|  |  |
| --- | --- |
| **TECHNICKÁ UNIVERZITA V KOŠICIACH**  **FAKULTA ELEKTROTECHNIKY A INFORMATIKY** | |
| **SCALE-INVARIANT FEATURE TRANSFORM**  **KEY POINTS DESCRIPTORS**  **Zadanie z predmetu Počítačové videnie** | |
|  | |
| **2019** | **Gonda Tomáš, Bc.**  **Kobularčík Peter, Bc.**  **Vasilko Matej, Bc.** |

**Znenie zadania:**

Cieľom zadania je vytvorenie vlastnej implementácie algoritmu SIFT (Scale-Invariant Feature Transform) kľúčových bodov deskriptorov v ľubovoľnom programovacom jazyku. Následne je potrebné zavolať vstavanú funkciu, ktorá má rovnakú funkcionalitu, vykonať porovnania, experimenty a zdokumentovať všetky spomínané úkony.

**SIFT (scale-invariant feature transform)**

Tento algoritmus je určený pre detekciu a popísanie základných rysov obrázku. Pozostáva z troch hlavných častí, a to:

* Detekcia kľúčových bodov (*ang. keypoint detection*)
* Vytvorenie deskriptorov pre kľúčové body (*ang. keypoint descriptors*)
* Spájanie základných rysov obrázku (*ang. feature matching*)

**Detekcia kľúčových bodov**

Ako prvé je potrebné vytvoriť škálovateľný priestor, ktorý nám identifikuje najvýraznejšie vlastnosti obrázka a zároveň ignoruje šum. Na zníženie šumu v obrázku použijeme techniku Gaussovského rozmazania tak, že pre každý pixel v obrázku vypočíta hodnotu na základe susedných pixelov, ako môžeme vidieť na obrázku.



Obr. 1 Gaussian blur

Po tomto kroku musíme zabezpečiť nezávislosť od rozsahu. To sa vykonáva opakovaným zmenšením obrázku a rozmazávaním a vytvoríme tak vyššie spomínaný škálovateľný priestor. Tieto obrázky budeme medzi sebou odčítavať, čím získame Difference of Gaussian, z ktorých určíme kľúčové body. Tieto body nájdeme s pomocou lokálneho maxima a minima, ktoré získame porovnávaním každého pixelu s okolitými pixelmi a s pixelmi s predchádzajúcej a nasledujúcej oktávy – dokopy ho porovnáme s 26 pixelmi. Následne vyberieme stabilné kľúčové body.

**Kľúčové body deskriptorov**

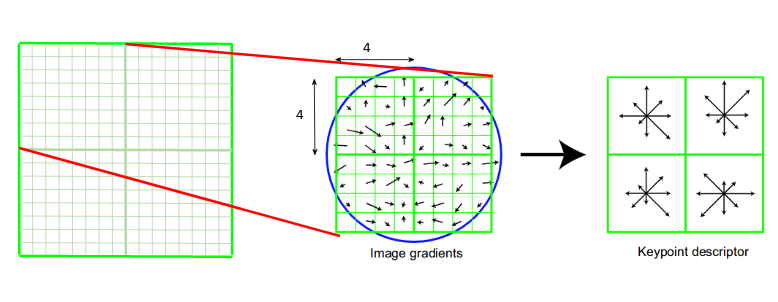
Pri vytváraní deskriptorov musíme najskôr vypočítať gradienty pre osi x a y. Tieto vypočítame tak, že pre os x odčítame hodnoty vpravo a vľavo od požadovaného pixelu a absolútna hodnota tohto rozdielu je náš gradient pre os x. Podobný postup opakujeme aj pre os y. S týmito hodnotami dokážeme vypočítať magnitúdu a orientáciu každého pixelu v obrázku.

Vzorec pre magnitúdu, kde Gx a Gy sú gradienty os:

Vzorec pre orientáciu:

Z vypočítaných hodnôt vytvoríme histogram orientovaných gradientov (ang. Histogram of Oriented Gradients). Podľa jeho vrcholu vieme určiť orientáciu pixelu. Ak je vrcholov viac, vygenerujeme ďalší kľúčový bod.

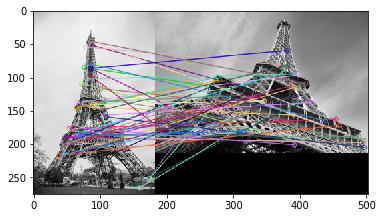
Deskriptor vytvoríme tak, že pre každý kľúčový bod vezmeme blok okolitých 16x16 pixelov. Tento blok rozdelíme na 4x4 podčasti a pre každý vytvoríme HoG. S týmito hodnotami dokážeme deskriptor vizualizovať, ako môžeme vidieť na obrázku.



