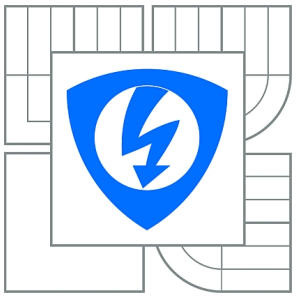
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

KLASIFIKACE TYPŮ VOZIDEL METODOU DYNAMICKÉHO BORCENÍ ČASU

DYNAMIC TIME WARPING FOR VEHICLE CLASSIFICATION

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

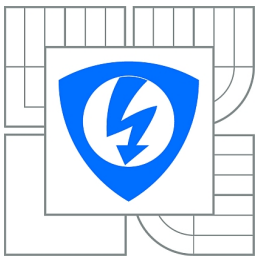
AUTOR PRÁCE — ALIAKSEI HALACHKIN

AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR HONZÍK, Ph.D.

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student:	Aliaksei Halachkin	***ID:***	152018
Ročník:	3	***Akademický rok:***	2014/2015

NÁZEV TÉMATU:

Klasifikace typů vozidel metodou dynamického borcení času

POKYNY PRO VYPRACOVÁNÍ:

Cílem bakalářské práce je seznámit se metodou dynamického borcení času (DTW) a jejími variantami, metodu naprogramovat a testovat na reálných datech a nakonec vytvořit laboratorní přípravek využívající metodu DTW pro klasifikaci typu vozidla podle zjištěného profilu.
1. Zpracujte rešerši různých variant algoritmu DTW a příkladů jeho praktického využití.
2. Naprogramujte metodu DTW a použijte ji na poskytnutých reálných datech na klasifikaci typu vozidla podle profilu získaného z laserové scanneru.
3. Vyhodnoťte přesnost dosažených výsledků.
4. Vytvořte laboratorní přípravek, na kterém bude možné uvedenou metodu prezentovat. Z praktických důvodů je možné nahradit měření profilu laserem alternativním způsobem.

DOPORUČENÁ LITERATURA:

da Costa Filho, A.C.B., de Brito Filho, J.P., de Araujo, R.E., Benevides, C.A.: Infrared-Based System for Vehicle Classification. In: 2009 SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference (IMOC), pp. 537–540 (2009).

Termín zadání:	9.2.2015	***Termín odevzdání:***	25.5.2015

Vedoucí práce: Ing. Petr Honzík, Ph.D.
Konzultanti bakalářské práce:

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato práce se věnuje metodě borcení času. Během práce byla napsaná C/Python knihovna, která je použita na klasifikaci typů vozidel podle profilů. Testování se provádělo na reálných datech z laserového skeneru. Algoritmus byl porovnán s korelací a Euklidovskou vzdáleností. Nakonec byl vytvořen laboratorní přípravek, který demonstruje rozpoznávání vozidel metodou borcení času.

KLÍČOVÁ SLOVA

DTW, metoda borcení času, rozpoznávaní typů vozidel, Python, Cython

ABSTRACT

This thesis focuses on the dynamic time warping. During the work was written C/Python library. Using this library, the algorithm was subsequently applied for the vehicle classification based on their shapes. Testing had been performed on real data from a laser scanner. Then the algorithm had been compared to the correlation and Euclidean distance. Finally, laboratory model had been created, which demonstrates vehicle recognition using dynamic time warping.

KEYWORDS

DTW, dynamic time warping, vehicle type recognition, Python, Cython

HALACHKIN, Aliaksei *Klasifikace typů vozidel metodou dynamického borcení času*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2015. 52 s. Vedoucí práce byl Ing. Petr Honzík, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Klasifikace typů vozidel metodou dynamického borcení času“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu semestrální práce panu Ing. Petru Honzíkovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci, společnosti Camea s.r.o. za poskytnutá data a Jakubu Korbelovi za gramatickou opravu.

Brno ....

(podpis autora)

OBSAH

ÚVOD

Metoda borcení času je široce používaný algoritmus, který se využívá například v rozpoznávání řeči, analýze EKG záznamů, rozpoznávání podpisů, analýze trhu atd. Tato práce stanovila cíl odpovědět na otázku, jestli se dá použít metoda dynamického borcení času pro rozpoznávání vozidel.

Práce je rozdělena v následujícím tvaru. První tři kapitoly jsou víc teoretické. Metoda borcení času je jeden ze základních algoritmů dynamického programování, proto první kapitola slouží jako stručný úvod do dynamického programování. Ve druhé kapitole se čitatel seznámí s klasickým algoritmem dynamického borcení času a jeho možnými nastaveními. Jsou popsané krokové funkce, globální omezení, váhy a normalizace. Třetí kapitola se věnuje aplikaci algoritmu. Jsou probrané tři obory, ve kterých má algoritmus největší popularitu: biomedicína, rozpoznávání řeči a rozpoznávání podpisu. Taky je analyzována témata této práce – klasifikace typu vozidel.

Další kapitoly jsou více praktické. Čtvrtá kapitola uvádí možnosti naprogramované C/Python knihovny, obsahuje návod pro překlad a popisuje její rozhraní. Pátá kapitola popisuje, jak byl algoritmus borcení času aplikován pro klasifikaci typů vozidel. Uvádí se úspěšnost klasifikace pomocí metody borcení času a její porovnání s korelací a Euklidovskou vzdáleností. V šesté kapitole je popsán vytvořený laboratorní přípravek.

1 DYNAMICKÉ PROGRAMOVÁNÍ

Dynamické programování je jedna z metod optimalizace. To je způsob řešení úloh, který spočívá v rozkladu složité úlohy na několik jednodušších úloh. Hlavní myšlenka dynamického programování je prostá. Z pravidla je pro řešení úlohy potřeba řešit podúlohy, potom zkombinovat řešení dílčích úkolů do jednoho společného řešení. Často je většina z podúloh stejná. Cílem dynamického programování je řešit každou podúlohu jenom jednou, čímž se sníží náročnost řešení. Problémy se dělí na menší podproblémy, dokud se nedojde do triviálního případu, který se řeší v konstantním čase. V dynamickém programování se používají následující vlastnosti úlohy: [1]:

- podúlohy se opakují
- optimální struktura

Úloha má optimální strukturu, když její optimální řešení lze sestavit z optimálního řešení triviálních případů. Řešení problémů se dělí na tři základní kroky [1]:

- definování podúloh
- nalezení opakování, které se vztahuje na podproblémy
- řešení bázových případů

Typické problémy, které se řeší pomocí dynamického programování, jsou Fibonacciho posloupnost, násobení matic, problém batohu, nejdelší společná podposloupnost, nejkratší cesty a jiné. Typické algoritmy pro takové úlohy jsou rekurzivní, což pro relativně velká vstupní data vede ke zpomalení programu a přetečení zásobníku. Hlavní přínos dynamického programování je tedy nahrazení pomalého rekurzivního algoritmu na řešení v polynomiálním množství času. Algoritmy dynamického programování jsou polynomiální v čase, pokud počet podproblémů lze vyjádřit polynomem.

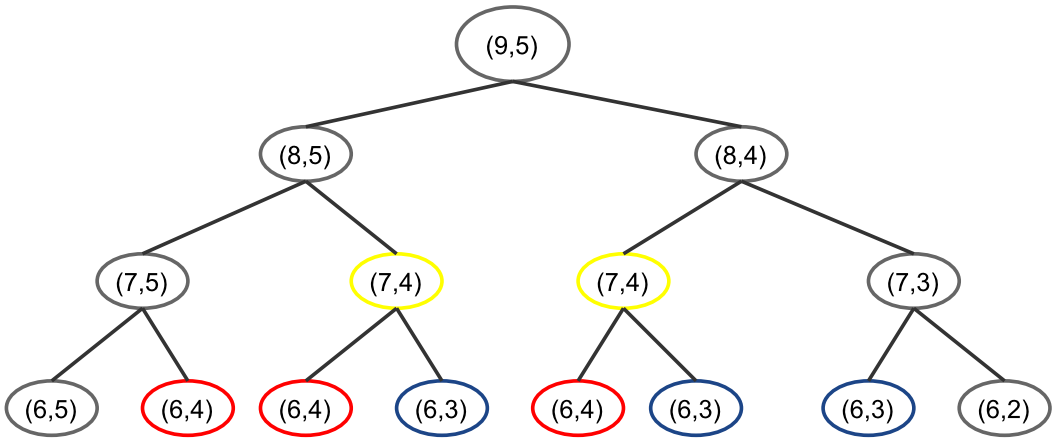
Příkladem nevýhody rekurzivního algoritmu může sloužit algoritmus pro počítání kombinace bez opakování [2]. Počet kombinace k-té třídy z n prvků bez opakování se dá vyjádřit matematický následovně:

$$C_k = \binom{n}{k} = \frac{n!}{k!(n-k)!} = \binom{n-1}{k} + \binom{n-1}{k-1} \tag{1.1}$$

Jedno z možných rekurzivních řešení je algoritmus 1. Kromě hluboké rekurze při velkém rozdílu $n - k$, z obrázku 1.1, který znázorňuje průběh rekurze algoritmu 1, je vidět, že funkce se několikrát rekurzivně volá se stejnými vstupními hodnotami, čímž se snižuje efektivita. Dynamické programování je schopno řešit tento problém pomocí pamatování mezních výpočtů. Přístup s použitím dynamického programování je uložení mezních výpočtů do matice. Jedno z možných řešení je algoritmus 2. Rekurzivní algoritmus má výpočetní složitost $O(2^{(n-k)})$, algoritmus dynamického programování má složitost jenom $O(nk)$.

Algoritmus 1: Počítáni kombinace bez opakování, rekurzivní řešení

```
1 def NchooseK(n,k):
      Data: n - rozměr množiny, k - rozměr podmnožiny
      Result: počet kombinace k-té třídy z n prvků bez opakování
2     if k = 0:
3         return 1
4     elif n = k:
5         return 1
6     else:
7         return  NchooseK(n − 1,k) +  NchooseK(n − 1,k − 1)
```



Obr. 1.1: Několik kroků výpočtu $\binom{9}{5}$.

Algoritmus 2: Počítáni kombinace bez opakování

```
1 def NchooseK(n,k):
      Data: n - rozměr množiny, k - rozměr podmnožiny
      Result: počet kombinace k-té třídy z n prvků bez opakování
2     C(0, 0) ⟵ 1
3     C(i, 0) ⟵ 1 pro i = [1 : n + 1]
4     C(0, j) ⟵ 0 pro j = [1 : k + 1]
5     for i ← 1 to n + 1:
6         for j ← 1 to k + 1:
7             C(i, j) ⟵ C(i − 1, j − 1) + C(i − 1, j)
8     return C(n,k)
```

2 DTW

Kapitola je zaměřena na popis principu metody DTW (metoda borcení času - překlad anglického názvů Dynamic Time Warping, zkrátka DTW), je popsané klasické DTW a varianty DTW (krokové funkce, globální omezení cesty, normalizace, váhy).

2.1 Klasické DTW

2.1.1 Formulace problému

Jedním z problémů, pro který byla úspěšně použita metoda dynamického programování, je porovnání dvou sekvencí. Krátký popis problému vypadá následovně. Na začátku se musí vybrat metrika, na základě které se dál sestavuje hodnoticí dvourozměrná matice. Každý prvek v matici odpovídá hodnotě porovnání příslušných prvků sekvencí. Celá matice představuje tedy všechny možné kombinace porovnání prvků dvou sekvencí. Cílem je najít optimální cestu od prvního prvku matice do posledního, kde se za optimální cestu považuje cesta s minimálním součtem hodnot prvků. Následuje formální popis problému [3, 4].

Distanční funkce zvolené metriky v kontextu dynamického programování se nazývá nákladová funkce. Za nákladovou funkci se často volí Manhattan vzdálenost:

$$c \in \mathbb{R} : \ c(a,b) = |a-b| \ \ a,b \in \mathbb{R}, \tag{2.1}$$

Nechť jsou dány dvě sekvence $r \in \mathbb{R}^N$ a $q \in \mathbb{R}^M$. Na základě zvolené metriky se sestavuje lokální nákladová matice:

$$C \in \mathbb{R}^{N \times M} : \ c_{ij} = c(r_i, \ q_j) \tag{2.2}$$

kde: $i \in \ [0 : N-1]$ a $j \in \ [0 : M-1]$ jsou indexy lokální nákladové matice. Transformační funkce w vyjadřuje zarovnání dvou sekvencí a definuje transformační cestu p.

$$w_l = (i_l, j_l) \ pro \ l \in [0 : L-1] \tag{2.3}$$

$$p = [w_0 : w_{L-1}] \tag{2.4}$$

Transformační cesta musí splňovat tři podmínky:

1. hraniční podmínka: $w_0 = (0,0)$ a $w_{L-1} = (N-1, M-1)$
2. cesta musí byt monotonní: $i_0 \leq i_1 \leq ... \leq i_{L-1}$ a $j_0 \leq j_1 \leq ... \leq j_{L-1}$
3. maximální rozměr kroku je jeden: $w_{l+1} - w_l \in \Big\{(0,1),(1,0),(1,1)\Big\}$

První dvě podmínky platí vždy, když třetí podmínka platí v rámci klasického DTW. Kumulována vzdálenost (vzdálenost DTW) je definovaná následovně:

$$c_p = \sum_{k=0}^{L-1} C(i_k, j_k),\ (i_k, j_k) \in p \tag{2.5}$$

Optimální transformační cesta je cesta s nejmenší kumulovanou vzdáleností. Metoda borcení času potřebuje dvě sekvence jako vstupní veličiny. Výstupem je optimální cesta a vzdálenost dvou posloupností. Počet cest s rozměrem matice roste exponenciálně. Nalezení optimální cesty je kombinatorický optimalizační problém, který se řeší hrubou silou v exponenciálním čase. DTW za pomocí dynamického programování nabízí řešení se složitostí $O(NM)$.

