Metapost a Houghova transformace

Michal Dobeš, Miloslav Závodný, PřF UP, Olomouc

Abstract. The Hough transform is a powerful tool for searching shapes within images. The aim of this article is to explain the principles of the Hough transform and to provide a real example. MetaPost is a powerful language for producing figures for documents to be printed on PostScript printers. In the example, we show possibilities of MetaPost to programme the Hough transform.

Úvodem

Houghova transformace je metoda umožňující hledat křivky, procházející danými body. Používá se zejména k rozpoznávání těchto křivek ve vhodně upravených obrazech. Je použitelná pro jakoukoliv křivku, kterou lze jednoznačně vyjádřit ve tvaru $f(\boldsymbol{x}, \boldsymbol{p}) = 0$, kde $\boldsymbol{x}$ je vektor souřadnic a $\boldsymbol{p}$ vektor parametrů. Počet parametrů $\boldsymbol{p}$ ovlivňuje časovou složitost metody, takže pro praktické realizace je rozumné použít maximálně tři parametry.

Nechť má hledaná křivka k parametrů, tj. $\boldsymbol{p} = (p_1, \ldots, p_k)$. Výpočet pomocí Houghovy transformace provádíme tak, že prostor parametrů $(p_1, \ldots, p_k)$ (tzv. Houghův prostor) rozdělíme na části, tzv. diskretizujeme, a pro každý bod a pro všechna $p_1, \ldots, p_{k-1}$ reprezentující odpovídající diskrétní část dopočteme z rovnice křivky p_k. Takto vypočtené „křivky“ evidujeme – každou nalezenou pozici $(p_1, \ldots, p_k)$ zaznamenáme v příslušné diskrétní oblasti. Parametry $(p'_1, \ldots, p'_k)$ té pozice, která je nejčetnější, určují křivku $f(\boldsymbol{x}, \boldsymbol{p}') = 0$, na které leží největší počet bodů s přesností danou jemností rozdělení Houghova prostoru. Stručně řečeno: *Houghova transformace spočívá ve vytvoření diskretizovaného Houghova prostoru reprezentovaného akumulátorem, v němž se akumulují evidence o počtu bodů ležících na příslušné křivkce. Hledaná křivka je dána parametry odpovídající pozici maximální hodnoty v akumulátoru (s přesností danou jemností rozdělení Houghova prostoru).* Maximální hodnota v akumulátoru tedy udává počet bodů na nalezené křivce (byla-li hledána jedna).

V článku bychom také chtěli obrátit pozornost čtenáře na program MetaPost, který je určen k „rýsování“ pérovek a kreslení obrázků (i ba-

revných), jejichž *prvky jsou zadány analyticky*. Program je součástí instalace TEXu, ale může pracovat na TEXu nezávisle.

MetaPost je programovacím jazykem, obsahujícím základní struktury (větvení `if`, cyklus se známým počtem průchodů `for`, nekonečný cyklus `forever` i podmínku `exitif` pro ukončení cyklu). K rozmanitým datovým typům užívaným v MetaPostu patří kromě číselných proměnných `numeric`, typ `pair` reprezentující uspořádanou dvojici (pro bod nebo vektor), `path` reprezentující „čáru“, `color` deklarující barvu, `transform` deklarující proměnnou pro geometrickou transformaci, `picture` deklarující proměnnou obsahující MetaPostový obrázek, `path` reprezentující „čáru“, `boolean` pro „pravdu“ či „nepravdu“ a `pen` reprezentující tvar špičky kreslícího pera. Samozřejmostí je možnost deklarovat pole.

Pokud proměnnou předem nedeklarujete nebo není-li typu `numeric`, pak vám MetaPost neporozumí. Výjimkou jsou indexované proměnné `z`, `x` a `y` (mohou být i „čárkované“). Proměnná `z1` je automaticky považována za dvojici o složkách `x1` a `y1`, např. `z4=(1,2)` je totéž co `x4=1` a `y4=2`. Pozornost zasluhuje proměnná `whatever`, která představuje neznámou.

