REFERAT
PROIECT GC

DATE:07.04.2023

STUDENT'S NAME:
MARIA ANDREI

Contents

1 Etapa de design a obiectului grafic

1.1 Sursa imaginii si implementarea in MATLAB

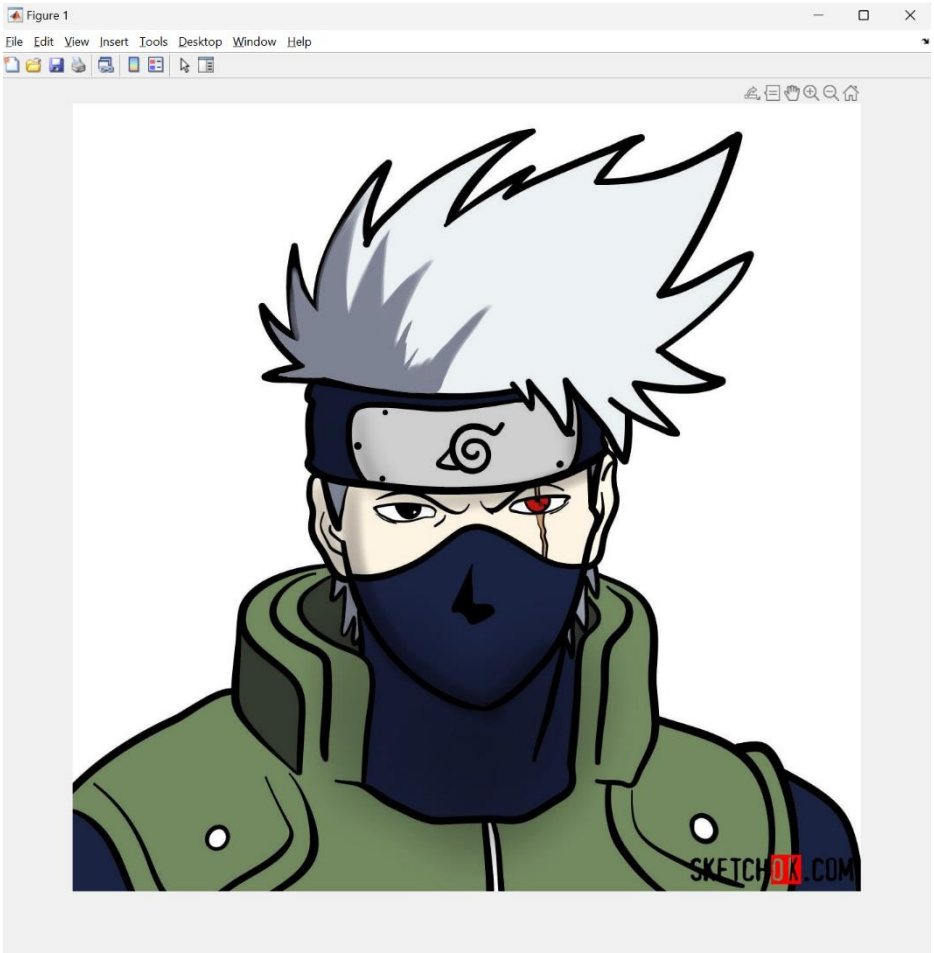
Imaginea este preluata de pe internet si implementata intr-o fereastra grafica a MATLAB-ului (Figure) pentru a usura determinarea punctelor de control.

Sursa imagine: ***Apasati pentru a vizualiza***

Pentru a putea importa imaginea intr-o fereastra grafica din MATLAB este necesar mai intai ca imaginea sa fie salvata in acelasi director. In continuare, utilizam urmatoarele comenzi:

```
1   %clear, clc
2   figure(1)
3   img = imread('kakashi_bezier2.png');
4   imshow(img);
```

Astfel, reusim sa incadram imaginea intr-o fereastra de tip *figure:*



Observatii:

Este important sa notam faptul ca dimensiunea ferestrei grafice coincide cu dimensiunea imaginii importate. Spre exemplu, pentru imaginea folosita dimensiunea este de *1200 x 1200* pixeli. De aici rezulta ca si fereastra *figure* va avea dimensiunea de *1200 x 1200*.

Un alt lucru extrem de important este faptul ca punctul de coordonate (0, 0) este pozitionat in ***coltul din stanga-sus.*** Acest lucru are mare importanta in prelucrarea mai departe a coordonatelor finale.

1.2 Determinarea punctelor de control si a curbelor de tip spline

Pentru curbele obisnuite de grad n, acestea au fost determinate prin incercare introducereii unor puncte de control cu ***ginput()*** care se estimeze curba corespondenta desenului.

Spre exemplu , sa luam o curba din parul personajului:

Punctele au fost introduse estimativ cu ***ginput(),*** iar apoi curba a fost trasata cu ajutorul unei *functii de calcul a unei curbe Bezier*.
(Vezi: reprezentare_bezier_prototype.m)

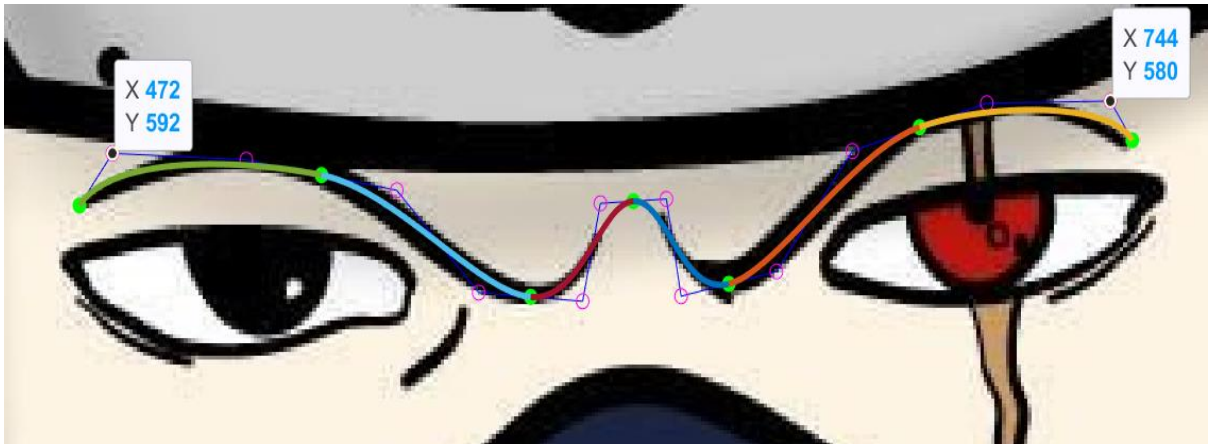
De asemenea, coordonatele obtinute sunt salvate in variabile corepunzatoare pentru a putea fi reprezentate intr-o figura separata.

Pentru cazul curbelor spline de continuitate G1 acestea au fost realizate cu ajutorul algoritmului ***F-Mill.***

(Vezi: FMill.m si reprezentare_Fmill.m)

Datele de intrare au fost de asemenea introduse estimativ avand in vedere modul de prelucrare al algoritmului. Astfel, ca si date de intrare au fost selectate punctele de inflexiune de pe curba precum si al doilea si penultimul punct de control al curbei.

Spre exemplu, sprancenele personajului au fost trasate astfel:

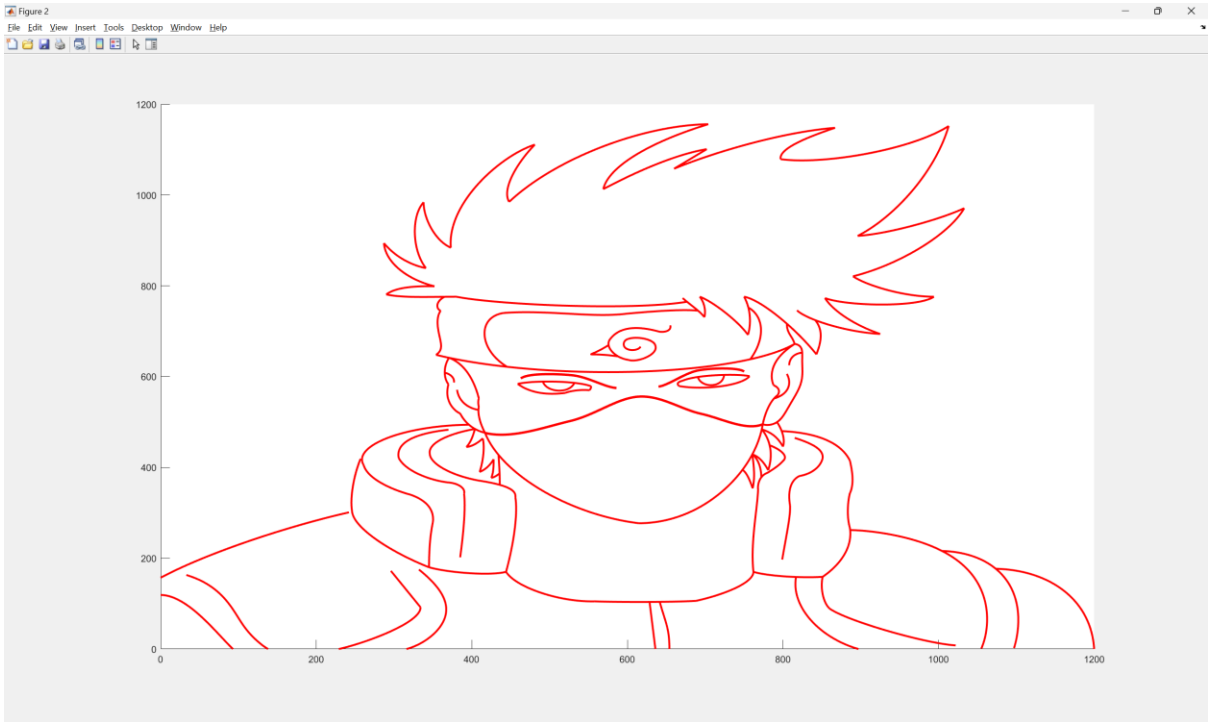


Punctele verzi sunt punctele de inflexiune de pe curba iar cele ale caror coordonate sunt vizibile sunt al doilea, respectiv penultimul punct de control. Astfel, cruba trasata este cea aproximata prin F-Mill.

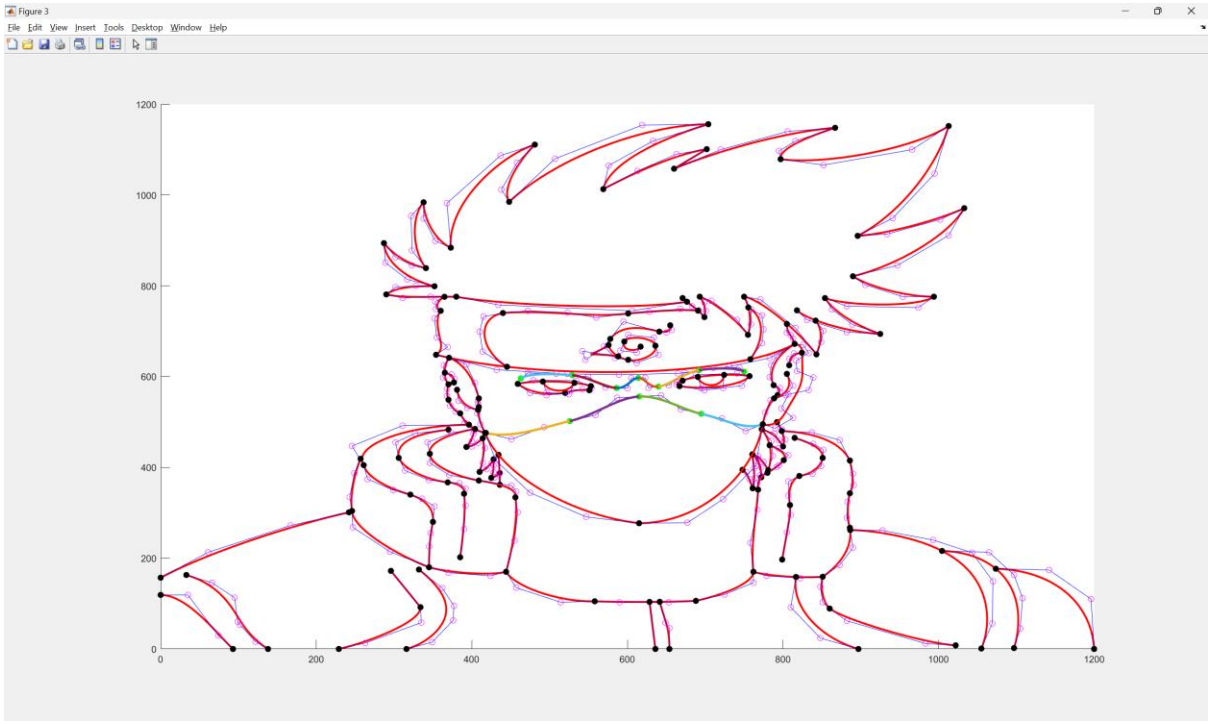
1.3 Multimea curbelor ce alcatuiesc figura si tipul acestora

