# 8. Sledování objektu (tracking). Formulace úlohy, standardní algoritmy (např. Kanade-Lucas tracker)

# Sledování objektů

Sledování (*video tracking*) je **proces** určení polohy **pohybujícího se objektu v čase** použitím **kamery**. Algoritmus vytvořený pro sledování **analyzuje video snímky** a určuje polohu objektu v jednotlivých snímcích.

 Jednoduchými **modely** pohybu pro sledování můžou být např.

* Planární objekt s modelem pohybu představující 2D transformaci obrazu (affiní transformaci nebo homografii) objektu.
* Rigidní 3D objekt s modelem pohybu závisející na jeho 3D poloze a orientaci.
* Objekt se **zajímavými** **body** (např. Harrisovy body) nebo oblastmi (např. MSER) v obraze, které jsou jednotlivě sledovány, to vytváří robustnost vůči zákrytům.
* Nerigidní (deformovatelné) objekty aproximovány na povrchu síťovým grafem. Pohyb objektu je určený pozicí hran síťového grafu.

Mezi nejznámější sledovací algoritmy patří **korelace**, **Mean-shift**, **Kanade-Lucas-Tomasi** (KLT) tracking, Kalman tracking, particle filtering, (sekvenční) linearní prediktory atd. Seznáníme se sledováním pomocí korelace a KLT trackingem.

# Sledování pomocí korespondencí nebo korelace

Již jsme se naučili hledat korespondence mezi obrázky. Úloha sledování je velmi podobná a hledá **korespondenci** **objektu** (výřezu, oblasti, významného bodu s okolím) **známého** v **jednom** **obraze** videosekvence v **neznámém**(ých) obraze(ch) ve videoesekvenci **následujícím**(ch). Rozdílem proti hledání korespondencí je možnost předpokládat jen **malé změny objektu** (polohové, deformační, fotometrické…) **mezi jednotlivými snímky**, avšak změny v průběhu **celé** **videosekvence** však mohou být **značné**. Výběr sledovaného objektu se provádí zpravidla ručně nebo automatickou detekcí.

**Korespondence**

Jsme schopni nalézt korespondující body mezi po sobě **jdoucími obrázky** **sekvence** a stejně tak i **homografii** **dvou** **snímků** se známými korespondujícími body. Vhodnou kombinací pro první pokusy se zdají být harrisovy body a dct popis. Zde můžeme při hledání korespondencí zkusit přidat omezení, že **vzdálenost** dvou korespondujících bodů nemůže být moc velká (snímky jdou v sekvenci po sobě, scéna se nemohla fyzicky příliš změnit).

Použité metody necháme na vašem uvážení a můžete pochopitelně i různě experimentovat. Minimem pro splnění této úlohy jsou funkční funkce, které jsou schopny **udržet a transformovat výřez skrz celou sekvenci**.

**Sledování pomocí korelace**

Sledování pomocí korelace hledá **korespondence okolí významných bodů objektu** ne na základě jejich **zjednodušeného popisu** (SIFT atd.), ale přímo pomocí **porovnání těchto okolí**. Měřítkem, jak moc si okolí odpovídají, může být **lineární korelační koeficient**.

**Korelační koeficient** (také nazývaný normalized cross-correlation, zkratka NCC) je definován jako

 , kde  je obrazový **vzor** a  vstupní **obraz**.

* **Výhody**: NCC je invariantní vůči lineárním transformacím jasu.
* **Nevýhody**: NCC není invariantní ke změnám **měřítka** obrazu, jeho **rotaci** a **perspektivní** distorzi. V některých aplikacích může být uváděna nevýhoda výpočetní náročnost.

