Математички факултет
Универзитет у Београду

# Примјена пројективне геометрије на отклањање дисторзије на фотографијама

Мастер рад

Студент
Јелена Дробњак

Београд,
2019.

# Садржај

# Увод

Свеопшта дигитализација довела је до тога да обрада података помоћу рачунара буде саставни дио живота скоро сваког појединца. Обрада дигиталних фотографија, осим из забаве и ради личног задовољства, има бројне примјене. На примјер, помоћу двије фотографије неког објекта, могуће је реконструисати негов положај и димензије.

Са развојем рачунарске графике, расте и потреба за ефикаснијим методама обраде фотографија. Одговори и рјешења за сва питања у рачунарству и програмирању налазе се у различитим математичким дисциплинама. Конкретно, обрада фотографија и цијела област рачунарског вида темељи се на геометрији, а централно мјесто заузима пројективна геометрија.

Тродимензионални свијет посматран кроз објектив губи читаву једну димензију и зато су предмети у даљини све мањи и мањи, паралелност се губи, немогуће је измјерити дужину и сл. Ипак, пројективна геометрија може да одгонетне и објасни много тога. Тема рада је отклањање пројективне дисторзије. То подразумијева извијесно „исправљање" дијела фотографије како би она изгледала вјеродостојније.

Прве двије главе представљају увод и мотивацију за рад. Ту је објашњено шта је то пројективна дисторзија и шта је математички модел фотоапарата (камере).

Трећа и четврта глава су посвећене пројективној геометрији и пројективним пресликавањима равни. Затим су детаљно описани неки алгоритми за одређивање пројективног пресликавања.

Седма глава даје објашњење како је могуће отклонити пројективну дисторзију (механички и софтверски), док је на самом крају изложен код програма који отклања ову врсту дисторзије са произвољне фотографије.

Овом приликом желим да се захвалим свом ментору, проф. др Срђану Вукмировићу и члановима комисије, др Тијани Шукиловић и др Весни Маринковић. Својим коментарима и сугестијама они су помогли при изради овог рада и учинили га љепшим и бољим.

# 1 Аберације на фотографијама

Приликом фотографисања могуће је да дође до неких нежељених резултата. Љубитељи фотографије сусрећу се са разним проблемима које произвођачи камера све успјешније превазилазе. Одступања онога што видимо од призора на фотографији називају се аберације (лат. *aberare* - одлутати) или дисторзије (лат. *distorsio* - искривљење, изопачење). Она углавном настају из оптичих разлога, нпр. због лоших или погрешно постављених сочива. Такве аберације се зато и називају оптичке аберације.

Неке од њих су:

- Хроматска аберација (дисторзија боја)
- Кома аберација
- Бураста дисторзија
- Јастучаста дисторзија.

(а) Бураста дисторзија

(б) Јастучаста дисторзија

(ц) Хроматска аберација

Слика 1: Примјери дисторзије на фотографијама (*www.nikon.rs*)

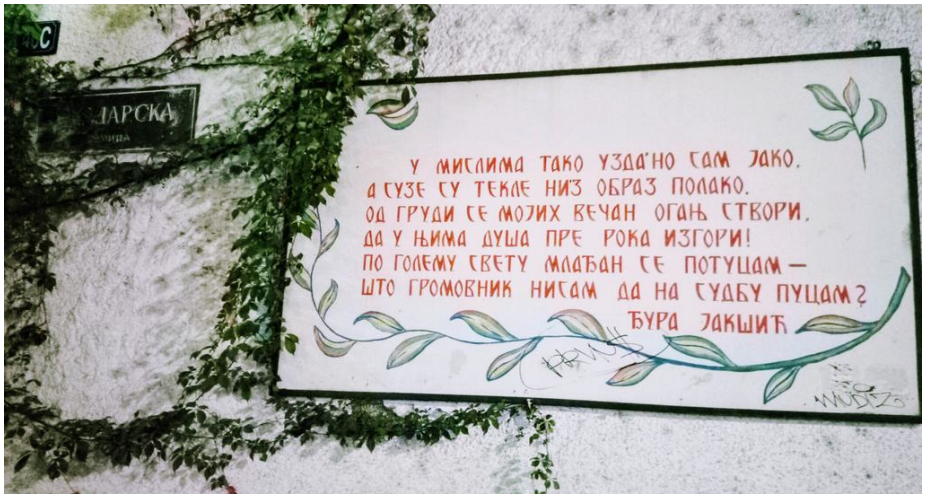
Хроматска и кома аберација настају због особина преламања свјетлости док на бурасту и јастучасту дисторзију (радијалне дисторзије) утиче искључиво квалитет сочива. Примјери неких од споменутих аберација могу се видјети на слици 1.

