```python
def ispraviSliku():
    tacke = []

    #odabrane tacke na slici
    for i in range(4):
        x = float(tackeE[i].get().strip().split(' ')[0])
        y = float(tackeE[i].get().strip().split(' ')[1])

        tacke.append([x, y, 1])

    #izracunavanje koordinata pravougaonika u cija temena treba da se preslikaju odabrane tacke na slici
    slike = []

    #p1 i p2 su pomocne promenljive koje predstavljaju duzinu ili sirinu odabranog cetvorougla
    #tj. cetvorougla koji definisu 4 odabrane tacke na slici
    p1 = math.sqrt((tacke[0][0]-tacke[3][0])**2 + (tacke[0][1]-tacke[3][1])**2)
    p2 = math.sqrt((tacke[1][0] - tacke[2][0])**2 + (tacke[1][1] - tacke[2][1])**2)

    #sirina pravougaonika je aritmeticka sredina sirina stranica originalnog cetvorougla
    sirina = round((p1+p2)/2)

    p1 = math.sqrt((tacke[0][0]-tacke[1][0])**2 + (tacke[0][1]+tacke[1][1])**2)
    p2 = math.sqrt((tacke[3][0]-tacke[2][0])**2 + (tacke[3][1] - tacke[2][1])**2)

    #duzina pravougaonika je aritmeticka sredina duzina stranica originalnog cetvorougla
    duzina = round((p1+p2)/2)

    s1 = dimensions[1] - sirina
    s1 = round(s1/2)

    s2 = dimensions[0] - duzina
    s2 = round(s2/2)

    slike.append([s2, s1, 1])
    slike.append([dimensions[0]-s2, s1, 1])
    slike.append([dimensions[0]-s2, dimensions[1]-s1, 1])
    slike.append([s2, dimensions[1]-s1, 1])

    #odgovarajuca matrica preslikavanja se dobija uz pomoc modifikovanog DLT algoritma
    P = dlt_normalize(tacke, slike)

    #racunanje inverza dobijenog preslikavanja
    P_inv = np.linalg.inv(P)

    #izdvajanje piksela ucitane slike
    old_pixels = image.load()

    #otvaranje nove slike koja je na pocetku crna
    image_new = PIL.Image.new("RGB", dimensions, "#000000")
    new_pixels = image_new.load()

    #prolazenje kroz sve piksele
    for i in range(dimensions[0]):
        for j in range(dimensions[1]):
            #za svaki piksel (i, j) se racuna piksel koji se sa originalne slike preslikava u njega
            x_coor, y_coor, z_coor = np.dot(P_inv, [i, j, 1])

            x = round(x_coor/z_coor)
            y = round(y_coor/z_coor)

            #ako koordinate tog piksela "iskacu" van originalne slike, piksel (i, j) ostaje crn
            #inace se piksel (i, j) postavlja na vrednost dobijenog piksela (x, y)
            if (x < 0 or x >= dimensions[0]):
                continue
            elif (y < 0 or y >= dimensions[1]):
                continue
            else:
                new_pixels[i, j] = old_pixels[x, y]


#cuvanje i prikazivanje slike sa ispravljenom projektivnom distorzijom
image_new.save("nova.bmp")
image_new.show()
```