Jak jsme již uvedli, metoda je vlastně obdobou **hledání korespondencí** - v každém **snímku** nalezněte **harrisovy body** a na základě **porovnání jejich okolí** za pomoci **korelačního koeficientu** se je pokuste **spárovat**, najít **korespondence**. Velikost uvažovaného okolí nastavíme parametrem a přesnost sledování je vhodné porovnat pro různá nastavení. S předpokladem jen malých posunů mezi následujícími snímky je zbytečné, dokonce nevhodné **počítat korelace** mezi **všemi dvojicemi okolí**. Pro každý bod v prvním obrázku stačí spočítat korelace jen s body v druhém obrázku do určité vzdálenosti, protože vzdálené body nemohou korespondovat. Tuto vzdálenost nastavíme parametrem a její velikost je závislá na charakteru pohybu v konkrétní sekvenci. Pro naši úlohu doporučujeme hodnotu asi 30px.

Pro spárování dvojic bodů s vysokým korelačním koeficientem použijte např. známého principu vzájemně nejbližších korespondencí - tedy, že výřez imgPrev z **předchozího** obrázku je nejbližší výřezu imgNew ze všech popisů v **aktuálním** obrázku jen tehdy, pokud nejbližší výřez imgNew z aktuálního obrázku je nejbližší výřezu imgPrev z předchozího ze všech předchozích.

Pokud k bodu z předchozího obrázku není nalezen korespondující bod v aktuálním obrázku, je tento bod **zahozen** a dále se již pro sledování neuvažuje.

# Kanade-Lucas-Tomasi Tracking (KLT tracker)

KLT se snaží měřit similaritu pomocí kvadrátu normy, která určuje **kvadratickou chybu mezi obrazovým vzorem a vstupním obrazem**. Minimalizace chyby je dosahováno **iterativně** pomocí **gradientní metody**.

**Algoritmus:**

Představme si obrazový vzor , který získáme vybráním okénka v čase , kde okénko překrývá námi sledovaný objekt.  je sloupcový vektor souřadnic v obrazu . V čase  se námi sledovaný objekt posune do jiné části obrazu. V čase  vybereme okénko , které je ve stejné pozici jako v čase . Uvažujme množinu dovolených translací  Gradientní metodou (Newton-Raphson) se okénko  v každé iteraci posunuje tak, aby byla minimalizována kvadratická chyba mezi vzorem a vstupním obrazem.

Optimální posunutí  okénka minimalizuje nepodobnost obrazového vzoru a vstupního obrazu

 Předpokládejme nejlepší změnu posunutí o , tímto (1) přejde ve tvar



Tento výraz se snažíme minimalizovat vzhledem k . Nelineární výraz (2) je linearizován Taylorovým rozvojem prvního řádu na tvar



kde  je gradient v obrazu vypočtený z  Člen  je Jacobiho matice (symetrická matice prvních parciálních derivací) translací ve směru  a . Výsledná Jacobiho matice se získá



K nalezení minima výrazu (3) parciálně derivujeme podle . Poté explicitně vyjádříme posunutí



kde  je aproximace Hessovy matice používaná v Gauss-Newtonově gradientní metodě. Je třeba také poznamenat, že Gauss-Newtonova metoda řeší nelineární regresi (nelineární metodu nejmenších čtverců) a minimalizuje tak tvar (3). Povšimněme si, že aproximace Hessovy matice v této nelineární regresi je rovna autokorelační matici (Harrisově matici),tj. skalárním součinem prvních parciálních derivací. Z toho vyplývá, že nejlepší je **trackovat body v malém okolí okolo Harrisových bodů**.



Použitím výrazu (4) se v (5) a (6) zjednodušší . Krok (5) je vypočítáván v každé iteraci a pro další iteraci je updatováno celkové posunutí



Konvergenci můžeme ukončit nastavením **maximálního počtu iterace**, nebo ukončovací **podmínku** ve tvaru



**Poznámka:**

takže T(x) je okénko kolem bodu x v obrázku (ten vzor, který sledujeme), I(W(x;p)) je okénko v obrázku, které získáme tak, že **původní okénko** na souřadnicích x **posuneme** o vektor p

**bod 1)** ten vztah představuje chybu, odečteš od sebe **jasy** v obou okénkách (posunutého a toho původního vzoru), rozdíly dáš na druhou a sečteš ... prostě **chyba**, která odpovídá **Eukleidovské vzdálenosti**, ideálně by chyba měla být nulová, pak víme, že posunutí p je správné, ve vztahu 2 rozepíšeme p jako p + delta p, kde delta p symbolizuje přírustek (změnu p) pro následující snímek, zpočátku je tedy p = 0 a s následujícími snímky k němu něco přičítáš a posouváš okénko