MetaPost umožňuje definovat procedury, tzv. makra, a to i rekurzivní. Jistým omezením MetaPostu je to, že počítá s nejmenší možnou hodnotou 1/65 536 – není prvotně určen k numerickým výpočtům, ale ke kreslení. Bohatou dokumentaci k MetaPostu lze nalézt na internetu, je samozřejmě také součástí instalace (včetně uživatelské příručky). Vzhledem k tomu, že jazyk MetaPostu vychází z jazyka METAFONTu, lze k jeho studiu použít i manuály METAFONTu.

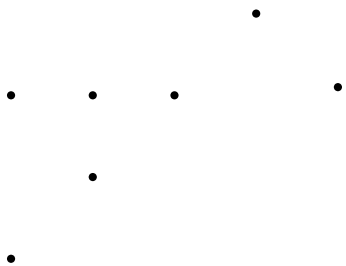
Možnosti MetaPostu ukážeme na příkladech hledání úsečky a kružnice v obraze daném několika body pomocí Houghovy transformace.

Hledání úsečky

Máme K bodů. Z těchto bodů je třeba vybrat ty body, které s jistou přesností leží na nějaké přímce – tvoří úsečku. Rovnici přímky, na níž úsečka leží, uvažujeme ve tvaru $r = x\cos\varphi + y\sin\varphi$, kde parametr r je vzdálenost přímky od počátku souřadnicového systému a φ úhel, který svírá kolmice z počátku souřadnicového systému na přímku s kladným směrem osy x (viz [2]). (Pro kladné φ a záporné r prochází přímka 3. kvadrantem.) Pracujeme tedy se dvěma parametry r a φ, jejich minimální a maximální hodnoty jsou v rozsahu $-90^\circ \le \varphi < 90^\circ$ a $-D \le r \le D$, kde D je vzdálenost mezi protilehlými rohy obrazu.

Rozdělíme prostor (φ, r) na $M \times N$ diskrétních částí, $i = 1, \ldots, M$, $j = 1, \ldots, N$, a hodnoty parametrů φ a r pro tyto části označíme φ_i a r_j. Akumulátorem bude dvourozměrné pole $A[\varphi_i, r_j]$ o velikosti $M \times N$, budeme v něm evidovat počet výskytů dvojic $[\varphi_i, r_j]$ nalezených pro body při daném φ_i. Přesnost, tj. maximální odchylka bodu od nalezené přímky, je dána jemností rozdělení prostoru, tj. počtem buněk M a N určujících rozměry akumulátoru. Přesnost zřejmě ovlivňuje i výsledek hledání přímky, při hrubším dělení by nalezená přímka mohla procházet jinými body, zvláště je-li bodů málo jako v našem příkladu.

Příklad. Máme 7 bodů o souřadnicích [0;0], [0;2], [1;1], [1;2], [2;2], [3;3], [4;2,1] (obr. 1). Nalezněte přímku, na níž leží nejvíce bodů.



Obr. 1

Aplikujeme výše uvedené na případ hledání přímky (viz [2]):

1. Diskretizujeme prostor (φ, r) na (φ_i, r_j), kde $i = 1, \ldots, M$ a $j = 1, \ldots, N$, tj. zvolíme dělení $\Delta\varphi$ a Δr. Vynulujeme akumulátor $A[\varphi_i, r_j]$ pro všechny hodnoty φ_i, r_j, tj. pro $i = 1, \ldots, M$, $j = 1, \ldots, N$.

2. Projdeme celý obraz a pro každý bod $[x_k, y_k]$, $k = 1, 2, \ldots, K$, a pro každou hodnotu φ_i, $i = 1, \ldots, M$, vypočteme $r_j = x_k \cos\varphi_i + y_k \sin\varphi_i$ a inkrementujeme akumulátor na pozici $[\varphi_i, r_j]$, $A[\varphi_i, r_j] := A[\varphi_i, r_j] + 1$.