*Desenul final este format din aproximativ **130** de curbe de mai multe tipuri. Majoritatea curbelor sunt curbe Bezier de grad **3**, insa sunt folosite si curbe Bezier de grad **2**, **4**, **5**, **7** sau chiar si **12**. De asemenea, sunt prezente si multe curbe de tip Spline atat de continuitate G0, cat si G1. Cele de continuitate G1 sunt doar in numar de doua si sunt colorate diferit pentru a putea fi observate pe figura.*

1.3.1 Desenul final fara puncte de control



1.3.2 Desenul final cu puncte de control



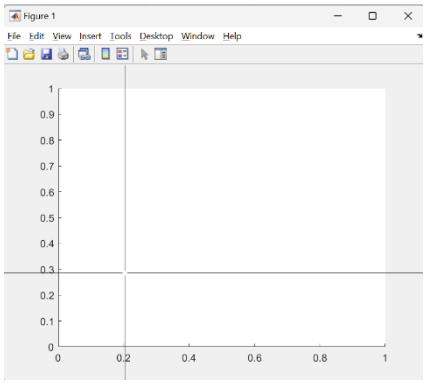
2 Rezolvarea proiectului

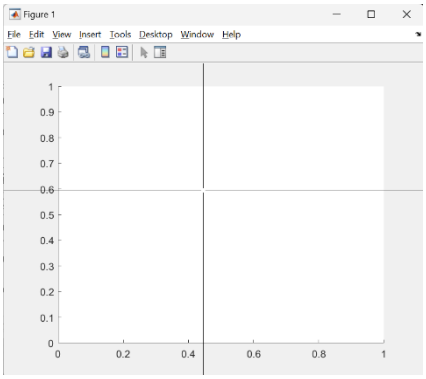
2.1 Date de intrare

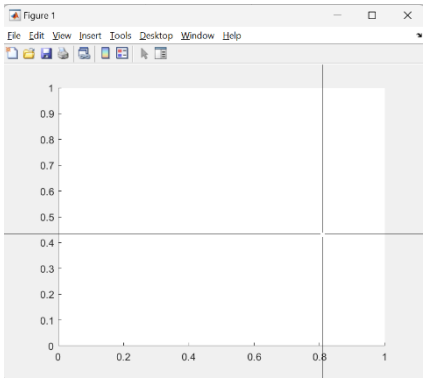
Ca date de intrare este necesar sa cunoastem toate punctele de control corespunzatoare curbelor Bezier din desen sau punctele de inflexiune impreuna cu al doilea si penultimul punct de control in cazul curbelor spline realizate cu F-Mill.

Modalitatea de introducere a acestor puncte este prin utilizarea comenzii ***ginput(n)*** care permite preluarea coordonatele unor puncte aflate la pozitia unde se executa un click cu mouse-ul. Aceste puncte sunt apoi afisate intr-o matrice de atatea linii cate puncte au fost preluate si doua coloane. Prin transpunerea acestei matrici obtinem o matrice de 2 linii, unde pe prima linie se afla coordonatele abscisei, iar pe a doua coordonatele ordonatei punctelor.

Exemplu: b = ginput(3)








```
>> b = b'

b =

    0.2035    0.4462    0.8518
    0.2831    0.5944    0.4484
```

!!! Obtinem coordonatele punctelor unde au fost executate click-urile in fereastra grafica

2.2 Date de iesire

Ca date de iesire se vor obtine coordonatele punctelor Bezier ale tuturor curbelor din desen. Aceste coordonate sunt salvate in matrici corespunzatoare punctelor de control asociate si sunt apoi utilizate pentru a putea afisa grafic aceste puncte.

2.3 Algoritmii utilizati

- Precum am mentionat, pentru fiecare curba, se porneste de la o matrice cu puncte de control care definesc curba

- Cu ajutorul polinoamelor lui Bernstein se determina reteaua de puncte de pe curba pe care dorim sa o reprezentam

Intregul program este rezolvat in mod modular, apeland la o varietate de functii ce indeplinesc cate o functie importanta in rezolvare. Fisierele de tip M-file principale ce realizeaza reprezentarea finala sunt *Drawing.m* si *Drawing_with_control_points.m*. Aceste fisiere apeleaza urmatoarele functii relevante:

1. ***Bernstein(t, n)*** – care returneaza polinoamele Bernstein de grad n pentru parametrul t care genereaza un spatiu liniar de 100 de puncte din intervalul [0, 1]

2. ***combinari(n, k)*** – returneaza rezultatul combinariilor de *n luate cate k* -> $\binom{n}{k}$

3. ***reprezentare_bezier(b, n, s)*** – returneaza punctele Bezier ale unei curbe de grad `$n$` cu punctele de control din `$b$` si o reprezinta cu sau fara poligonul de control al acesteia in functie de parametrul `$s$`. Daca *s = 'pc'* atunci se reprezinta si poligonul de control. Daca `$s$` este orice altceva atunci nu se va reprezenta poligonul de control

4. ***Fmill(p,b1, bn)*** – returneaza punctele de control calculate ale unei curbe spline de continuitate G1 pornind de la punctele de inflexiune (*p*) si al doilea (*b1*) si penultimul (*bn*) punct de control al curbei

5. ***reprezentare_fmill****(p, b1, bn, s)* – reprezinta grafic curba specifica punctelor de control realizate cu functia *Fmill*. De

asemenea, in functie de parametrul `*s*` aflam daca vrem sa reprezentam functia cu poligonul de control sau fara acesta. In plus, `s` controleaza modul in care se deseneaza curba in functie de cele doua curbe spline care se afla in desenul final. Daca *s = 'e' (eyebrows)* stim ca vrem sa reprezentam sprancenele din desen motiv pentru care nu afisam a 3-a si a 4-a sectiune din curba pentru a nu desena sprancenele lipite. Daca *s = 'm' (mask)* reprezentam curba complet dar fara poligonul de control, iar daca *s* este orice altceva reprezentam intreaga curba impreuna cu punctele de control.

6. ***Casteljau(tp, n, bo)*** – Calculeaza punctul de pe o curba de grad ***n*** cu poligonul de control dat prin punctele din ***bo*** la *pozitia* ***tp*** de pe curba. De asemenea, returneaza o matrice ***b*** ce contine toate coordonatele poligoanelor intermediare obtinute prin metoda lui Casteljau

7. ***reprezentare_casteljau(b, n)*** – reprezinta grafic poligoanele determinate din functia *Casteljau(tp, n, bo) cu culori diferite si marcheaza punctele din aceste poligoane precum si punctul final cautat ce se afla pe curba de grad n cu care lucram*

8. ***conversie_coordonate(bi)*** – Precum am mentionat anterior, in fereastra grafica in care s-a importat imaginea sursa si de unde se obtin coordonatele punctelor de control (datele de intrare), punctul de coordonata (0, 0) este pozitionat in coltul din stanga sus iar dimensiunea acestei imaginii este de *1200 x 1200*. Asadar, daca mutam punctul (0, 0) in stanga jos atunci abscisa ramane aceasi dar ordonata devine ***1200 – valoarea_initiala.*** Acesta este rolul acestei functii, de a face conversia coordonatelor pentru un poligon de control

9. ***afin_invariatie(bi, coef)*** – afiseaza modul in care o curba bezier ***(bi)*** se modifica atunci cand doua puncte de control consecutive sunt modificate cu o anumita valoare ***(coef)***

2.4 Computation

1. *Obtinerea polinoamelor Bernstein de grad n – functia* ***Bernstein(t, n)****:*

- *polinoamele lui Bernstein au fost calculate prin formula* $B_k = \binom{n}{k} \cdot (1-t)^{n-k} \cdot t^k$, *unde* $k = 0, n$ *si* $t \in [0,1]$

- *pentru a putea salva polinoamele* $B_0, B_1, B_2, \dots, B_n$ *in variabile cu nume corespunzator s-a folosit functia* ***eval***

- *S-a creat variabila B de n+1 linii si 100 de coloane (deoarece t = linspace(0,1, 100) in care fiecare linie este corespondentul unui polinom Bernstein.*

  Spre exemplu: Pentru polinoamele Bernstein de grad 3, B ar fi de forma: B = [B0; B1; B2; B3]

```
function [B] = Bernstein(t, n)
B = [];
for k = 0:n
    eval(['B' num2str(k) ' = combinari(n, ' num2str(k) ') .* (1 - t) .^ (n
- ' num2str(k) ') .* t .^ ' num2str(k) ';']);
    B = [B; eval(['B' num2str(k)])];
end
end
```

2. *Reprezentarea curbei bezier cu punctele de control si polinoamele lui Bernstein – functia* ***reprezentare_bezier(b, n, s)*** *:*

- *Prin inmultirea punctelor de control si a polinoamelor Bernstein de grad n putem obtine multimea de puncte ce alcatuiesc curba poligonului nostru de control:* ***C = b * B***

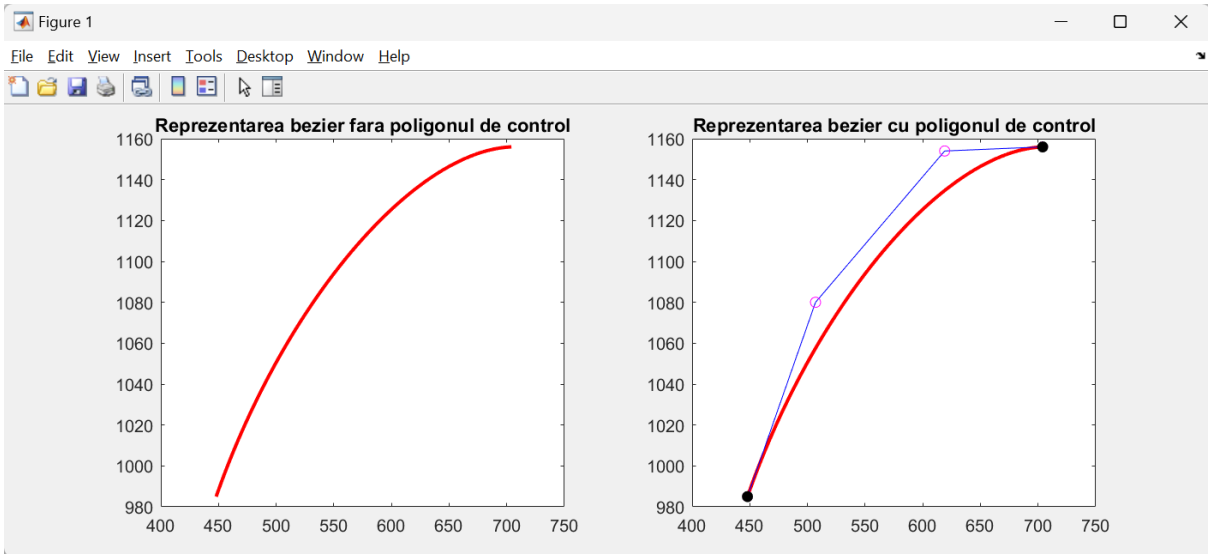
- *Cu aceasta functie il determinam pe C si reprezentam grafic curba*

- *In functie de variabila* ***'s'*** *stabilim si daca dorim ca reprezentarea sa includa trasarea poligonului de control si evidentierea punctelor de control. Daca* ***s = 'pc'*** *atunci le ilustram si pe cele mentionate anterior, iar in caz contrar decat curba. Punctele poligonului de control sunt realizate cu cerculete goale iar punctele de pe curba ce au proprietatea de interpolare Lagrange sunt ilustrate cu cerculete pline*

```
function [C] = reprezentare_bezier(b, n, s)
t = linspace(0, 1);
B = Bernstein(t, n);
C = b * B;
plot(C(1, :), C(2, :), 'r-', 'LineWidth', 2)

if(s == 'pc')
    hold on
    plot(b(1, :), b(2, :), 'b-o', 'MarkerEdgeColor', 'magenta')
    plot(C(1, 1), C(2, 1), 'ko', 'MarkerFaceColor','black')
    plot(C(1, end), C(2, end), 'ko', 'MarkerFaceColor','black')
end

end
```

Exemplu:

```
b = [448 507 619 704; 215 120 46 44];
b = conversie_coordonate(b);
subplot(121)
reprezentare_bezier(b, length(b) - 1, '-'); title('Reprezentarea bezier fara
poligonul de control')
subplot(122)
reprezentare_bezier(b, length(b) - 1, 'pc'); title('Reprezentarea bezier cu
poligonul de control')
```



3. *Aproximarea unei curbe spline de continuitate G1 prin interpolarea F-Mill: functia* ***FMill(p, b1, bn)*** *:*