**bod 3)** z předchozího snímku ty znáš to původní okénko I(W(x;p)), ale v novém snímku neznáš to posunuté okénko I(W(x;p+delta p)), takže co s tím? provedeš aproximaci ze **znalosti gradientu**, totiž každou funkci lze aproximovat následujícím způsobem: **F(x+d) = F(x) + d \* derivace F(x)**, takže oni tady to neznámé okénko I(W(x;p+delta p)) nahradí jako součet známého okénka a derivace, tedy I(W(x;p)) + grad I \* dW/dp \* delta p, takže to neznámé můžu vyjádřit pomocí toho co znám - znám **současné okénko** a **přičtu k němu nějaké gradienty**, které jsem taky schopen spočítat ze současného okénka

takže v bodě 3) je pak ta funkce přepsaná s touhle aproximací a tuto chybu chceme minimalizovat a odtud si pak vyjádří čemu by se mělo rovnat delta p

gradienty řeknou jak z toho pak vypočteš posun, dělá se to gradientní metodou, představ si to takhle

máš původní okénko v novém snímku a musíš ho někam posunout, podle toho kam ho posuneš si můžeš spočítat chybu, takže některým směrem je posouvat výhodněji než jiným směrem, ty si gradientní metodou určíš směr, kam je to nejvýhodněji ho posunout a tam ho posuneš.

Já si to představuju jakoby ve 3D grafu .. na nějaké pozici x,y máš chybu třeba E, když se posouváš nějakým směrem jdeš jakoby do **kopce** (zvětšuješ chybu), když jdeš jiným směrem jdeš **dolů** (snižuješ chybu).. prostě takové pohoří, no a **gradient** ti říká, kterým směrem máš jít nejstrměji dolů, abys minimalizoval chybu a tam to okénko posuneš.

Dále se vysvětluje, jak v okolí najdeš to minimum a nějak se to počítá, v kroku 4 si vyjádřili derivaci dW/dp, v kroku 5 potom jak se vypočítá to co hledáme delta p a k tomu vyjádření použili H hessovu matici a v 6 je jak se spočítá

v 7) updatuješ p podle spočítaného delta p a pak se testuje konvergence v 8) která zjišťuje zdali je posunutí dostatečně malé, pokud by bylo velké, tak jsme nejspíš detekovali něco jiného.. třeba objekt zmizel ze záběru, tak nám to posunutí skočí někam hodně vedle..

Samotný KLT algoritmus lze sumarizovat v několika krocích:

Zvol vzorek T(x) v okolí Harrisova bodu x v minulém obrázku xPrev, nastav P=[0;0] a iteruj:

1. Vyber vzorek  z obrázku imgNew v okolí harrisova bodu s uvažováním aktuálního posunutí **P**
2. Vypočti obrazovou chybu .
3. Vypočti gradienty  s translací  .
4. Aproximuj Hessovu matici  skalárním součinem  .
5. Odhadni translaci 
6. Updatuj parametry použitím 
7. Testuj konvergenci 

Nastav novou pozici Harrisova bodu .

Pokud algoritmus nezkonverguje v daném počtu kroků, je nejlepší tento bod z dalšího sledování **vyřadit**, protože s velkou pravděpodobností jsme nalezli jiný než skutečný posun.

Zjednodušené ilustrační schéma algoritmu na příkladu trackování auta je níže na obrázku:



# Mean-shift

Algoritmus Mean shift, který spadá do oblasti **jádrového sledování**. Algoritmy této oblasti se vyznačují svou nízkou výpočetní náročností a relativní jednoduchostí. Algoritmy pracují s **barevnou** **reprezentací** **cíle** a nevyžadují žádný externí mechanismus pro určení polohy. Barevnou reprezentací cíle se rozumí, že objekt, který má být algoritmem sledován, je vymodelován pomocí **barevného** **prostoru**. Algoritmus již nepracuje s vybraným obrazem jako takovým, nýbrž s jeho modelem. Princip algoritmu je obsažen v jeho názvu, Mean shift algoritmus hledá na každém snímku **střed** neboli **těžiště** (průměr) modelu, které reprezentuje daný objekt.