3. Po projití celého obrazu určuje každá hodnota v akumulátoru $A[\varphi_i, r_j]$ počet bodů ležících na přímce s parametry $[\varphi_i, r_j]$. Hodnoty parametrů pro maximální hodnotu v akumulátoru určují přímku, na které se vyskytuje (s danou tolerancí) největší počet bodů (obr. 2).

Nejen pro uživatele Metapostu může být zajímavý následující kód pro obr. 2b:

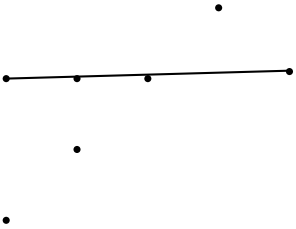
```
beginfig(1);
z1=(0,0); z2=(0,2); z3=(1,1); z4=(1,2); z5=(2,2); z6=(3,3);
z7=(4,2.1); % dané izolované body
path p;
```

```
numeric akumulator[][], pocet, kroku, krokr, D, r, uhel;
numeric maxx, maxy, celkemu, celkemr, maxim, umax, rmax;
% 1. pocet bodů a přesnost dělení akumulátoru
pocet=7; kroku=0.2; krokr=0.048; % konstanty
% nalezení maximálních hodnot souřadnic pro výpočet D
maxx:=0; maxy:=0; % přiřazení
for i=1 upto pocet:
 if maxx<x[i]: maxx:=x[i]; fi;
 if maxy<y[i]: maxy:=y[i]; fi;
endfor;
D=sqrt(maxx*maxx+maxy*maxy);
% zjistíme dimenzi pole akumulator[1..celkemu][1..celkemr]
celkemu:=0; celkemr:=0;
for i=-D step krokr until D: celkemr:=celkemr+1; endfor;
for i=-90 step kroku until (90-kroku):
  celkemu:=celkemu+1;
endfor;
% nulujem akumulátor
for i=1 upto celkemu:
 for j=1 upto celkemr:
   akumulator[i][j]:=0;
 endfor;
endfor;
% 2. pro každý bod 1..pocet a každý úhel vypočteme r
for k=1 upto pocet:
  for i=1 upto celkemu:
  % cyklus ukončíme, až najdeme pozici pro inkrementaci
  konec:=0; % zatím nekonči
  r:=x[k]*cosd ((i-1)*kroku-90)+y[k]*sind ((i-1)*kroku-90);
  % a na příslušné pozici inkrementujeme akumulátor
   for j=1 upto celkemr:
   if r<=j*krokr-D:
     akumulator[i][j]:=akumulator[i][j]+1; konec:=1;
   fi;
   exitif konec=1;  % skonči
   % po inkrementaci jdeme k dalšímu úhlu
   endfor;
  endfor;
endfor;
```

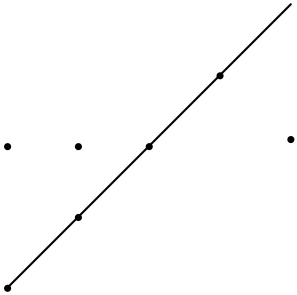
```
% 3. nalezneme maximum v akumulátoru, poznamenáme si pozici
umax:=0; rmax:=0; maxim:=0;
for i=1 upto celkemu:
  for j=1 upto celkemr:
   if akumulator[i][j]>maxim:
      maxim:=akumulator[i][j]; umax:=i; rmax:=j;
   fi;
  endfor;
endfor;
% můžeme si nechat vypsat na obrazovku nalezené hodnoty
show maxim; show umax; show rmax;
% z pozice vypočteme úhel 'uhel' a délku 'r'
uhel:=((umax-1)*kroku-90);  show uhel;
r:=((rmax-1)*krokr-D);  show r;
% vypočteme přímku
z'100=(r*cosd uhel,r*sind uhel);  % jeden bod přímky
% z parametrické rovnice vypočteme krajní body úsečky
x'101=maxx;  %nejpravější bod obrázku pomocí 'x'
z'101=(r*cosd uhel,r*sind uhel)
  +whatever*(-sind uhel,cosd uhel);
x'102=0;  %nejlevější bod obrázku pomocí 'x'
z'102=(r*cosd uhel,r*sind uhel)
  +whatever*(-sind uhel,cosd uhel);
p:=z'101--z'102;  % úsečka p
% kreslicímu peru zmenšíme hrot, zvětšíme celý obrázek
pickup pencircle scaled (0.03);
draw p;  % nakreslíme úsečku a pak i body 1..pocet
pickup pencircle scaled (0.1);
for i=1 upto pocet: drawdot z[i]; endfor;
% vzhledem k plánovanému rozlišení 600 dpi a protože jsme
% pracovali se souřadnicemi pixelů 0..4 aktuální obrázek
%  'currentpicture' vhodně zvětšíme
currentpicture:=currentpicture scaled 5mm;
endfig; % a je to!
```