Obr. 2 SIFT descriptors

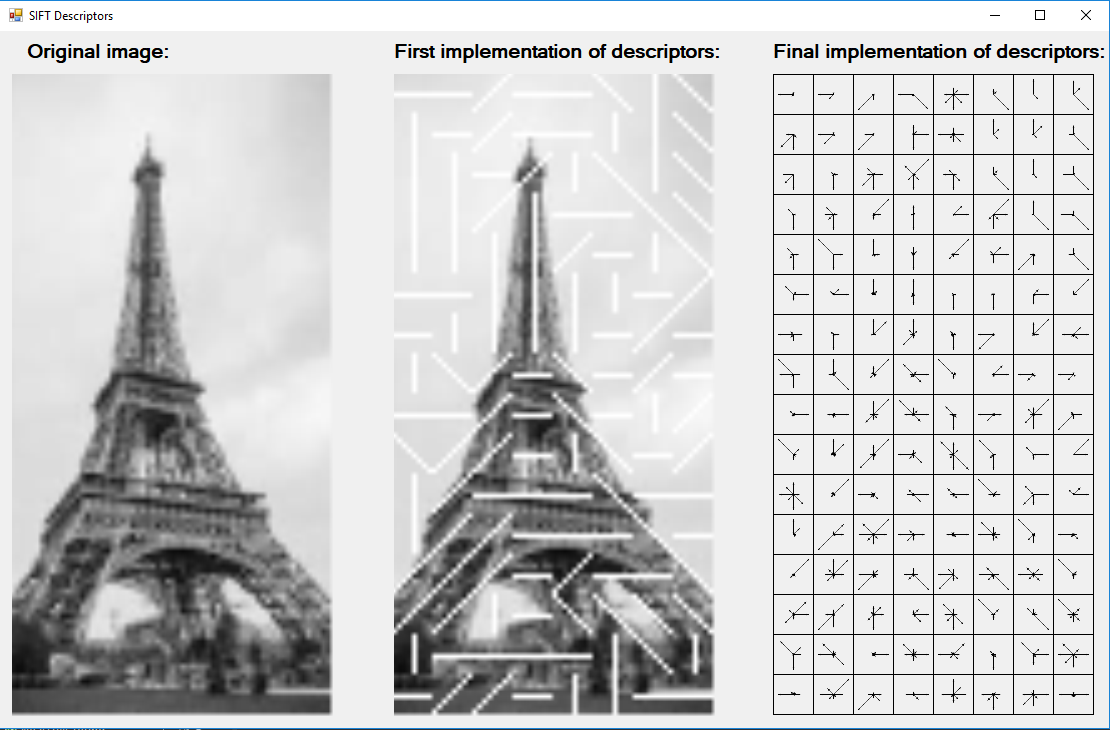
**Spájanie kľúčových bodov**

Spájanie kľúčových bodov je posledná časť algoritmu SIFT. Pre dva rôzne obrázky sú vytvorené kľúčové body a ich unikátne deskriptory. Podľa týchto údajov vieme spojiť kľúčové body a toto spojenie zobraziť. Býva konvenciou obmedziť počet zobrazených spojení - kvôli prehľadnosti.