2.1.2 Algoritmus DTW

Algoritmus DTW zavádí pojem globální nákladové matice (jiný název je matice kumulativních vzdáleností) D. První řádek a sloupec jsou definovány následovně:

$$D(i, 0) = \sum_{k=0}^{i} c(r_k, q_0) \text{ pro } i \in [0 : N-1] \tag{2.6}$$

$$D(0, j) = \sum_{k=0}^{j} c(r_0, q_k) \text{ pro } j \in [0 : M-1] \tag{2.7}$$

Zbývající prvky se spočítají následujícím způsobem:

$$D(i, j) = min\Big\{D(i-1, j),\ D(i, j-1),\ D(i-1, j-1)\Big\} + c(r_i, q_j) \tag{2.8}$$

Klíčovou vlastností matice kumulativních vzdáleností je, že každý její prvek vyjadřuje optimální kumulativní vzdálenost, oříznutých do pozice daného prvku v matici posloupnosti r a q(důkaz lze najít v [3, str. 72]). Tedy:

$$D(i, j) = c_p^*(r(0 : i), q(0 : j)), \text{ kde:} \tag{2.9}$$

$$c_p^*(r(0 : i), q(0 : j)) = min\Big\{c_p(r(0 : i), q(0 : j)),\ p \text{ je transformační cesta}\Big\} \tag{2.10}$$

Poslední prvek matice kumulativních vzdáleností *D(N-1,M-1)* je optimální vzdálenost mezi posloupnostmi a jeden z výstupů algoritmu DTW. Druhý výstup je transformační cesta p, která se dá najít pomocí zpětného trasování z bodu *(M-1,N-1)* do *(0,0)*. Transformační cesta a optimální vzdálenost v případě Manhattan

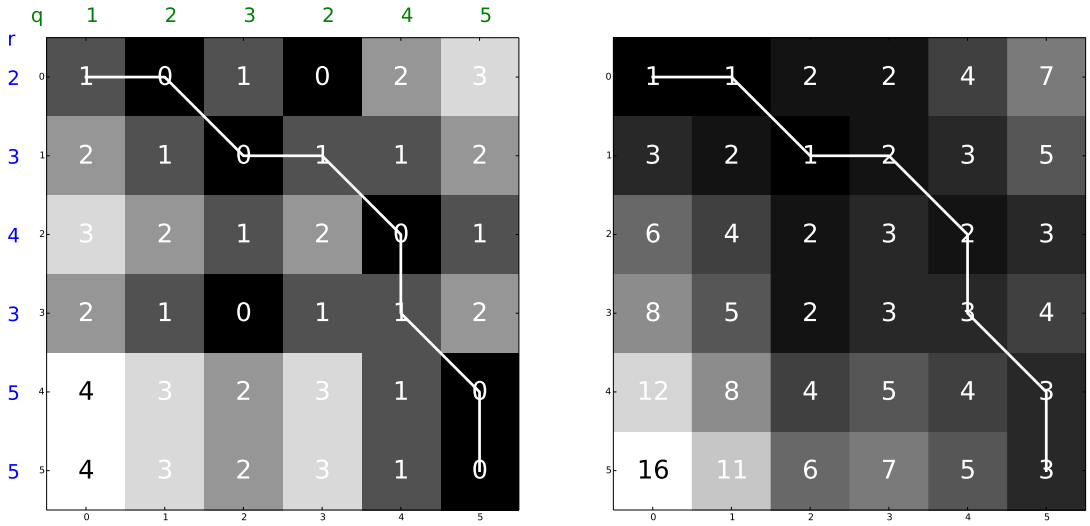
vzdálenosti jsou svázaný následujícím vztahem:

$$dtw(r,q) = D(N-1, M-1) = \sum_{k=0}^{L-1} |r(i_k) - q(j_k)|, \ (i_k, j_k) \in p \tag{2.11}$$

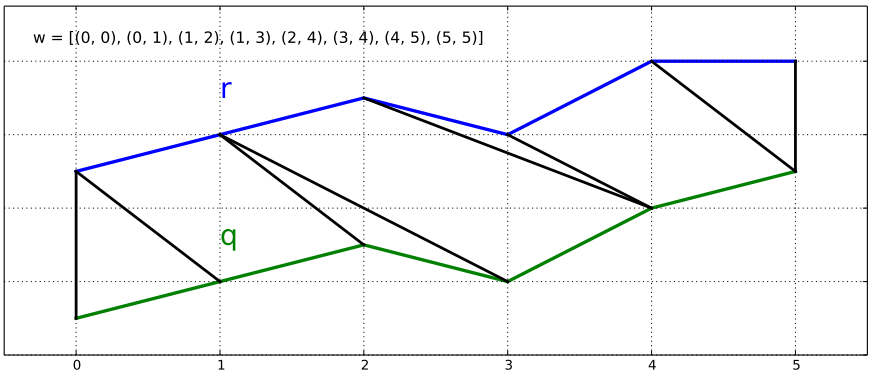
Optimální cesta není deterministická, jako extrémní příklad lze uvést nulové posloupnosti r a q, kde libovolná cesta bude optimální. Optimální cesta závisí na volbě Manhattan nebo Euklidovské vzdálenosti. Pokud je potřeba použit Euklidovskou vzdálenost, je potřeba změnit distanční funkci 2.1: $c(a,b) = (a-b)^2$. Pak vzdálenost DTW bude:

$$dtw(r,q) = \sqrt{D(N-1, M-1)} = \sqrt{\sum_{k=0}^{L-1} (r(i_k) - q(j_k))^2}, \ (i_k, j_k) \in p \tag{2.12}$$

Pro optimalizaci algoritmu se používá druhá mocnina Euklidovské vzdálenosti, kde není potřeba počítat odmocninu (má smysl při hledání subsekvencí).



Obr. 2.1: Lokální a globální nákladové matice s optimální cestou



Obr. 2.2: Znázornění zarovnání dvou sekvencí

Algoritmus zpětného trasování v případě rekurzivního výpočtu globální nákladové matice podle vztahu 2.1.2 může být jednoduchý gradientní sestup. Ovšem při použití modifikovaných krokových funkcí nebo váh (kapitola 2.2) je potřeba obecnějšího zpětného trasování, příkladem může sloužit algoritmus 3.

Algoritmus 3: Algoritmus DTW

```
1  def DTW(r,q):
       Data: r ∈ ℝ^N, r ∈ ℝ^M vstupní posloupnosti r a q
       Result: d, p distance DTW a optimální cesta
2      UP ← 0, LEFT ← 1, DIAG ← 2
3      D(0,0) ← |r(0) − q(0)|
4      for j ← 1 to M:
5          D(0,j) ← |r(0) − q(j)| + D(0,j−1)
6          Direction(0,j) ← LEFT
7      for i ← 1 to N:
8          D(i,0) ← |r(i) − q(0)| + D(i−1,0)
9          Direction(i,0) ← UP
10     for i ← 1 to N:
11         for j ← 1 to M:
12             d ← |r(i−1) − q(j−1)|
13             D(i,j) ← d + min{D(i−1,j), D(i,j−1), D(i−1,j−1)}
14             Direction(i,j) ← argmin{D(i−1,j) + d, ...
15                     D(i,j−1) + d, D(i−1,j−1) + d}
16     i ⟵ M, j ⟵ N
17     p ⟵ new list
18     p.append((i,j))
19     while i > 0 and j > 0:
20         if i = 0:
21             j ← j − 1
22         elif j = 0:
23             i ← i − 1
24         else:
25             if Direction(i,j) = UP:
26                 i ← i − 1
27             elif Direction(i,j) = LEFT:
28                 j ← j − 1
29             else:
30                 i ← i − 1, j ← j − 1
31             p.append((i,j))
32     return D(M−1, N−1), p
```

2.2 Varianty DTW

Optimální cesta často nevyjadřuje reálnou situaci, proto počet možných cest se uměle omezuje nebo se zavádějí jiné modifikace pro lepší přizpůsobení algoritmu realitě. V dalších podkapitolách jsou popsané možnosti změny krokové funkce, zavedení globálního omezení, vah a normalizace vzdálenosti DTW.

2.2.1 Kroková funkce

Jedno z možných nastavení algoritmu DTW je kroková funkce. Krokovou funkcí se myslí vztah 2.1.2, který zadává způsob sestavení globální nákladové matice. Jiný název je lokální omezení. Počet typů možných krokových funkcí je neomezený, známé funkce, které prokázaly dobré výsledky v určitých úlohách, se dají rozdělit podle autorů. Existují klasifikace podle Rabiner-Juang [5], Sakoe-Chiba [6], a Rabiner-Myers [7] [8]. Komplikované krokové funkce zavádí změny v algoritmu a značně zvětšují náročnost výpočtu matice kumulativních vzdáleností.

Motivací ke změně krokové funkce jsou problém degradace transformační cesty v některých případech nebo fakt, že diagonální krok stojí dva kroky nediagonálních. Degradace cesty spočívá v její příliš velké strmosti. Důsledkem je porovnání bodů jedné sekvence s příliš velkým počtem bodů druhé sekvence a zanedbávání ostatních bodů.

Při použití krokové funkce odlišné od 2.1.2 nelze hledat optimální cestu gradientním sestupem a mění se definice kumulované vzdálenosti 2.5, která závisí na krokové funkci.

Krokové funkce lze rozdělit na funkce neomezující a omezující lokální strmost cesty a taky na symetrické a nesymetrické. Funkce neomezující strmost cesty zaplňují celou globální matici. Symetrické krokové funkce jsou nezávislé na volbě řádkových a sloupcových indexů, $dtw(r, q) = dtw(q, r)$. Dvě široce používané symetrické neomezující krokové funkce jsou definovány následovně:

$$D(i,j) = min \begin{cases} D(i-1,j) + c(r_i, q_j) \\ D(i,j-1) + c(r_i, q_j) \\ D(i-1,j-1) + 2c(r_i, q_j) \end{cases} \tag{2.13}$$

$$D(i,j) = min\Big\{D(i-1,j), D(i,j-1)\Big\} + c(r_i, q_j) \tag{2.14}$$

První kroková funkce nutí DTW udělat krok doleva nebo nahoru, když diagonální krok stojí víc. Druhá kroková funkce nepovoluje diagonální krok vůbec. Kroková funkce 2.13 není časově reverzibilní, výsledek se mění pří invertování pořadí prvků v sekvenci. Funkce 2.14 a 2.1.2 jsou naopak časově reverzibilní.

Pří omezení strmosti cesty určité prvky matice D zůstanou prázdné. Za příklad funkce omezující strmost cesty může sloužit následující vztah [3, str.75]:

$$D(i,j) = min\Big\{D(i-1,j-1), D(i-2,j-1), D(i-1,j-2)\Big\} + c(r_i, q_j) \quad (2.15)$$

Když existuje předpoklad o určitém maximálním možném zpoždění v signálech, používají se funkce podobné 2.15. Tady DTW vždycky udělá krok doleva, tím se eliminuje její příliš velká strmost. Pro danou krokovou funkci je cesta ohraničená přímkami: $j < 2i$, $i < 2j$, $i \geq N - 2(M - j)$ a $j > M - 2(N - i)$. Pro implementaci zpětného trasování takových funkcí je potřeba uvažovat, že cesta musí spojovat body maticí D, které se používají pro rekurzivní výpočet. V daném případě jsou povolené předcházející body $(i-1, j-1)$, $(i-2, j-1)$, $(i-1, j-2)$. Matici D je potřeba rozšířit o dva řádky a sloupce tak, že $D(-1 : N, -1 : 1) = \infty$ a $D(-1 : 1, -1 : M) = \infty$.

2.2.2 Váhy

Váhami se myslí prvky vektoru (a, b, c) ze vztahu 2.16. Váhy tvoří podmnožinu množiny krokových funkcí s tím rozdílem, že ovlivňují jenom *cenu* kroků, prvky v matici D potřebné pro rekurzivní výpočet se nemění. Například vektor $(1, 1, 1)$ vede ke klasickému DTW, vektor $(1, 1, 2)$ je případ 2.13 a $(1, 1, \infty)$ odpovídá 2.14.

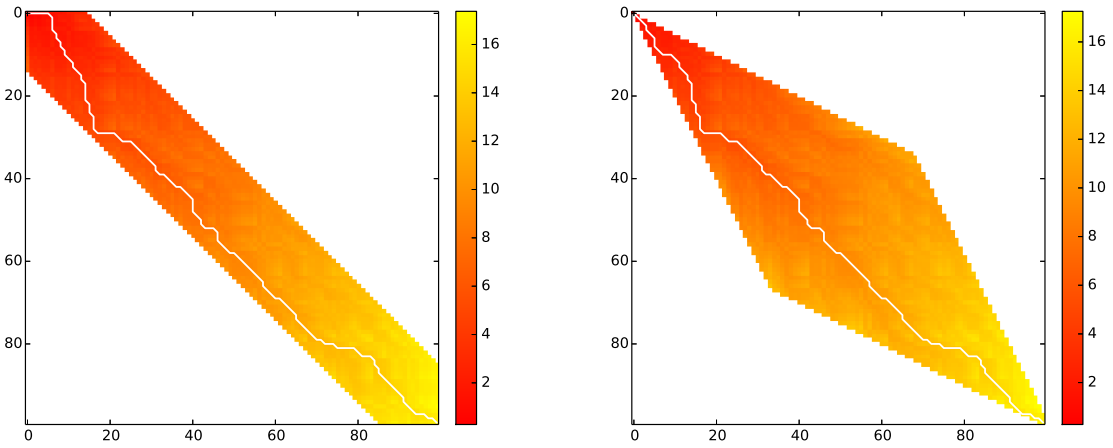
$$D(i,j) = min \begin{cases} D(i-1,j) + a \cdot c(r_i, q_j) \\ D(i,j-1) + b \cdot c(r_i, q_j) \\ D(i-1,j-1) + c \cdot c(r_i, q_j) \end{cases} \quad (2.16)$$

2.2.3 Globální omezení cesty

Globální omezení cesty spočívá v zákazu určitých prvků nákladové matice vytvořením tzv. *okna*, ve kterém se může nacházet cesta. Důvody podporující využití globálního omezení cesty jsou dva. Zaprvé je to zrychlení algoritmů, když není potřeba „otvírat" každý prvek v matici. Ovšem vhodná implementace může být poměrně složitá pro typy omezení složitější než 2.18. Zadruhé je možné zlepšení kvality výsledků, když je odhad o velikostech možného zpoždění v naměřených signálech.

Dva populární druhy globálního omezení cesty jsou *Sakoe-Chiba band* [6] a *Itakura parallelogram* [3]. První pásmo je určeno vztahem:

$$|i - j| < R \text{ , kde } R \in \mathbb{Z} \text{ je šířka dovoleného pásma} \quad (2.17)$$



(a) Sakoe-Chiba band

(b) Itakura parallelogram

Obr. 2.3: Globální omezení cesty

Definice 2.17 má nevýhodu v tom, že je funkční, jenom když rozdíl délek je menší než velikost okna $|M - N| < R$, proto se zavádí obecnější definice. Následující typ okna se nazývá podle jména autoru článku[9] – *Palival adjustment window*:

$$|\frac{M}{N}i - j| < R \tag{2.18}$$

Během této práce bylo zjištěno, že pokud v databázi jsou sekvence různé délky, je dobře zobecnit R pro libovolné délky posloupností a šířku pásma R definovat přes délku referenční sekvencí: $R = pN$. Možná implementace je ukázaná v algoritmu 4.

Algoritmus 4: Výpočet globální nákladové matice s globálním omezením

```
1  def costMatrix(r,q,p):
       Data: r ∈ ℝ^ℕ, r ∈ ℝ^𝕄, p ∈ ℝ  vstupní posloupnosti a p = w/N, kde w je
             šířka pásma
       Result: D ∈ ℝ^{ℕ×𝕄}  globální nákladová matice
2      for i ← 0 to N + 1:
3          for j ← 0 to M + 1:
4              D(i, j) ← ∞
5      w = pN
6      D(0, 0) ← 0
7      s = M/N
8      for i ← 1 to N + 1:
9          for j ← max{1, round(s(i − w))} to min{M, round(s(i + w))}:
10             D(i, j) ← abs(r(i − 1) − q(j − 1)) + ...
11                       min{D(i − 1, j), D(i, j − 1), D(i − 1, j − 1)}
12     return D(1 : N + 1, 1 : M + 1)
```

Okno *Itakura parallelogram* představuje všechny možné cesty, které mají strmost mezi $1/S$ a S. Například použitím krokové funkce 2.15 dostaneme pásmo se stejnou formou, jak pro $S = 2$ [3, str. 76]. Poměr délek vstupních posloupností v tomto případě musí ležet v rozmezí $\frac{M}{N} \in (\frac{1}{2}, 2)$.

