Sledování pohybujících se částic

Bc. Ivo Pešák

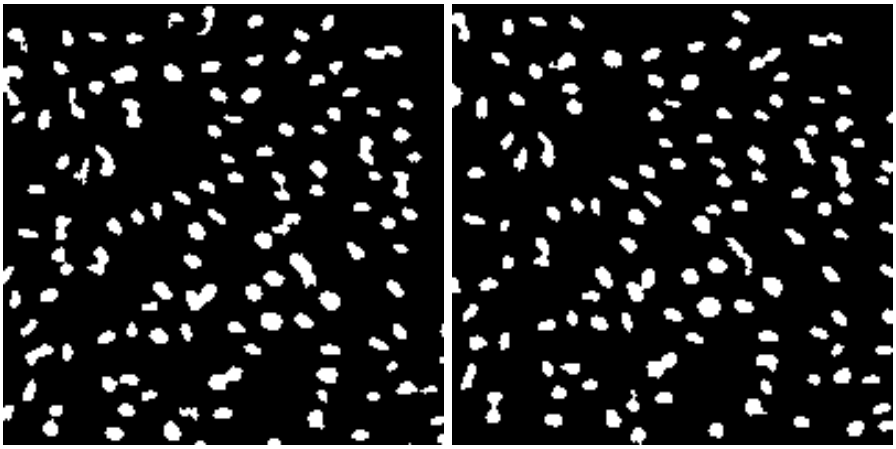
Listopad 2022

Abstrakt

Cílem dokumentu je popsat fungování naivního algoritmu pro sledování částic mezi skupinou snímků. Algoritmus dále počítá průměrnou rychlost částic a dokáže částečně redukovat dopad třesu mezi snímky.

1 Vstup

Vstupem algoritmu jsou snímky částic, které byly zpracovány černobílým filtrem s prahem aktivace. Díky tomuto je obrázek lehčí pro zpracování. Obrázky musí být načítány v takovém pořadí, v jakém byly foceny, aby byl výpočet správný.



Obrázek 1: Příklad dvou snímku na vstupu

2 Detekce částic

Jako první krok algoritmu se detekují částice v obrázku. Jelikož barvy na něj jsou pouze černá a bílá, tak je detekce jednodušší. Stačí nám kontrolovat pouze dvě hodnoty. Obrázek se nejprve načte jako bitmapa. Kontrolujeme postupně

všechny pixely a pokud narazíme na pixel bílý, prohledáme jeho okolí pro nalezení částice. Prohledávání kontroluje sousedící pixely a všechny bílé přidá do prohledávací fronty. Pokud je fronta prázdná, tak je detekce konkrétní částice ukončena a částice je detekována a přidána do kolekce částic času *T*. Aby nedocházelo k opakované detekci částice, tak je přítomna i maska která obsahuje informaci jaké pixely již byly prohledány. Pro lepší pozdější kalkulace je vypočten také centroid částice z pozic, přes které se rozléhá. Částice je pak v programu reprezentována následovně.

```
class Particle
{
    List<List<Vector2>> Positions;
    List<Vector2> Centroids;
    int TimeOfCreation;
}
```

Listing 1: Reprezentace částice

Positions i *Centroids* je reprezentováno jako List (kolekce), jelikož obsahuje data v čase. Pokud se tedy částice vyskytuje ve třech snímcích, tak kolekce obsahují tři informace. *Positions* je kolekce bodů v čase, na kterých se částice nachází. Z těchto bodů je vypočten již dříve zmíněný centroid. Uložena je také informace, ve kterém čase byla částice stvořena. Časem se rozumí snímek. Tedy částice detekovaná poprvé na prvním snímku má hodnotu *TimeOfCreation* rovnou nule. Částice detekovaná poprvé na desátém snímku má hodnotu *TimeOfCreation* rovnou devíti.

Částice jsou pak ukládány do kolekce obdobně jako *Positions*. Tedy kolekce kolekcí, kde pozice vnitřní kolekce označuje výskyt v čase.