Све ове неправилности могу да се отклоне софтверски или коришћењем посебних додатака. Осим тога, данас се производе довољно добри објективи да оптичке аберације на фотографијама буду у потпуности непримјетне.

## 1.1 Пројективна дисторзија

Осим оптичке, приликом фотографисања јавља се и пројективна дисторзија. Она нема везе са сочивима тј. оптиком. Настаје када сензор који прима свјетлосне зраке (или раније филм) није паралелан равни у којој се налази објекат који фотографишемо (слика 2).

Ова појава се озбиљно разматра још од ренесансе. У сликарству је од тада позната као преспектива. Примјећено је да се у људском видном пољу тродимензионални свијет који нас окружује не понаша по правилима еуклидске геометрије, већ предмети изгледају све мањи што су даљи од ока посматрача. Паралелне прве (нпр. шине) могу да се сијеку на хоризонту.

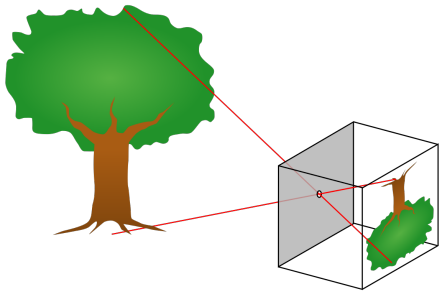



Слика 2: Примјер пројективне дисторзије. Правоугана плоча на фотографији нема облик правоугаоника.

Када фотографишемо неки објекат, а раван у којој се он налази није паралелна равни фотоапарата (сензора), жељена раван ће се на раван фотографије пресликати неким пројективним пресликавањем. Дакле, да бисмо могли да говоримо о отклањању пројективне дисторзије, прво се морамо упознати са геометријом која објашњава њено настајање. Зато ће у наредним поглављима бити ријечи о централној пројекцији и пројективној геометрији.

## 2 Централна пројекција као модел фотоапарата

Још од давнина, људи су примјетили да у мрачној просторији може да се види слика неког предмета из спољашњости. То се дешава уколико кроз малу рупу унутра допиру свјетлосни зраци. Међутим, слика предмета пројектованог на тај начин је заротирана за 180° (тј. предмет видимо окренут наопачке) и доста нејасна (слика 3). Познато је да су то Кинези уочили још прије више од 3500 година. Касније је та просторија у Европи прозвана **камера опскура** (лат. *camera obscura* - мрачна соба). Претече данашње фотографије настале су управо захваљујући мрачној комори у којој су помоћу разних фотоосјетљивих супстанци забиљежени „отисци" спољног свијета.



Слика 3: Камера опскура (*www.wikipedia.org*)

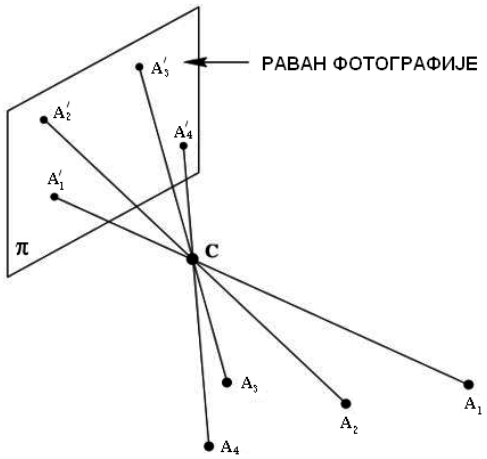
Овај феномен представља најједноставнији облик фотоапарата и назива се још и пинхол модел (*pinehole* - игличаста рупа). По том принципу функционишу и савремени фотоапарати само што је одавно мрачна кутија замијењена комплекснијом апаратуром, а умјесто просте „рупице" свјетлост улази кроз читав низ посебно дизајнираних сочива, па је слика много јаснија и вјеродостојнија.