2.2.4 Normalizace

Optimální kumulativní vzdálenost 2.10 lze normalizovat počtem kroků cesty:

$$c_p^{opt}(r,q) = \frac{D(M-1, N-1)}{L} \text{ , kde L je délka optimální cesty } p \text{ 2.4} \qquad (2.19)$$

Někdy není potřeba počítat délku cesty, třeba pro krokovou funkci 2.14 je zřejmé, že $L = M + N$. Normalizace vzdálenosti DTW počtem kroků může mít negativní vliv. Větší počet kroků po menších vzdálenostech může v součtu vést k příliš dlouhé cestě, normalizace počtem kroků však může vést k tomu, že tato cesta bude považována za nejkratší možnou. Proto se často pro krokovou funkci zavádí *normalizační faktor* a finální vzdálenost se normalizuje hodnotou závislou na délkách vstupních posloupností r a q. Například pro všechny krokové funkce z kapitoly 2.2.1 je normalizační faktor roven $N + M$.

3 APLIKACE DTW

Algoritmus DTW se teoreticky dá použit na cokoliv, co je možné vyjádřit ve tvaru posloupností čísel nebo znaků. Především má popularitu v oblastech rozpoznávání řeči, biomedicíně, rozpoznávání podpisů a rukopisů, vyhledávání v textu, dolování sekvenčních dat, rozpoznávání pohybu těla. Předpokladem úspěšného použití DTW je splnění dvou podmínek:

- testovací a referenční řady musí mít stejnou vzorkovací frekvenci
- hodnoty sekvencí musí mít stejnou prioritu

V následujících podkapitolách jsou popsané nejrozšířenější možnosti aplikace metody borcení časů.

3.1 Biomedicína

Použití DTW v biomedicíně se dělí na dva druhy: zarovnání DNA a RNA sekvencí a analýza EKG.

3.1.1 Analýza RNA a DNA sekvencí

RNA a DNA sekvence nejsou časové řady a nejsou číslicové posloupnosti ale posloupnosti znaků. Použití DTW ve smyslu borcení času tady není možné. Používají se algoritmy Smith-Waterman a Needleman-Wunsch, oba jsou založené na principech dynamického programování a jsou velmi podobné DTW.

DNA má abecedu $\sum = \{A, C, T, G\}$, a RNA $\sum = \{A, C, U, G\}$. Úlohou je provést sekvenční alignment (zarovnání) a nalézt podobnost sekvencí. Z těchto dat se dá dostat informace například o evolučních vztazích. Pro sekvenční alignment jsou definované tři operátory: mazání (deletion), vložení (insertion) a výměna (substitution). Zarovnané sekvence musí mít stejnou délku. Znaky musí mít stejné pořadí jako u originálu a prázdné prvky „–" se nesmí nacházet pod sebou.

Sekvenční alignment dvou sekvencí $X = ACCT$ a $Y = TACGGT$ lze provést následovně [10]:

-	A	C	-	C	T
T	A	C	G	G	T

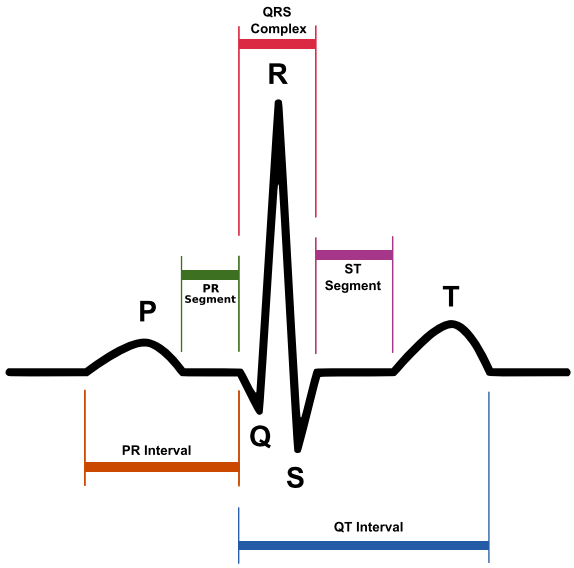
Pro daný případ v první sekvenci jsou vymazané první a čtvrtý prvek a nahrazen pátý. Samozřejmé pro dané sekvence existuje víc možností jak provést zarovnání. Počet možných variant je velký, zajímavý je ale jenom optimální alignment. Hodnocení zarovnání se provádí analogický klasickému DTW, podle lokální nákladové matice. Sestavit tuto matici lze různými způsoby. Lze hodnotit podle operátorů, například

často se zavádí hodnoticí systém, kde shoda dostává +1, nahrazení a mazání dostává −1. Druhá pokročilejší varianta je sestavení celé hodnoticí matice prvky, které jsou závislé na konkrétních znacích. Optimální zarovnání je zarovnání, které dostává nejvíc bodů podle hodnoticí matice. Dál algoritmus je úplně stejný jako v klasickém DTW, sestavuje se globální nákladová matice a vypočítá se optimální cesta (v tomto kontextu „optimální" znamená s největší cenou).

3.1.2 Analýza EKG

Ke druhé oblasti biomedicíny, kde DTW má potenciál využití, patří analýza EKG signálů [12].[13].

Analýza EKG je v dnešní době aktuální problém a spočívá v klasifikaci srdečních cyklů a rozpoznávání anomálií, které jsou například tachykardie(zvýšená frekvence srdce), bradykardie(snížená frekvence srdce) nebo extrasystoly(aktivace komor mimo pravidelný rytmus). V praxi má automatická klasifikace vyznám nejspíš při monitorování práce srdce pomocí holteru. Holter je zaznamenáván po dobu 24 hodin. Klasifikace cyklů zaprvé uspoří lékaři čas, protože se může dívat už přímo na anomální části záznamu a zadruhé slouží ke spuštění alarmů při anomální situaci. Problém má kritický význam, když se jedná o defibrilátory, které mohou začít elektroterapie po chybném hlášení.



Obr. 3.1: Běžný průběh signálu EKG [Převzato z *Wikipedia.org*]

EKG záznam je nasnímaný průběh napětí, proto je potřeba jeho předzpracování. Signál se filtruje pro odstranění šumu a driftu. Dál je EKG signál potřeba segmentovat na cykly. Segmentaci lze provést prahováním, za předpokladů, že poměr PR a

QT intervalů je konstantní a celou periodu lze určit z délky mezi R vlnami [12] [13]. Zároveň se používá DTW pro subsekvenci, tedy v segmentace není potřeba (algortmus je popsán v [3, str. 79]). V analýze EKG se používá taky DDTW[1] algoritmus. DDTW oproti DTW pracuje s derivací signálů. Používá se globální omezení ve tvaru 2.17.

3.2 Rozpoznávání řeči

Rozpoznávání řeči je stále otevřený problém. Nejrozšířenější nástroje jsou skryté Markovské modely a DTW. DTW v rozpoznávání řeči se používá pro zpracování povelů s relativně malým slovníkem. Model na základě DTW je závislý na mluvčím.

Předzpracování akustického signálu je podobné jako pro EKG záznamy, spočívá ve filtraci a segmentaci. Kromě segmentací řečového signálu do slov se při zpracování používají tzv. překrývající se okna, která mají délky řadové 5-10ms. Existují tři možnosti aplikace algoritmu DTW [11].

První spočívá v použití přímo číslicových hodnot dvou signálů. Vzorkovací frekvence audio snímače je vysoká (typická hodnota 44 kHz). Sekvence z důvodu výpočetní náročnosti je potřeba redukovat v rozměrech, smazat některé prvky nebo aproximovat. Úloha je pak ideální pro algoritmus borcení času, slovo může byt roztažené v čase (všichni mluví s různou rychlostí) nebo posunuté, tyto dva problémy je algoritmus schopen řešit. Problém ale existuje v amplitudě, ten se řeší normalizací signálů nebo použitím DDTW.

Alternativa prvního způsobu – výpočet spektra signálu pomocí diskrétní Fourierové transformaci a následná aplikace DTW na složky spektra. Frekvenční spektrum se upravuje na energetické, které je vstupem pro algoritmus borcení času.

Aplikace DTW přímo na vzorkovaný akustický signál nebo jeho frekvenční spektrum je ale stále neefektivní z pohledu výpočetní náročnosti. Pokročilejší metoda je použití mel-frekvenčních cepstrálních koeficientů[2]. Teoretický základ a postup výpočtu MFCC je možné najít například v [14]. Celý postup výpočtu distancí mezi signály je následující:

- transformace signálů z časové oblasti do frekvenční – FFT
- převod Hertzů na Mely
- výpočet energií pro každou Mel-frekvenci a převod těchto energií do logaritmické osy
- pomocí kosinové transformace (DCT) naležení mel-frekvenčních cepstrálních koeficientů

[1] Derivative Dynamic Time Warping
[2] angl. Mel-frequency cepstrum coefficients, zkr. MFCC

- na vstup DTW je možné dávat ohraničený počet MFCC

Použití MFCC je velmi časově efektivní, protože sekvence máji délku maximálně pár desítek prvků.

3.3 Rozpoznávání podpisů

Metody ověření podpisů se dělí na dva druhy: dynamické a statické. Dynamický nebo taky online metoda předpokládá, že člověk se podepisuje na grafickém tabletu a extrakce hodnot je v reálném čase. Statická nebo offline metoda je založena jenom na obrázku podpisu, když člověk se podepisuje na papír a podpis se následně digitalizuje. Z pohledu rozpoznávání je online varianta výhodnější, protože se dá dostat víc charakteristik než jenom dvourozměrná matice. Online jsou obvykle k dispozici koordináty $x(t)$ a $y(t)$, tlak pera na tablet a azimut.

Článek [15] se věnuje problému volby frekvence vzorkování pří online rozpoznávání. Předpoklad úspěšného použití DTW je stejná vzorkovací frekvence. V článku je popsána modifikace DTW pro případ různých vzorkovacích frekvencí. Rozdíl oproti klasickému DTW je v sestavení globální nákladové matice [15, str.655]:

$$D(i,j) = min \begin{cases} D(i-1, j-(F-1)) + c(r_i, q_j) \\ D(i-1, j-F) + c(r_i, q_j) \\ D(i-1, j-(F+1)) + c(r_i, q_j) \end{cases} \tag{3.1}$$

Kde $F = \frac{f_q}{f_r}$ vyjadřuje poměr testovací vzorkovací frekvence a referenční.

3.4 Klasifikace typů vozidel

Práce [16] se věnuje klasifikaci typů vozidel z profilů naměřených pomocí snímače založeném na infračerveném záření a DTW. Autoři uvádějí možnost využití takového způsobu rozpoznávání, ovšem vzhledem k rozměru databází profilů neuvádějí konkrétní úspěšnost metody.

Práce [17] aplikuje algoritmus DTW na elektromagnetické profily vozidel. Elektromagnetické profily jsou závislé na kovové hmotnosti vozidla. Vozidla byla rozdělena do 9-ti tříd (každá třída obsahovala 54 profilů), autoři udávají nejlepší výsledek při použití metody DTW, přičemž klasifikační chyba se pohybovala v rozmezí 1,3% až 50,9%.

4 KNIHOVNA DTW

Během této práce byla naprogramovaná knihovna pro práci s metodou borcení času[1]. Knihovna je napsaná pro Python. Volba Python souvisí s jeho vysokou mírou abstrakce, velkou komunitou a popularitou v dolování dat, kde se využívá DTW.

V průběhu vývoje knihovny se rychle ukázalo, že implementovat DTW v „čistém" Pythonu nebude to správné řešení z pohledu efektivity (Python je skriptovací programovací jazyk, který má dynamickou typovou kontrolu). Bylo zjištěno, že výpočetní část programu musí být v nízkoúrovňovém jazyce, za který byl zvolen jazyk C. Psaní C-rozšíření v Pythonu je běžná praxe pro matematické výpočty, velká část matematických nástrojů má realizaci v C. Celkem je pět populárních možností psaní C-rozšíření pro Python:

- nativní Python API
- Cython
- ctypes - externí knihovna, která umožňuje použití datových typů z jazyka C
- Pyrex
- SWIG

Ve výsledku byla zvolená druhá varianta. Při použití C-rozšíření rychlost výpočtu oproti Python-implementaci vzrostla cca 100 krát, pro vstupní sekvence délkou 1000 prvků je čas výpočtu cca 40ms a pro délky 8000 prvků je 2s. Testování se provádělo na počítači s procesorem Intel Core i5-3337U 1.8 GHz. Rychlost závisí na konkrétním nastavením DTW.

4.1 Cython

Cython[18] je programovací jazyk, který se snaží propojit to nejlepší z C/C++ a Pythonu. Má velmi podobnou syntaxi jako Python, ale ve srovnání s Pythonem se kód v Cythonu kompiluje a má statickou typovou kontrolu. Cython zároveň podporuje skoro libovolný Python-kód, což tvoří z něho neplnou realizaci Python. Cython umožňuje optimalizovat Python-kód použitím statických datových typů. Ve výsledku kód v Cythonu má rychlost skoro stejnou jako kód v C, protože v podstatě je to kód v C jenom s „syntaktickém cukrem". Cython také umožňuje jednoduché volání kódu z C/C++ v Pythonu a naopak. Poslední vlastnost byla klíčová. Hlavní myšlenkou bylo psát nezávislý kód v C a pak použít Cython jako rozhraní pro C v Pythonu.

[1] <https://github.com/honeyext/cdtw>

4.2 Návod na použití

4.2.1 Překlad Cython

Zdrojové soubory Cython mají koncovku `.pyx`, z nich Cython překladač generuje C-kód, který již obsahuje Python API. Z C-kódu překladač jazyka C vytváří sdílenou knihovnu `.so` v Linux nebo `.pyd` ve Windows. Je důležité překládat knihovnu stejným překladačem, který byl použit pro překlad Pythonu. Knihovna je napsaná pro standard C99, překlad byl otestován pomocí GCC 4.8.1 a MS Visual C 2008, ovšem překlad by měl fungovat libovolným překladačem, který podporuje C99. Další závislosti jsou následující (uvedené verze byli použité při vývoji a testování):

- Python 2.7.6
- Cython 0.22
- numpy 1.9.2 [2]
- matplotlib 1.3.1 [3]

Překlad nastavuje soubor `setup.py`, který při použití GCC musí nastavit proměnnou `ext_modules` následovně:

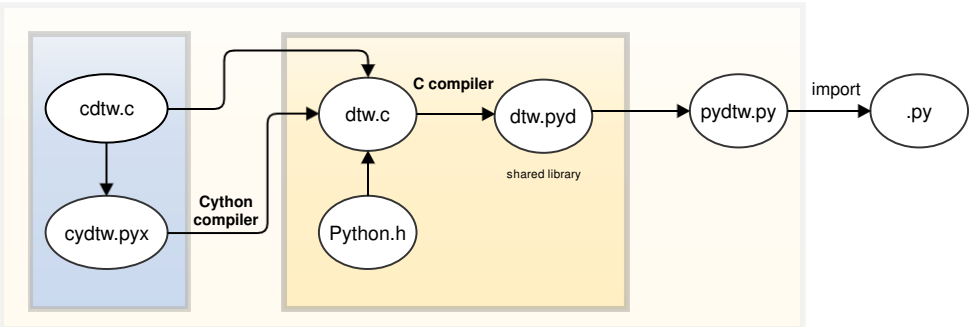
```
ext_modules = [Extension("cydtw",  sourcefiles,
                         include_dirs=[numpy.get_include()],
                         extra_compile_args=["-std=c99", "-w"]
                         )]
```

Překlad pak lze spustit ze složky se zdrojovými soubory pomocí příkazu:

```
python setup.py build_ext --inplace
```

Pro překlad pomocí MS Visual C do `Extension` nedávat parametr `extra_compile_args` a překlad spustit příkazem:

```
python setup.py build_ext --compiler=msvc
```