Na obr. 2a je přímka nalezená s přesností danou krokem dělení Houghova prostoru $\Delta\varphi = 0{,}2^\circ$ a $\Delta r = 0{,}1$ mm, vypočtené hodnoty jsou $\varphi = -88{,}4^\circ$ a $r = -1{,}99$ mm. Na obr. 2b je přímka nalezená s přesností danou krokem dělení $\Delta\varphi = 0{,}2^\circ$ a $\Delta r = 0{,}048$ mm, vypočtené hodnoty

jsou $\varphi = -45{,}0°$ a $r = 0{,}007$ mm. V prvním případě tak byl vzat v úvahu i bod [4;2,1] a vzhledem k způsobu prohledávání akumulátoru má první nalezená přímka (obr. 2a) menší odchylku od $-90°$ než druhá (obr. 2b), i když v akumulátoru je stejné číslo 4.



Obr. 2a

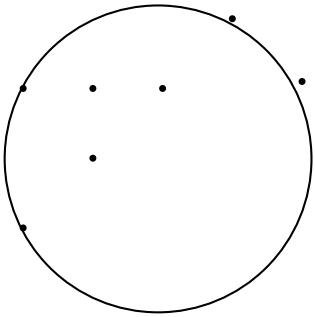


Obr. 2b

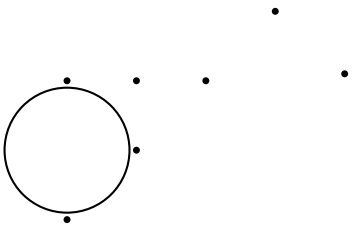
Hledání kružnice

Houghovu transformaci nyní předvedeme na hledání kružnice dané rovnicí ve středovém tvaru $(x - p_x)^2 + (y - p_y)^2 = p_r^2$, kde parametry p_x, p_y, p_r jsou po řadě souřadnice středu kružnice, třetí parametr je její poloměr. Přidržíme se nejčastějšího postupu, kdy volíme p_x, p_y a počítáme p_r. Výpočetní složitost lze redukovat na základě jistého předpokladu o rozsahu parametrů.

Opět použijeme našich 7 bodů. Za předpokladu, že střed kružnice leží uvnitř obrázku s body a že její poloměr není větší než vzdálenost mezi protilehlými rohy obrazu, lze našimi body například proložit kružnici na obr. 3.



Obr. 3a



Obr. 3b

Přesnost jejího nalezení na obr. 3a je dána diskretizací prostoru (p_x, p_y, p_r) pro dělení dané krokem pro evidenci souřadnic středu kružnice $\Delta p_x = 0{,}045$, $\Delta p_y = 0{,}045$ a poloměru $\Delta p_r = 0{,}1$. Na kružnici leží (s danou přesností) 4 body $z1 = [0, 0]$, $z2 = [0, 2]$, $z6 = [3, 3]$, $z7 = [4, 2.1]$.

