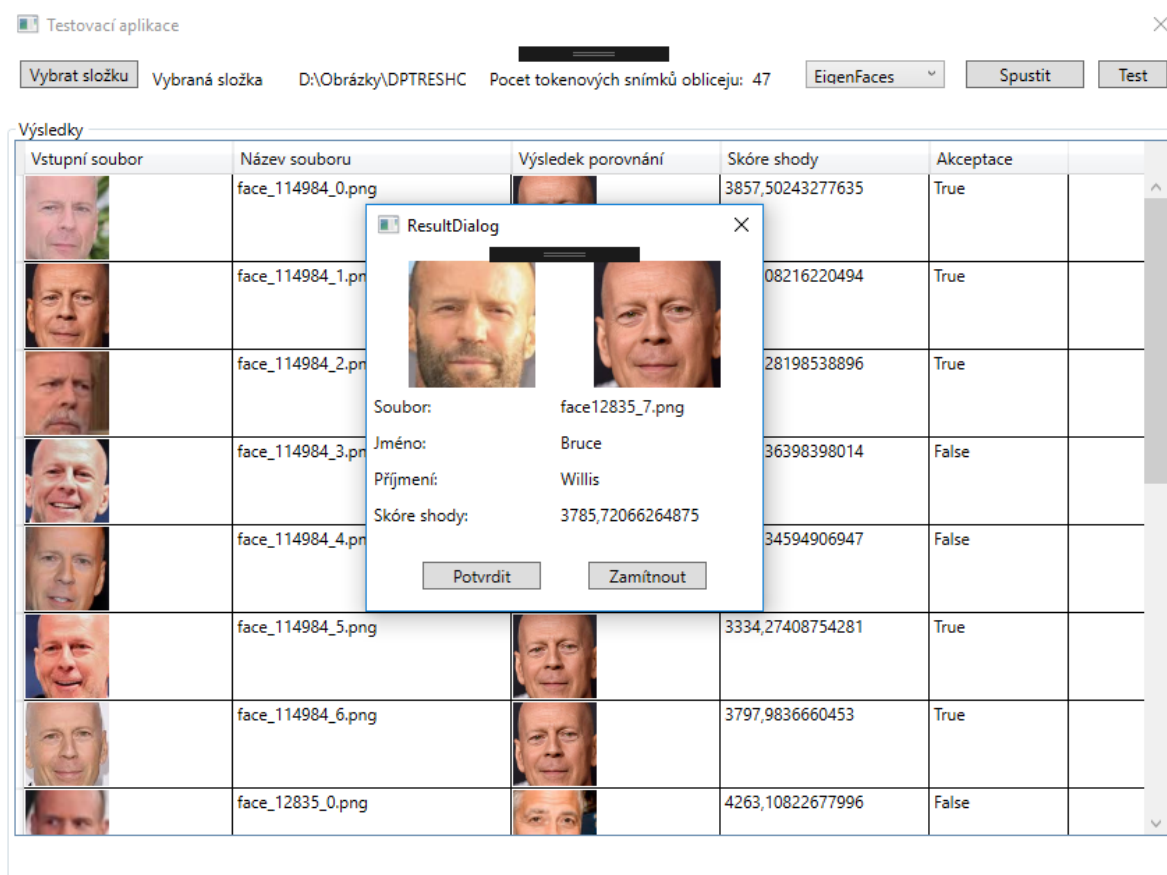
### 4.1 Hledání prahových hodnot

Tento režim aplikace plně navazuje na poslední oddíl předchozí kapitoly, kde bylo konstatováno, že správné určení prahové hodnoty je zcela zásadní pro správné vyhodnocení porovnávacího procesu. Než začne interakce s touto aplikací, je třeba mít předpřipravená data v podobě tokenových obrázků. Tyto je možné (buď po jednom nebo hromadně) uložit z aplikace pro zpracování obrazu (oddíl 3).

