Robotika: Robot Bosch písařem

Ondřej Čermák, Pavel Jisl

9. ledna 2002

Kapitola 1

Robot Bosch písařem

1.1 Zadání úlohy

Úkolem úlohy je naučit robota BOSCH psát. Uživatel pak nastaví velikost, případně orientaci fontu, délku řádky nebo pracovní plochu a začne psát. Robot bude psát předepsaná písmena.

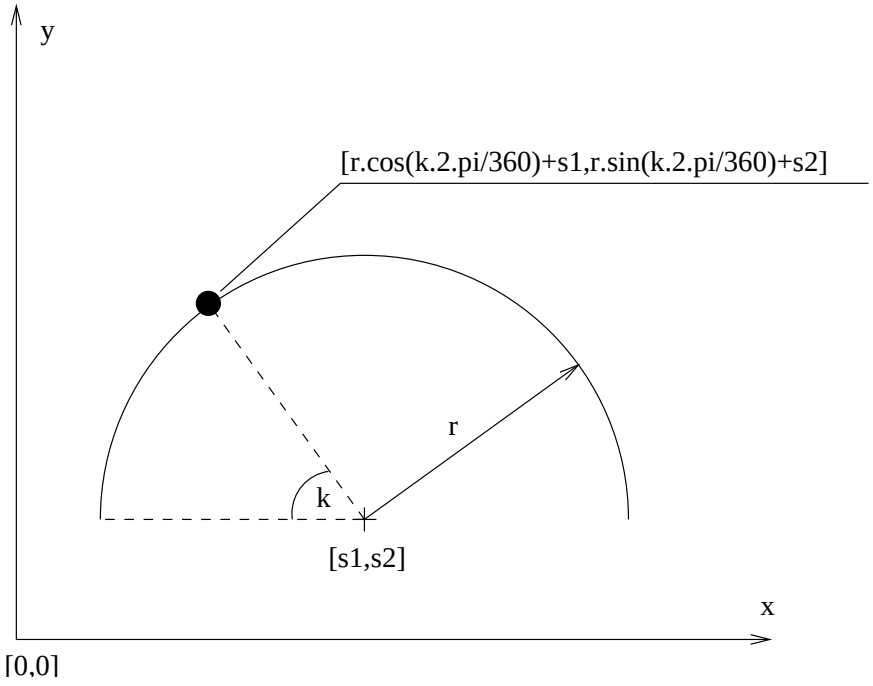
1.2 Řešení problému primitiv

Po prozkoumání abecedy jsme došli k závěru, že všechna písmena lze složit z primitiv úsečka a oblouk. Vzhledem k tomu, že problém úsečky jsme již řešili v předchozí úloze (Řešní IKU pro robota Bosch), zaměřili jsme se na kreslení oblouku.

Oblouk (v podstatě výsek kružnice) lze jednoduše počítat pomocí goniometrických funkcí. Z toho naše řešení vycházelo. Vytvořili jsme funkci `oblouk(robot, r, s, f, smer)`. Parametry jsou samozřejmě identifikace používaného robota (`robot`), dále poloměr oblouku (`r`), souřadnice středu (`s=[x,y]`), velikost počátečního a koncového úhlu (`f=[u1,u2]`) a směr kreslení oblouku.

Po zkontrolování polohy bodů, zda vyhovují IKU, se již propočítávají vlastní body oblouku a ihned se pohybuje ramenem robota.

Pro zlepšení výstupu se bere v potaz také poloměr oblouku a pomocí něj se koriguje krok, po kterém se jednotlivé body oblouku počítají.



1.3 Řešení problému psaní písem

Pro psaní písmen jsme vytvořili funkci `napis(robot,slovo,m,r,fi,e,delka,pocetr)`. Její parametry jsou:

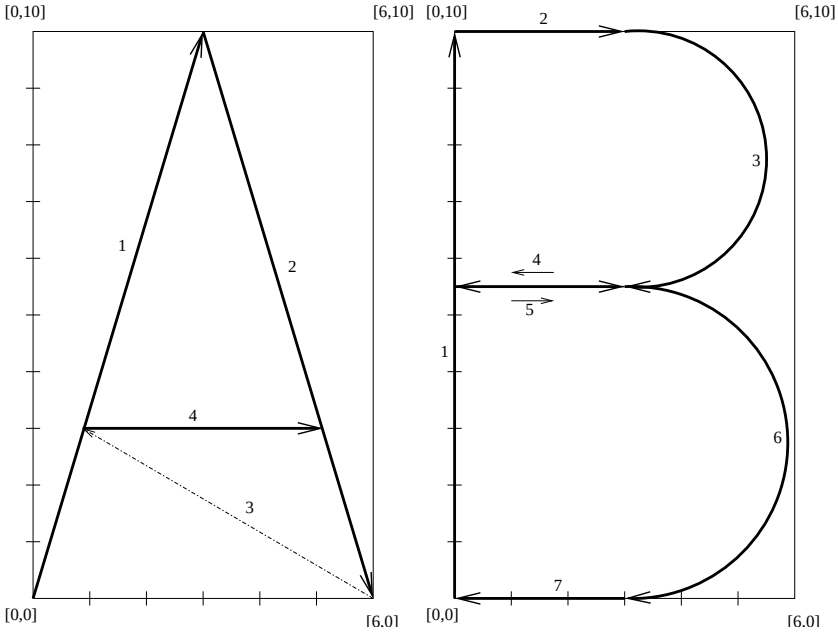
- `robot` - určení robota
- `slovo` - slovo, které má robot ve svém pracovním prostoru napsat
- `m` - matice počátku plochy, kde se má psát (např. `[-100,250]`)
- `r` - velikost písmen
- `fi` - úhel natočení písmen
- `e` - velikost mezery mezi jednotlivými znaky
- `delka` - délka řádky
- `pocetr` - počet řádek