Obr. 3b ukazuje nalezenou kružnici pro $\Delta p_x = 0{,}05$, $\Delta p_y = 0{,}05$ a $\Delta p_r = 0{,}45$. I na této kružnici leží (s danou přesností) 4 body $z1 = [0, 0]$, $z2 = [0, 2]$, $z3 = [1, 1]$, $z4 = [1, 2]$.

Je zřejmé, že tyto extrémní výsledky našeho algoritmu jsou dány velmi malým počtem bodů. Při velkém počtu bodů, které jsou systematicky rozmístěny, pracuje algoritmus mnohem lépe. Houghova transformace je určena pro vyhledávání křivek v reálných obrazech. Naším cílem však bylo seznámit čtenáře s jejím principem a možnostmi MetaPostu.

Kód pro obr. 3a je ještě jednodušší než v případě hledání úsečky:

```
beginfig(2);
path p;
numeric  pocet, D, akumulator[][][];
numeric celkemsx, celkemsy, celkemr, kroksx, kroksy, krokr,
 maxim, sxmax, symax, rmax, sx, sy, sr;
z1=(0,0); z2=(0,2); z3=(1,1); z4=(1,2); z5=(2,2); z6=(3,3);
z7=(4,2.1);
pocet=7;
kroksx=0.045; kroksy=0.045; krokr=0.1;  %přesnost
maxx:=0; maxy:=0;
for i=1 upto pocet:
 if maxx<x[i]: maxx:=x[i]; fi;
 if maxy<y[i]: maxy:=y[i]; fi;
endfor;
D=sqrt(maxx*maxx+maxy*maxy);  % omezení pro poloměr
% zjistíme dimenzi pole
 % akumulator[1..celkemsx][1..celkemsy][1..celkemr]
celkemsx:=0; celkemsy:=0; celkemr:=0;
for i=0 step kroksx until maxx:
 celkemsx:=celkemsx+1;
endfor;
for i=0 step kroksy until maxy:
 celkemsy:=celkemsy+1;
endfor;
```

```
for i=0 step krokr until D: celkemr:=celkemr+1; endfor;
% nulujem akumulátor
for i=1 upto celkemsx:
  for j=1 upto celkemsy:
   for k=1 upto celkemr:
     akumulator[i][j][k]:=0;
   endfor;
  endfor;
endfor;
%pro každý bod 1..pocet a každý střed vypočteme r
for m=1 upto pocet:
 for i=1 upto celkemsx:
  for j=1 upto celkemsy:
   konec:=0;   % pro ukončení cyklu až najdeme pozici
   r:=sqrt((x[m]-i*kroksx)*(x[m]-i*kroksx)
    +(y[m]-j*kroksy)*(y[m]-j*kroksy));
   %a na příslušné pozici inkrementujeme akumulátor
   for k=1 upto celkemr:
    if r<=k*krokr:
     akumulator[i][j][k]:=akumulator[i][j][k]+1; konec:=1;
    fi;
   exitif konec=1;  % po inkrementaci jdeme dál
   endfor;
  endfor;
 endfor;
endfor;
% nalezneme maximum v akumulátoru a poznamenáme si pozici
sxmax:=0; symax:=0; rmax:=0; maxim:=0;
for i=1 upto celkemsx:
 for j=1 upto celkemsy:
  for k=1 upto celkemr:
   if akumulator[i][j][k]>maxim:
   maxim:=akumulator[i][j][k]; sxmax:=i; symax:=j; rmax:=k;
   fi;
  endfor;
 endfor;
endfor;
show maxim; show sxmax; show symax; show rmax;
sx:=((sxmax-1)*kroksx); sy:=((symax-1)*kroksy);
```

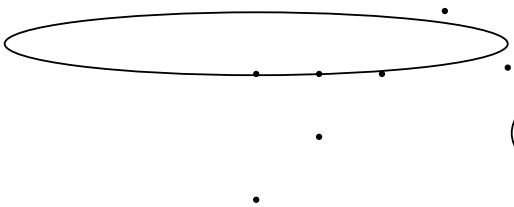
```
r:=((rmax-1)*krokr);
show sx; show sy; show r;
% nakreslíme kružnici
pickup pencircle scaled (0.03);
p:=fullcircle scaled 2r shifted (sx,sy);  draw p;
% nakreslíme body
pickup pencircle scaled (0.1);
for i=1 upto pocet: drawdot z[i]; endfor;
currentpicture:=currentpicture scaled 5mm;
endfig;
```