Myšlenka této aplikace je zpracování co možná největšího množství těchto testovacích dat dle vybraného biometrického algoritmu. Samozřejmostí je také předem naplněná databáze osob, vůči kterým bude docházet k porovnávání. Pro každé porovnání bude následně vyžadován zásah operátora (uživatele), který bude muset potvrdit, zda algoritmus shodu vyhodnotil správně či nikoliv.

Prvním krokem je tedy výběr složky, ve které se připravené tokenové obrázky nacházejí. Následně dojde k naplnění seznamu názvu souborů z vybrané složky, přičemž vybírány jsou pouze soubory ve formátu png. Po výběru biometrického algoritmu již stačí jen stisknout tlačítko *Spustit* a vyhodnocovat všechny vzniknuvší shody. Změna prahové hodnoty má vliv na ukazatele FRR a FAR, proto je potřeba určit, jaký je ideální stav. Toto se odvíjí od případu užití a způsobu nasazení, jelikož například u zabezpečené budovy, kde systém umožňuje přístup do skladu utajovaných informací jsou jiné preference než u obyčejného docházkového systému firmy.



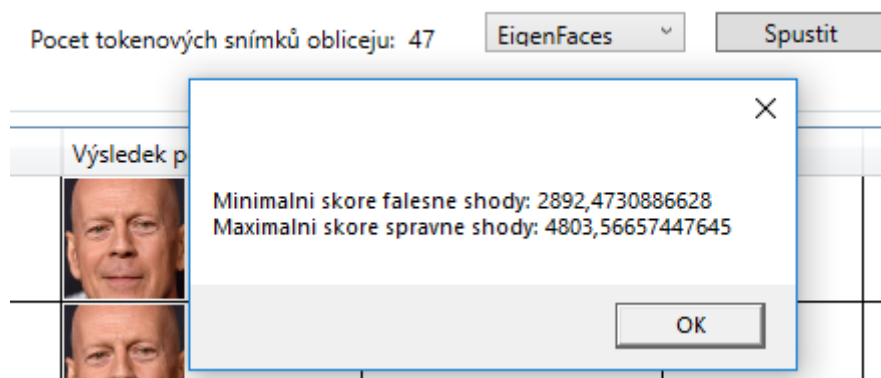


**Obrázek 11 - Průběh hledání prahových hodnot**

Na Obrázek 11 je možné vidět, jakým způsobem je řešeno grafické uživatelské rozhraní aplikace. Stejně jako u ostatních aplikací se veškeré ovládací prvky nachází v horní části. V levé části se nachází tlačítko pro výběr složky, ve které se nacházejí testovací data. Po vybrání se zobrazí hned vedle tohoto tlačítka vybrané umístění a počet tokenových snímků obličejů. Následuje combo box pro výběr biometrického porovnávacího algoritmu, pro který chceme prahovou hodnotu nalézt a tlačítko *Spustit*. Poslední tlačítko (Test) je ovládacím prvkem druhé části této aplikace, které bude věnován následující oddíl.

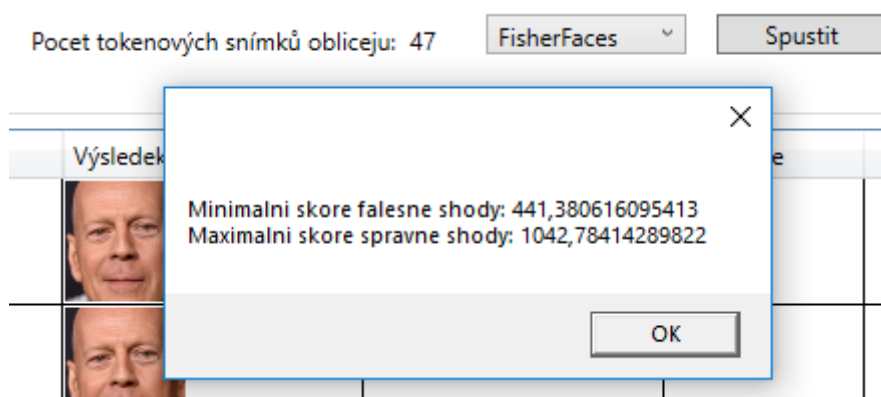
Již v oddíle 3, konkrétně na Obrázek 9 bylo řečeno, že hodnoty shody pro jednotlivé algoritmy se poměrně dost liší, proto je nutné provést hledání optimálních hodnot pro každý algoritmus zvlášť. Aplikace má podobný průběh jako aplikace pro zpracování obrázku (je přeskočena část detekce, jelikož se předpokládá práce s tokenovými obrázky). Vstupní token je tedy převeden do odstínů šedé barvy a je zavolána funkce Predict vybraného biometrického algoritmu. Výsledek je poslán do nového dialogového okna, kde je vyžadováno vyhodnocení ze strany operátora.