Obr. 4.1: Diagram vazeb mezi C, Cython a Python kódem

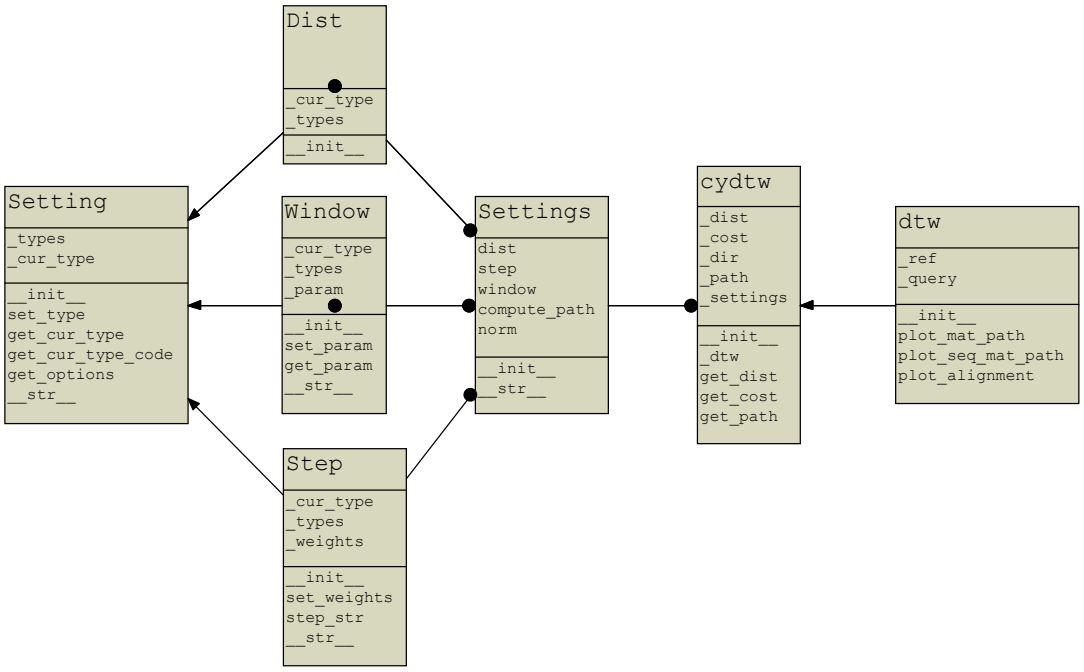
[2] <www.numpy.org>
[3] <http://matplotlib.org/>

4.2.2 Rozhraní

V následujícím odstavci je popsané rozhraní pro Python. Nicméně všechny možnosti, kromě grafiky, jsou také dostupné i v jazyku C. V Pythonu a C je všechno shrnuto do jedné funkce. Knihovna podporuje:

- **distanční funkce**:
  - Euklidovskou vzdálenost
  - Manhattan vzdálenost
- **krokové funkce**:
  - základní krokové funkce (vzorce 2.1.2, 2.7 a 2.8)
  - všechny krokové funkce podle klasifikace Sakoe-Chiba [6]
- **váhy**
- **globální omezení**:
  - Palival adjustment window(což zahrnuje Sakoe-Chiba band)
  - Itakura parallelogram
- **normalizaci**
- **kreslící nástroje**(jenom v Python)

Uvnitř se sestavuje knihovna z kompozice všech nastavení do třídy `Settings`. Instance třídy `Settings` je součástí třídy `cydtw`. Třída `dtw` dědí od třídy `cydtw` a je hlavním vstupem knihovny. Z pohledu uživatele se `dtw` chová jako funkce vracející instanci třídy `dtw` (vytváření instance `dtw` znamená výpočet DTW).

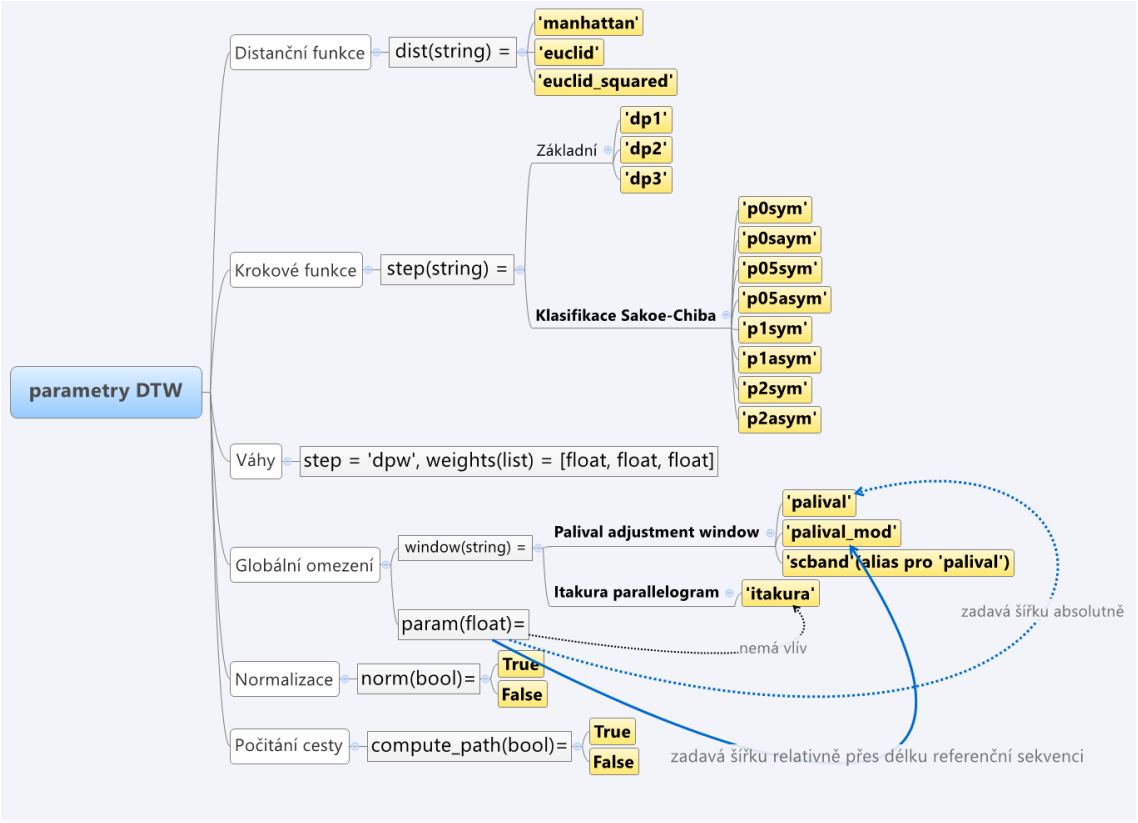


Obr. 4.2: UML diagram knihovny

Definice `dtw` vypadá následovně:

```
dtw(reference, query, Settings(dist='manhattan', step='dp2',
                               window='nowindow', param=0.0,
                               norm=False, compute_path=False))
```

Následující diagram znázorňuje možnosti volby parametrů `dtw`.



Obr. 4.3: Parametry `dtw`

Definice krokové funkce lze zobrazit v Python konzole následujícím způsobem:

```
>import pydtw
>print pydtw.Step().step_str('p05sym')
  * Sakoe-Chiba classification p = 0.5, symmetric step pattern:
  *  min(
  *      cost_matrix[i-1][j-3] + 2d(r[i],q[j-2]) + d(r[i],q[j-1]) + d(r[i],q[j]),
  *      cost_matrix[i-1][j-2] + 2d(r[i],q[j-1]) + d(r[i],q[j]),
  *      cost_matrix[i-1][j-1] + 2d(r[i],q[j]),
  *      cost_matrix[i-2][j-1] + 2d(r[i-1],q[j]) + d(r[i],q[j]),
  *      cost_matrix[i-3][j-1] + 2d(r[i-2],q[j]) + d(r[i-1],q[j]) + d(r[i],q[j])
  *      )
```

5 DTW PRO KLASIFIKACI TYPŮ VOZIDEL

Kapitola popisuje, jak byl během této práce algoritmus DTW použit pro klasifikaci typů vozidel podle profilů, metody, které byly použity pro rozpoznávání, metody testování modelu a nakonec porovnání DTW oproti alternativním metodám porovnání sekvencí.

5.1 Databáze vozidel

CAMEA spol. s r.o.[1] poskytla data o vozidlech, které obsahovaly:

- záznamy z laserového 3D skeneru SICK
- záznamy hmotnosti, rychlosti, počtu os a délky vozidla – WIM
- foto vozidel

Skener SICK má frekvence řezu 100 Hz, naměřená data byla ve tvaru dvourozměrných matic. SW skeneru už řeší segmentaci jednotlivých vozidel.

Bylo řešeno, že 3D data se budou redukovat do profilů. Důvodů je několik. Zaprvé, je to víc praktické, snímání profilu auta je levnější než skenovat celé vozidlo. Zadruhé, je to velká redukce dat, což je výhodné z pohledu efektivity. Zatřetí, cíl prací byl prozkoumat jednorozměrné DTW právě na bočních profilech.

Z těchto dat byla ručně vytvořena databáze, která obsahuje profil vozidla, typ vozidla a identifikační číslo podle času, kdy auto projelo. Bylo zpracováno cca 110 hodin trafiku na dálnici z jednoho pruhu. Vozidla s chybnou 3D rekonstrukcí nebo neurčitým typem (typ byl určen převážně z fotografií, což pro noční záznamy je problematické) nebyla zahrnuta do databáze. Motorky a některé zvláštní typy vozidel (například traktor) nebyly zahrnuty do testování vzhledem k jejich malému počtu.

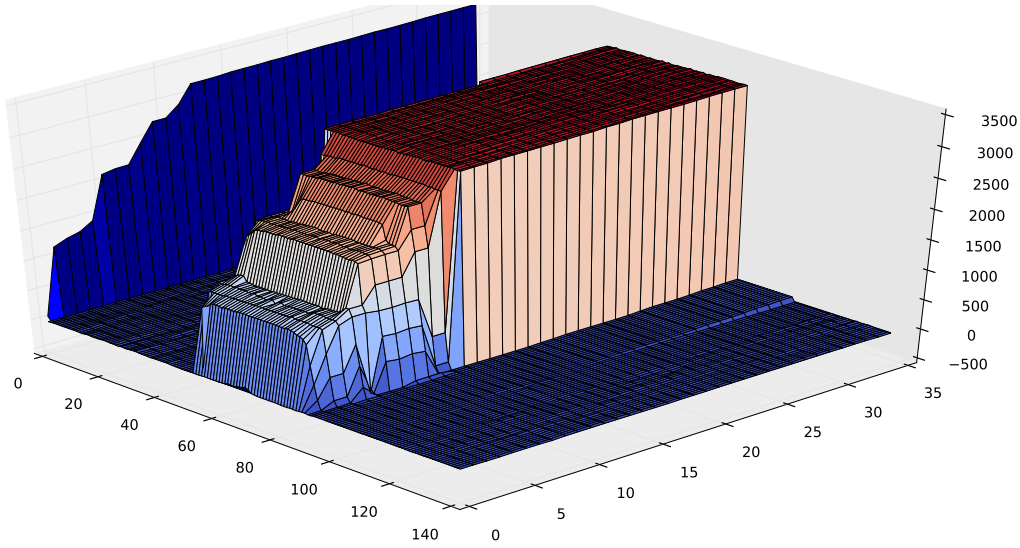
Celkem databáze obsahuje 2535 aut. Profil vozidla je definován jako maximální příčná výška. Většina profilů má délku v rozmezí 10-200 prvků (závisí na rychlosti a délce vozidla). Vozidla byla rozdělena do 6-ti tříd. Všechna osobní auta byla při-

typ vozidla	počet
sedan	684
hatchback	716
dodávkový	323
nákladní	121
kamion	644
autobus	47

Tab. 5.1: Počet jednotlivých typů aut v databázi

[1] `http://www.camea.cz`

řazena do sedanu nebo hatchbacku, proto je třeba tyto názvy chápat v obecném smyslu, vozidla typu sedan jsou sice sedany, ale hatchback zahrnuje také crossovery a džípy.



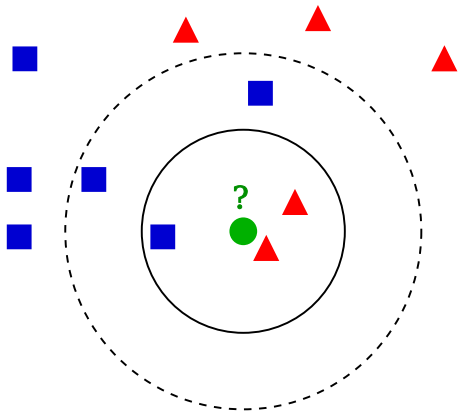
Obr. 5.1: 3D model vozidla a jeho profil

5.2 Algoritmus k-nejbližších sousedů

Samotné porovnání vzorků už je vyřešeno (DTW), problém, který zůstává – jak přiřadit prvek do určité třídy. Jinak řečeno, jaké použít rozhodovací pravidlo. V této práci byl postaven model na základě algoritmu k-nejbližších sousedů[2].

Algoritmus k-nejbližších sousedů patří k učení založenému na instancích, které funguje následovně. Neznámý vzorek se porovnává se všemi známými. Porovnání se provádí na základě atributů vzorků, v daném případě atributy jsou body profilu. Dále známé vzorky se seřazují od nejbližšího k nejvzdálenějšímu. Pak se provádí hlasování k-prvních prvků (k-nejbližších sousedů), každý prvek hlasuje za svou třídu, neznámý prvek se přiřazuje do třídy s největším počtem hlasů. Jako příklad může sloužit obrázek 5.2, kde zelený kruh pro $k = 3$ se klasifikuje jako modrý čtverec a pro $k = 5$ jako červený trojúhelník.

[2]k-nearest neighbors algorithm, zkratka KNN



Obr. 5.2: Algoritmus k-nejbližších sousedů [Převzato z *Wikipedia.org*]

5.3 Validace

Během práce bylo zkoušeno několik validačních metod. Na začátku byl použit holdout, kde rozměry trénovací a testovací množiny měly poměr 1:2. S optimalizací knihovny DTW přišla možnost k efektivnějšímu využití dat. Holdout byl použit s náhodnou stratifikací a opakoval se tolikrát, kolik dovolil výpočetní čas. Ovšem i při relativně velkém počtu opakování, výsledky měly příliš nedeterministický charakter. Určitě je možné rozdělit vozidla víc něž na 6 typů. Při náhodné stratifikaci se vyskytují případy, když v trénovací množině chybí prvky některých subtypů, které všichni dostali do testovací množiny. To působí nerealisticky vysoký rozptyl výsledků. Nakonec metoda, která vyřešila tento problém, je křížová validace.