Сви фотоапарати, почев од камере опскуре па до данас, као и људско око, функционишу по једноставном математичком моделу. Посматрани предмет се на сензор (некада зид мрачне собе) преноси **централном пројекцијом** као што је приказано на слици 4.

**Дефиниција 1.** *Нека је задата раван $\pi$ и тачка $C$ ван ње.* ***Централна пројекција*** *је пресликавање које сваку тачку простора $A \neq C$ слика у тачку $A'$ равни $\pi$ тако да важи $A' = \pi \times CA$. Тачка $C$ назива се* ***центар пројекције****, а раван $\pi$* ***пројекцијска раван****.*

**Напомена 1.** *Симбол $\times$ из претходне дефиниције је ознака за пресјек два геометријска објекта (у конкретном случају, пресјек праве и равни).*

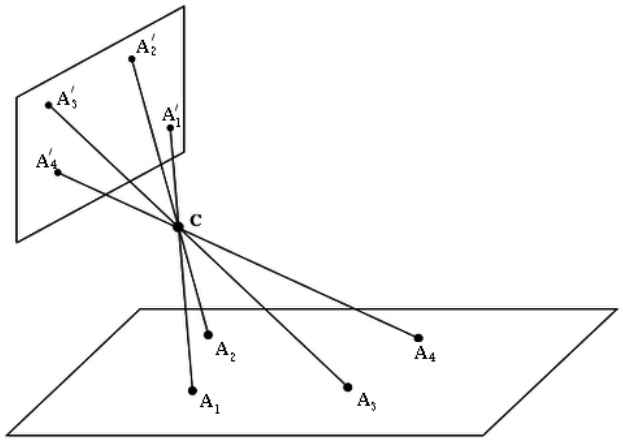
Централна пројекција није пројективно пресликавање, није чак ни бијекција. Међутим, у неким специјалним случајевима, она има све особине




Слика 4: Централна пројекција као модел фотоапарата (*Hartley R.*, *Zisserman A.*)

пројективног пресликавања. Баш ти случајеви ће нам бити посебно значајни јер се пројективна дисторзија, као и њено отклањање, темељи управо на њима.

О пројективној геометрији и пројективним пресликавањима ће бити више ријечи у главама 3 и 4. Засад је довољно знати да су пројективна пресликавања бијекције које чувају колинеарност.

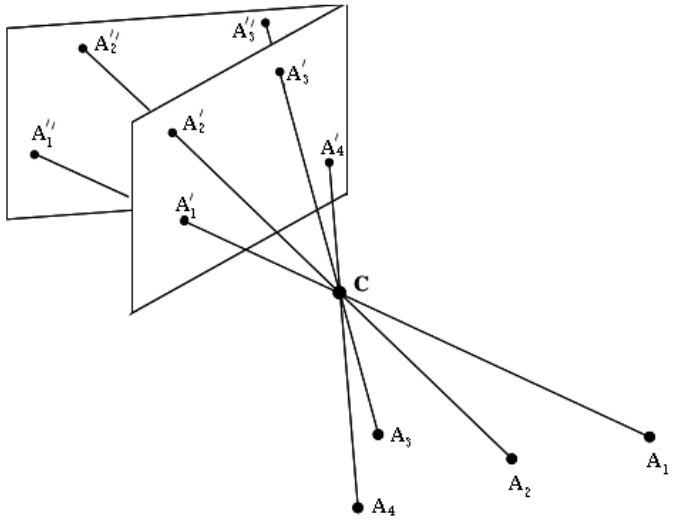



Слика 5: Централна пројекција једне равни (*Hartley R.*, *Zisserman A.*)

Уколико посматрамо рестрикцију централне пројекције на једну раван

простора, то пресликавање јесте пројективно. Јединствено пројективно пресликавање повезује четири копланарне тачке простора и исте те тачке на фотографији (слика 5).

Такође, ако четири тачке простора (не морају бити копланарне) пресликамо централном пројекцијом са истим центром на двије различите равни (слика 6), може се пронаћи пројективно пресликавање између тих слика.