Algoritmus může tedy být shrnut do následujících šesti kroků:

1) Výběr polohy sledovací oblasti. Před zahájením sledování je tedy nutné vybrat na snímku objekt, který má být algoritmem sledován. Tento objekt se umístí do **sledovací oblasti** (oblast ve tvaru obdélníku nebo elipsy). Výpočet a zálohování histogramu cíle. Ze sledovací oblasti, kterou je vybraný cíl obklopen, je vypočítán histogram - plní funkci modelu sledovaného objektu. Algoritmus bude sledovat takovou **barevnou** **reprezentaci**, která byla sestavena při inicializaci cíle. Z tohoto důvodu je **cílový**

**histogram** uchováván a po celou dobu sledování se **nemění**. Kromě histogramu cíle je nutné také uchovat polohu vybraného sledovaného objektu. Poloha se uchovává v podobě **souřadnic** **středu** sledovací oblasti, která je vztažena k rozměrům daného snímku. Tento celý krok je krokem inicializačním a následuje přechod na nový snímek.

2) V druhém kroku se předpokládá, že je zpracováván další snímek sekvence, v této fázi je již pravděpodobné, že se vybraný cíl může nacházet na poloze **posunuté** oproti poloze zachycené v předchozím snímku. V současném snímku je sledovací oblast umístěna na stejnou polohu, v jaké se nacházela v předchozím kroku. Ze sledovací oblasti, která pravděpodobně nyní **neobklopuje** celý

sledovaný objekt, se vypočítá **histogram** **kandidáta** na cíl stejným způsobem, jakým byl vypočítán

histogram cíle. Histogram cíle je rovněž potřeba uchovat. V tuto chvílí pracuje algoritmus s dvěma

histogramy, **statickým** **histogramem** **cíle** a **histogramem** **kandidáta**.

3) Z histogramu cíle a histogramu kandidáta, provést výpočet souřadnic těžiště a výslednou polohu zálohovat.

4) Ve čtvrtém kroku se použijí souřadnice těžiště z kroku předchozího, k **vystředění** sledovací oblasti na novou polohu. Vzdálenost mezi původní polohou **středu** sledovací oblasti a nově vypočítanou polohou **těžiště** se nazývá **Mean shift vektor**. Po posunutí sledovací oblasti o tento vektor, se **opět provede výpočet histogramu kandidáta** a získá se **nové těžiště**. Celá procedura posouvání na pozici nově získaného těžiště se provádí, dokud je vektor různý od nuly, nebo dokud je větší než předepsaná minimální vzdálenost. Tato podmínka je podmínkou konvergence a cyklus provádí posouvání a znovu vystřeďování na pozici těžiště, dokud podmínka konvergence není splněna nebo dokud není dosaženo maximálního počtu iterací algoritmu.

5) Opakovat kroky 3 až 4, dokud nedojde ke konvergenci nebo dokud není dosažen maximální počet povolených iterací.

6) Při ukončení cyklu vystřeďování se předpokládá, že **cíl** byl **nalezen**, jeho poloha je zaznamenána a algoritmus očekává další snímek ke zpracování. Dále pokračuje krokem 2 a probíhá dokud není přerušen vnějším zásahem nebo dokud není snímková sekvence u konce.

**Příklad:**

|  |  |
| --- | --- |
| Ukázka Mean shift iterací je zobrazena na obrázku. Je zde zobrazen přechod ze snímku 32na snímek 33, kdy snímek 32 by mohl představovat snímek **inicializační**. Dále je na následujícím snímku 33, možné pozorovat posouvání sledovací oblasti při jednotlivých iteracích, kdy sedmá iterace je závěrečná.V této chvíli algoritmus nalezl těžiště sledovaného objektu a očekává další snímek sekvence ke zpracování. | Mean shift iterace při přechodu mezi dvěma snimky. |

**Mean Shift procedure**

****