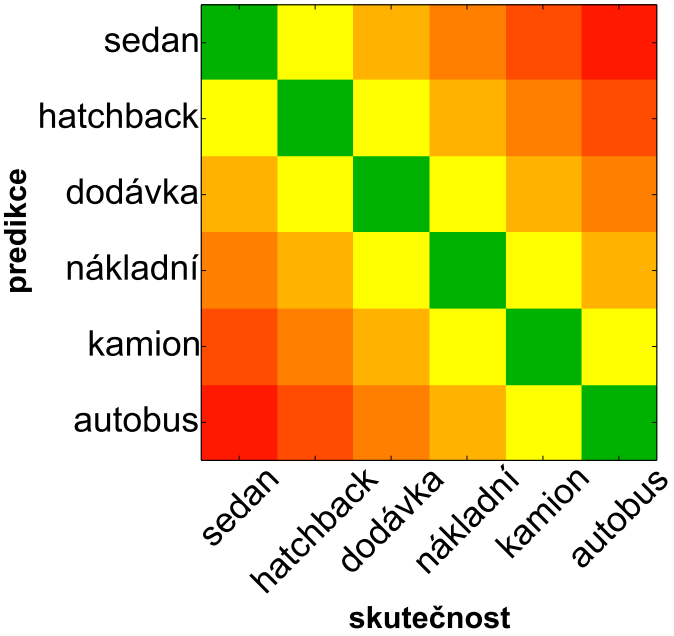
5.3.1 Křížová validace

Křížová validace[3] je metoda používaná pro odhad přesnosti modelů. Pro validaci se vstupní množina dat rozděluje na k testovacích disjunktních podmnožin stejné velikosti. Pro každou vytvořenou testovací podmnožinu se zbytek dat používá jako trénovací a celý proces se k-krát opakuje. V této práci byl použit limitní případ k-ární křížové validace – leave-one-out, kde testovací množiny tvoří jenom jeden vzorek. Leave-one-out byl zvolen vzhledem k optimálnímu využití dat. Pro odhad rozptylů se používala zároveň varianta 10-fold se stratifikací.

[3] k-fold cross-validation

5.4 Matice záměn

Matice záměn je způsob znázornění výsledků klasifikace. Indexy matice se označují jako jednotlivé třídy, řádkový index označuje predikci modelů a sloupcový skutečný typ vzorků. Prvky ležící na hlavní diagonále tedy vyjadřují správnou klasifikaci. Obrázek 5.3 znázorňuje matici, přičemž pořadí indexu je zvoleno tak, že čím dál prvek leží od hlavní diagonály, tím závaznější je chyba. Například prvek (0,5) znamená, že model klasifikoval vozidlo jako autobus, které ve skutečnosti bylo sedanem. Závažnost chyby je graficky znázorněna barvou. Nebyla informace o konkrétních požadavcích, proto při výpočtu úspěšnosti všechny prvky matice záměn měly stejnou prioritu.



Obr. 5.3: Matice záměn

Pokud označíme matici záměn $M \in \mathbb{R}^{NxN}$, kde N je počet tříd, celková přesnost modelu[4] je rovna součtu prvků hlavní diagonály lomeno součet všech prvků matice:

$$ACC = \frac{\sum_{i=1}^{N} M(i,i)}{\sum_{i=1}^{N} \sum_{j=1}^{j=N} M(i,j)} \tag{5.1}$$

Pro detailnější popis úspěšnosti modelu se používaly ještě dvě charakteristiky:

- citlivost(sensitivity) nebo True Positive Rate – TPR
- fall-out nebo False Positive Rate – FPR

[4] overall accuracy

TPR a FPR jsou definovaný pro binární klasifikaci, pokud se jednalo o klasifikaci vozidel na 6 typů, je potřeba převést každou třídu do binární oblasti.

Příklad toho převodu pro sedan je možné popsat následovně:

$$TP = M_{binary}(0,0) = M_{multi}(0,0) \tag{5.2}$$

$$FP = M_{binary}(0,1) = \sum_{i=1}^{N} M_{multi}(0,i) \tag{5.3}$$

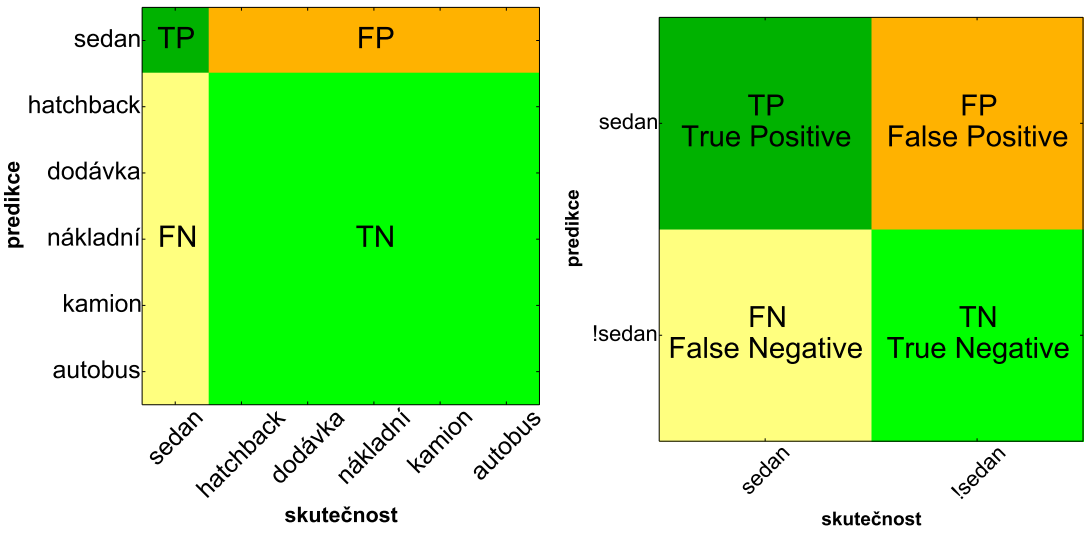
$$FN = M_{binary}(1,0) = \sum_{i=1}^{N} M_{multi}(i,0) \tag{5.4}$$

$$TN = M_{binary}(1,1) = \sum_{i=0}^{N}\sum_{j=0}^{N} M_{multi}(i,j) - TP - FP - FN \tag{5.5}$$

Analogický postup lze provést pro ostatní třídy. TPR a FPR jsou potom definované následovně:

$$TPR = \frac{TP}{TP + FN} \tag{5.6}$$

$$FPR = \frac{FP}{FP + TN} \tag{5.7}$$

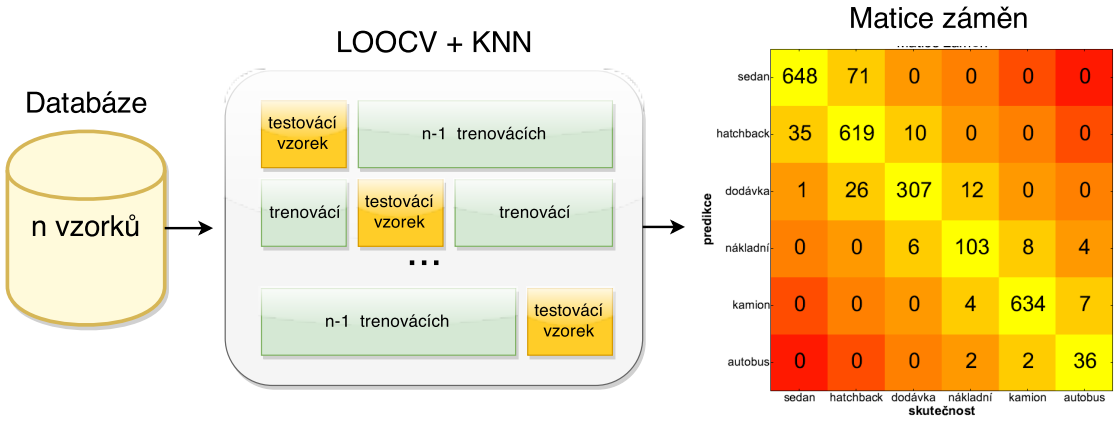


(a) Matice záměn pro všechny třídy (b) Binární matice záměn pro sedan

Obr. 5.4: Převod klasifikace do binární

5.5 Analýza výsledků

Nejlepší výsledek při klasickém DTW(distanční funkce – Manhattan vzdálenost, kroková funkce 2.8) prokázal KNN s počtem sousedů rovném $k = 3$. Přičemž přesnost pro daný případ je rovna $ACC = 92,6\%$ a matice záměn vypadá následovně:

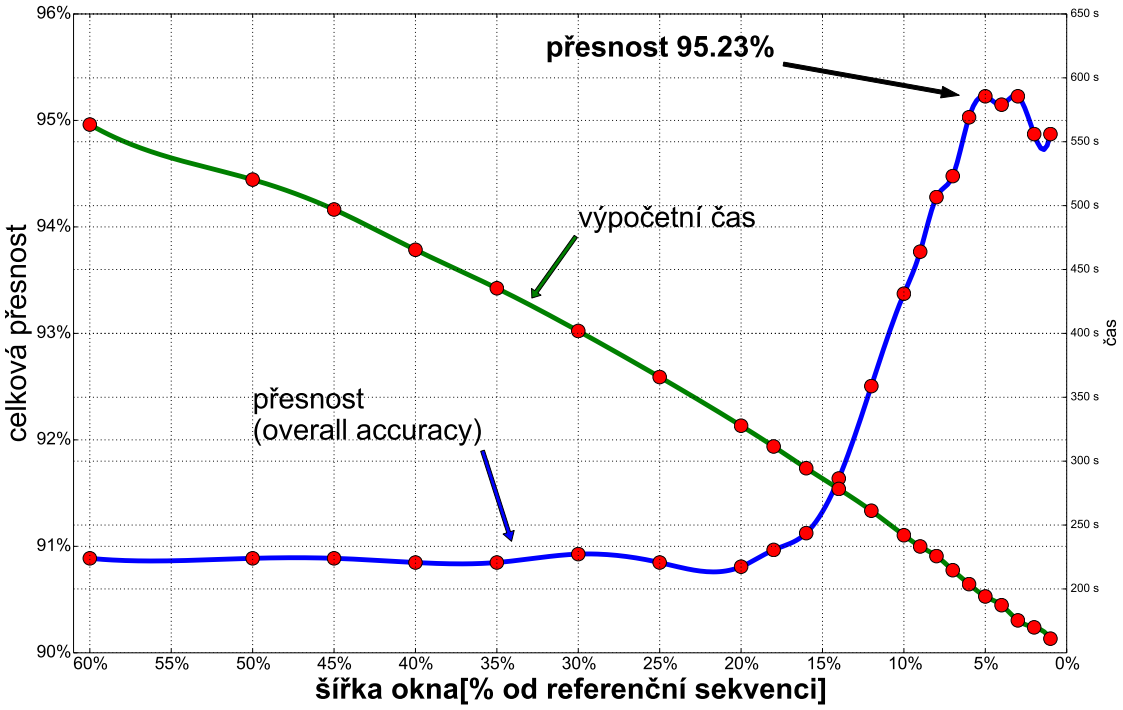


Obr. 5.5: Model na základě klasického DTW a KNN, k = 3

Přesnost 92,6% zní dobře, když naivní klasifikátor na základě porovnání vozidel podle maximální výšky dával přesnost jenom 80%. Ale pokud algoritmus DTW má velice široké možnosti nastavení, vzniká otázka, jestli se dá najít lepší varianta DTW něž klasická. Normalizace DTW s ohledem na rozptyl výrazně neovlivňovala úspěšnost modelů. Ovšem s ohledem na fakt, že délka vozidel souvisí s jejich typem, i když souvislost se může ztratit, pokud délka naměřených profilů závisí na rychlosti vozidel, další testování se provádělo bez normalizace DTW.

Krokové funkce odlišné od klasické neměly pozitivní vliv na úspěšnost modelu. Navíc složitější krokové funkce zvyšují výpočetní čas. Kroková funkce 2.14, která z pohledu výpočetní náročnosti je lepší, měla výrazně negativní charakter z pohledu úspěšnosti.

Globální omezení cesty DTW mělo pozitivní vliv na model. Což splňuje teoretický předpoklad, vozidla za dobu průjezdu pod skenerem výrazně nemění svou rychlost (daný skener se nacházel na dálnici) a signál ze skeneru už byl segmentován. Tedy nelinearita a časové zpoždění v naměřených datech bylo skoro zanedbatelné. Navíc globální omezení snižuje výpočetní čas, protože není potřeba počítat všechny prvky globální nákladové matice. Šířka okna při globálním omezení se definovala přes délku referenční sekvence, jak je to popsané v odstavci 2.2 a algoritmu 4. Maxima přesnosti bylo dosáhnuto při šířce okna v 3% a 5% od délky referenční sekvence, kde přesnost byla $ACC = 95,32\%$. Závislost mezi přesností modelu a šířkou znázorňuje následující graf:



Obr. 5.6: Vliv globálního omezení na úspěšnost modelů

Jiné nastavení DTW, které by prokázalo lepší úspěšnost, nebylo nalezeno. Výpočetní čas ukázaný na grafu 5.6 je čas počítání distanční matice vozidel. Pokud se používala symetrická kroková funkce, kde $dtw(r, q) = dtw(r, q)$, není potřeba počítat celou matici, ale jenom rozdílné dvojice. Počet těchto dvojic vyjadřuje kombinace bez opakování, tedy počet porovnání (kolikrát se volala funkce `dtw`) je roven:

$$C_2(n) = \binom{n}{2} = \frac{n!}{2!(n-2)!} = \frac{(n+1)n}{2} \tag{5.8}$$

Databáze obsahuje 2535 profilů vozidel a $C_2(2535) = 3214380$. Pokud rozdělíme čas počítání distanční matice na tohle číslo a násobíme počtem profilů v databázi minus jeden, dostaneme přibližný čas pro klasifikaci jednoho vozidla:

$$t = \frac{250s}{3214380} 2534 = 0.197s \tag{5.9}$$

0.197s při rychlosti 120 km/h odpovídá cca 6m. V takové těsné blízkosti vozidla by na dálnici neměli jet. Ovšem systém má prostor k optimalizaci. Knihovna DTW pracuje s datovým typem `double`, když výstup ze skeneru je reprezentován celými čísly. Převod knihovny na `integer` má potenciální sníženi času počítaní o násobek. Jestli navíc vzít v úvahu, že počítání prvků distanční matice vozidel jsou nezávislé a tento výpočet lze provádět paralelně, 197ms lze ještě několikrát podělit. Z výše

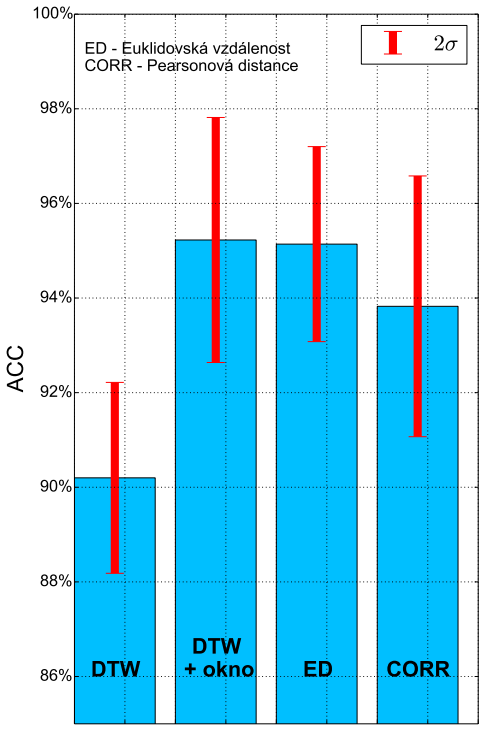
uvedeného plyne, že systém může fungovat v reálném čase i na levném laptopu.