Tato funkce nejdříve otestuje parametry, zda je možné v určené ploše text zapsat, zda písmeno, část písmene nebo textu neleží mimo pracovní oblast robota a pak podle znaku provádí operace kreslení. Pro kreslení znaku používáme následující funkce:

- `presunup(robot,bodB)` - přesun ramena do zadaného bodu se zvednutým pisátkem
- `nahoru(robot)` - zvednutí pisátka
- `primka2(robot,bodA,bodB)`- funkce pro nakreslení úsečky mezi dvěma zadanými body
- `oblouk(robot,r,s,f,smer)` - funkce pro nakreslení oblouku se zadaným středem, poloměrem a mezními úhly

Písmena jsou vytvořena v rastru 10x6. Hodnoty byly zvoleny z potřeby změny velikosti písmen. Pomocí předchozích funkcí jsme vytvořili konstrukci `switch`, ve které vykreslujeme jednotlivá písmena.

Pro větší názornost jsme postup při kreslení znaku znázornili na dvou písmenech, písmenech A a B. Zde jsou vidět v praxi všechny funkce pro kreslení.



1.3.1 Písmeno A

Písmeno A je jeden z nejjednodušších případů (samozřejmě kromě triviálních písmen I a L), kde lze vidět kreslení úseček a přesunu pisátka robota do nové polohy bez kreslení. V jednoduchosti lze zapsat část kódu v Matlabu takto:

```
presunup(robot,transform(n,fi)+m);
primka2(robot,transform(n,fi)+m,transform(r*[3,10]+n,fi)+m);            (1)
primka2(robot,transform(r*[3,10]+n,fi)+m,transform(r*[6,0]+n,fi)+m);    (2)
nahoru(robot);
presunup(robot,transform(r*[0.9,3]+n,fi)+m);                            (3)
primka2(robot,transform(r*[0.9,3]+n,fi)+m,transform(r*[5.1,3]+n,fi)+m); (4)
nahoru(robot);
```

1.3.2 Písmeno B

Na písmenu B lze zase vidět, jak se vytvoří písmeno z úseček a oblouků a to bez zvednutí pisátka. Následující kód ukazuje ve zjednodušené podobě postup kreslení.

```
presunup(robot,transform(r*[0,0]+n,fi)+m);
primka2(robot,transform(r*[0,0]+n,fi)+m,transform(r*[0,10]+n,fi)+m);    (1)
primka2(robot,transform(r*[0,10]+n,fi)+m,transform(r*[3,10]+n,fi)+m);   (2)
oblouk(robot,r*2.25,transform(r*[3,7.75]+n,fi)+m,[90,-90]+fi,-1);       (3)
primka2(robot,transform(r*[3,5.5]+n,fi)+m,transform(r*[0,5.5]+n,fi)+m); (4)
primka2(robot,transform(r*[0,5.5]+n,fi)+m,transform(r*[3,5.5]+n,fi)+m); (5)
oblouk(robot,r*2.75,transform(r*[3,2.75]+n,fi)+m,[90,-90]+fi,-1);       (6)
primka2(robot,transform(r*[3,0]+n,fi)+m,transform(n,fi)+m);             (7)
nahoru(robot);
```

1.4 Zhodnocení

Pokusili jsme se naučit psát robota Bosch. Tato úloha spočívala ve vyřešení inversní kinematické úlohy robota Bosch a poté vytvoření funkcí pro kreslení úsečky a oblouku. Tyto funkce byly vytvořeny a úspěšně otestovány na reálném robotu.

Úloha pro nakreslení znaků spočívala hlavně ve zvektorizování jednotlivých písmen, odečtení rozměrů a použitých primitiv pro kreslení a napsání příslušných částí kódu. Vzhledem k nedostatku času však nebylo možno zvektorizovat všechna písmena. Přednostně byla vytvořena písmena, jejichž tvar je názorný a lze na nich vidět postup tvoření písmen pomocí jednotlivých primitiv.