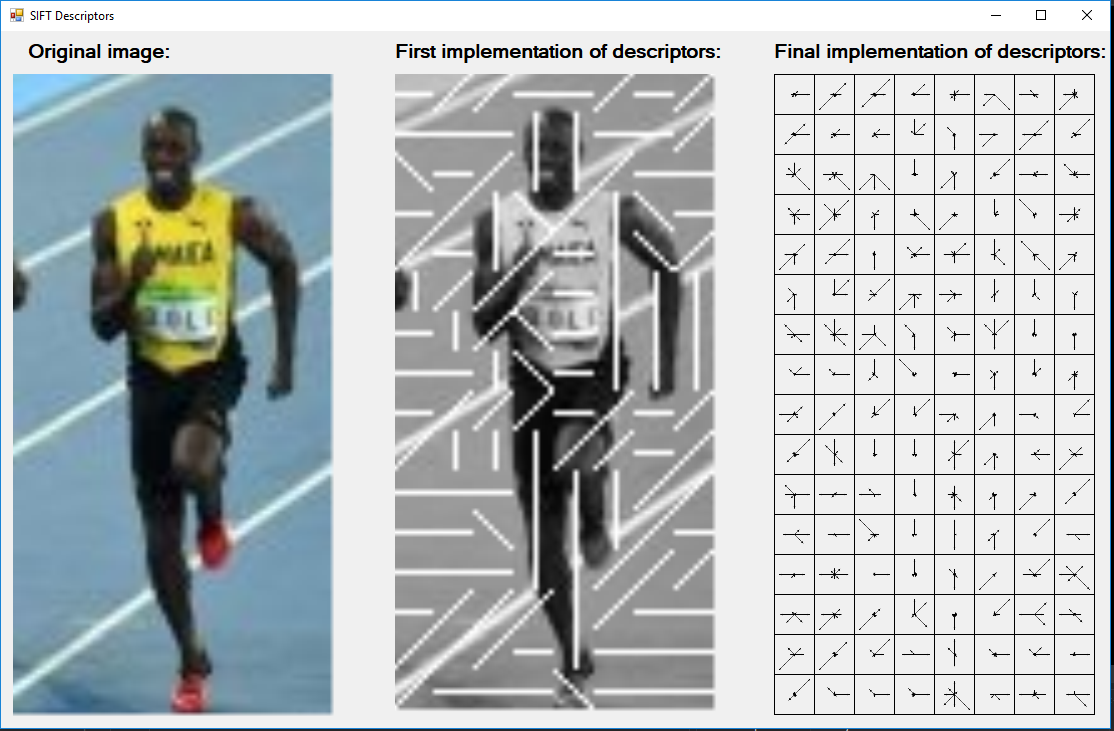


Obr. 3 SIFT feature matching

**Vizualizácia našej implementácie SIFT algoritmu:**



Obr. 4 SIFT naša vizualizácia

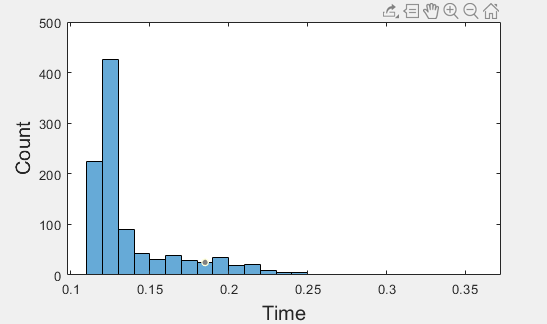


Obr. 5 SIFT - naša vizualizácia 2

**Experimenty**

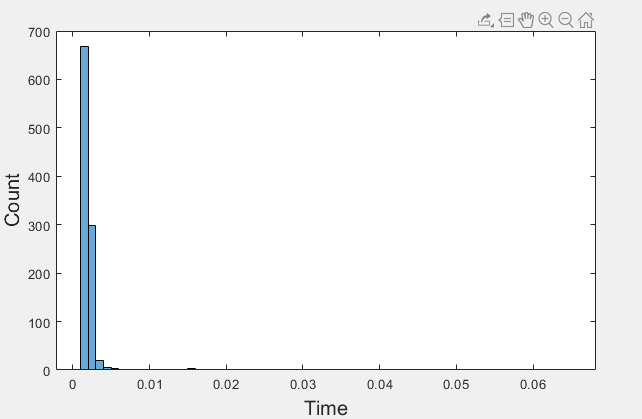
Dôležitou súčasťou dokumentácie sú experimenty, ktoré boli vykonané na našej implementácií algoritmu ako aj na vstavanej funckii, ktorá bola zavolaná z knižnice EmguCV. Bolo vykonaných 1000 meraní, z ktorých každá iterácia merala čas od spustenia programu po jeho ukončenie.

Na nasledujúcom obrázku môžeme vidieť distribúciu hodnôt nameraných na našej implementácii.



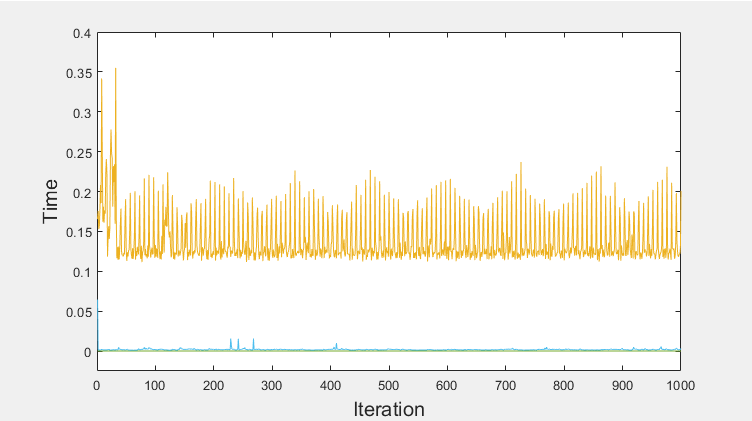
Obr. 6 Distribúcia nameraných hodnôt našej implementácie

Po zavolaní vstavanej funkcie z knižnice a vykonaní 1000 iterácií môžeme vidieť namerané hodnoty (hlavne čas) v nasledujúcom grafe. Môžeme vidieť, že vstavaná funkcia sa vykoná podstatne rýchlejšie, ako naša implementácia.



Obr. 7 Distribúcia nameraných hodnôt vstavanej funkcie

Pre lepšiu predstavu sú na nasledujúcom obrázku zobrazené hodnoty meraní oboch implementácií, pričom oranžovou farbou sú vykreslené časy našej funkcie a modrou farbou sú znázornené časy pre vstavanej funkcie.



Obr. 8 Grafické porovnanie časov vstavanej funkcie a našej implementácie

**Porovnanie časov**

- Priemerná a celková rýchlosť vykonania vstavanej funkcie pri 1000 iteráciách:

* Priemerný čas: 00.0020000 [sec]
* Celkový čas: 01.9503145 [sec]

-Priemerná a celková rýchlosť vykonania našej implementácie pri 1000 iteráciách:

* Priemerný čas: 00.1380000 [sec]
* Celkový čas: 02:18.1386334 [min]