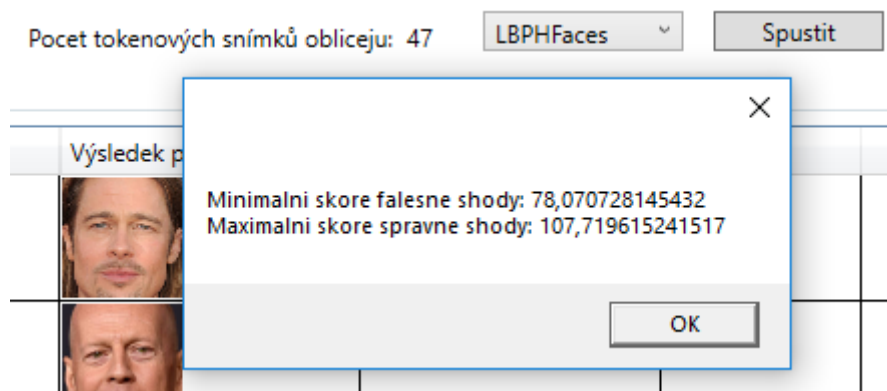
**Obrázek 12 - Výsledek hledání prahové hodnoty pro algoritmus EigenFaces**

Podstatným výsledkem hledání prahů je nalezení dvou hodnot – minimální hodnoty, kterou algoritmus nesprávně vyhodnotil jako shodu a maximální hodnotu, pro kterou algoritmus správně vyhodnotil shodu. Z těchto dvou hodnot je následně možné vytvořit novou prahovou hodnotu. Pokud by se například jednalo o vysoký požadavek na zabezpečení, byla by prahová hodnota nastavena dle minimálního skóre falešné shody. V takovém případě by ze všech vstupních 47 tokenových obrázků nedošlo ani k jednomu falešnému přijetí.



**Obrázek 13 - Výsledek hledání prahové hodnoty pro algoritmus FisherFaces**

Přestože by bylo možné docílit stavu, kdy ze všech vstupních dat by nedošlo ani k jednomu falešnému přijetí, negativním důsledkem by byl zvýšený výskyt falešného odmítnutí. Jak je vidět na Obrázek 12 i Obrázek 13, vyskytují se případy, kde skóre shody je vyšší, než při kterém došlo k nesprávnému párování. U takovýchto případů by při nastavení prahové hodnoty dle minimálního skóre falešné shody docházelo k falešnému zamítnutí.



**Obrázek 14 - Výsledek hledání prahové hodnoty pro algoritmus LBPH**

Vzhledem k velkým závislostem ukazatelů FRR a FAR na nastavení prahové hodnoty často dochází k opakovanému testování. Častou praktikou u systémů, které využívají zmíněné biometrické algoritmy, proto je na základě procesu hledání využití obou získaných hodnot (minimální skóre falešné shody i maximální skóre správné shody) a nastavení prahové hodnoty na hodnotu přesně mezi těmito hodnotami. Pro algoritmus **EigenFaces** je tedy na základě procesu hledání optimální prahová hodnota **3 847**, pro algoritmus **FisherFaces** **741** a pro algoritmus **LBPH** **92,5**.