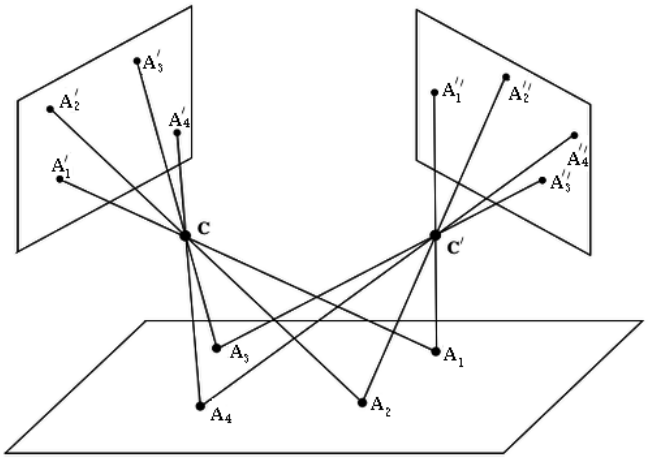



Слика 6: Двије централне пројекције са истим центром (*Hartley R., Zisserman A.*)

Ова чињеница се користи приликом прављења панорама. Центар фотоапарата се не мијења, прави се неколико фотографија док ротирамо фотоапарат (тако да су пројекцијске равни различите). Зато се сваке двије сусједне фотографије могу надовезати помоћу пројективних пресликавања.

Чак и ако промијенимо центар централне пројекције (центар фотоапарата), али посматрамо тачке које јесу копланарне, између тако добијених слика поново постоји пројективно пресликавање (слика 7). Ово је последица чињенице да је композиција два пројективна пресликавања пројективно пресликавање.

Ове чињенице су значајне за компјутерски вид (енг. *computer vision*) јер је са пројективним пресликавањима много лакше радити него са централном пројекцијом.




Слика 7: Случај када се фотографише иста раван из два центра (*Hartley R., Zisserman A.*)

# 3 Геометрија пројективне равни

Како ће у овом раду бити ријечи о отклањању пројективне дисторзије, потребно је увести неке основне појмове пројективне геометрије.

Пројективну раван можемо да схватимо на више начина. Један од њих је помоћу хомогених координата.

## 3.1 Хомогене координате

**Дефиниција 2.** ***Реална пројективна раван*** $\mathbb{RP}^2$ *је скуп тачака облика* $(x_1 : x_2 : x_3)$*, гдје су* $x_1, x_2$ *и* $x_3$ *реални бројеви такви да нису истовремено сви нула. То јест,*

$$\mathbb{RP}^2 = \left\{ (x_1 : x_2 : x_3) \,|\, x_1, x_2, x_3 \in \mathbb{R}, x_1^2 + x_2^2 + x_3^2 \neq 0 \right\}.$$

Дакле, тачке пројективне равни записујемо помоћу три координате. Тачки $A(x_1 : x_2 : x_3)$ одговара вектор $\vec{A} = (x_1, x_2, x_3)$ који зовемо **вектор представник** тачке $A$.

Праву пројективне равни представљамо једначином

$$p : ax_1 + bx_2 + cx_3 = 0.$$

Њене координате су уређена тројка $[a : b : c]$. Вектор $\vec{p} = (a, b, c)$ је вектор представник праве $p$.

Будући да $\vec{p} = (a, b, c)$ и $\vec{q} = (\alpha a, \alpha b, \alpha c), (\alpha \neq 0)$ представљају исти вектор, закључујемо да координате праве нису јединствене. Множењем координата скаларом различитим од нуле добићемо више представљања једне исте праве. Због тога ове координате називамо **хомогене координате**.

Аналогно, све ово важи и за тачке, тј. тачку $A(x_1 : x_2 : x_3)$ можемо записати и као $A(\alpha x_1 : \alpha x_2 : \alpha x_3)$ при чему је $\alpha$ произвољан реалан број различит од нуле.

У пројективној геометрији (као и у афиној и еуклидској), тачка $A$ ће припадати правој $p$ ако и само ако њене координате задовољавају једначину праве. Однос тачке и праве можемо испитати и помоћу њихових вектора представника (користећи скаларни производ) о чему говори следећа теорема.