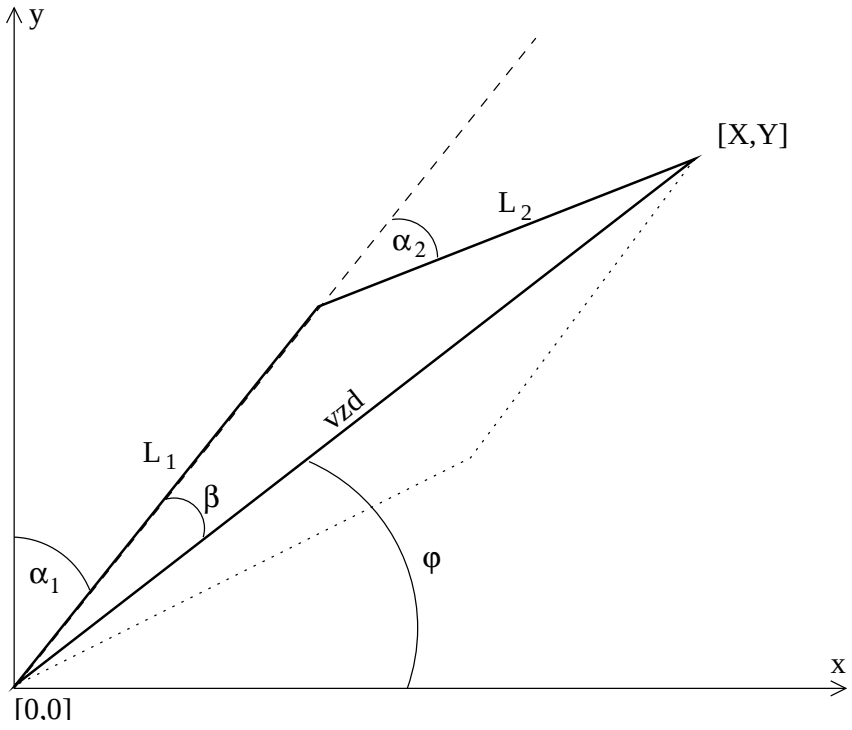
Kapitola 2

Přílohy

2.1 Inverzní kinematická úloha

2.1.1 Vyřešení inverzní kynematické úlohy

Nákres souřadnic, pro výpočet potřebných úhlů a vzdáleností:



Z tohoto schématu pak dostáváme vzorec pro výpočet vzdálenosti

$$vzd^2 = L_1^2 + L_2^2 + 2L_1L_2cos\alpha_2,$$

z čehož po jednoduché úpravě dostáváme

$$\cos\alpha_2 = \frac{vzd^2 - L_1^2 - L_2^2}{2L_1L_2}$$

a následně i výpočet vlastního úhlu α_2

$$\alpha_2 = \pm \arccos\left(\frac{vzd^2 - L_1^2 - L_2^2}{2L_1L_2}\right).$$

Pro výpočet úhlu β vycházíme z rovnice

$$L_2^2 = vzd^2 + L_1^2 - 2.vzd.L_1 cos\beta,$$

kde po úpravě dostáváme výraz pro výpočet úhlu β

$$\beta = \arccos\left(\frac{vzd^2 + L_1^2 - L_2^2}{2.vzd.L_1}\right).$$

Úhly φ a α_1 pak vypočteme pomocí vztahů

$$\varphi = \arctan\left(\frac{Y}{X}\right)$$

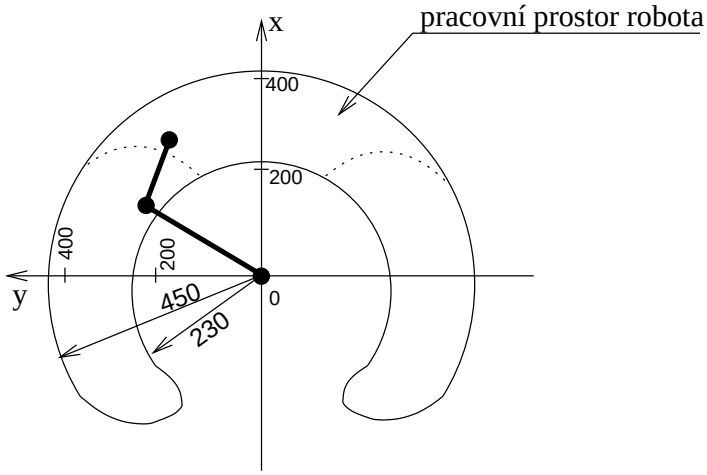
a

$$\alpha_1 = 90^o - \varphi \mp \beta.$$

Tento postup platí pro pravou polorovinu souřadné soustavy. Pro levou polorovinu je výpočet shodný (je s pravou polorovinou symetrická podle osy) a proto stačí předchozí výsledky hodnotou -1.

Pro výpočet těchto hodnot je použita funkce, napsaná v programovacím jazyku programu Matlab. Jeho výpis je v příloze č. 1.

2.1.2 Řešení pracovního prostoru



Z nákresu je vidět, že je třeba vyřešit úlohu pro dvě různé IKU. Pro to slouží funkce `mojeikt`, která počítá jednotlivé úhly natočení ramen robota. Pokud je zadaný bod `[X,Y]` záporný, řešíme druhou IKU. Ta však spočívá pouze v násobení -1 a změně pořadí hodnot úhlů v matici úhlů.

Pro přesun po přímce slouží funkce `primka`. Zde jsou také ošetřeny veškeré chybové stavy.