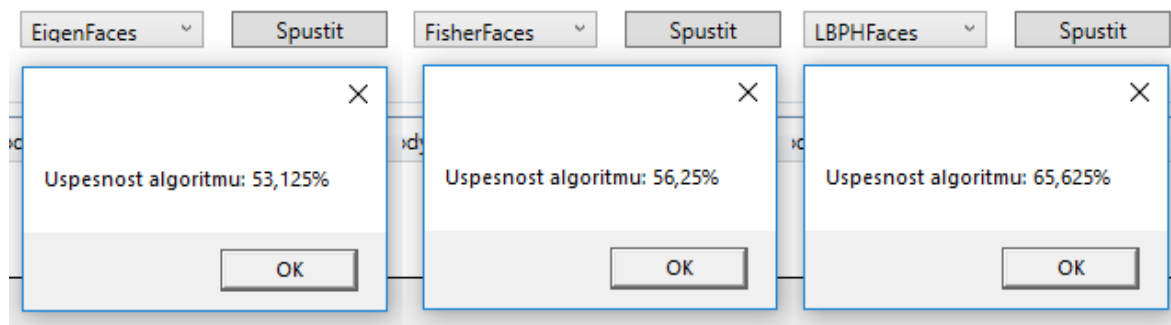
Takto nastavenou hodnotu je samozřejmě potřeba otestovat a v případě nepříznivých výsledků upravit. Samozřejmě je ideální minimalizovat oba chybové ukazatele (FRR i FAR), ale jak již bylo opakovaně zmíněno, snížením jednoho dochází ke zvýšení druhého.

## **4.2 Testování biometrických algoritmů**

Před samotným zahájením testování je třeba mít nastavenou prahovou hodnotu, abychom pracovali s „ostrou“ verzí pro každý algoritmus. Dalším předpokladem je trochu upravená sada testovacích tokenových fotografií, a to tím způsobem, že první část názvu každého souboru (před dělicím znakem \_) obsahuje identifikátor osoby, ke které fotografie patří. Tohoto lze rovněž dosáhnout pomocí upravené funkce aplikace pro zpracování obrázku, kde je v kontextové nabídce možnost „*Uložit pro test*“. Účelem je dosažení plně automatizovaného průběhu testování, kde je výsledek testování porovnán právě s první částí názvu souboru.

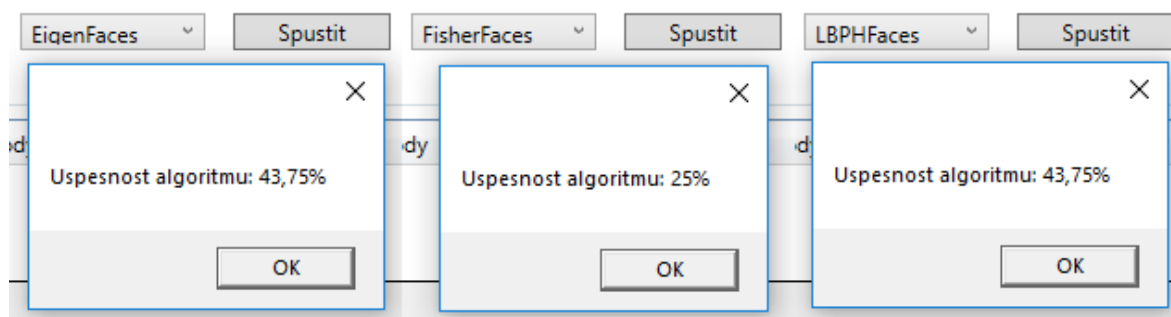
Začátek procesu je obdobný procesu hledání prahových hodnot. V horní části aplikace je stisknuto tlačítko pro výběr adresáře, po čemž následuje načtení názvů souborů do seznamu. Po zvolení požadovaného biometrického algoritmu k otestování je stisknuto tlačítko Test (pravá horní část aplikace, viz Obrázek 11). Toto testování tedy probíhá automaticky a délka běhu tohoto procesu tedy závisí na objemu testovaných dat (i na objemu referenčních dat, uložených

v databázi). Po dokončení testování se zobrazí informační okno s výslednou procentuální úspěšností (procento správně vyhodnocených snímků daným algoritmem).