Závěrem

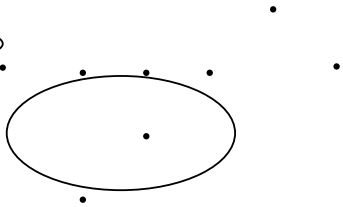
Přestože se za rozumné považuje hledat křivku určenou maximálně třemi parametry, ukážeme ještě Houghovu transformaci na hledání elipsy s osami rovnoběžnými se souřadnicovými osami (obr. 4). Elipsa je určena čtyřmi parametry:

$$\frac{(x-p_x)^2}{p_a^2}+\frac{(y-p_y)^2}{p_b^2}=1$$

Dodatečnými podmínkami je to, že střed elipsy $[p_x, p_y]$ leží uvnitř obrázku s body a že poloosa p_b není větší než výška tohoto obrázku, poloosa p_a není větší než šířka původního obrázku (obr. 4a), resp. jeho polovina (obr. 4b). MetaPost se s touto úlohou také vyrovná.



Obr. 4a



Obr. 4b

Obr. 4a ukazuje nalezenou elipsu při diskretizaci prostoru parametrů (p_x, p_y, p_a, p_b) dané krokem $\Delta p_x = 0{,}08$, $\Delta p_y = 0{,}08$, $\Delta p_a = 0{,}5$, $\Delta p_b = 0{,}5$. Na elipse leží (s danou přesností) 5 bodů.

Obr. 4b ukazuje nalezenou elipsu při diskretizaci (p_x, p_y, p_a, p_b) dané krokem $\Delta p_x = 0{,}05$, $\Delta p_y = 0{,}05$, $\Delta p_a = 0{,}3$, $\Delta p_b = 0{,}3$. Na elipse leží (s danou přesností) 4 body.

Kód je analogický jako v případě hledání kružnice. Uvedeme jen část, naplňující akumulátor. Do proměnné `P` uložíme podíl $(y-p_y)^2/p_b^2$, testujeme, zda nacházíme elipsu

$$\frac{(x-p_x)^2}{p_a^2}+\frac{(y-p_y)^2}{p_b^2}=1 \Longrightarrow P=1-\frac{(x-p_x)^2}{p_a^2}=\frac{(y-p_y)^2}{p_b^2}>0$$

a v kladném případě vypočteme p_b. Elipsu

```
p:=fullcircle xscaled 2a yscaled 2b shifted (sx,sy);
```

nakreslí příkaz `draw p;`

```
% pro každý bod 1..pocet, každý střed (sx, sy) a každé a
% vypočteme b
for m=1 upto pocet:
 for i=1 upto celkemsx:
  for j=1 upto celkemsy:
   for k=1 upto celkema:
    konec:=0;
    P:=1-(((x[m]-i*kroksx)/(k*kroka))
      *((x[m]-i*kroksx)/(k*kroka)));
    if P>0: b:=abs(y[m]-j*kroksy)/ sqrt(P);
     for l=1 upto celkemb:
     if b<=l*krokb:
      akumulator[i][j][k][l]:=akumulator[i][j][k][l]+1;
      konec:=1;
     fi;
     exitif konec=1;
     endfor;
    fi;
   endfor;
  endfor;
 endfor;
endfor;
```

Literatura

[1] Dobeš, M.: *Zpracování obrazu a algoritmy v C#.* BEN – technická literatura, Praha, 2008.

[2] Dobeš, M., Závodný, M.: Vyhledávání v obraze a Houghova transformace. *Rozhledy matematicko-fyzikální* **85**, č. 1, (2010), str. 17–22.