- *Pentru a satisface datele necesare determinarii unei curbe spline G1 cu n portiuni de curba este necesar sa cunoastem n+1 puncte din curba finala. Asadar, prin intermediul lui p, introducem aceste n+1 puncte care sunt corespunzatoare punctelor de inceput si final ale unei portiuni de curba. De asemenea, mai este necesar sa cunoastem al doilea punct de control al primei portiuni, precum si penultimul punct de control al ultimei portiuni pentru a determina forma curbei de inceput si final*

- *Cu ajutorul acestor cunostiinte putem sa determinam restul punctelor de control, intrucat cunoastem faptul ca aceasta curba repsecta continuitatea de tip G1, ceea ce inseamna ca penultimul si ultimul punct al unei portiuni de curba trebuie sa formeze un segment de dreapta cu cel de al doilea punct de control al urmatoarei portiuni (mai exact: punctele respective sa fie coliniare).*

```
function [b] = FMill(p, b1, bn)
n = max(size(p));
l = zeros(2, n - 2);
a = b1(1, 1); e = b1(2, 1);
c = bn(1, 1); d = bn(2, 1);
for j = 1:(n-2)
    l(:, j) = p(:, j+2) - p(:, j);
end
ultim = 3*(n-1) + 1;
b = ones(2, ultim);
b(:, 1) = p(:, 1);
b(:, 2) = [a; e];
b(:, 3) = p(:, 2) - (1/6) * l(:, 1);
for k = 1 : (n - 3)
    b(:, 3*k+1) = p(:, k+1);
    b(:, 3*k+2) = p(:, k+1) + (1/6) * l(:, k);
    b(:, 3*k+3) = p(:, k+2) - (1/6) * l(:, k+1);
end
b(:, 3*(n-2)+1) = p(:, n-1);
b(:, 3*(n-2)+2) = p(:, n-1) + (1/6) * l(:, n-2);
b(:, 3*(n-2)+3) = [c, d];
b(:, ultim) = p(:, n);
end
```

4. *Reprezentarea curbei cu ajutorul punctelor de control calculate cu algoritmul Fmill() - functia* ***reprezentare_Fmill(p, b1, bn, s)*** *:*

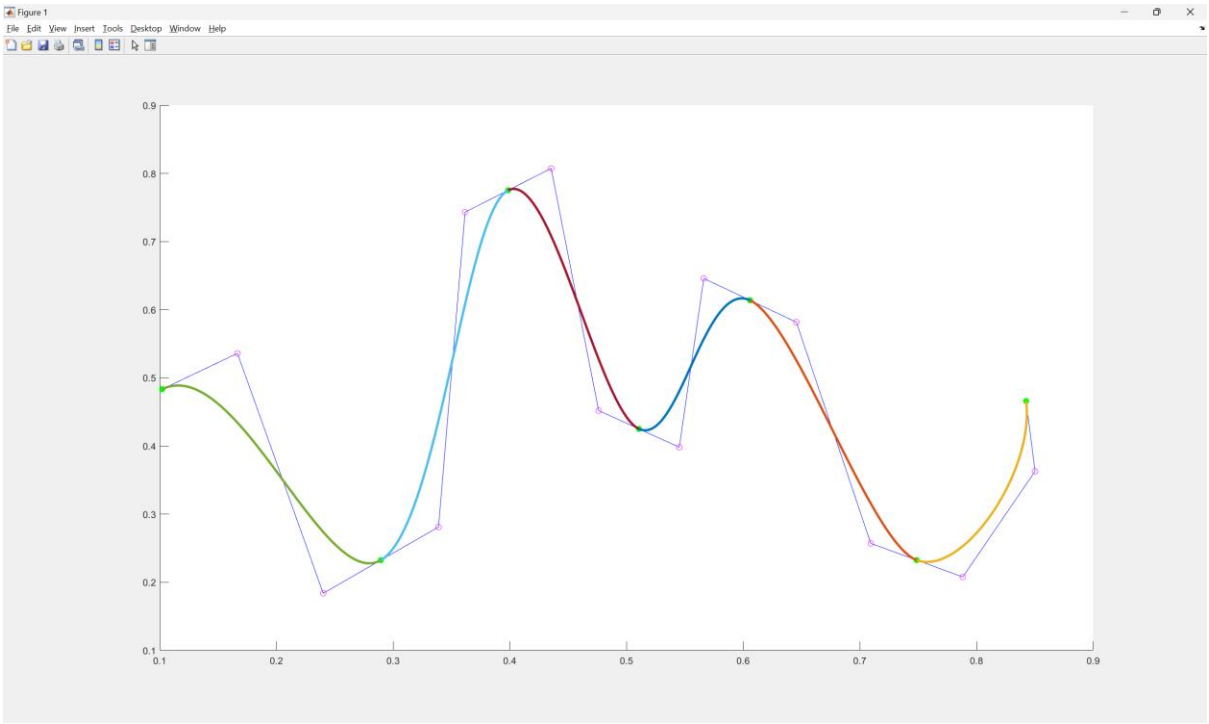
- *reprezentam fiecare portiune de curba si punctele acesteia de control pe care le-am calculat*

- *In aceast functie, parametrul* ***'s'*** *ne spune mai multe despre cum am vrea sa desenam curba si poligoanele acesteia. Daca s = 'e' (eyebrows) stim ca vrem sa reprezentam sprancenele din desen motiv pentru care nu afisam a 3-a si a 4-a sectiune din curba pentru a nu desena sprancenele lipite. Daca s = 'm' (mask) reprezentam curba complet dar fara poligonul de control, iar daca s este orice altceva reprezentam intreaga curba impreuna cu punctele de control.*

```
function reprezentare_Fmill(p, b1, bn, s)
b = FMill(p, b1, bn);
n = max(size(p));
ultim = 3*(n-1) + 1;
i = 1;
nr = 1;
ng = n-1;

if (s ~= 'e' && s~= 'm')
while nr <= ng
    hold on
    plot(b(1, i:i+3), b(2, i:i+3), 'bo-', 'MarkerEdgeColor','magenta');
    plot(b(1, i), b(2, i), 'go', 'MarkerFaceColor','green')
    plot(b(1, i+3), b(2, i+3), 'go', 'MarkerFaceColor','green')
    nr = nr + 1;
    i = i + 3;
end
end
t = linspace(0, 1);
B = Bernstein(t, 3);
m = length(t);
x = zeros(2, m);
i = 1;
nr = 1;
while nr <= ng
    x = b(:, i:i+3) * B;
    if(s == 'e')
        if(nr ~= 3 && nr ~= 4)
            plot(x(1, :), x(2, :), 'r-','LineWidth', 2.5);
        end
    elseif(s == 'm')
        plot(x(1, :), x(2, :), 'r-','LineWidth', 2.5);
    else
        plot(x(1, :), x(2, :),'LineWidth', 2.5);
    end
    nr = nr + 1;
    i = i + 3;
    end
end
```

Am dat mai devreme ca si exemplu realizarea sprancenelor din desen cu acest algoritm. Sa incercam acum sa vizualizam acest algoritm cu niste puncte date aleatoriu folosind comanda ***ginput()*** *atat pentru punctele de pe curba cat si pentru cele doua puncte de control ce ne sunt necesare. Vom realiza acest lucru pentru o curba cu 6 portiuni:*

```
p = ginput(7); p = p'
b1 = ginput(1); b1 = b1'
bn = ginput(1); bn = bn'
hold on
reprezentare_Fmill(p, b1, bn, '-');
```



5. *Aflarea unui punct de pe o curba–functia* ***Castlejau(tp, n, bo)*** *:*

- *Daca dorim sa aflam un punct pozitionat la o anumita pozitie pe curba, algoritmul lui Casteljau este extrem de util. Sa spunem ca dorim sa aflam care este punctul din mijloc al unei curbe ale carei puncte de control sunt cunoscute. Ca sa realizam acest lucru unim punctele de mijloc ale dreptelor polignului de control. Dupa aceasta unim la fel mijloacele celor doua noi drepte obtinute si tot asa pana cand reducem totul la un punct. Acel punct este cel pe care il cautam*
-

- *Functia mea primeste ca date de intrare `**tp**` care mentioneaza la ce portiune de curba dorim sa aflam punctul, `**n**` care este gradul curbei si **'bo'** care reprezinta coordonatele polignoului de control al curbei. Trebuie sa aflam coordonatele de control ale urmatoarelor nivele pana ajungem la nivelul final **(nivelul n).** Astfel, functia calculeaza matriciile corespunzatoare coordonatelor fiecarui nivel (b1, b2, ..., bn). In continuare le concateneaza pe toate in matricea b = [b1; b2; ... ; bn]*

```
function [b] = Casteljau(tp, n, b0)
b = [];
for i = 1:n
    eval(['b' num2str(i) ' = zeros(2, n - ' num2str(i) ' + 1);']);
    for j = 1:(n - i + 1)
        eval(['b' num2str(i) '(:, j) = b' num2str(i-1) '(:,j) .* (1-tp) +
b' num2str(i-1) '(:, j + 1) .* tp;']);
    end
end
for i = 1:n
    eval(['b = [b, b' num2str(i) '];' ])
end
```

6. *Reprezentarea dreptelor determinate de Castlejau de pe fiecare nivel – functia **reprezentare_casteljau(b, n)** :*

- *in algoritmul anterior am mentionat ca salvam in matricea b toate coordonatele poligoanelor de pe fiecare nivel din structura sistolica.*

- *ce este de mentionat, este faptul ca fiecare nivel va avea cu un punct mai putin decat cel anterior. Tinem cont de acest lucru atunci cand reprezentam, pentru a parcurge elementele din matrice corespunzator. Daca pe nivelul zero am avut 4 puncte, atunci pe nivelul 1 vom avea 3 puncte. Asa ca incepem cu primele 3 puncte de matricea b si le reprezentam, dupa care urmatoarele 2 si le reprezentam, iar apoi ultimul.*

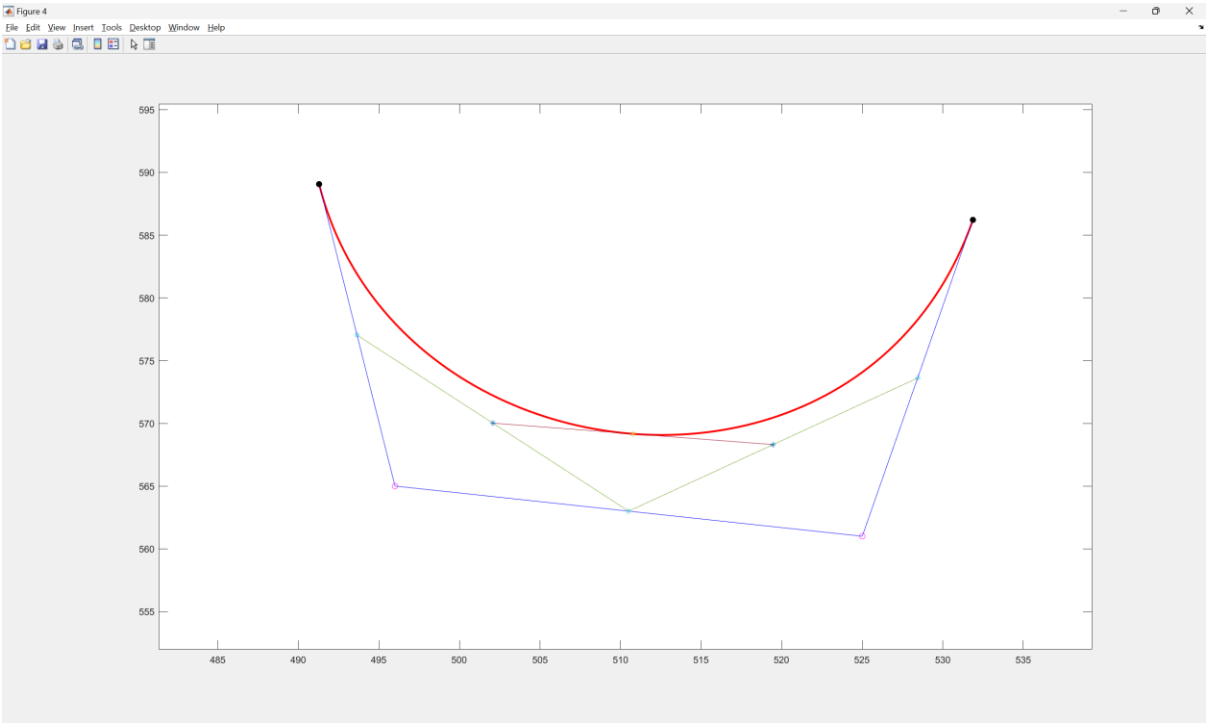
- *Asadar, algoritmul va arata in urmatorul fel: **(Vezi urmatoarea pagina)***

```
function [] = reprezentare_casteljau(b, n)
nr = 1;
for i = 1:n
     eval(['b' num2str(i) '= [];']);
    for j = 1:(n - i + 1)
        eval(['b' num2str(i) '= [b' num2str(i) ', b(:, ' num2str(nr) ')];']);
%bi = [bi, b(:, nr)];
        nr = nr + 1;
    end
    eval(['plot(b' num2str(i) '(1, :), b' num2str(i) '(2, :))']); %plot(bi(1,:),
bi(2,:))
    eval(['plot(b' num2str(i) '(1, :), b' num2str(i) '(2, :), ''*'' )']);
%plot(bi(1,:), bi(2,:), '*')
end
```

