LINEÁRNÍ APROXIMACE TVARU PROJEKCE NA LIDSKÉ ROHOVCE

M. Polčák

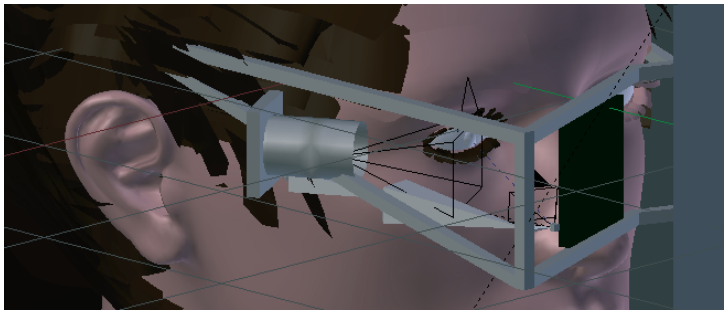
Katedra fyziky, fakulta Elektrotechnická, ČVUT

Abstrakt

Lidské oho je jedním z nejrychlejších zpětnovazebních systémů, kterými člověk disponuje. Během jeho pozorování s cílem určení pozice, na kterou je zaostřeno vzniká několik nepřesností. Za jednu z hlavních je považována chyba projekce kulové plochy rohovky na 2D snímač (kameru nebo fotoaparát). Následující článek se zabývá jednou z možností korekce tohoto vlivu.

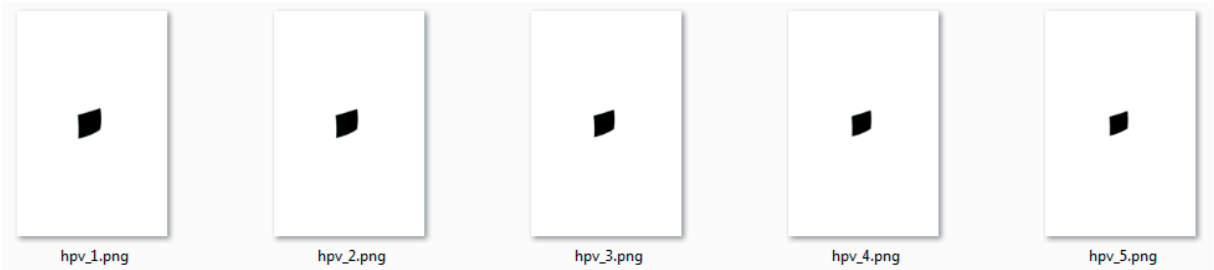
1 Trénovací množina

Pro tuto konkrétní metodu je nutným předpokladem přístup k 3D modelačnímu software, ve kterém je možné celou scénu rekonstruovat. Je tedy třeba vytvořit poměrově správný model lidského oka, pozorovaného předmětu a pozorujícího senzoru. Výstupem takto připravené simulace, která může vypadat kupříkladu jako na ilustraci č.1., je trénovací množina. Ta zobrazujíce trajektorii, kterou oko při pozorování předmětu opsalo. V tomto příkladě počítáme s pozorováním displeje PC o poměru stran 4:3 ve vzdálenosti 50 – 100 cm se zafixovanou polohou senzoru.



Ilustrace 1: 3D model potřebný pro vygenerování trénovací množiny.

Takto vytvořená trénovací množina je zobrazena na ilustraci č.2. Jednotlivé prvky jsou reálné poměrné zmenšeniny pozorovaného předmětu. V tomto případě se nejedná o zmenšeniny ve 2D obrazu prostředí, ale o zmenšeniny pozorovaného objektu v 3D simulaci. Jsou proto silně neproporcionální, neboť je v nich zahrnut příspěvek kulového zakřivení lidského oka. V případě požadavku na co nejpřesnější aproximaci, můžeme krok zmenšení 3D objektu přiblížit 0 a tím získat téměř spojitou funkci. Nicméně tento postup je nevhodný, protože vede ke zbytečnému generování velkého množství dat za cenu zpřesnění výpočtu v řádu setin.

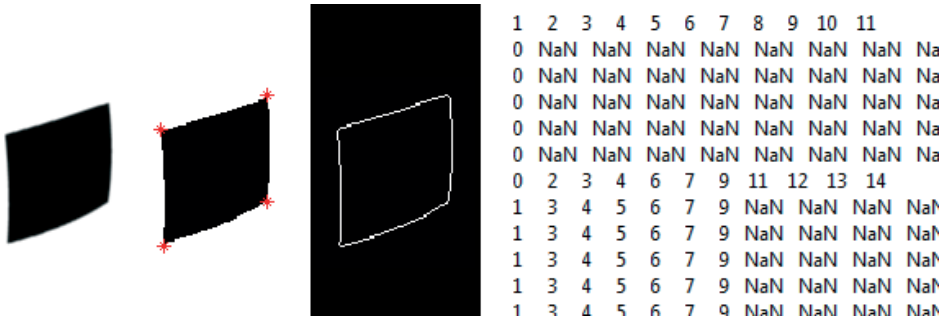


Ilustrace 2: Zmenšující se projekce trajektorie pozorování.