**Теорема 1.** *Нека су $\vec{p}$ и $\vec{A}$ вектори представници праве $p$ и тачке $A$, редом. Тачка $A$ припада правој $p$ ако и само ако важи*

$$\vec{p} \cdot \vec{A} = \vec{A} \cdot \vec{p} = 0. \tag{1}$$

*Доказ.* Нека је једначина праве $p : ax_1 + bx_2 + cx_3 = 0$, а тачка има координате $A(x : y : z)$. Дакле, вектори представницу су дати са $\vec{p} = (a, b, c)$, а $\vec{A} = (x, y, z)$. Тачка $A$ ће припадати датој правој ако и само ако важи

$$ax + by + cz = 0.$$

Векторски записано, то је управо $\vec{p} \cdot \vec{A} = 0$. Такође важи и $\vec{p} \cdot \vec{A} = \vec{A} \cdot \vec{p}$ због комутативности скаларног производа. Јасно је да једнакост (1) не зависи од вектора представника јер је $(\alpha\vec{p}) \cdot (\beta\vec{A}) = \alpha\beta(\vec{p} \cdot \vec{A})$, тако да је тврђење доказано. □

Користећи векторе представнике, можемо одредити праву која садржи двије задате тачке, као и пресјек двије праве.

**Теорема 2.** *Вектор представник праве $p$ која садржи тачке $A$ и $B$ дат је са $\vec{p} = \vec{A} \times \vec{B}$.*

**Напомена 2.** *Симбол $\times$ је у овом случају ознака за векторски производ.*

*Доказ.* (Геометријски) Права $p$ садржи тачке $A$ и $B$ па мора да буде $\vec{p} \cdot \vec{A} = 0$ и $\vec{p} \cdot \vec{B} = 0$. То значи да је вектор $\vec{p}$ нормалан и на вектор $\vec{A}$ и на вектор $\vec{B}$. Зато $\vec{p}$ и јесте векторски производ вектора $\vec{A}$ и $\vec{B}$. □

*Доказ.* (Алгебарски) Тражимо вектор $\vec{p} = (a, b, c)$ за који важи

$$x_a a + y_a b + z_a c = 0 \tag{2}$$

$$x_b a + y_b b + z_b c = 0 \tag{3}$$

при чему је $\vec{A} = (x_a, y_a, z_a)$, а $\vec{B} = (x_b, y_b, z_b)$. Једначине (2) и (3) представљају услов да дате тачке припадају правој $p$. Када ријешимо систем (2)-(3) добићемо вектор $\vec{p}$, а самим тим и координате праве $p$. Систем има три непознате ($a$, $b$ и $c$), а само двије једначине. Међутим, то није проблем

јер вектор представник праве (као и тачке) није јединствен.

Кренимо сада да рјешавамо систем.

$$\begin{array}{l} x_a a + y_a b + z_a c = 0 \\ x_b a + y_b b + z_b c = 0 \end{array} \begin{array}{l} \\ \leftarrow + \end{array} \begin{array}{l} \\ \rightarrow -\frac{x_b}{x_a} \end{array}$$

$$\begin{array}{lll} x_a a + & y_a b + & z_a c = 0 \\ & \left(y_b - \dfrac{x_b y_a}{x_a}\right) b + & \left(z_b - \dfrac{x_b z_a}{x_a}\right) c = 0 \end{array}$$

Одавде је лако израчунати да је $c = \dfrac{x_b y_a - x_a y_b}{x_a z_b - x_b z_a} \cdot b$. Затим се, уз мало рачуна, добије да је $a = \dfrac{y_b z_a - y_a z_b}{x_a z_b - x_b z_a} \cdot b$. Тако смо нашли читаву класу вектора представника праве $p$: $\vec{p} = \left(\dfrac{y_b z_a - y_a z_b}{x_a z_b - x_b z_a} \cdot b, b, \dfrac{x_b y_a - x_a y_b}{x_a z_b - x_b z_a} \cdot b\right)$, $b \in \mathbb{R}$. Фиксирањем било ког реалног броја $b$, добићемо један вектор представник наше праве. Згодно је узети $b = x_b z_a - x_a z_b$. Тада права $p$ има координате $[y_a z_b - y_b z_a : x_b z_a - x_a z_b : x_a y_b - x_b y_a]$.