Sa testam acest lucru pentru o dreapta din desenul nostru:

Avem fisierul ***Casteljau_aplicat.m*** *care ne exemplifica acest lucru:*

```
%% Casteljau aplicat
clear, clc
figure(4)

e5 = [491.289 496 525 531.867; 610.939 635 639 613.785];
e5 = conversie_coordonate(e5);
reprezentare_bezier(e5, length(e5) - 1, 'pc');

b = Casteljau(1/2, length(e5) - 1, e5);
reprezentare_casteljau(b, length(e5) - 1);

axis([450 650 500 650])
```



7. *Functia de conversie a coordonatelor pentru a transpune punctul (0, 0) din stanga-sus in stanga-jos – functia* ***conversie_coordonate(bi)*** *:*

- *este introdus poligonul de control iar apoi coordonatele sunt convertite folosind formula:* ***1200 – valoarea_initiala***

```
function [bf] = conversie_coordonate(bi)
bi(2, :) = 1200 - bi(2, :);
bf = bi;
end
```

8. *Ilustrarea proprietatii de afin invariatie a unei curbe prin modificare a cate doua puncte de control consecutive din poligon de control – functia* ***afin_invariatie(bi, coef)*** *:*

- *dorim sa observam cum se modifica curba atunci cand sunt modificate doua puncte de control consecutive si cum transformarea poligonului de control afecteaza si transformarea curbei respective*

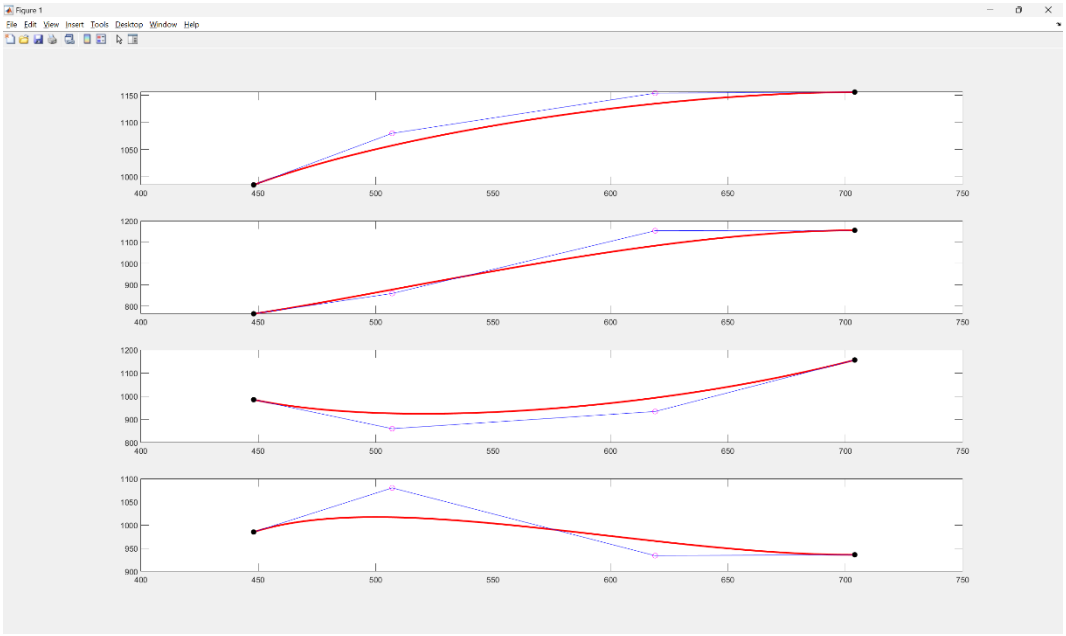
- *exemplificam acest lucru prin modificarea punctelor i si i+1 unde i = 1, n-1 (n – gradul curbei). Prin intermediul unui coeficient ales, crestem sau scadem cursiv aceste doua puncte si observam vizual schimbariile ce au loc*

```
function [] = afin_invariatie(bi, coef)
subplot(4,1,1)
reprezentare_bezier(bi, length(bi) - 1, 'pc');
nr = 2;
hold on
for i = 1:length(bi)-1
    b = bi;
    subplot(4, 1, nr)
    b(2, i) = b(2, i) + coef;
    b(2, i+1) = b(2, i+1) + coef;
    reprezentare_bezier(b, length(b) - 1, 'pc');
    nr = nr + 1;
end
```

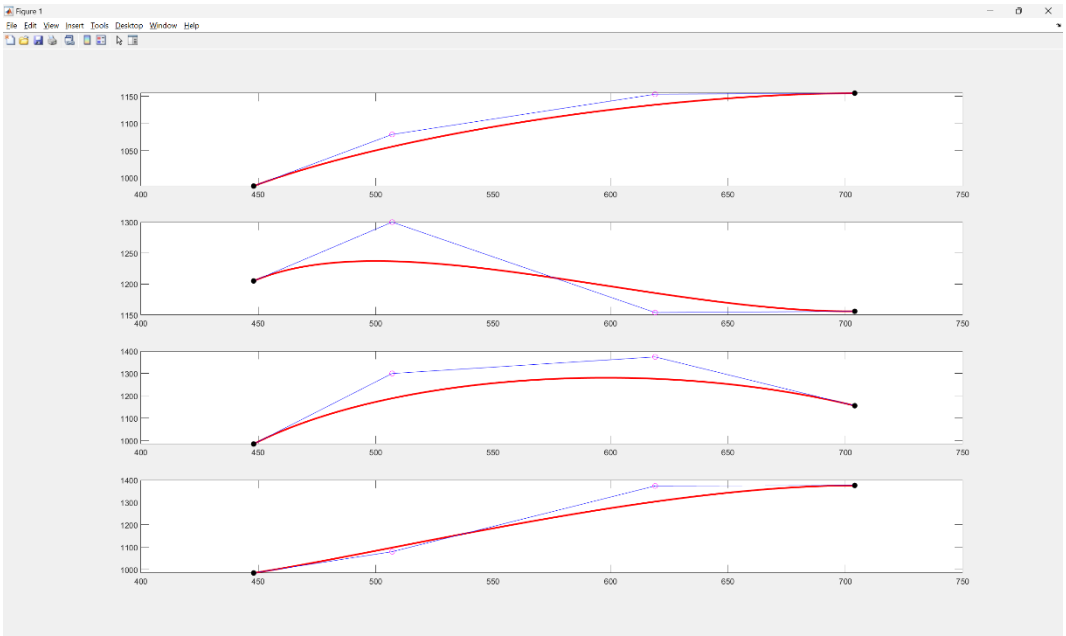
Sa luam ca exemplu prima curba pe care o avem din reprezentarea parului personajului si un coeficient egal cu:

I. ***coef = -220***

```
b1 = [448 507 619 704; 215 120 46 44];
b1 = conversie_coordonate(b1);
afin_invariatie(b1, coef);
```



II. *coef = +220*



9. *Functia de desenare a parului – functia* ***hair(s)*** *:*

```
function [] = hair(s)
b1 = [448 507 619 704; 215 120 46 44];
b1 = conversie_coordonate(b1);
reprezentare_bezier(b1, length(b1) - 1, s);

b2 = [704 633 576 569; 44 81 135 187];
b2 = conversie_coordonate(b2);
reprezentare_bezier(b2, length(b2) - 1, s);

b3 = [569 613 663 702; 187 147 110 99];
b3 = conversie_coordonate(b3);
reprezentare_bezier(b3, length(b3) - 1, s);

b4 = [702 660; 99 142];
b4 = conversie_coordonate(b4);
reprezentare_bezier(b4, length(b4) - 1, s);

b5 = [660 720 806 867; 142 100 60 52];
b5 = conversie_coordonate(b5);
reprezentare_bezier(b5, length(b5) - 1, s);

b6 = [867 816 795 797; 52 81 103 121];
b6 = conversie_coordonate(b6);
reprezentare_bezier(b6, length(b6) - 1, s);

b7 = [797 852 966 1013; 121 134 100 48];
b7 = conversie_coordonate(b7);
reprezentare_bezier(b7, length(b7) - 1, s);

b8 = [1013 995 941 896; 48 153 251 290];
b8 = conversie_coordonate(b8);
reprezentare_bezier(b8, length(b8) - 1, s);

b9 = [896 934 1002 1033; 290 286 254 229];
b9 = conversie_coordonate(b9);
reprezentare_bezier(b9, length(b9) - 1, s);

b10 = [1033 1013 947 890; 229 289 355 379];
b10 = conversie_coordonate(b10);
reprezentare_bezier(b10, length(b10) - 1, s);

b11 = [890 906 954 994; 379 398 424 424];
b11 = conversie_coordonate(b11);
reprezentare_bezier(b11, length(b11) - 1, s);

b12 = [994 974 882 854; 424 448 445 427];
b12 = conversie_coordonate(b12);
reprezentare_bezier(b12, length(b12) - 1, s);

b13 = [854 863 889 925; 427 452 485 506];
b13 = conversie_coordonate(b13);
reprezentare_bezier(b13, length(b13) - 1, s);

b14 = [925 882 831 818; 506 502 475 454];
b14 = conversie_coordonate(b14);
reprezentare_bezier(b14, length(b14) - 1, s);
```

```
% Primul punct de control al lui b15 a fost calculat folosind algoritmul lui De
Casteljau pentru t = 2/3
%Castlejau(2/3, 3, b14); -> 841.96 si 476.69

b15 = [841.96 852 849 843; 476.69 499 524 551];
b15 = conversie_coordonate(b15);
reprezentare_bezier(b15, length(b15) - 1, s);

b16 = [843 816 771 750; 551 493 430 424];
b16 = conversie_coordonate(b16);
reprezentare_bezier(b16, length(b16) - 1, s);

b17 = [750 759 759 755; 424 452 485 508];
b17 = conversie_coordonate(b17);
reprezentare_bezier(b17, length(b17) - 1, s);

b18 = [755 741 708 693; 508 475 433 424];
b18 = conversie_coordonate(b18);
reprezentare_bezier(b18, length(b18) - 1, s);

b19 = [693 699 701 699; 424 437 455 469];
b19 = conversie_coordonate(b19);
reprezentare_bezier(b19, length(b19) - 1, s);

b20 = [699 692 683 671; 469 454 445 427];
b20 = conversie_coordonate(b20);
reprezentare_bezier(b20, length(b20) - 1, s);

b21 = [676.587 606 434 380; 435.085 455 442 424];
b21 = conversie_coordonate(b21);
reprezentare_bezier(b21, length(b21) - 1, s);

b22 = [380 347 311 290; 424 424 427 419];
b22 = conversie_coordonate(b22);
reprezentare_bezier(b22, length(b22) - 1, s);

b23 = [290 302 328 352; 419 403 400 401];
b23 = conversie_coordonate(b23);
reprezentare_bezier(b23, length(b23) - 1, s);

b24 = [352 317 289 287; 401 386 349 306];
b24 = conversie_coordonate(b24);
reprezentare_bezier(b24, length(b24) - 1, s);

b25 = [287 302 323 341; 306 337 355 361];
b25 = conversie_coordonate(b25);
reprezentare_bezier(b25, length(b25) - 1, s);

b26 = [341 323 322 338; 361 322 246 216];
b26 = conversie_coordonate(b26);
reprezentare_bezier(b26, length(b26) - 1, s);

b27 = [338 338 353 373; 216 252 301 316];
b27 = conversie_coordonate(b27);
reprezentare_bezier(b27, length(b27) - 1, s);

b28 = [373 368 437 481; 316 218 113 89];
b28 = conversie_coordonate(b28);
```

```
reprezentare_bezier(b28, length(b28) - 1, s);

b29 = [481 458 438 448; 89 129 188 215];
b29 = conversie_coordonate(b29);
reprezentare_bezier(b29, length(b29) - 1, s);

b30 = [404.111 402 400 393; 715.296 726 742 755];
b30 = conversie_coordonate(b30);
reprezentare_bezier(b30, length(b30) - 1, s);

b31 = [393 400 408 414; 755 754 746 736];
b31 = conversie_coordonate(b31);
reprezentare_bezier(b31, length(b31) - 1, s);

b32 = [414 417 414 410; 736 757 790 810];
b32 = conversie_coordonate(b32);
reprezentare_bezier(b32, length(b32) - 1, s);

b33 = [410 419 425 428; 810 804 791 782];
b33 = conversie_coordonate(b33);
reprezentare_bezier(b33, length(b33) - 1, s);

b34 = [428 428 428 425; 782 796 811 823];
b34 = conversie_coordonate(b34);
reprezentare_bezier(b34, length(b34) - 1, s);

b35 = [425 429 435 436; 823 822 815 812];
b35 = conversie_coordonate(b35);
reprezentare_bezier(b35, length(b35) - 1, s);

b36 = [792.302 799 803 800; 700.207 717 739 754];
b36 = conversie_coordonate(b36);
reprezentare_bezier(b36, length(b36) - 1, s);

b37 = [800 795 789 772.73; 754 740 725 716.047];
b37 = conversie_coordonate(b37);
reprezentare_bezier(b37, length(b37) - 1, s);

b38 = [772.73 785 787 781; 716.047 742 775 806];
b38 = conversie_coordonate(b38);
reprezentare_bezier(b38, length(b38) - 1, s);

b39 = [781 775 768 760.562; 806 787 775 771.333];
b39 = conversie_coordonate(b39);
reprezentare_bezier(b39, length(b39) - 1, s);

b40 = [760.562 763 764 761; 771.333 802 823 846];
b40 = conversie_coordonate(b40);
reprezentare_bezier(b40, length(b40) - 1, s);

b41 = [761 757 753 747.963; 846 826 811 805.261];
b41 = conversie_coordonate(b41);
reprezentare_bezier(b41, length(b41) - 1, s);

neck1 = [434.28 436 435.919; 772.256 812 838.406];
neck1 = conversie_coordonate(neck1);
reprezentare_bezier(neck1, length(neck1) - 1, s);

neck2 = [772.165 772 768 760.562; 821.794 800 781 771.333];
```

```
neck2 = conversie_coordonate(neck2);
reprezentare_bezier(neck2, length(neck2) - 1, s);
```