5.6 Porovnání DTW s alternativními metodami

Metoda DTW byla porovnaná s korelací a Euklidovskou vzdáleností, model na základě třech nejbližších sousedů zůstal stejný. Korelace (Personův korelační koeficient) má opačný smysl oproti DTW a Euklidovskou vzdáleností, její význam není vzdálenost, kterou potřebuje KNN, ale podobnost. Proto byla použita Personová vzdálenost, definice které je následující:

$$d(r,q) = 1 - \rho(r,q) \tag{5.10}$$

Pokud obor hodnot korelace leží v rozmezí $\rho(r,q) \in < -1;1 >$, Pearsonova vzdálenost leží v $d(r,q) \in < 0;2 >$. Euklidovská vzdálenost a korelace potřebují sekvenci stejné délky, proto všechny profily v databázi byly převzorkovány na stejnou délku. Převzorkování se provádělo pomocí interpolace lineárním splajnem:



Obr. 5.7: Porovnání modelů

$$S_i(x) = f_i + \frac{f_{i+1} - f_i}{x_{i+1} - x_i} \cdot (x - x_i), \; i = 0, 1 \; ... \; n-1 \tag{5.11}$$

Lineární splajn propojuje dva sousední body přímkou. Pro převzorkování sekvence délkou n se sestavuje $n-1$ rovnice přímek. Pokud má sekvence délku n a je potřeba jí převzorkovat na délku m, vypočítá se vektor $i' \in R^m$:

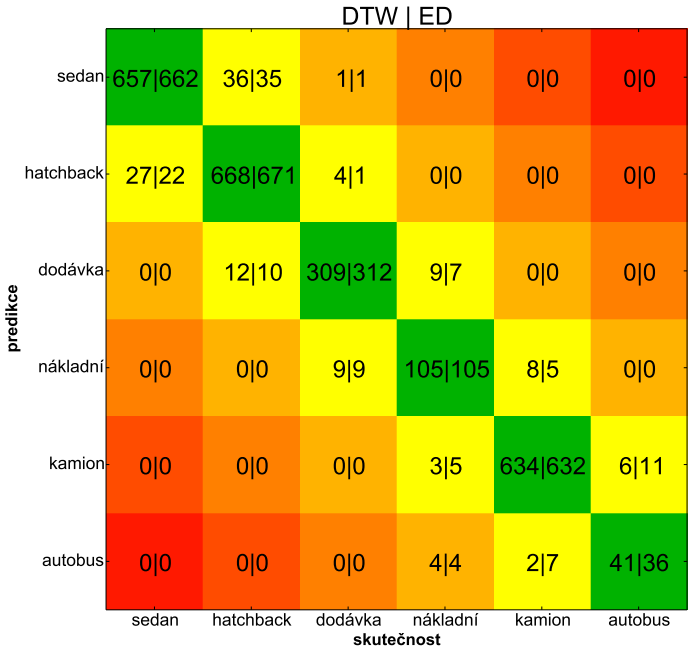
$$i'_k = k \cdot \frac{n}{m}, \; k = 0, 1 \; ... \; m-1 \tag{5.12}$$

Prvky tohoto vektoru se dosazují v příslušný splajn. Takovým způsobem byly všechny profily normovány na délku v 50 prvků (hodnota 50 měla nejlepší přesnost pro Euklidovskou vzdálenost). Na obrázku 5.5 jsou znázorněny výsledky porovnání,

velikost chyby je určena jako směrodatná odchylka z 10-fold křížové validace:

$$\sigma = \sqrt{\frac{1}{10}\sum_{i=1}^{10}(ACC_i - \mu)^2}, \text{ kde } \mu = \frac{1}{10}\sum_{k=1}^{10} ACC_k \tag{5.13}$$

Z grafu je vidět, že mezi metodami s ohledem na rozptyl není velký rozdíl v přesnosti. Při použití korelace, autobus a kamion model se mohl považovat za osobní vozidla. Všechny prvky matice záměn měly stejnou prioritu, ale v praxi by to mohlo být velkým mínusem. Bližší pohled na výsledky DTW s globálním omezením a Euklidovskou vzdálenost poskytuje následující matice záměn a tabulka chyb.

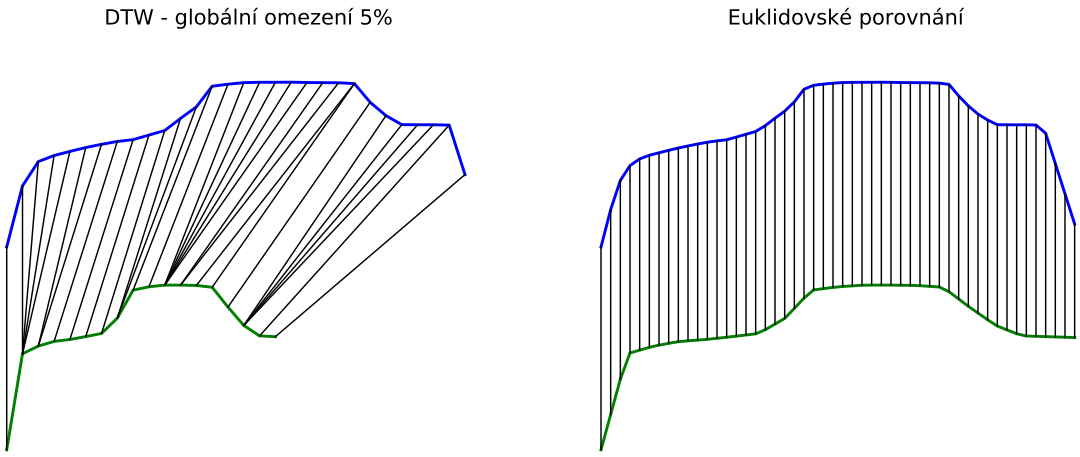


Obr. 5.8: Matice záměn klasifikace pomocí DTW a Euklidovské vzdálenosti

	DTW		ED	
typ vozidla	**FPR**[%]	**FNR**[%]	**FPR**[%]	**FNR**[%]
sedan	2,00	3.95	1.94	3.21
hatchback	1.70	6.70	1.26	6.28
dodávkový	0.95	4.33	0.77	3.40
nákladní	0.70	13.22	0.58	13.22
kamion	0.48	1.55	0.85	1.86
autobus	0.24	12.77	0.44	23.40

Tab. 5.2: TPR a FPR klasifikace pomocí DTW a Euklidovské vzdálenosti

Podobnost výsledků Euklidovské vzdálenosti a DTW spočívá především v tom, že DTW při globálním omezení s šířkou okna 5% od referenční sekvence je skoro lineární porovnání, které provádí Euklidovská vzdálenost. Lze uvést extrémní případ – omezení cesty DTW do diagonály globální nákladové matice. Jestli sekvence budou mít stejnou délku, DTW bude počítat právě Euklidovskou vzdálenost. Pokud ale sekvence mají různou délku, postupy převzorkování a následný výpočet Euklidovské vzdálenosti a přímý výpočet vzdálenosti pomocí DTW jsou rozdílné. Rozdíl na příkladu dvou sedanů znázorňuje následující obrázek.



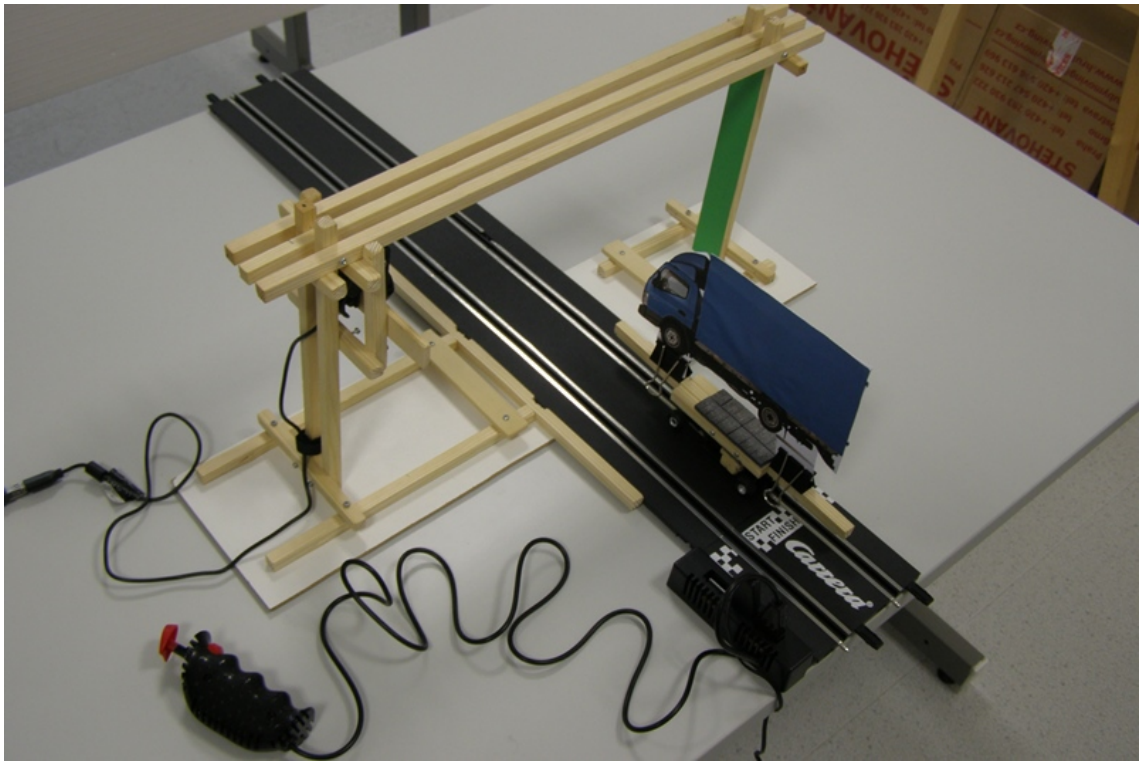
Obr. 5.9: Porovnání podle DTW a Euiklidovské vzdáleností

6 LABORATORNÍ PŘÍPRAVEK

6.1 Koncept

V rámci této práce byl vytvořen laboratorní přípravek, který sleduje dva cíle. Zaprvé, přípravek imituje mýtnou bránu a demonstruje metodu navrženou na reálných datech, ale už na datech umělých. Zadruhé, přípravek by měl sloužit jako pomůcka ve výuce ke generování dat pro DTW.

Měření pomoci laserového skeneru bylo nahrazeno kamerou. Vozidla simulují profily, které jsou vystřihnuté z papíru. Pro přípravek byla zakoupena dětská autodráha a sestavena konstrukce, která imituje mýtnou bránu a zároveň slouží jako držák kamery. Auto (které bylo v soupravě autodráhy) bylo modifikováno pro účely přípravku. Do něho se dá jednoduše vkládat profily vozidel. Původní rychlost autodráhy byla omezena pomocí modelářského závaží.



Obr. 6.1: Laboratorní přípravek

K dispozici byla poskytnuta webová kamera Logitech C905, která podporuje snímání v rozlišení 640 x 480 prvků při 30 FPS. Zpracování obrazu se provádělo za pomocí knihovny OpenCV[1] v Python. Grafické rozhraní bylo vytvořeno za pomoci

[1] <http://opencv.org/>

knihovny PyQt[2].

Následující kapitoly popisují způsob snímaní profilu pomocí kamery a související SW.

6.2 Snímání profilu vozidla pomocí kamery

Napsaný program nastavuje rozlišení 640 x 480 prvků a 30 FPS. Rychlost vozidel je relativně vysoká, proto před zpracováním obrazu se nastavuje minimální možná hodnota expozice kamery. Datasheet kamery neobsahuje informaci o rozsahu hodnot expozice, ale na bázi empirického výzkumu bylo zjištěno, že minimu odpovídá hodnota -8 parametru `CV_CAP_PROP_EXPOSURE`. Kvůli nízké expozice je potřeba v dostatečném osvětlení přípravku. Osvětlení v laboratoři je k tomu dostačující. Ovladače kamery mají autofokus, který má negativní vliv na snímání profilu projíždějícího vozidla. Ten nejde vypnout pomocí nástrojů OpenCV, ale tuhle volbu obsahuje související SW kamery.

Snímání profilu vozidla z celého obrazu bylo vyloučeno. Hlavní důvod je ekvidistantnost bodů v obrázku, což je v rozporu s cíli přípravku:

- analogie s měřením výšky vozidel na dálnice
- demonstrace metody DTW
- generace dat pro výukovou úlohu DTW.

Místo toho bylo rozhodnuto simulovat chování řádkové kamery, z dvourozměrné matice obrazu používat jenom jeden sloupec, z každého snímku ukládat jenom jeden bod profilu. I když z určitého pohledu je taková metoda velká redukce (podmnožina použití celého obrazu), ale sestavení profilu z několika snímků zachovává informaci o čase, což je klíčový faktor pro metodu borcení času. Vzorkování o frekvenci 30 Hz je v daném případě mínusem, ale pro rozpoznávaní je dostačující.

Segmentovat vozidlo pomáhá nalepená zelená páska, proto není žádoucí, aby vozidla byla taky zelená. Snímky se čtou barevně, v BGR tvaru. První krok po načtení snímku je určení prahů pro segmentaci pozadí. Snímek se převádí na HSV[3] barevný model. V OpenCV[4] H, S, V jsou celá čísla, přičemž rozsahy hodnot jsou následující:

$$H \in [0 : 179] \tag{6.1}$$

$$S \in [0 : 255] \tag{6.2}$$

$$V \in [0 : 255] \tag{6.3}$$

[2] <`http://http://www.riverbankcomputing.com/`>

[3] hue-saturation-value

[4] byla použitá Python-OpenCV 2.4.0

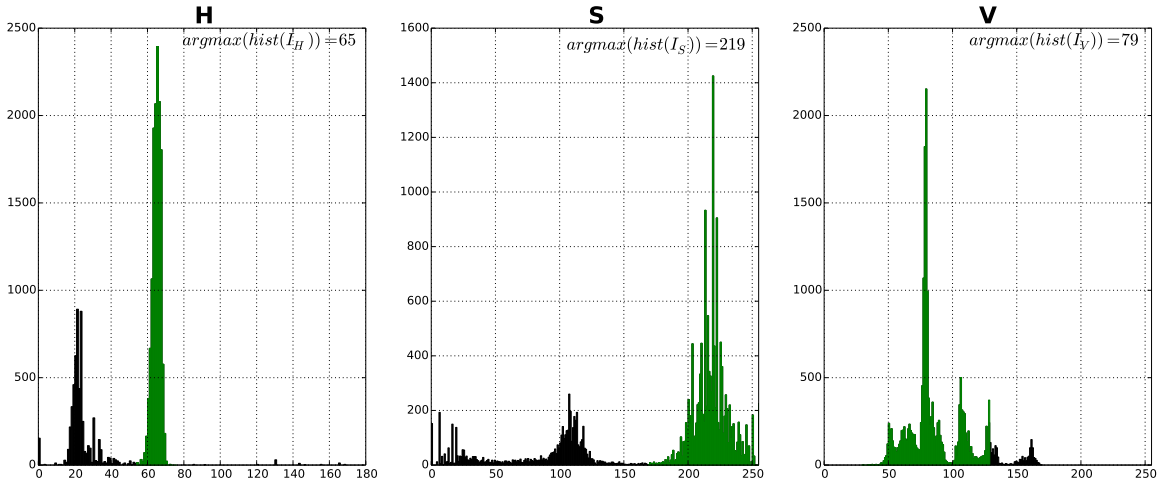
Prahy se nastavují automaticky z prvního snímku. Snímek se ořezává tak, aby zelená páska zabírala většinu plochy snímku. Dál se vypočítají histogramy pro H, S, V hodnoty. Pokud zelená páska má nejvíc plochy, na histogramu jí odpovídají oblasti kolem globálních maxim. Dolní a horní meze prahů se určují na základě těchto hodnot. Pokud označíme HSV obrázek I_{HSV} a funkci histogramu $h(x)$, prahy jsou definované následovně:

$$H_{lower} = \operatorname*{argmax}_x(h_{I_H}(x)) - 10; \quad H_{upper} = \operatorname*{argmax}_x(h_{I_H}(x)) + 10; \tag{6.4}$$

$$S_{lower} = \operatorname*{argmax}_x(h_{I_S}(x)) - 50; \quad S_{upper} = \operatorname*{argmax}_x(h_{I_S}(x)) + 50; \tag{6.5}$$

$$V_{lower} = \operatorname*{argmax}_x(h_{I_V}(x)) - 50; \quad V_{upper} = \operatorname*{argmax}_x(h_{I_V}(x)) + 50, \tag{6.6}$$

přičemž hodnoty 10 a 50 jsou zjištěny empiricky.



