```python
def naivniAlgoritam(originali, slike):

    #matrica sistema originalnih tacaka
    matricaSistema1 = np.array([
        [originali[0][0], originali[1][0], originali[2][0]],
        [originali[0][1], originali[1][1], originali[2][1]],
        [originali[0][2], originali[1][2], originali[2][2]]
    ])

    #koordinate cetvrte tacke
    rezultat1 = np.array([originali[3][0], originali[3][1], originali[3][2]])

    #resavanje sistema jednacina
    resenjeSistema1 = np.linalg.solve(matricaSistema1, rezultat1)

    #izdvajanje sva 3 resenja
    alfa1 = resenjeSistema1[0]
    beta1 = resenjeSistema1[1]
    gama1 = resenjeSistema1[2]

    #formiranje odgovarajuce matrice preslikavanja uz pomoc dobijenih koeficijenata
    kolona11 = np.array([alfa1*originali[0][0], alfa1*originali[0][1], alfa1*originali[0][2]])
    kolona12 = np.array([beta1*originali[1][0], beta1*originali[1][1], beta1*originali[1][2]])
    kolona13 = np.array([gama1*originali[2][0], gama1*originali[2][1], gama1*originali[2][2]])

    p1 = np.column_stack([kolona11, kolona12, kolona13])

    #racunanje inverza prve matrice preslikavanja
    p1 = np.linalg.inv(p1)

    #matrica sistema tacaka koje predstavljaju slike originalnih tacaka
    matricaSistema2 = np.array([
        [slike[0][0], slike[1][0], slike[2][0]],
        [slike[0][1], slike[1][1], slike[2][1]],
        [slike[0][2], slike[1][2], slike[2][2]]
    ])

    rezultat2 = np.array([slike[3][0], slike[3][1], slike[3][2]])

    #resavanja drugog sistema
    resenjeSistema2 = np.linalg.solve(matricaSistema2, rezultat2)

    alfa2 = resenjeSistema2[0]
    beta2 = resenjeSistema2[1]
    gama2 = resenjeSistema2[2]

    #formiranje odgovarajuce matrice preslikavanja uz pomoc dobijenih koeficijenata
    kolona21 = np.array([alfa2*slike[0][0], alfa2*slike[0][1], alfa2*slike[0][2]])
    kolona22 = np.array([beta2*slike[1][0], beta2*slike[1][1], beta2*slike[1][2]])
    kolona23 = np.array([gama2*slike[2][0], gama2*slike[2][1], gama2*slike[2][2]])

    p2 = np.column_stack([kolona21, kolona22, kolona23])

    #racunanje konacne matrice preslikavanja
    p = np.dot(p2, p1)

    return p
```