Са друге стране, израчунајмо шта је векторски производ $\vec{A} \times \vec{B}$.

$$\begin{vmatrix} i & j & k \\ x_a & y_a & z_a \\ x_b & y_b & z_b \end{vmatrix} = (y_a z_b - y_b z_a)i + (x_b z_a - x_a z_b)j + (x_a y_b - x_b y_a)k$$

Дакле, $\vec{A} \times \vec{B} = (y_a z_b - y_b z_a, x_b z_a - x_a z_b, x_a y_b - x_b y_a)$, а то јесте вектор представник праве $p$. Из свега наведеног слиједи да је $\vec{p} = \vec{A} \times \vec{B}$, што је требало доказати.

□

На исти начин може се доказати и следећа теорема.

**Теорема 3.** *Нека су вектори $\vec{p}$ и $\vec{q}$ вектори представници правих $p$ и $q$ редом. Нека је $A$ тачка пресјека ових правих. Тада важи: $\vec{A} = \vec{p} \times \vec{q}$.*

**Напомена 3.** *Из наведене теореме се јасно види зашто се иста ознака користи за векторски производ и пресјек геометријских објеката.*

Сада знамо да нађемо пресјек правих и праву одређену двијема тачкама користећи векторски производ. Оно што је важно примјетити (као последицу претходне теореме) је да се у пројективној равни сваке двије праве сијеку.

## 3.2 Проширена афина раван

Хомогеним координатама можемо представити и тачке афине равни. На примјер, тачка $A(x, y)$ има хомогене координате $A(x : y : 1)$, али одговара јој и било која уређена тројка $(x_1 : x_2 : x_3)$ за коју важи да је $x = \frac{x_1}{x_3}$ и $y = \frac{x_2}{x_3}$. Права у афиној равни

$$p : ax + by + c = 0$$

има хомогене координате $[a : b : c]$.

**Примјер 1.** *Дата је права $p : 3x + 2y - 2 = 0$ и тачка $A(0, 1)$ у афиној равни. Наћи њихове хомогене координате и показати да тачка $A$ припада правој $p$.*

*Рјешење.* Права $p$ има хомогене координате $[3 : 2 : -2]$, а тачка $A$ $(0 : 1 : 1)$. Пошто је $3 \cdot 0 + 2 \cdot 1 + (-2) \cdot 1 = 0$, дата тачка припада правој $p$. △

Јасно је да свакој тачки афине равни одговара нека тачка пројективне равни (исто важи и за праве), али обрнуто не важи. Пројективних тачака има више. Пројективне тачке облика $X(x_1 : x_2 : 0)$ не могу да се запишу афиним координатама. Њих називамо **бесконачно далеке тачке**. Све оне припадају истој правој пројективне равни $p : x_3 = 0$ (правој са координатама $[0 : 0 : 1]$). Називамо је **бесконачно далека права**. Као ни тачке од којих је сачињена, ни њу не можемо представити у афиној равни.

**Примјер 2.** *Одредити пресјек правих $p : x_1 - x_3 = 0$ и $q : x_1 + 2x_3 = 0$.*

*Рјешење.* Пресјек правих у пројективној равни одређујемо помоћу векторског производа њихових вектора представника. $\vec{p} = (1, 0, -1)$, а $\vec{q} = (1, 0, 2)$.

$$\begin{vmatrix} i & j & k \\ 1 & 0 & -1 \\ 1 & 0 & 2 \end{vmatrix} = 0 \cdot i - 3 \cdot j + 0 \cdot k$$

Дакле, праве $p$ и $q$ сијеку се у тачки $A$ чији је вектор представник $\vec{A} = (0, -3, 0)$, тј. $A(0 : -3 : 0)$.