*10. Functia de desenare a bandanei – functia **headband(s)** :*

```
function [] = headband(s)
hb1 = [364.856 353 354 360; 424.158 435 451 455];
hb1 = conversie_coordonate(hb1);
reprezentare_bezier(hb1, length(hb1) - 1, s);

hb2 = [360 352 355 369 354; 455 472 514 535 552];
hb2 = conversie_coordonate(hb2);
reprezentare_bezier(hb2, length(hb2) - 1, s);

hb3 = [354 432 523 589 665 731 783 815; 552 585 594 596 593 578 559 528];
hb3 = conversie_coordonate(hb3);
reprezentare_bezier(hb3, length(hb3) - 1, s);

hb4 = [815 813 810 805 805; 528 517 509 503 484];
hb4 = conversie_coordonate(hb4);
reprezentare_bezier(hb4, length(hb4) - 1, s);

hb5 = [445.229 414 410 415 440; 578.338 545 501 468 460];
hb5 = conversie_coordonate(hb5);
reprezentare_bezier(hb5, length(hb5) - 1, s);

hb6 = [440 472 523 560 601; 460 455 459 470 461];
hb6 = conversie_coordonate(hb6);
reprezentare_bezier(hb6, length(hb6) - 1, s);

hb7 = [601 628 668 690.782; 461 457 451 454.678];
hb7 = conversie_coordonate(hb7);
reprezentare_bezier(hb7, length(hb7) - 1, s);

hb8 = [755.457 773 775 773 758; 447.721 465 496 526 561.735];
hb8 = conversie_coordonate(hb8);
reprezentare_bezier(hb8, length(hb8) - 1, s);
```

*11. Functia de desenare a simbolului–functia **headband_symbol(s)**:*

```
function [] = headband_symbol(s)
hold on
h1 = [617 612 591 596; 534 547 543 523];
h1 = conversie_coordonate(h1);
reprezentare_bezier(h1, length(h1) - 1, s);

h2 = [596 601 634 636; 523 508 514 532];
h2 = conversie_coordonate(h2);
reprezentare_bezier(h2, length(h2) - 1, s);

h3 = [636 640 612 601; 532 552 570 563];
h3 = conversie_coordonate(h3);
reprezentare_bezier(h3, length(h3) - 1, s);

h4 = [601 584 570 578; 563 559 533 517];
h4 = conversie_coordonate(h4);
reprezentare_bezier(h4, length(h4) - 1, s);
```

```
h5 = [578 595 641; 517 479 501];
h5 = conversie_coordonate(h5);
reprezentare_bezier(h5, length(h5) - 1, s);

h6 = [641 650 657 655; 501 503 497 487];
h6 = conversie_coordonate(h6);
reprezentare_bezier(h6, length(h6) - 1, s);

h7 = [575.842 546 542 588.146; 530.519 563 544 555.467];
h7 = conversie_coordonate(h7);
reprezentare_bezier(h7, length(h7) - 1, s);
```

*12. Functia de desenare a sprancenelor – functia **eyebrows(s)** :*

```
function [] = eyebrows(s)
p = [463 529 586 614 640 692 750; 604 597 625 603 622 586 589];
p = conversie_coordonate(p);
x1 = [472; 592]; x1 = conversie_coordonate(x1);
x2 = [744; 580]; x2 = conversie_coordonate(x2);
hold on
reprezentare_Fmill(p, x1, x2, s);
```

*13. Functia de desenare a mastii – functia **facemask(s, s1)** :*

```
function [] = face_mask(s, s1)
hold on
p = [417 526 615 695 774; 724 698 644 682 705];
p = conversie_coordonate(p);
x1 = [451; 738]; x1 = conversie_coordonate(x1);
x2 = [752; 720]; x2 = conversie_coordonate(x2);
reprezentare_Fmill(p, x1, x2, s1);
hold on
f1 = [417 431 475 547 615; 724 785 856 909 923];
f1 = conversie_coordonate(f1);
reprezentare_bezier(f1, length(f1) - 1, s);

f2 = [615 677 723 765 774; 923 922 870 797 705];
f2 = conversie_coordonate(f2);
reprezentare_bezier(f2, length(f2) - 1, s);
```

*14. Functia de desenare a urechiilor – functia **ears(s)** :*

```
function [] = ears(s)
u1 = [774 790 813 798 807 833 839 818 817 833 826 815;
    705 714 691 670 639 630 602 588 565 548 530 528];
u1 = conversie_coordonate(u1);
reprezentare_bezier(u1, length(u1) - 1, s);

u2 = [815 792 785 783 788; 528 554 575 601 619];
u2 = conversie_coordonate(u2);
reprezentare_bezier(u2, length(u2) - 1, s);

u3 = [788 796 796 793; 619 625 634 641];
u3 = conversie_coordonate(u3);
reprezentare_bezier(u3, length(u3) - 1, s);
```

```
u4 = [793 784 777 774; 641 652 679 705];
u4 = conversie_coordonate(u4);
reprezentare_bezier(u4, length(u4) - 1, s);

u5 = [418 403 391 385; 724 719 702 681];
u5 = conversie_coordonate(u5);
reprezentare_bezier(u5, length(u5) - 1, s);

u6 = [385 379 372 370; 681 677 664 651];
u6 = conversie_coordonate(u6);
reprezentare_bezier(u6, length(u6) - 1, s);

u7 = [370 368 368 370; 651 643 627 617];
u7 = conversie_coordonate(u7);
reprezentare_bezier(u7, length(u7) - 1, s);

u8 = [370 363 364 370.761; 617 602 575 558.553];
u8 = conversie_coordonate(u8);
reprezentare_bezier(u8, length(u8) - 1, s);

u9 = [370.761 393 404 409; 558.553 573 614 648];
u9 = conversie_coordonate(u9);
reprezentare_bezier(u9, length(u9) - 1, s);

u10 = [409 408 408 409; 648 653 660 667];
u10 = conversie_coordonate(u10);
reprezentare_bezier(u10, length(u10) - 1, s);

u11 = [409 407 411 417; 667 681 705 724];
u11 = conversie_coordonate(u11);
reprezentare_bezier(u11, length(u11) - 1, s);

u12 = [824.32 814 808 808; 547.301 549 561 575];
u12 = conversie_coordonate(u12);
reprezentare_bezier(u12, length(u12) - 1, s);

u13 = [805 813 804 788.551; 594 620 641 647.996];
u13 = conversie_coordonate(u13);
reprezentare_bezier(u13, length(u13) - 1, s);

u14 = [365.331 374 378 377; 591.445 595 605 613];
u14 = conversie_coordonate(u14);
reprezentare_bezier(u14, length(u14) - 1, s);

u15 = [381 383 394 408.13; 629 653 669 673.173];
u15 = conversie_coordonate(u15);
reprezentare_bezier(u15, length(u15) - 1, s);
```

*15. Functia de desenare a ochiilor – functia **eyes(s)** :*

```
function [] = eyes(s)
e1 = [459 475 496 520; 616 636 641 636];
e1 = conversie_coordonate(e1);
reprezentare_bezier(e1, length(e1) - 1, s);

e2 = [520 527 542 551; 636 631 628 630];
e2 = conversie_coordonate(e2);
reprezentare_bezier(e2, length(e2) - 1, s);

e3 = [551 552 554 553; 630 629 624 621];
e3 = conversie_coordonate(e3);
reprezentare_bezier(e3, length(e3) - 1, s);

e4 = [553 534 479 459; 621 609 608 616];
e4 = conversie_coordonate(e4);
reprezentare_bezier(e4, length(e4) - 1, s);

e5 = [491.289 496 525 531.867; 610.939 635 639 613.785];
e5 = conversie_coordonate(e5);
reprezentare_bezier(e5, length(e5) - 1, s);

e6 = [667 699 747 757; 621 630 620 599];
e6 = conversie_coordonate(e6);
reprezentare_bezier(e6, length(e6) - 1, s);

e7 = [757 741 688 671; 599 593 596 609];
e7 = conversie_coordonate(e7);
reprezentare_bezier(e7, length(e7) - 1, s);

e8 = [671 664 663 667; 609 612 617 621];
e8 = conversie_coordonate(e8);
reprezentare_bezier(e8, length(e8) - 1, s);

e9 = [690.521 695 723 724.3; 600.809 626 625 596.201];
e9 = conversie_coordonate(e9);
reprezentare_bezier(e9, length(e9) - 1, s);
```