Obr. 6.2: Histogram pro určení prahů

Následně obrázek se ořezává do jednoho sloupce uprostřed zelené pásky a provádí se prahování následujícím způsobem:

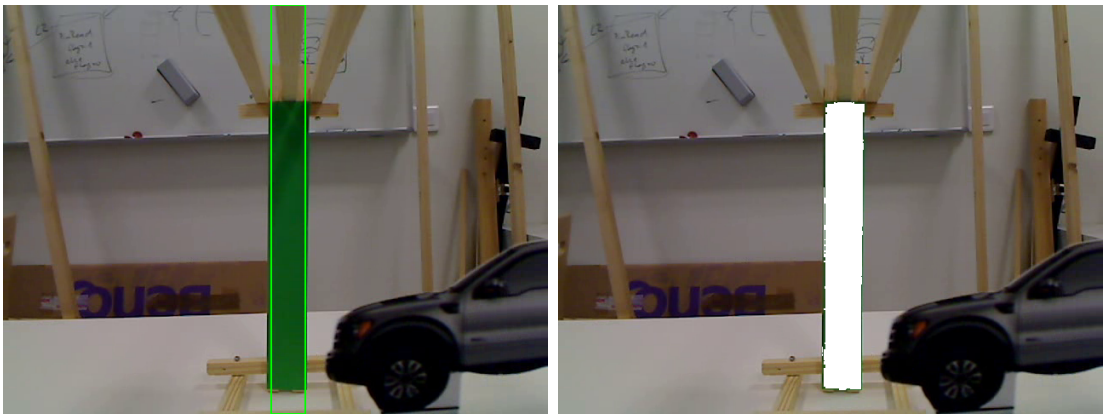
$$I_{HSV}(i,j,:) = \begin{cases} (0,0,255), & \text{jestli } b(i,j) = True \\ I_{HSV}(i,j,:), & \text{jestli } b(i,j) = False \end{cases} \tag{6.7}$$

, kde $b(i,j)$:

$$\begin{aligned} b(i,j) = H_{lower} \leq I_{HSV}(i,j,0) \leq H_{upper} \wedge \\ S_{lower} \leq I_{HSV}(i,j,1) \leq S_{upper} \wedge \\ V_{lower} \leq I_{HSV}(i,j,2) \leq V_{upper} \end{aligned} \tag{6.8}$$

Výsledek takového postupu je ukázán na obrázku 6.3. Na původním obrázku je

zelenou čarou označena oblast, podle které se určují prahy z histogramu. Je nutné upozornit, že obrázek je jenom ilustrační, v programu se neprahuje celý obraz, ale jenom jeden sloupec uprostřed, platí to i pro další ilustraci.



(a) Původní snímek (b) Snímek po prahování

Obr. 6.3: Prahování

Po oříznutí snímku a prahování v HSV oblasti se obraz převádí na šedotónový tvar. Hodnoty obrazu se invertují:

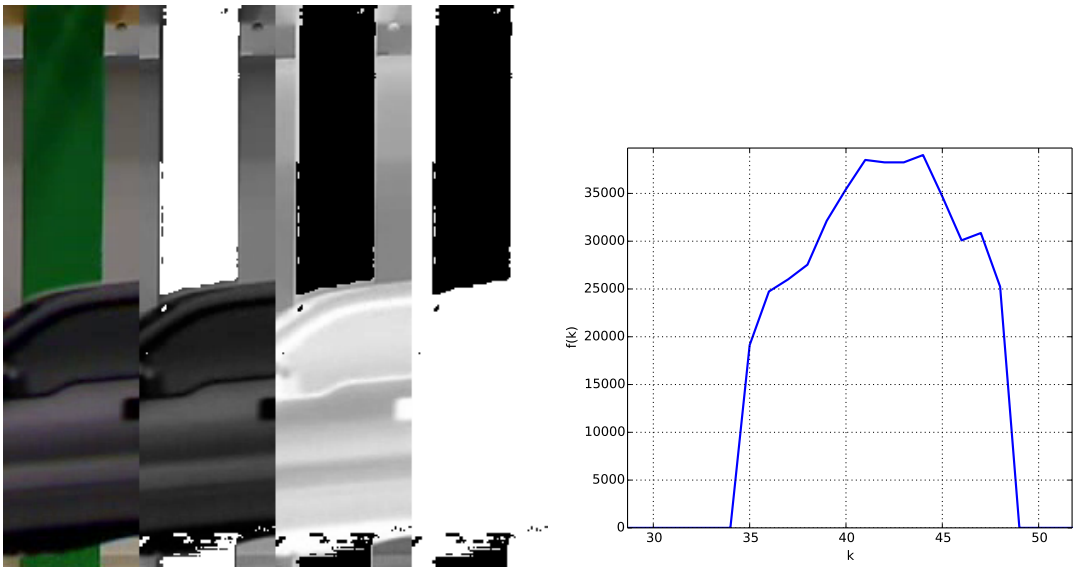
$$I_{gray}(i,j) = 255 - I_{gray}(i,j) \tag{6.9}$$

A následně se znovu prahují do binárního tvaru. Při uvažování o úspěšném minulém prahování, tento práh může mít libovolnou hodnotu větší než 0:

$$I_{bin}(i,j) = \begin{cases} 255, & \text{jestli } I_{gray}(i,j) \geq 10 \\ 0, & \text{jestli } I_{gray}(i,j) < 10 \end{cases} \tag{6.10}$$

Nakonec poslední krok – výpočet z obrazu jednoho bodu profilu. Za bod profilu se považuje součet pixelu ve sloupci uprostřed zelené pásky. S ohledem na pozici kamery, střed pásky je roven skoro středu celého obrazu. Pokud Ik je k-atý snímek, funkce profilů je:

$$f(k) = \sum_{i=0}^{N-1} I_{bin_k}(i, 335) \tag{6.11}$$



(a) Kroky zpracování obrazu

(b) Profil vozidla

Obr. 6.4: Měření profilu

6.3 Rozpoznávání

Pro demonstrační a testovací účely bylo vyříznuto z papíru 6 různých vozidel. Z těchto vozidel byla naměřena trénovací data. Každé vozidlo má v databázi 5 profilů. Rozpoznávaní je implementováno podobně jako při reálných datech z 3D skeneru. Používá se ale jenom 1 nejbližší soused a klasické DTW. Naměřené profily vstupují do DTW bez segmentace, proto globální omezení DTW má v daném případě negativní vliv.

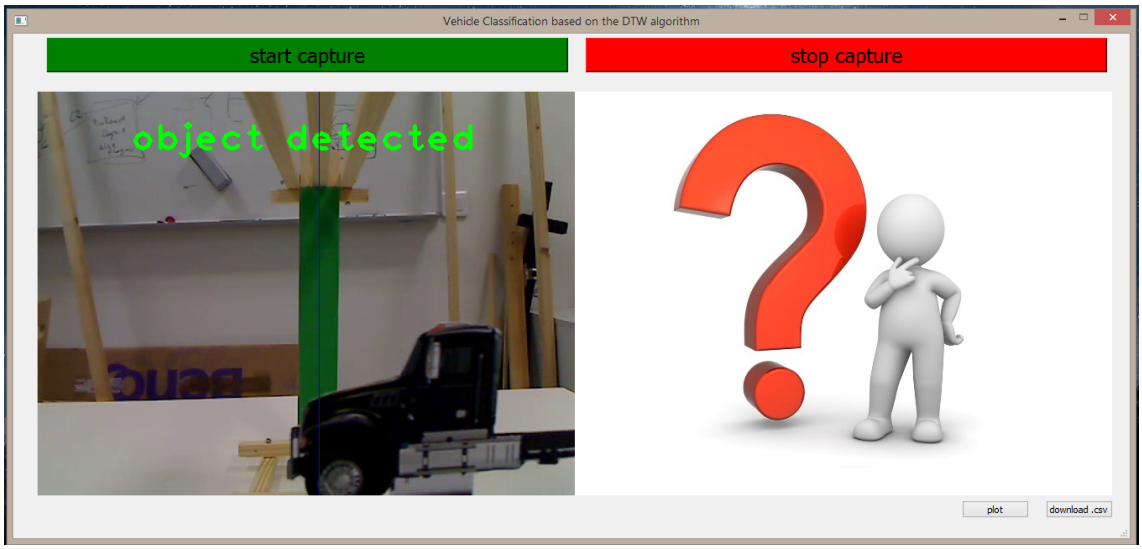
Úspěšnost testovaná na naměřených datech je 100%. Ve skutečnosti to určitě tak není, klíčový faktor je v podmínkách měření, hlavně v osvětlení. Příliš vysoká rychlost taky někdy působila chyby v rozpoznávaní, protože relativně nízká vzorkovací frekvence nezajišťuje dostatek informací. Pomocí přípravku se dá otestovat změna rychlosti vozidla v průběhu měření profilu. Například je možné zpomalit nebo zrychlit, dokonce i zastavit vozidlo pod branou a program ho stejně rozpozná.

6.4 Grafické rozhraní

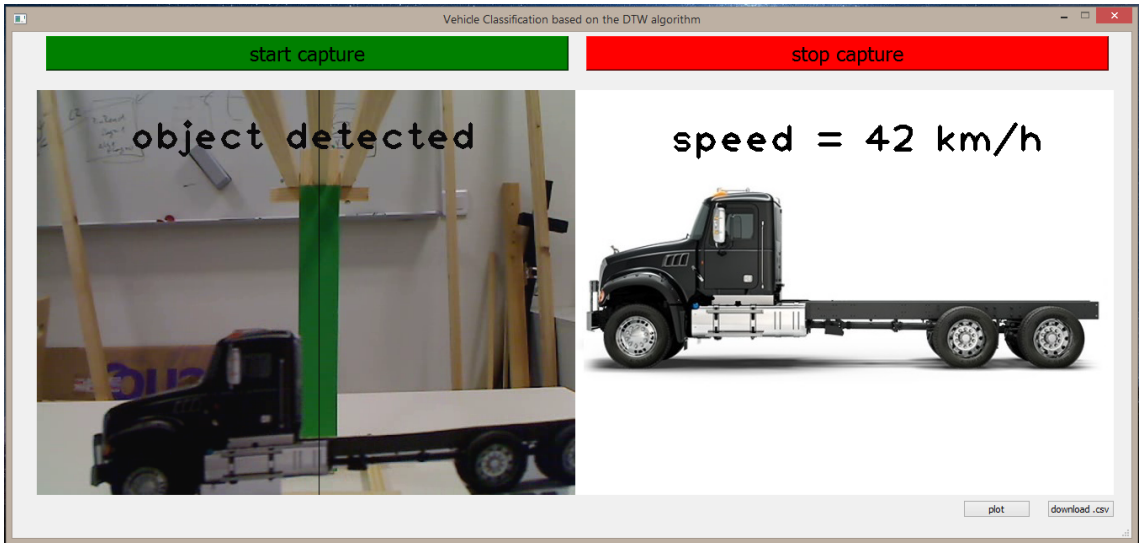
Grafické rozhraní je vizuálně rozděleno na dvě části. Zleva se ukazuje aktuální výstup kamery, zprava po projetí vozidla ukazuje výsledek klasifikace. Zároveň s rozpoznáváním, program měří průměrnou rychlost vozidla. Ta sleduje jenom demonstrační účel, nelze jí chápat jakou realistickou. Výpočet rychlosti se provádí přes počet namě-

řených bodů profilu, délku vozidla (předpokladem je, že vozidlo se rozpozná správně) a vzorkovací frekvence kamery.

Uživatel by měl stisknout tlačítko **start capture** a projet po autodráze. Po průjezdu vozidla není potřeba tisknout tlačítko **stop capture**, program automaticky zastaví snímání a ukáže výsledek klasifikace. Tlačítko **stop capture** slouží v případě, když vozidlo pod branou neprojelo, nebo program ho nedetekoval. Zároveň s výsledkem klasifikace program ukazuje snímek vozidla pod branou. Nasnímaný profil vozidla se dá vykreslit pomocí tlačítka **plot** nebo uložit na disk v csv formátu pomocí tlačítka **download .csv**.



(a) Program v době průjezdu vozidla



(b) Program po průjezdu vozidla

Obr. 6.5: Grafické rozhraní programu

7 ZÁVĚR

V práci byl popsán algoritmus DTW, jeho možné varianty, nastavění a oblasti použití algoritmu. Byla naprogramována C/Python knihovna, která se dá využívat v jazyku C a Python. Použití kombinace C a Python byla nutnost, která umožnila z jedné strany jednoduchou manipulaci s daty a použití výhod vysoké abstrakce jazyka Python a zároveň využívat výkon jazyka C. Nebyl problém používat náročný algoritmus leave-one-out křížové validace, efektivně používat poskytnutá data a podrobně analyzovat vliv různých nastavení DTW.

Implementovaný algoritmus byl následně použit na data z laserového skeneru, přičemž se jedná o reálná data ze 110 hodinového záznamu z provozu. Během testování modelu na základě KNN a DTW bylo zjištěno, že globální omezení cesty DTW má výrazně pozitivní vliv na úspěšnost klasifikace. Nejlepší dosáhnutá úspěšnost je 95,2% pří použití třech nejbližších sousedů, DTW bez normalizace, klasickou krokovou funkcí a globálním omezením typu *Sakoe-Chiba band/Palival adjustment window* s šířkou rovnou 5% od referenční sekvence. Důvodem úspěšnosti globálního omezení cesty DTW je specifikum poskytnutých dat. Laserový skener se nacházel na dálnici, kde vozidla jedou skoro konstantní rychlostí, nelinearita v naměřených datech měla zanedbatelnou roli. To se potvrdilo porovnáním DTW z Euklidovskou vzdáleností, kde nebyly zaznamenány vysoké rozdíly v úspěšnosti. Použití Euklidovské vzdálenosti a korelace vyžaduje předzpracování profilů, sekvence je potřeba normalizovat na stejnou délku, DTW může pracovat s různými délkami. Při výběru mezi DTW a Euklidovskou vzdáleností je potřeba vzít v úvahu, že i při silném omezení cesty, DTW je stále kvadratický algoritmus, když výpočet Euklidovské vzdálenosti je lineární v čase. Výhodou DTW je velký počet stupňů volnosti. Model na základě DTW se dá nastavovat a optimalizovat, když v případě korelace a Euklidovské vzdálenosti je jediný stupeň – předzpracování signálů. Klíčovou výhodou DTW je jeho nelineární porovnání, které má velký potenciál, pokud by se laserový skener nacházel na křižovatce nebo v jakémkoliv jiném místě, kde vozidla se mění svou rychlost nebo zastavují.

Uvedenou výhodu DTW je možné prozkoumat pomocí vytvořeného laboratorního přípravku, který simuluje mýtnou bránu. Vozidla imitují profily vystřihnuté z papíru, které se měří pomocí webové kamery. Napsaný program zpracovává snímky kamery a rozpoznává projetá vozidla na základě DTW a KNN.