Ако посматрамо дате праве у афиној равни, јасно је да су оне паралелне (обје су паралелне $y$ оси).

$$p : x = 1$$

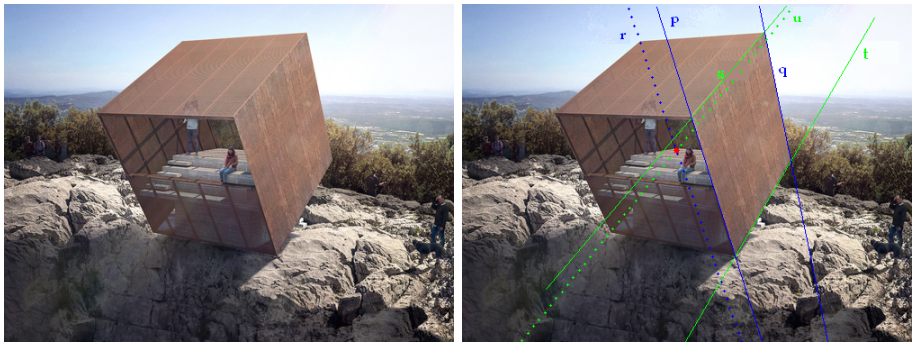
$$q : x = -2$$

Значи, њихов пресјек у афиној равни не постоји, али у проширеној афиној равни, то је бесконачно далека тачка $A(0 : -3 : 0)$. △

Сада можемо замислити афину раван и једну издвојену праву која јој не припада (бесконачно далека права). То је још један начин да разумијемо пројективну раван: као проширену афину раван. Ипак, у проширеној афиној равни, бесконачно далеке тачке су другачије, издвојене, док су у пројективној равни све тачке (као и праве) равноправне.

**Примјер 3.** *Одређивање непознатог тјемена.*

Понекад је потребно да знамо координате тјемена неке фигуре које се на фотографији не види. На примјер, то је корисно при $3D$ реконструисању објекта помоћу фотографија. На слици 8 (а) приказана је коцка којој се једно тјеме не види (заклоњено је остатком коцке). Уколико је потребно да знамо координате тог тјемена, можемо их одредити уз помоћ закона који важе у пројективној геометрији.




(а) Слика 1 (*www.archdaily.com*) (б) Слика 2

Слика 8: Одређивање координата тјемена коцке

Слика 8 (б) представља илустрацију идеје на који начин то можемо урадити. Назовимо плаве линије $p$ и $q$, зелене $s$ и $t$, непознату плаву линију $r$, а зелену $u$. Хомогене координате правих $p, q, s$ и $t$ лако је израчунати јер видимо тјемена коцке која их одређују (па самим тим можемо да очитамо њихове координате), а користећи векторски производ. У конкретном случају (слика 8), те координате су следеће:

$$p[154 : -48 : -33222]$$

$$q[143 : -29 : -47149]$$

$$s[95 : 81 : -34431]$$

$$t[106 : 62 : -49478].$$

До рјешења ћемо доћи захваљујући чињеници да се праве $p$, $q$ и $r$ сијеку у истој тачки јер су у стварности паралелне (да су којим случајем биле паралелне и на фотографији свакако би се сјекле у истој бесконачно далекој тачки). Пресјек правих $p$ и $q$ налазимо као векторски производ њихових вектора представника.
$\vec{p} \times \vec{q} = \vec{A} = (-1299714, -2510200, -2398)$

Како сада знамо координате двије тачке праве $r$ (тачка $A$ и једно тјеме коцке које се види), добијамо и њене координате.
$r[2411882 : -827308 : -441221126]$

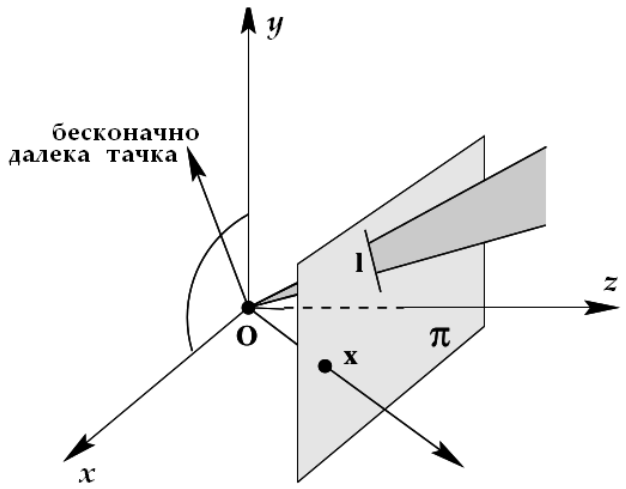
На исти начин добијамо и координате праве $u$.
$u[-1732812 : -1430852 : 646186724]$

Тражена тачка $M$ је пресјек правих $r$ и $u$ и има координате
$M(238 : 162 : 1)$, тј. на фотографији јој одговара пиксел са координатама $(238, 162)$.