*16. Functia de desenare a gulerului jachetei – functia **jacket_collar(s)** :*

```
function [] = jacket_collar(s)
v1 = [396.292 311 246 261; 705.927 708 753 795];
v1 = conversie_coordonate(v1);
reprezentare_bezier(v1, length(v1) - 1, s);

v2 = [261 266 299 321; 795 826 850 860];
v2 = conversie_coordonate(v2);
reprezentare_bezier(v2, length(v2) - 1, s);

v3 = [321 336 352 350; 860 869 886 920];
v3 = conversie_coordonate(v3);
reprezentare_bezier(v3, length(v3) - 1, s);

v4 = [350 347 345 345; 920 945 974 1020];
v4 = conversie_coordonate(v4);
```

```
reprezentare_bezier(v4, length(v4) - 1, s);

v5 = [345 295 247 246; 1020 986 932 896];
v5 = conversie_coordonate(v5);
reprezentare_bezier(v5, length(v5) - 1, s);

v6 = [246 243 249 257; 896 865 813 781];
v6 = conversie_coordonate(v6);
reprezentare_bezier(v6, length(v6) - 1, s);

v7 = [370 327 302 306; 717 725 745 779];
v7 = conversie_coordonate(v7);
reprezentare_bezier(v7, length(v7) - 1, s);

v8 = [306 314 345 369; 779 807 828 833];
v8 = conversie_coordonate(v8);
reprezentare_bezier(v8, length(v8) - 1, s);

v9 = [369 384 392 390; 833 835 847 858];
v9 = conversie_coordonate(v9);
reprezentare_bezier(v9, length(v9) - 1, s);

v10 = [390 392 390 385; 858 884 936 998];
v10 = conversie_coordonate(v10);
reprezentare_bezier(v10, length(v10) - 1, s);

v11 = [404.111 374 343 346; 715.296 724 745 770];
v11 = conversie_coordonate(v11);
reprezentare_bezier(v11, length(v11) - 1, s);

v12 = [346 346 371 409; 770 790 817 829];
v12 = conversie_coordonate(v12);
reprezentare_bezier(v12, length(v12) - 1, s);

v13 = [409 451 457 456; 829 840 853 866];
v13 = conversie_coordonate(v13);
reprezentare_bezier(v13, length(v13) - 1, s);

v14 = [456 459 455 444; 866 899 961 1030];
v14 = conversie_coordonate(v14);
reprezentare_bezier(v14, length(v14) - 1, s);

v15 = [345 370 424 444; 1020 1032 1039 1030];
v15 = conversie_coordonate(v15);
reprezentare_bezier(v15, length(v15) - 1, s);

v16 = [444 457 514 558; 1030 1065 1097 1095];
v16 = conversie_coordonate(v16);
reprezentare_bezier(v16, length(v16) - 1, s);

v17 = [558 590 653 688; 1095 1097 1097 1094];
v17 = conversie_coordonate(v17);
reprezentare_bezier(v17, length(v17) - 1, s);

v18 = [688 725 762 762; 1094 1080 1054 1030];
v18 = conversie_coordonate(v18);
reprezentare_bezier(v18, length(v18) - 1, s);

v19 = [782.851 794 807 801; 751.275 758 773 784];
```

```
v19 = conversie_coordonate(v19);
reprezentare_bezier(v19, length(v19) - 1, s);

v20 = [801 795 792 780; 784 795 799 812];
v20 = conversie_coordonate(v20);
reprezentare_bezier(v20, length(v20) - 1, s);

v21 = [780 771 767 768; 812 817 835 849];
v21 = conversie_coordonate(v21);
reprezentare_bezier(v21, length(v21) - 1, s);

v22 = [768 767 758 762; 849 893 966 1030];
v22 = conversie_coordonate(v22);
reprezentare_bezier(v22, length(v22) - 1, s);

v23 = [815 839 852 851; 735 748 761 779];
v23 = conversie_coordonate(v23);
reprezentare_bezier(v23, length(v23) - 1, s);

v24 = [851 849 838 821; 779 797 814 819];
v24 = conversie_coordonate(v24);
reprezentare_bezier(v24, length(v24) - 1, s);

v25 = [821 807 807 809; 819 831 862 883];
v25 = conversie_coordonate(v25);
reprezentare_bezier(v25, length(v25) - 1, s);

v26 = [809 810 805 799; 883 905 943 1003];
v26 = conversie_coordonate(v26);
reprezentare_bezier(v26, length(v26) - 1, s);

v27 = [798.332 837 873 886; 720.005 723 740 785];
v27 = conversie_coordonate(v27);
reprezentare_bezier(v27, length(v27) - 1, s);

v28 = [886 890 891 886; 785 814 839 857];
v28 = conversie_coordonate(v28);
reprezentare_bezier(v28, length(v28) - 1, s);

v29 = [886 883 882 886; 857 876 911 933];
v29 = conversie_coordonate(v29);
reprezentare_bezier(v29, length(v29) - 1, s);

v30 = [886 890 873 851; 933 977 1015 1041];
v30 = conversie_coordonate(v30);
reprezentare_bezier(v30, length(v30) - 1, s);

v31 = [851 822 779 762; 1041 1044 1039 1030];
v31 = conversie_coordonate(v31);
reprezentare_bezier(v31, length(v31) - 1, s);
```

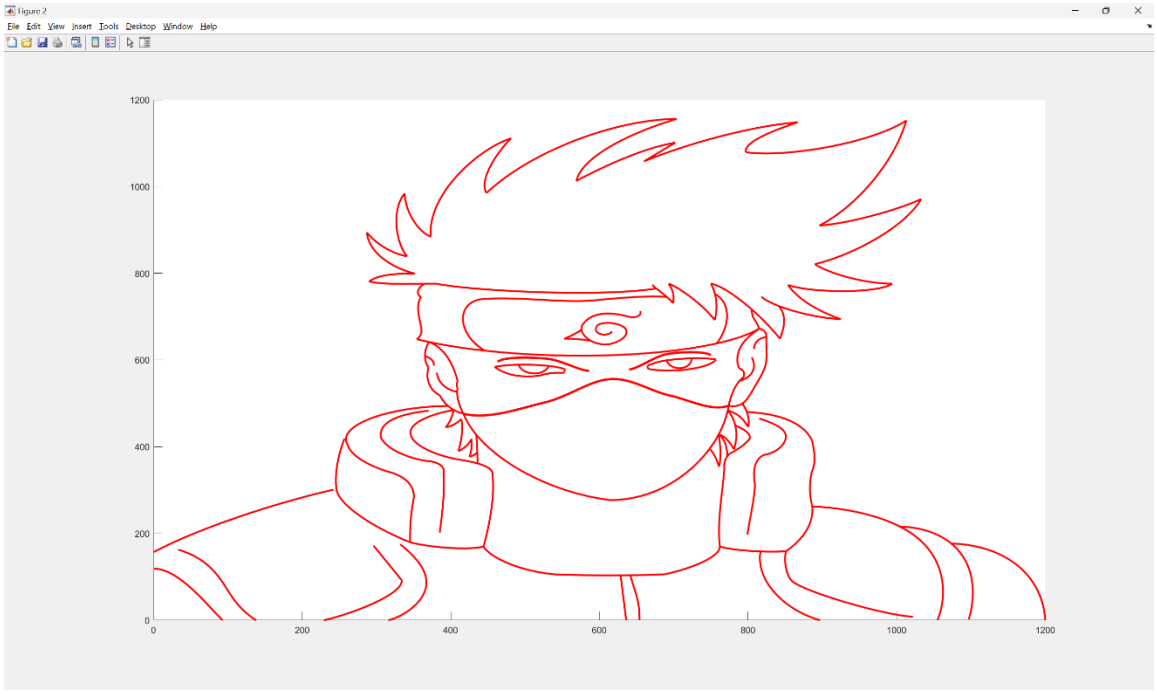
*17. Functia de desenare a vestei – functia **Vest(s)** :*

```
function [] = Vest(s)
V1 = [242 167 61 0; 899 928 987 1043];
V1 = conversie_coordonate(V1);
reprezentare_bezier(V1, length(V1) - 1, s);

V2 = [0 35 74 93; 1081 1081 1170 1200];
V2 = conversie_coordonate(V2);
reprezentare_bezier(V2, length(V2) - 1, s);

V3 = [33 66 95 99 102 122 138; 1037 1054 1087 1140 1147 1184 1200];
V3 = conversie_coordonate(V3);
reprezentare_bezier(V3, length(V3) - 1, s);

V4 = [296 334; 1028 1108];
V4 = conversie_coordonate(V4);
reprezentare_bezier(V4, length(V4) - 1, s);

V5 = [334 335 263 229; 1108 1142 1187 1200];
V5 = conversie_coordonate(V5);
reprezentare_bezier(V5, length(V5) - 1, s);

V6 = [332 362 377 376 349 316; 1025 1066 1105 1137 1185 1200];
V6 = conversie_coordonate(V6);
reprezentare_bezier(V6, length(V6) - 1, s);

V7 = [628.458 636; 1096.33 1200];
V7 = conversie_coordonate(V7);
reprezentare_bezier(V7, length(V7) - 1, s);

V8 = [641.548 649 654 654; 1096.13 1142 1154 1200];
V8 = conversie_coordonate(V8);
reprezentare_bezier(V8, length(V8) - 1, s);

V9 = [816.76 810 848 897; 1041.36 1108 1176 1200];
V9 = conversie_coordonate(V9);
reprezentare_bezier(V9, length(V9) - 1, s);

V10 = [851 848 851 860; 1041 1063 1097 1111];
V10 = conversie_coordonate(V10);
reprezentare_bezier(V10, length(V10) - 1, s);

V11 = [860 882 983 1022; 1111 1138 1188 1192];
V11 = conversie_coordonate(V11);
reprezentare_bezier(V11, length(V11) - 1, s);

V12 = [886.4 928 993 1043 1070 1069 1055;
       938 939 960 988 1051 1144 1199];
V12 = conversie_coordonate(V12);
reprezentare_bezier(V12, length(V12) - 1, s);

V13 = [1004.47 1065 1097 1108 1105 1097;
       984.05 987 1037 1088 1155 1198];
V13 = conversie_coordonate(V13);
reprezentare_bezier(V13, length(V13) - 1, s);
```

```
V14 = [1073.56 1142 1196 1200; 1023.25 1026 1090 1200];
V14 = conversie_coordonate(V14);
reprezentare_bezier(V14, length(V14) - 1, s);
```

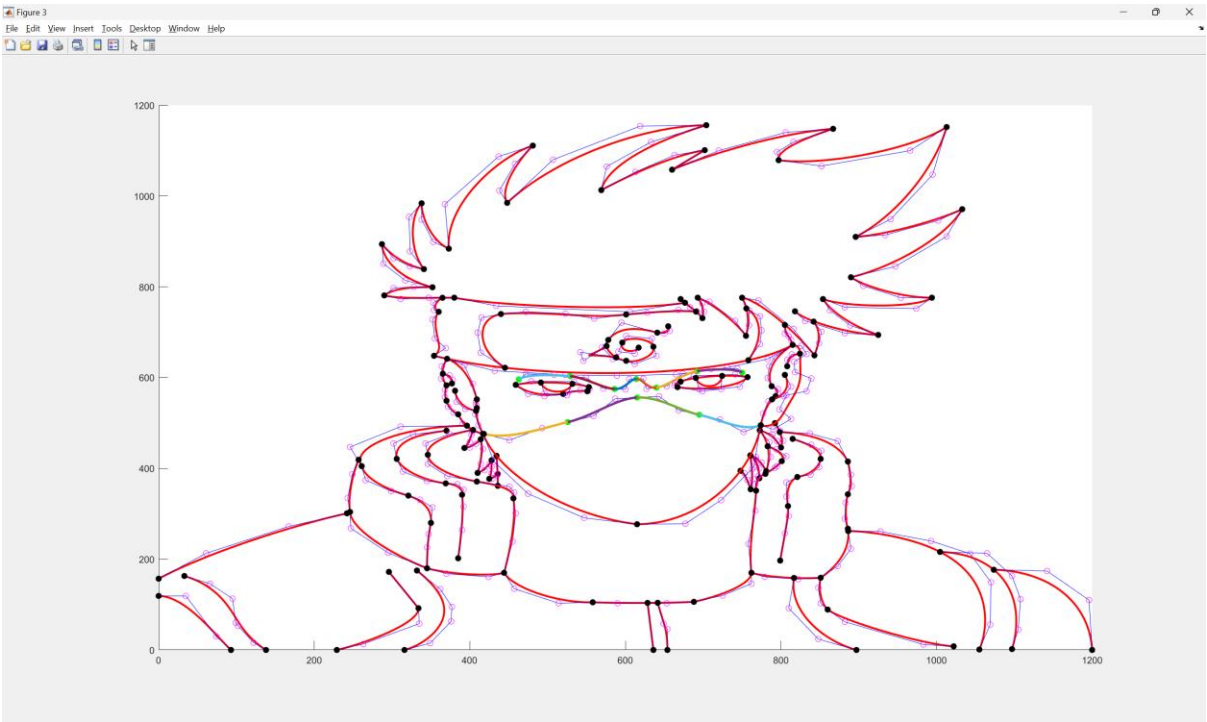
18. Functia de desenare a intregii imagini fara poligoanele de control – fisierul ***Drawing.m*** *:*

```
clear, clc
figure(2)
hold on
%% Drawing the hair with spline interpolation
hair('-');
%% Drawing the headband and the metal plate on it
headband('-');
%% Drawing of the headband sign
headband_symbol('-');
%% Drawing the eyebrows with F-Mill
eyebrows('e');
%% Drawing the face mask
face_mask('-','m');
%% Drawing the ears
ears('-');
%% Drawing the eyes
eyes('-');
%% Drawing the jacket collar
jacket_collar('-');
%% Drawing the vest
Vest('-');
%%
axis([0 1200 0 1200])
```



19. Functia de desenare a intregii imagini impreuna cu poligoanele de control – fisierul ***Drawing_with_control_points.m*** *:*

```
clear, clc
figure(3)
hold on
%% Drawing the hair with spline interpolation with control points
hair('pc');
%% Drawing the headband and the metal plate on it with control points
headband('pc');
%% Drawing of the headband sign with control points
headband_symbol('pc');
%% Drawing the eyebrows with F-Mill with control points
eyebrows('-');
%% Drawing the face mask with control points
face_mask('pc','-');
%% Drawing the ears with control points
ears('pc');
%% Drawing the eyes with control points
eyes('pc');
%% Drawing the jacket collar with control points
jacket_collar('pc');
%% Drawing the vest with control points
Vest('pc');
%%
axis([0 1200 0 1200])
```