2 Zpracování dat

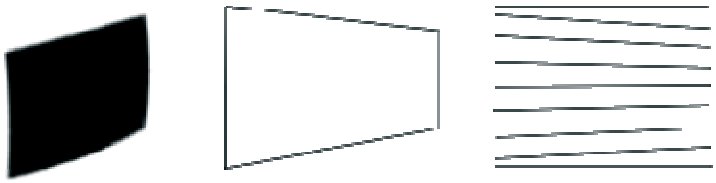
Po nahrání obrázků do matlabu je možné přistoupit k samotnému zpracování dat. Pro tento úkon doporučuji klasickou funkci imread('path'). Dalším krokem je převedení trénovacích dat z RGB stupnice na bitové pole (černo - bílá). Vlivem toho dojde ke ztrátě kvality, nicméně získáme možnost

použít standardní, předpřipravené funkce pro zpracování obrazu pomocí programu Matlab. Prvním úkonem je detekce rohů pomocí funkce corner('image',number), která se v poskytnutém obraze pokusí najít zadaný počet rohů. Tento úkon není nutný, je možné si napsat vlastní detektor, nebo zvolit ručně výchozí body analýzy. Po určení rohů je vhodné detekovat také hrany. K tomuto úkonu použijeme opět implicitní funkci edge('image'). V tomto případě je ale nutné si dát pozor na odchylku, která mezi funkcemi na detekci rohů a hran vzniká. Detekované rohy totiž nemusejí nutně tvořit podmnožinu detekovaných hran. Doporučuji proto napsat jednoduchou opravnou podmínku, která jednokrokově vyhodnotí, zda detekovaný roh není mimo hranu a provede opravu. Toto je možné díky faktu, že funkce corner() nikdy nedetekuje roh dále než jeden pixel od detekované hrany. Jinak by se musela oprava řešit použitím cyklu. Jakmile máme připraveny základní detekované body, můžeme přistoupit k jejich klasifikaci a následnému přemapování na výstupní matici. Cílem klasifikace je určit, které body leží na které hraně. Toho se dá velice snadno dosáhnout právě díky znalosti již detekovaných rohů a hran. Stačí zjistit funkcí max('vector') maximum v bitovém poli hran pro všechny posloupnosti mezi detekovanými rohy. Nejdříve vytvoříme čtyři menší matice zvlášť pro každou hranu a následně na každou z nich aplikujeme již zmíněnou funkci max(). Funkce ale funguje pouze na vertikální vektory, proto je ještě před aplikací potřeba 3 ze 4 matic narotovat, nebo transponovat tak, aby každá matice obsahovala svoji hranu v detekovatelné pozici. Jakmile jsou hrany klasifikovány můžeme přistoupit k jejich namapování, jako je uvedeno na ilustraci č.3



Ilustrace 3: Převedení trénovací množiny do bitového pole, dále detekce rohů, detekce hran a ukázka neúplné výstupní matice.

Problém nastává v okamžiku, kdy jsou protilehlé hrany různě dlouhé. Připisujeme to vychýlené poloze senzoru ve vertikální ose od pozorovaného oka. Pouze v případě že je senzor umístěn přímo naproti oku by tento problém nenastal. V této fázi přichází na řadu interpolace, kdy na kratší hraně uměle přidáme body, které by tam ve skutečnosti neexistovaly. Ačkoli to nemusí být z prvního pohledu viditelné, přesnost výsledku to neovlivní. Doplnění je vhodné pouze pro následující detekci ve vzniklé matici. Ilustrace č.4 se snaží zmíněnou interpolaci zobrazit.



3 Lineární aproximace

Poslední a zároveň nejjednodušší částí výpočtu tvoří lineální aproximace. Díky zhotovení proporcionální matice, která má v ose x namapovány sloupce s y hodnotami a naopak, stačí pouze poměrově dopočítat hodnoty zbývající. Problém při aproximaci nastává pouze v okolí rohů. Je proto vhodné linearizaci opakovat ve dvou krocích, kdy nejdříve dojde k vyplnění rovnoběžných ploch a vzniku prázdných oken v okolí rohů. V druhé fázi již nic nebrání vyplnění zbývajících oken a dokončení vzniku matice.

4 Výsledky

Výsledkem postupu je jedna matice pro každou možnou pozici senzoru vůči oku a zároveň vůči pozorovanému předmětu. Je jasné, že abychom pokryli celý prostor, potřebovali bychom velkou množinu trénovacích dat. V reálné situaci se nicméně snažíme minimalizovat tuto potřebu co nejpřesnější fixací vztažných soustav oko – senzor a oko – pozorovaný předmět. Pro případ že by se nám nicméně nepodařilo stav fixovat, disponujeme možností vytvoření dostatečného množství trénovacích dat ke stanovení odpovídajících korekčních matic. V maticích se pak jednoduše, za pomoci algoritmu nejbližšího souseda, detekuje pozice bodu který nese informaci o skutečném místě zaostření zornice. Tato přesnost je nicméně dána rozlišením senzoru a jeho vzdáleností od pozorovaného objektu. Platí přímá úměra, že čím kvalitnější senzor použijeme, tím vyšší máme pravděpodobnost určení přesné pozice.

Marek Polčák
marek@polcak.eu , polcamar@fel.cvut.cz