**Obrázek 15 - Úspěšnosti algoritmů před nastavením prahové hodnoty**

Na Obrázek 15 je vidět stoupající tendence vzhledem k novějšímu algoritmu. Již v teoretické části bylo popisováno, že s postupem času se algoritmy stávaly dokonalejšími a spolehlivějšími. Přestože 65 % se může zdát jako poměrně nízká hladina úspěšnosti, je nutné práť v potaz různé podmínky, za kterých byly pořízeny vstupní snímky a relativně malý objem trénovačích setů, uložených v databázi. První testování bylo provedeno před nastavením prahových hodnot na hodnoty získané v předchozím oddílu.



**Obrázek 16 - Úspěšnosti algoritmů po nastavení prahových hodnot**

Z Obrázek 16 je patrný skoro až alarmující pokles procentuální úspěšnosti. Na první pohled by se mohlo zdát, že nastavením prahové hodnoty má opačný než kýžený účinek a že celý proces, popisovaný v předchozí kapitole je zbytečný. Opak je však pravdou, neboť přestože došlo ke zvýšení chyb typu False Reject (falešné odmítnutí, tedy nerozpoznání osoby, která je uložena v databázi), došlo ke značnému snížení chyb opačného typu, tedy False Accept (falešné přijetí – chybné přiřazení snímku osoby, které není v databázi k osobě, která se tam nachází).

Ve výchozím stavu jsou totiž algoritmy inicializovány s velmi vysokou hodnotou prahu, takže výsledkem rozpoznání nikdy není *Neznámá osoba*. Účelem není najít věrohodnou shodu, nýbrž nejbližší možnou. Tento postup je založen na popisovaném principu minimalizace

(Euklidovské) vzdálenosti bez dalších podmínek. Přidáním prahové hodnoty dojde k omezení maximální přípustné vzdálenosti, pro kterou je shoda považována za věrohodnou.

Bohužel však i u jedné osoby je pomocí různého úhlu a nasvícení pořídit 2 fotografie, které budou mít velmi vysokou hodnotu vzdálenosti (tzv. skóre shody), proto může dojít k falešnému zamítnutí (FR). Naopak nejsou výjimkou případy, kdy se jedna osoba natolik podobá jiné, že výsledná vzdálenost mezi snímky, pořízenými za podobných podmínek (světlo, úhel), je natolik nízká, že může projít pod hodnotou prahovou.

Možností řešení tohoto problému je více, z důvodu spolehlivosti a odolnosti vůči napadení je však většinou vyžadována součinnost operátora (uživatele – obsluhy). Existují systémy, ve kterých na konci určitého období dochází k manuální kontrole zamítnutých přístupů, přičemž pokud došlo k chybnému zamítnutí, je daný snímek osoby, která se již v databázi nachází, přidán do trénovacího setu za účelem možného vytvoření dokonalejšího vzoru – šablony dané osoby.

Jistou alternativou mohou být systémy, které mají nastavenou periodu, po které dochází k přidání do trénovacího setu. Tento přístup není založen na chybném zamítnutí, nýbrž naopak na správném přiřazení. Přičemž například při každém desátém správném přiřazení dojde k přidání správně rozpoznaného snímku do trénovacího setu pro daného uživatele. Tento proces je automatizovaný, což může představovat bezpečnostní riziko. Pokud by se totiž ve správný okamžik podařilo oklamat rozpoznávací systém, došlo by k zápisu podvržené fotografie do systému a postupně by pak bylo možné přetvořit uživatelskou šablonu na úplně jiný vzhled.