20. Functia de desenare a curbelor si punctelor de control impreuna cu imaginea suport – functia ***imagine.m*** *:*

```
%clear, clc
figure(1)
img = imread('kakashi_bezier2.png');
imshow(img);
%% Hair of the character overlaped with image
hold on
b1 = [448 507 619 704; 215 120 46 44];
reprezentare_bezier_prototype(b1, length(b1) - 1);
b2 = [704 633 576 569; 44 81 135 187];
reprezentare_bezier_prototype(b2, length(b2) - 1);
b3 = [569 613 663 702; 187 147 110 99];
reprezentare_bezier_prototype(b3, length(b3) - 1);
b4 = [702 660; 99 142];
reprezentare_bezier_prototype(b4, length(b4) - 1);
b5 = [660 720 806 867; 142 100 60 52];
reprezentare_bezier_prototype(b5, length(b5) - 1);
b6 = [867 816 795 797; 52 81 103 121];
reprezentare_bezier_prototype(b6, length(b6) - 1);
b7 = [797 852 966 1013; 121 134 100 48];
reprezentare_bezier_prototype(b7, length(b7) - 1);
b8 = [1013 995 941 896; 48 153 251 290];
reprezentare_bezier_prototype(b8, length(b8) - 1);
b9 = [896 934 1002 1033; 290 286 254 229];
reprezentare_bezier_prototype(b9, length(b9) - 1);
b10 = [1033 1013 947 890; 229 289 355 379];
reprezentare_bezier_prototype(b10, length(b10) - 1);
b11 = [890 906 954 994; 379 398 424 424];
reprezentare_bezier_prototype(b11, length(b11) - 1);
b12 = [994 974 882 854; 424 448 445 427];
reprezentare_bezier_prototype(b12, length(b12) - 1);
b13 = [854 863 889 925; 427 452 485 506];
reprezentare_bezier_prototype(b13, length(b13) - 1);
b14 = [925 882 831 818; 506 502 475 454];
reprezentare_bezier_prototype(b14, length(b14) - 1);
% Primul punct de control al lui b15 a fost calculat folosind algoritmul lui De
Casteljau pentru t = 2/3
%Castlejau(2/3, 3, b14); -> 841.96 si 476.69
b15 = [841.96 852 849 843; 476.69 499 524 551];
reprezentare_bezier_prototype(b15, length(b15) - 1);
b16 = [843 816 771 750; 551 493 430 424];
reprezentare_bezier_prototype(b16, length(b16) - 1);
b17 = [750 759 759 755; 424 452 485 508];
reprezentare_bezier_prototype(b17, length(b17) - 1);
b18 = [755 741 708 693; 508 475 433 424];
reprezentare_bezier_prototype(b18, length(b18) - 1);
b19 = [693 699 701 699; 424 437 455 469];
reprezentare_bezier_prototype(b19, length(b19) - 1);
b20 = [699 692 683 671; 469 454 445 427];
reprezentare_bezier_prototype(b20, length(b20) - 1);
b21 = [676.587 606 434 380; 435.085 455 442 424];
reprezentare_bezier_prototype(b21, length(b21) - 1);
b22 = [380 347 311 290; 424 424 427 419];
reprezentare_bezier_prototype(b22, length(b22) - 1);
b23 = [290 302 328 352; 419 403 400 401];
reprezentare_bezier_prototype(b23, length(b23) - 1);
```

```
b24 = [352 317 289 287; 401 386 349 306];
reprezentare_bezier_prototype(b24, length(b24) - 1);
b25 = [287 302 323 341; 306 337 355 361];
reprezentare_bezier_prototype(b25, length(b25) - 1);
b26 = [341 323 322 338; 361 322 246 216];
reprezentare_bezier_prototype(b26, length(b26) - 1);
b27 = [338 338 353 373; 216 252 301 316];
reprezentare_bezier_prototype(b27, length(b27) - 1);
b28 = [373 368 437 481; 316 218 113 89];
reprezentare_bezier_prototype(b28, length(b28) - 1);
b29 = [481 458 438 448; 89 129 188 215];
reprezentare_bezier_prototype(b29, length(b29) - 1);

b30 = [404.111 402 400 393; 715.296 726 742 755];
%b30 = conversie_coordonate(b30);
reprezentare_bezier_prototype(b30, length(b30) - 1);

b31 = [393 400 408 414; 755 754 746 736];
%b31 = conversie_coordonate(b31);
reprezentare_bezier_prototype(b31, length(b31) - 1);

b32 = [414 417 414 410; 736 757 790 810];
%b32 = conversie_coordonate(b32);
reprezentare_bezier_prototype(b32, length(b32) - 1);

b33 = [410 419 425 428; 810 804 791 782];
%b33 = conversie_coordonate(b33);
reprezentare_bezier_prototype(b33, length(b33) - 1);

b34 = [428 428 428 425; 782 796 811 823];
%b34 = conversie_coordonate(b34);
reprezentare_bezier_prototype(b34, length(b34) - 1);

b35 = [425 429 435 436; 823 822 815 812];
%b35 = conversie_coordonate(b35);
reprezentare_bezier_prototype(b35, length(b35) - 1);

b36 = [792.302 799 803 800; 700.207 717 739 754];
%b36 = conversie_coordonate(b36);
reprezentare_bezier_prototype(b36, length(b36) - 1);

b37 = [800 795 789 772.73; 754 740 725 716.047];
%b37 = conversie_coordonate(b37);
reprezentare_bezier_prototype(b37, length(b37) - 1);

b38 = [772.73 785 787 781; 716.047 742 775 806];
%b38 = conversie_coordonate(b38);
reprezentare_bezier_prototype(b38, length(b38) - 1);

b39 = [781 775 768 760.562; 806 787 775 771.333];
%b39 = conversie_coordonate(b39);
reprezentare_bezier_prototype(b39, length(b39) - 1);

b40 = [760.562 763 764 761; 771.333 802 823 846];
%b40 = conversie_coordonate(b40);
reprezentare_bezier_prototype(b40, length(b40) - 1);

b41 = [761 757 753 747.963; 846 826 811 805.261];
%b41 = conversie_coordonate(b41);
```

```
reprezentare_bezier_prototype(b41, length(b41) - 1);

neck1 = [434.28 436 435.919; 772.256 812 838.406];
%neck1 = conversie_coordonate(neck1);
reprezentare_bezier_prototype(neck1, length(neck1) - 1);

neck2 = [772.165 772 768 760.562; 821.794 800 781 771.333];
%neck2 = conversie_coordonate(neck2);
reprezentare_bezier_prototype(neck2, length(neck2) - 1);
%% Eyebrows represented with Fmill
p = [463 529 586 614 640 692 750; 604 597 625 603 622 586 589];
x1 = [472; 592];
x2 = [744; 580];
hold on
reprezentare_Fmill(p, x1, x2, '-');

%% Headband sign representation
hold on
h1 = [617 612 591 596; 534 547 543 523];
%h1 = conversie_coordonate(h1);
reprezentare_bezier_prototype(h1, length(h1) - 1);

h2 = [596 601 634 636; 523 508 514 532];
%h2 = conversie_coordonate(h2);
reprezentare_bezier_prototype(h2, length(h2) - 1);

h3 = [636 640 612 601; 532 552 570 563];
%h3 = conversie_coordonate(h3);
reprezentare_bezier_prototype(h3, length(h3) - 1);

h4 = [601 584 570 578; 563 559 533 517];
%h4 = conversie_coordonate(h4);
reprezentare_bezier_prototype(h4, length(h4) - 1);

h5 = [578 595 641; 517 479 501];
%h4 = conversie_coordonate(h4);
reprezentare_bezier_prototype(h5, length(h5) - 1);

h6 = [641 650 657 655; 501 503 497 487];
%h6 = conversie_coordonate(h6);
reprezentare_bezier_prototype(h6, length(h6) - 1);

h7 = [575.842 546 542 588.146; 530.519 563 544 555.467];
%h7 = conversie_coordonate(h7);
reprezentare_bezier_prototype(h7, length(h7) - 1);
%% Headband + metal plate
hold on
hb1 = [364.856 353 354 360; 424.158 435 451 455];
%hb1 = conversie_coordonate(hb1);
reprezentare_bezier_prototype(hb1, length(hb1) - 1);

hb2 = [360 352 355 369 354; 455 472 514 535 552];
%hb2 = conversie_coordonate(hb2);
reprezentare_bezier_prototype(hb2, length(hb2) - 1);

hb3 = [354 432 523 589 665 731 783 815; 552 585 594 596 593 578 559 528];
%hb3 = conversie_coordonate(hb3);
reprezentare_bezier_prototype(hb3, length(hb3) - 1);
```

```
hb4 = [815 813 810 805 805; 528 517 509 503 484];
%hb4 = conversie_coordonate(hb4);
reprezentare_bezier_prototype(hb4, length(hb4) - 1);

hb5 = [445.229 414 410 415 440; 578.338 545 501 468 460];
%hb5 = conversie_coordonate(hb5);
reprezentare_bezier_prototype(hb5, length(hb5) - 1);

hb6 = [440 472 523 560 601; 460 455 459 470 461];
%hb6 = conversie_coordonate(hb6);
reprezentare_bezier_prototype(hb6, length(hb6) - 1);

hb7 = [601 628 668 690.782; 461 457 451 454.678];
%hb7 = conversie_coordonate(hb7);
reprezentare_bezier_prototype(hb7, length(hb7) - 1);

hb8 = [755.457 773 775 773 758; 447.721 465 496 526 561.735];
%hb8 = conversie_coordonate(hb8);
reprezentare_bezier_prototype(hb8, length(hb8) - 1);
%% Face mask
hold on
p = [417 526 615 695 774; 724 698 644 682 705];
x1 = [451; 738];
x2 = [752; 720];
reprezentare_Fmill(p, x1, x2, '-');
hold on
f1 = [417 431 475 547 615; 724 785 856 909 923];
%f1 = conversie_coordonate(f1);
reprezentare_bezier_prototype(f1, length(f1) - 1);

f2 = [615 677 723 765 774; 923 922 870 797 705];
%f2 = conversie_coordonate(f2);
reprezentare_bezier_prototype(f2, length(f2) - 1);

%% The ears
u1 = [774 790 813 798 807 833 839 818 817 833 826 815;
    705 714 691 670 639 630 602 588 565 548 530 528];
%u1 = conversie_coordonate(u1);
reprezentare_bezier_prototype(u1, length(u1) - 1);

u2 = [815 792 785 783 788; 528 554 575 601 619];
%u2 = conversie_coordonate(u2);
reprezentare_bezier_prototype(u2, length(u2) - 1);

u3 = [788 796 796 793; 619 625 634 641];
%u3 = conversie_coordonate(u3);
reprezentare_bezier_prototype(u3, length(u3) - 1);

u4 = [793 784 777 774; 641 652 679 705];
%u4 = conversie_coordonate(u4);
reprezentare_bezier_prototype(u4, length(u4) - 1);

u5 = [417 403 391 385; 724 719 702 681];
%u5 = conversie_coordonate(u5);
reprezentare_bezier_prototype(u5, length(u5) - 1);

u6 = [385 379 372 370; 681 677 664 651];
%u6 = conversie_coordonate(u6);
reprezentare_bezier_prototype(u6, length(u6) - 1);
```