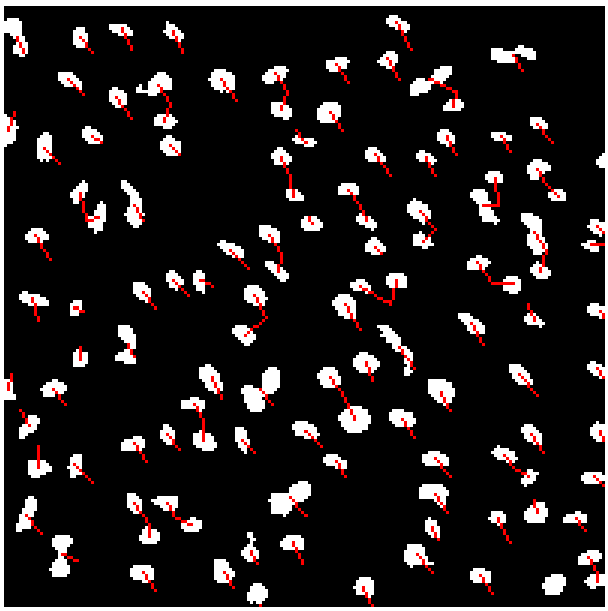
3 Detekce na minulém snímku

Pokud se proces detekce nachází v čase $T > 0$ (zpracováváme obrázek v pořadí po prvním), tak se můžeme pokusit k detekované částici na aktuálním obrázku, přiřadit částici na obrázku minulém. Toto nemá v čase $T = 0$ smysl, protože žádné částice v čase předchozím nebyly. Detekce probíhá na základě blízkosti centroidu. Prochází se centroidy částic v čase $T - 1$ a hledá se takový, který je nejbližší k centroidu aktuální detekované částice podle Euklidovské vzdálenosti. Je zde však omezení na nejdelší přijatelnou vzdálenost. Pokud není nalezen centroid se vzdáleností menší než je limit, tak je částice označená jako nová s hodnotou *TimeOfCreation* rovnou aktuálnímu času. V opačném případě je detekovaná částice nahrazena již vytvořenou z času $T - 1$ a její *Positions* a *Centroids* se rozšíří o aktuální pozice. Problém by mohl nastat v případě, když se jedna částice rozdělí na dvě. Tak by se dvě nové částice nahradily jednou starší a došlo by (z důvodu sdílené reference) k rozšíření jejich kolekcí o dva údaje (za každou částici jeden), namísto jednoho. Aby k tomuto nedošlo, tak je ještě v případě nalezení částice v čase $T - 1$ kontrolováno, jestli se tato částice už nenachází v kolekci částic času T (jiná částice ji již označila jako předka

a přidala její referenci). Pokud referenci obsahuje, tak se pouze vytvoří nová částice, ale se stejnými daty jako předek. Jde pouze o vytvoření nového objektu s kopií dat, aby nedošlo k problémům s referencemi a nechtěnému ovlivnění jiného objektu.

3.1 Výstup

Po zpracování každého snímku se vytvoří jeho kopie a vyznačí se na ní vypočtená cesta, kterou částice urazila. Soubor se pak uloží do složky *output*.



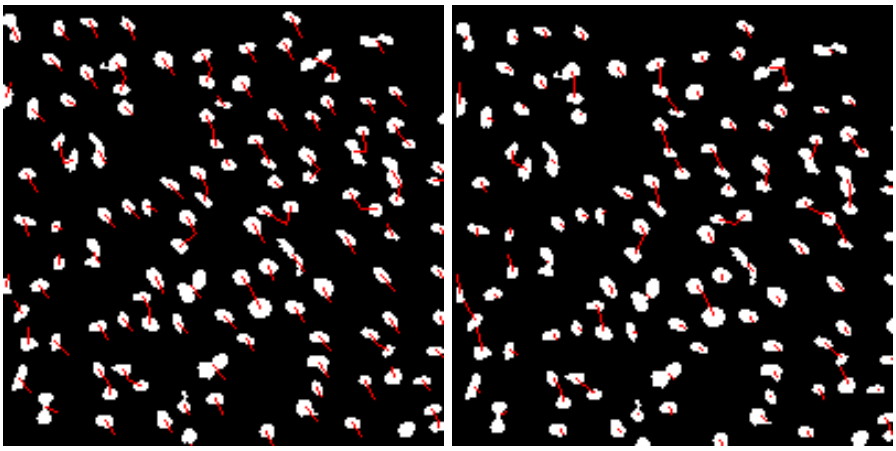
Obrázek 2: Výstup pro vstup z obrázku 1. Cesta je vyznačena červenou barvou

4 Výpočet charakteristik

Pro závěrečný výpočet průměrné rychlosti a uražené vzdálenosti se postupně prochází kolekce částic v čase. Protože jsou stejné částice v kolekci uložený ve více časech (stejná reference na objekt), tak se při procházení kolekce berou v potaz pouze ty částice, které byly stvořeny v čase aktuální čase iterace. Samotný výpočet pak prochází postupně všechny centroidy a počítá absolutní rozdíl mezi centroidem v čase T a T - *1*. Tyto rozdíly sečte a podělí délkou životnosti částice pro získání její průměrné rychlosti. Tento výpočet se provede pro všechny částice. Průměrné rychlosti se sečtou a podělí celkových počtem unikátních částic a vznikne nám průměrná rychlost částic na snímcích. Pro výpočet průměrné uražené vzdálenosti se vynásobí průměrná rychlost počtem snímků.

5 Stabilizace snímků

V případě že snímky na vstupu nejsou foceny stabilně a je v nich znát lehký třes, tak je možné je v algoritmu stabilizovat. Stabilizace probíhá tak, že po zpracování snímku a propojení částic se snímkem minulým se nepokračuje následujícím snímkem, ale. Proces stabilizace se provádí mezi detekcí částic na minulém snímku a zpracováním snímku následujícího. Z propojených částic se vypočítá průměrná rychlost v tomto snímku. Avšak počítá se neabsolutní, takže jde v podstatě o průměrný offset všech částic. Tento offset se ke všem částicím přičte a s takto upravenými daty pokračuje algoritmus dál. Stabilizace se takto provádí pro všechny snímky. Tento proces ale může vést ke špatnému výpočtu v případě, kdy na většině snímků třes není. Stabilizace bude odečítat offsety, které ve skutečnosti nejsou způsobeny třesem, ale skutečným pohybem částic.



Obrázek 3: Srovnání výstupu se stabilizací (vpravo) a bez

Algoritmus 1: Detekce částice na minulém snímku

```
Input: Cesta k souboru Path snímku času T a čas T
Result: Kolekce částic v čase T
bmp := NačtiBitmapuZeSouboru(Path);
výsledek := PrázdnáKolekce();
foreach pixel in bmp do
    částice := ZkusNajítČástici(pixel);
    if částice is not null then
        if T ≥ 1 then
            centroid := SpočtiCentroid(částice);
            rodič := ZkusNajitRodičeVčase(centroid, T - 1);
            if rodič is not null then
                if rodič not in výsledek then
                    PřidejCentroidČástici(rodič, centroid);
                    částice := rodič;
                else
                    nováČástice := VytvořPodleVzoru(rodič);
                    OdstraňPoslednícentroid(nováČástice);
                    PřidejCentroidČástici(nováČástice, centroid);
                    částice := nováČástice;
                end
            end
        end
        PřidejDoKolekce(výsledek, částice);
    end
end
return výsledek;
```