LITERATURA

[1] PARK Jaehyun. CS 97SI: Itroduction to programming contests [přednáška]. *Stanford University* [online]. In: Stanford online, 01.12.2014. Dostupné z URL: <http://web.stanford.edu/class/cs97si/04-dynamic-programming.pdf>

[2] NAKHLEH Luay. Algorithmic Thinking. Lecture Dynamic programming [video]. *Rice University* [online]. In: Coursera.org. [Vid. 01.12.2014]. Dostupné z URL: <https://class.coursera.org/algorithmicthink-001>

[3] MÜLLER, Meinard. *Information retrieval for music and motion.* Berlin: Springer, c2007, s. 70-84. ISBN 9783540740476. Dostupné z URL: <http://www.springer.com/978-3-540-74047-6>.

[4] SENIN, Pavel. *Dynamic time warping algorithm review.* 2008. Dostupné z URL: <http://www2.hawaii.edu/~senin/assets/papers/DTW-review2008draft.pdf>

[5] JUANG, Lawrence Rabiner; Biing-Hwang. *Fundamentals of speech recognition.* 2nd Indian Reprint. Delhi: Pearson Education, 2005. ISBN 81-297-0138-3.

[6] SAKOE, H. a S. CHIBA. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing.* 1978, vol. 26, issue 1, s. 43-49. DOI: 10.1109/ TASSP.1978.1163055. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1163055

[7] MYERS„ Cory S. *A comparative study of several dynamic time warping algorithms for speech recognition.* Massachusetts Institute of Technology. Dept. of Electrical Engineering and Computer Science, 1980. Dostupné z: http://dspace.mit.edu/bitstream/handle/1721.1/27909/07888629.pdf?sequence=1. Thesis M.S. MIT. Vedoucí práce Jae S. Lim.

[8] GIORGINO, Toni. Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package. *Journal of Statistical Software* [online]. 2009, Vol. 31, Issue 7 [cit. 2014-12-02]. Dostupné z: http://www.jstatsoft.org/v31/i07/paper

[9] PALIWAL, K.K., Anant AGARWAL a Sarvajit S. SINHA. A modification over Sakoe and Chiba's dynamic time warping algorithm for isolated word recognition. *Signal Processing.* 1982, vol. 4, issue 4, s. 329-333. DOI: 10.1016/0165-1684(82)90009-3. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/0165168482900093

[10] NAKHLEH Luay. Algorithmic Thinking. Dynamic Programming and Pairwise Sequence Alignment. *Rice University* [online]. In: Coursera.org. Dostupné z URL: <https://class.coursera.org/algorithmicthink-001>

[11] FURTUNA, Titus Felix. Dynamic Programming Algorithms in Speech Recognition. *Informatica Economica Journal.* 2008, 2(46). Dostupné z: http://revistaie.ase.ro/content/46/S%20-%20Furtuna.pdf

[12] JURAČKA, Zdeněk. *Rozpoznávací metody v oblasti biosignálů: Recognition methods for biosignals*. Brno: Vysoké učení technické, Fakulta elektrotechniky a komunikačních technologií, 2009. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=16384

[13] HUANG, B. a W. KINSNER. ECG frame classification using dynamic time warping. *IEEE CCECE2002. Canadian Conference on Electrical and Computer Engineering. Conference Proceedings (Cat. No.02CH37373)*. IEEE, 2002, s. 1105-1110. DOI: 10.1109/CCECE.2002.1013101. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1013101

[14] ČERNOCKÝ, H. *Zpracování řečových signálů* - studijní opora Elektronický text předmětu Zpracování řečových signálů. Fakulta informačních technologií, VUT v Brně. Aktualizováno 6. 12. 2006. Dostupné z: http://www.fit.vutbr.cz/study/courses/ZRE/public/opora/zre_opora.pdf

[15] MARTENS, R. a L. CLAESEN. Dynamic programming optimisation for on-line signature verification. *Proceedings of the Fourth International Conference on Document Analysis and Recognition*. IEEE Comput. Soc, 1997, s. 653-656. DOI: 10.1109/ICDAR.1997.620587. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=620587

[16] da Costa Filho, A.C.B., de Brito Filho, J.P., de Araujo, R.E., Benevides, C.A.: Infrared-Based System for Vehicle Classification. In: 2009 SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference (IMOC), pp. 537–540 (2009)

[17] TAVARES, Bruno, Ricardo AZEVEDO, Pedro MENDES JORGE, António SERRADOR a Tomé CANAS. Vehicle classification based on the magnetic profile. In: *Instituto Superior de Engenharia de Lisboa* [online]. 2013 [cit. 2015-05-05]. Dostupné z: http://www.adeetc.isel.pt/cetc13/Book%20of%20Abstracts/Oral%203C/cetc2013_submission_140.pdf

[18] Cython. BEHNEL, Stefan. *Cython: C-Extensions for Python* [online]. 2015 [cit. 2015-01-13]. Dostupné z: http://cython.org/

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

A KNIHOVNA DTW V C

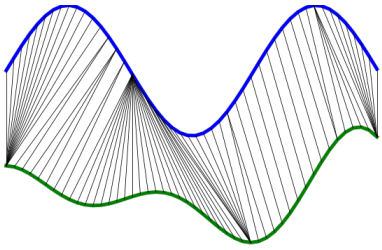
```
#include "cdtw.h"

int main()
{
    /*vstupni sekvence*/
    /*0 na zacatku nemaji vliv - specifikum implementace*/
    double r[] = {0,2,3,4,3,5,5};
    double q[] = {0,1,2,3,2,4,5};

    /*a jejich delky*/
    int len_ref = sizeof(r)/sizeof(r[0]);
    int len_query = sizeof(q)/sizeof(q[0]);

    /*nastaveni dtw*/
    struct t_dtw_settings dtw_settings ={true,      /*pocitani cesty*/
                                         _MANHATTAN,/*distancni funkce*/
                                         _DP2,      /*krokova funkce*/
                                         false,     /*okno*/
                                         0,         /*parameter okna*/
                                         false};    /*normalizace*/

    /*dodatecny rozmer matici*/
    /*pro _DP2 je 1 */
    dtw_settings.offset = extra_size(dtw_settings.dp_type);

    /*alokace globalni nakladove matice*/
    double *cost = (double*)malloc(sizeof(double)*(
                                (len_ref+dtw_settings.offset)*
                                (len_query+dtw_settings.offset)));

    /*alokace cesty*/
    struct t_path_element* path = (struct t_path_element*)malloc(
                                    sizeof(struct t_path_element)*
                                      (len_ref+len_query));
    int path_len = 0;

    double dtw_dist = cdtw(r, q,
                           len_ref-1,
                           len_query-1,
                           cost,
                           path, &path_len, /*v path_len
                                              bude realny rozmer cesty*/
                           dtw_settings);
    print_matrix(cost,
                 len_ref,
                 len_query);
    for (int i = 0; i<path_len; i++)
        printf("i: %d  j: %d\n", path[i].i, path[i].j);

    return 0;
}
```

B KNIHOVNA DTW V PYTHON

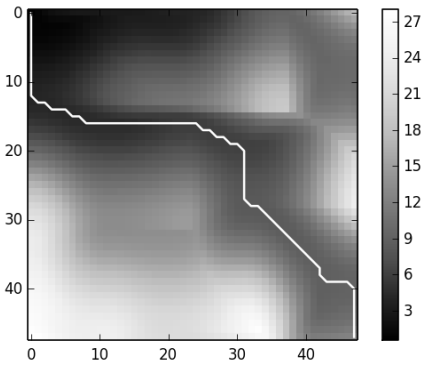
```
In [3]: import numpy as np
        import pydtw

In [4]: t = np.arange(0,3*np.pi,0.2)
        r = np.sin(t)
        q = np.cos(1.2*t+1) + np.sin(t)
```

Klasicke DTW

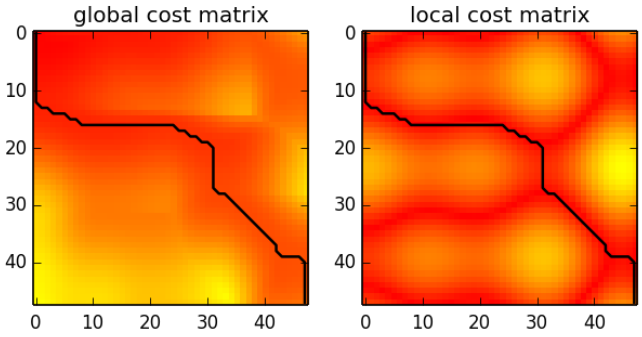
```
In [5]: d = pydtw.dtw(r,q,pydtw.Settings(dist = 'manhattan',
                                         step  = 'dp2',
                                         window = 'nowindow',
                                         compute_path = True))

In [6]: #distace
        d.get_dist()

Out[6]: 12.37014033416319

In [7]: #nastaveni prostredi
        %matplotlib inline
        import matplotlib as mpl
        mpl.rc("savefig", dpi=120)
        #alignment
        d.plot_alignment()
```



```
In [8]: #tisk krokove funkce
        print pydtw.Step().step_str('dp2')

*  Step patern dp2:
  *  min(
  *      cost_matrix[i][j-1]   +   d(r[i],q[j])
  *      cost_matrix[i-1][j]   +   d(r[i],q[j]),
  *      cost_matrix[i-1][j-1] +   d(r[i],q[j])
  *      )

In [9]: d.plot_mat_path()

Out[9]: <matplotlib.axes.AxesSubplot at 0x32fddb0>
```

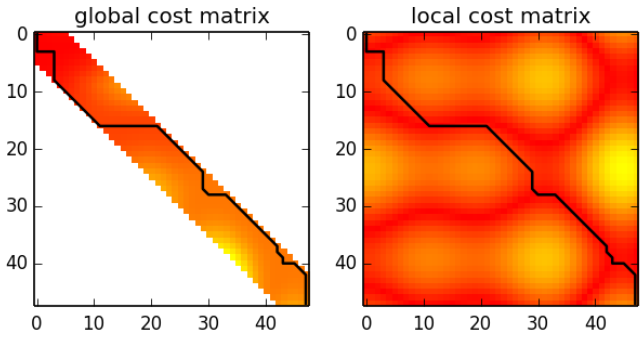



```
In [10]: #priklad definice vlastni kreslici funkce
         import matplotlib.pyplot as plt
         import matplotlib.cm as cm
         def plot_cost(d):
             ils = []
             jls = []
             for i,j in d.get_path():
                 ils.append(i), jls.append(j)
             fig = plt.figure()
             axl = fig.add_subplot(121)
             axr = fig.add_subplot(122)
             axl.plot(jls,ils, 'k', lw = 1.5)
             axr.plot(jls,ils, 'k', lw = 1.5)
             axl.imshow(d.get_cost(),#globalni nakladova matice
                        interpolation = 'nearest',
                        cmap = cm.autumn)
             dist_matrix = np.abs(r[..., np.newaxis] - q) #vypocet
                                                          #lokalni nakladove matice
             axr.set_title('local cost matrix')
             axl.set_title('global cost matrix')
             axr.imshow(dist_matrix, interpolation = 'nearest', cmap = cm.autumn)


         plot_cost(d)
```

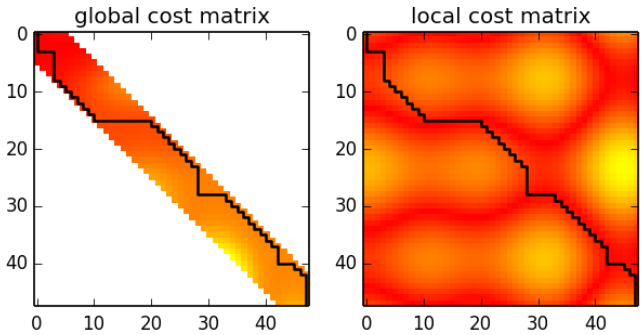


*Globalni omezeni

```
In [11]: d = pydtw.dtw(r,q,pydtw.Settings(dist = 'euclid',
                                           step  = 'dp1',
                                           window = 'palival_mod',
                                           param = 0.1, #sirka okna = 0.1*referencni sekvence
                                           compute_path = True))
         plot_cost(d)
```




```
In [12]: d = pydtw.dtw(r,q,pydtw.Settings(dist = 'euclid',
                                           step  = 'dp3',
                                           window = 'palival_mod',
                                           param = 0.1, #sirka okna = 0.1*referencni sekvence
                                           compute_path = True))
         plot_cost(d)
```



C VYBRANÉ VÝSLEDKY KLASIFIKACE

Testování se provádělo pomocí leave-one-out křížové validaci, počet sousedů byl 3.

	metoda	ED	MAXH	CORR	DTW					
	normalizace délky	50 prvků	-	50 prvků	-	-	-	-	-	-
nastavení DTW	distanční funkce	-	-	-	Euklidovská	Euklidovská	Euklidovská	Manhattan	Euklidovská	Manhattan
	kroková funkce	-	-	-	dp1	dp2	dp3	dp2	dp2	dp2
	okno	-	-	-	ne	ne	ne	ne	ano	ano
	šířka [%]	-	-	-	-	-	-	-	5	5
sedan	FPR [%]	1,94	11,51	2,05	5,78	5,24	13,02	3,89	2,00	2,32
	FNR [%]	3,22	20,76	3,51	7,31	6,43	5,26	5,26	3,95	4,39
	ACC [%]	97,71	85,83	97,55	93,81	94,44	89,07	95,74	97,48	97,12
hatchback	FPR [%]	1,26	9,43	1,92	3,30	2,80	2,14	2,47	1,70	1,92
	FNR [%]	6,28	32,26	6,56	20,11	18,02	39,11	13,55	6,70	6,98
	ACC [%]	97,32	83,85	96,77	91,95	92,90	87,42	94,40	96,88	96,65
dodávka	FPR [%]	0,77	3,10	1,76	2,17	1,99	2,53	1,72	0,95	0,81
	FNR [%]	3,41	9,60	5,88	5,57	4,33	2,48	4,95	4,33	4,33
	ACC [%]	98,90	95,91	97,71	97,40	97,71	97,48	97,87	98,62	98,74
nákladní	FPR [%]	0,58	2,49	0,58	0,87	0,75	0,95	0,75	0,70	0,70
	FNR [%]	13,22	29,75	28,93	16,53	18,18	20,66	14,88	13,22	15,70
	ACC [%]	98,82	95,96	98,07	98,38	98,42	98,11	98,58	98,70	98,58
kamion	FPR [%]	0,85	1,52	1,16	0,79	0,69	0,69	0,58	0,48	0,48
	FNR [%]	1,86	3,88	2,33	1,71	1,40	2,33	1,55	1,55	1,71
	ACC [%]	98,90	97,75	98,54	98,97	99,13	98,90	99,17	99,25	99,21
autobus	FPR [%]	0,44	0,44	0,20	0,28	0,28	0,52	0,16	0,24	0,28
	FNR [%]	23,40	61,70	27,66	31,91	25,53	44,68	23,40	12,77	10,64
	ACC [%]	99,13	98,17	99,29	99,13	99,25	98,66	99,41	99,53	99,53
	$\overline{\text{ACC}}$ [%]	98,42	92,91	97,91	96,44	96,84	94,61	97,44	98,37	98,26
	Overall ACC[%]	**95,38**	**80,51**	**93,96**	**89,82**	**90,93**	**84,77**	**92,58**	**95,23**	**94,91**

Legenda:

- ED – Euiklidovská vzdálenost
- MAXH – rozdíl výšek vozidel
- CORR – Personová vzdálenost