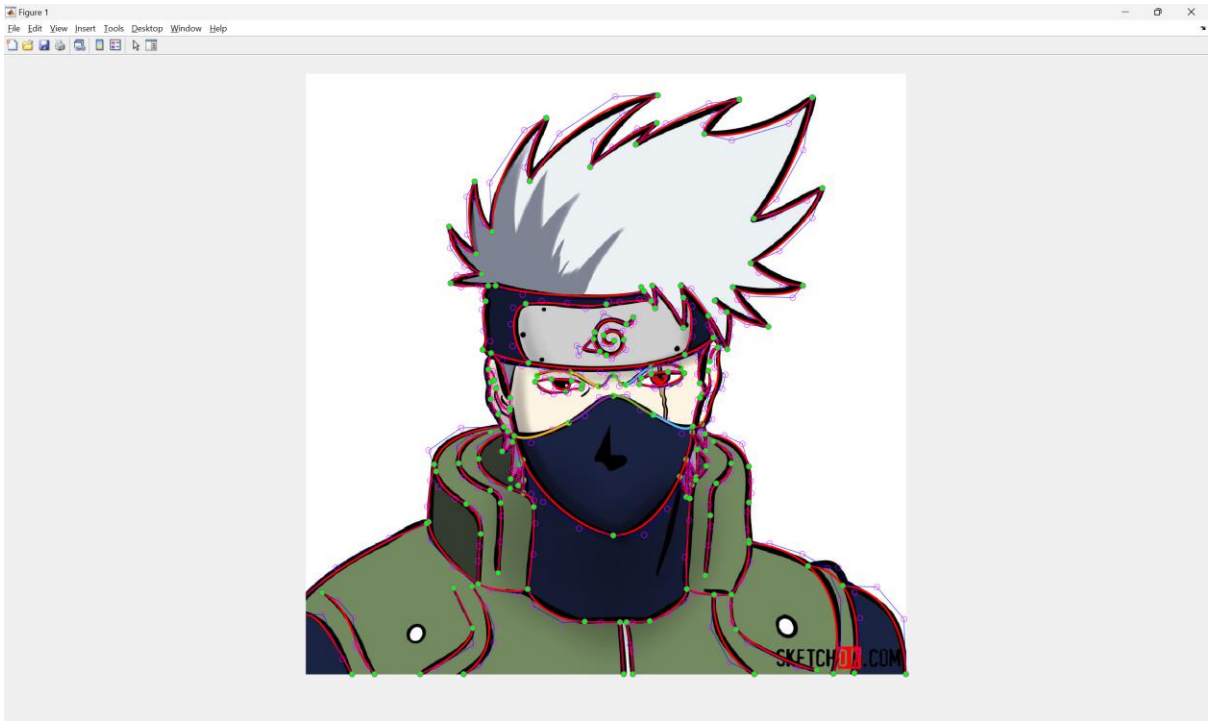
```
u7 = [370 368 368 370; 651 643 627 617];
%u7 = conversie_coordonate(u7);
reprezentare_bezier_prototype(u7, length(u7) - 1);

u8 = [370 363 364 370.761; 617 602 575 558.553];
%u8 = conversie_coordonate(u8);
reprezentare_bezier_prototype(u8, length(u8) - 1);

u9 = [370.761 393 404 409; 558.553 573 614 648];
%u9 = conversie_coordonate(u9);
reprezentare_bezier_prototype(u9, length(u9) - 1);

u10 = [409 408 408 409; 648 653 660 667];
%u10 = conversie_coordonate(u10);
reprezentare_bezier_prototype(u10, length(u10) - 1);

u11 = [409 407 411 417; 667 681 705 724];
%u11 = conversie_coordonate(u11);
reprezentare_bezier_prototype(u11, length(u11) - 1);

u12 = [824.32 814 808 808; 547.301 549 561 575];
%u12 = conversie_coordonate(u12);
reprezentare_bezier_prototype(u12, length(u12) - 1);

u13 = [805 813 804 788.551; 594 620 641 647.996];
%u13 = conversie_coordonate(u13);
reprezentare_bezier_prototype(u13, length(u13) - 1);

u14 = [365.331 374 378 377; 591.445 595 605 613];
%u14 = conversie_coordonate(u14);
reprezentare_bezier_prototype(u14, length(u14) - 1);

u15 = [381 383 394 408.13; 629 653 669 673.173];
%u15 = conversie_coordonate(u15);
reprezentare_bezier_prototype(u15, length(u15) - 1);
%% Eyes
e1 = [459 475 496 520; 616 636 641 636];
%e1 = conversie_coordonate(e1);
reprezentare_bezier_prototype(e1, length(e1) - 1);

e2 = [520 527 542 551; 636 631 628 630];
%e2 = conversie_coordonate(e2);
reprezentare_bezier_prototype(e2, length(e2) - 1);

e3 = [551 552 554 553; 630 629 624 621];
%e3 = conversie_coordonate(e3);
reprezentare_bezier_prototype(e3, length(e3) - 1);

e4 = [553 534 479 459; 621 609 608 616];
%e4 = conversie_coordonate(e4);
reprezentare_bezier_prototype(e4, length(e4) - 1);

e5 = [491.289 496 525 531.867; 610.939 635 639 613.785];
%e5 = conversie_coordonate(e5);
reprezentare_bezier_prototype(e5, length(e5) - 1);

e6 = [667 699 747 757; 621 630 620 599];
%e6 = conversie_coordonate(e6);
```

```
reprezentare_bezier_prototype(e6, length(e6) - 1);

e7 = [757 741 688 671; 599 593 596 609];
%e7 = conversie_coordonate(e7);
reprezentare_bezier_prototype(e7, length(e7) - 1);

e8 = [671 664 663 667; 609 612 617 621];
%e8 = conversie_coordonate(e8);
reprezentare_bezier_prototype(e8, length(e8) - 1);

%e9 = [689.59 692 725 728; 601.048 630 630 596];
e9 = [690.521 695 723 724.3; 600.809 626 625 596.201];
%e9 = conversie_coordonate(e9);
reprezentare_bezier_prototype(e9, length(e9) - 1);
%% Drawing the vest collar
v1 = [396.292 311 246 261; 705.927 708 753 795];
%v1 = conversie_coordonate(v1);
reprezentare_bezier_prototype(v1, length(v1) - 1);

v2 = [261 266 299 321; 795 826 850 860];
%v2 = conversie_coordonate(v2);
reprezentare_bezier_prototype(v2, length(v2) - 1);

v3 = [321 336 352 350; 860 869 886 920];
%v3 = conversie_coordonate(v3);
reprezentare_bezier_prototype(v3, length(v3) - 1);

v4 = [350 347 345 345; 920 945 974 1020];
%v4 = conversie_coordonate(v4);
reprezentare_bezier_prototype(v4, length(v4) - 1);

v5 = [345 295 247 246; 1020 986 932 896];
%v5 = conversie_coordonate(v5);
reprezentare_bezier_prototype(v5, length(v5) - 1);

v6 = [246 243 249 257; 896 865 813 781];
%v6 = conversie_coordonate(v6);
reprezentare_bezier_prototype(v6, length(v6) - 1);

v7 = [370 327 302 306; 717 725 745 779];
%v7 = conversie_coordonate(v7);
reprezentare_bezier_prototype(v7, length(v7) - 1);

v8 = [306 314 345 369; 779 807 828 833];
%v8 = conversie_coordonate(v8);
reprezentare_bezier_prototype(v8, length(v8) - 1);

v9 = [369 384 392 390; 833 835 847 858];
%v9 = conversie_coordonate(v9);
reprezentare_bezier_prototype(v9, length(v9) - 1);

v10 = [390 392 390 385; 858 884 936 998];
%v10 = conversie_coordonate(v10);
reprezentare_bezier_prototype(v10, length(v10) - 1);

v11 = [404.111 374 343 346; 715.296 724 745 770];
%v11 = conversie_coordonate(v11);
reprezentare_bezier_prototype(v11, length(v11) - 1);
```

```
v12 = [346 346 371 409; 770 790 817 829];
%v12 = conversie_coordonate(v12);
reprezentare_bezier_prototype(v12, length(v12) - 1);

v13 = [409 451 457 456; 829 840 853 866];
%v13 = conversie_coordonate(v13);
reprezentare_bezier_prototype(v13, length(v13) - 1);

v14 = [456 459 455 444; 866 899 961 1030];
%v14 = conversie_coordonate(v14);
reprezentare_bezier_prototype(v14, length(v14) - 1);

v15 = [345 370 424 444; 1020 1032 1039 1030];
%v15 = conversie_coordonate(v15);
reprezentare_bezier_prototype(v15, length(v15) - 1);

v16 = [444 457 514 558; 1030 1065 1097 1095];
%v16 = conversie_coordonate(v16);
reprezentare_bezier_prototype(v16, length(v16) - 1);

v17 = [558 590 653 688; 1095 1097 1097 1094];
%v17 = conversie_coordonate(v17);
reprezentare_bezier_prototype(v17, length(v17) - 1);

v18 = [688 725 762 762; 1094 1080 1054 1030];
%v18 = conversie_coordonate(v18);
reprezentare_bezier_prototype(v18, length(v18) - 1);

v19 = [782.851 794 807 801; 751.275 758 773 784];
%v19 = conversie_coordonate(v19);
reprezentare_bezier_prototype(v19, length(v19) - 1);

v20 = [801 795 792 780; 784 795 799 812];
%v20 = conversie_coordonate(v20);
reprezentare_bezier_prototype(v20, length(v20) - 1);

v21 = [780 771 767 768; 812 817 835 849];
%v21 = conversie_coordonate(v21);
reprezentare_bezier_prototype(v21, length(v21) - 1);

v22 = [768 767 758 762; 849 893 966 1030];
%v22 = conversie_coordonate(v22);
reprezentare_bezier_prototype(v22, length(v22) - 1);

v23 = [815 839 852 851; 735 748 761 779];
%v23 = conversie_coordonate(v23);
reprezentare_bezier_prototype(v23, length(v23) - 1);

v24 = [851 849 838 821; 779 797 814 819];
%v24 = conversie_coordonate(v24);
reprezentare_bezier_prototype(v24, length(v24) - 1);

v25 = [821 807 807 809; 819 831 862 883];
%v25 = conversie_coordonate(v25);
reprezentare_bezier_prototype(v25, length(v25) - 1);

v26 = [809 810 805 799; 883 905 943 1003];
%v26 = conversie_coordonate(v26);
reprezentare_bezier_prototype(v26, length(v26) - 1);
```

```
v27 = [798.332 837 873 886; 720.005 723 740 785];
%v27 = conversie_coordonate(v27);
reprezentare_bezier_prototype(v27, length(v27) - 1);

v28 = [886 890 891 886; 785 814 839 857];
%v28 = conversie_coordonate(v28);
reprezentare_bezier_prototype(v28, length(v28) - 1);

v29 = [886 883 882 886; 857 876 911 933];
%v29 = conversie_coordonate(v29);
reprezentare_bezier_prototype(v29, length(v29) - 1);

v30 = [886 890 873 851; 933 977 1015 1041];
%v30 = conversie_coordonate(v30);
reprezentare_bezier_prototype(v30, length(v30) - 1);

v31 = [851 822 779 762; 1041 1044 1039 1030];
%v31 = conversie_coordonate(v31);
reprezentare_bezier_prototype(v31, length(v31) - 1);
%% Vest
V1 = [242 167 61 0; 899 928 987 1043];
%V1 = conversie_coordonate(V1);
reprezentare_bezier_prototype(V1, length(V1) - 1);

V2 = [0 35 74 93; 1081 1081 1170 1200];
%V2 = conversie_coordonate(V2);
reprezentare_bezier_prototype(V2, length(V2) - 1);

V3 = [33 66 95 99 102 122 138; 1037 1054 1087 1140 1147 1184 1200];
%V3 = conversie_coordonate(V3);
reprezentare_bezier_prototype(V3, length(V3) - 1);

V4 = [296 334; 1028 1108];
%V4 = conversie_coordonate(V4);
reprezentare_bezier_prototype(V4, length(V4) - 1);

V5 = [334 335 263 229; 1108 1142 1187 1200];
%V5 = conversie_coordonate(V5);
reprezentare_bezier_prototype(V5, length(V5) - 1);

V6 = [332 362 377 376 349 316; 1025 1066 1105 1137 1185 1200];
%V6 = conversie_coordonate(V6);
reprezentare_bezier_prototype(V6, length(V6) - 1);

V7 = [628.458 636; 1096.33 1200];
%V7 = conversie_coordonate(V7);
reprezentare_bezier_prototype(V7, length(V7) - 1);

V8 = [641.548 649 654 654; 1096.13 1142 1154 1200];
%V8 = conversie_coordonate(V8);
reprezentare_bezier_prototype(V8, length(V8) - 1);

V9 = [816.76 810 848 897; 1041.36 1108 1176 1200];
%V9 = conversie_coordonate(V9);
reprezentare_bezier_prototype(V9, length(V9) - 1);

V10 = [851 848 851 860; 1041 1063 1097 1111];
%V10 = conversie_coordonate(V10);
```

```
reprezentare_bezier_prototype(V10, length(V10) - 1);

V11 = [860 882 983 1022; 1111 1138 1188 1192];
%V11 = conversie_coordonate(V11);
reprezentare_bezier_prototype(V11, length(V11) - 1);

V12 = [886.4 928 993 1043 1070 1069 1055;
       938 939 960 988 1051 1144 1199];
%V12 = conversie_coordonate(V12);
reprezentare_bezier_prototype(V12, length(V12) - 1);

V13 = [1004.47 1065 1097 1108 1105 1097;
       984.05 987 1037 1088 1155 1198];
%V13 = conversie_coordonate(V13);
reprezentare_bezier_prototype(V13, length(V13) - 1);

V14 = [1073.56 1142 1196 1200; 1023.25 1026 1090 1200];
%V14 = conversie_coordonate(V14);
reprezentare_bezier_prototype(V14, length(V14) - 1);
%% trail and error tests for Bezier and Fmill
%p = ginput(5); p = p'
%x = ginput(1); x = x'
%y = ginput(1); y = y'
hold on
%reprezentare_Fmill(p, x, y);

%b = ginput(4);
%b = b'
%reprezentare_bezier_prototype(b, length(b) - 1);
```



Functia de desenarea si determinare a curbelor bezier fara conditii ale punctelor de control – functia ***reprezentare_bezier_prototype.m*** *:*

```
function [C] = reprezentare_bezier_prototype(b, n)
t = linspace(0, 1);
B = Bernstein(t, n);
C = b * B;
hold on
plot(C(1, :), C(2, :), 'r-', 'LineWidth', 2)
plot(b(1, :), b(2, :), 'b-o', 'MarkerEdgeColor', 'magenta')
plot(C(1, 1), C(2, 1), 'mo', 'MarkerFaceColor','green')
plot(C(1, end), C(2, end), 'mo', 'MarkerFaceColor','green')
end
```