**Напомена 4.** *Будућу да све наведено важи и за тачке и праве у еуклидској равни, умјесто израза „проширена афина раван", може се користити и „проширена еуклидска раван".*

## 3.3 Скуп правих и равни у еуклидском простору

Будући да тачке у пројективној равни записујемо помоћу три координате, можемо да их замишљамо и као прамен правих простора $\mathbb{R}^3$ које пролазе кроз координатни почетак (слика 9). Тада би праве пројективне равни могле да се виде као равни простора $\mathbb{R}^3$ које садрже координатни почетак. Раван $x_3 = 0$, тј. $O_xO_y$ раван, представља бесконачно далеку праву, а све праве које јој припадају бесконачно далеке тачке.




Слика 9: Праве и равни еуклидског простора као модел тачака и правих пројективне равни (*Hartley R.*, *Zisserman A.*)

## 3.4 Аксиоматски приступ

Пројективна раван може се увести и схватити и чисто аксиоматски. Пројективна раван је одређена само условима да двије тачке одређују тачно једну праву и да се сваке двије праве сијеку у јединственој тачки (то је

главна разлика у односу на еуклидску раван). Потребно је још само обезбиједити да буде недегенерисана. Зато се уводи и аксиома која каже да постоје четири тачке у општем положају (видјети дефиницију 4).

Иако на први поглед дјелује да оваквим заснивањем ништа није речено и прецизирано, ове три аксиоме потпуно дефинишу пројективну раван о којој је било ријечи у претходним поглављима.

# 4 Пројективна пресликавања равни

**Дефиниција 3.** ***Пројективно пресликавање*** *је бијекција пројективне равни $\mathbb{R}\mathcal{P}^2$ на себе (или неку другу пројективну раван) при чему се колинеарне тачке сликају у колинеарне тачке.*

Зато се може представити помоћу $3 \times 3$ матрице $P$ која је регуларна, тј. $det(P) \neq 0$. Тачка $A(x_1 : x_2 : x_3)$ слика се у тачку $A'(x_1' : x_2' : x_3')$ на следећи начин:

$$\lambda \begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{4}$$

што краће записујемо $\lambda A' = PA$. При томе је $\lambda \in \mathbb{R} \setminus \{0\}$ произвољан број који указује на то да су координате хомогене.

Будући да су пројективна пресликавања једнозначно одређена регуларним $3 \times 3$ матрицама, јасно је да она чине групу у односу на операцију композиције пресликавања (јер матрице које их одређују чине групу у односу на множење матрица). Због тога знамо и да је композиција два пројективна пресликавања пројективно пресликавање.

Пошто је пројективно пресликавање $P$ бијекција, увијек постоји инверзно пресликавање $Q$ чија је матрица $Q = P^{-1}$. Тако да важи $\lambda A = QA'$.

Пројективним пресликавањем (помоћу његове матрице) можемо пресликати и праве пројективне равни. Уколико тачка $A$ припада правој $p$ ($A \in p$), важиће и $A' \in p'$, при чему је $A'$ слика тачке $A$, $p'$ слика праве $p$ при пресликавању $P$.

Значи, $A' = PA$. Знамо да је $A^T p = 0$ (јер $A$ припада правој $p$) као и да важи $(A')^T p' = 0$. Одавде имамо да је

$$0 = A^T p = (P^{-1}A')^T p = (A')^T (P^{-1})^T p.$$

Дакле, $p' = (P^{-1})^T p$, што је формула пресликавања праве.

Због хомогености, матрица $P$ и $\lambda P$ дефинишу исто пресликавање. То значи да је само осам елемената матрице $P$ независно. Пошто вектор представник тачке има два међусобно независна елемента, слиједи да су потребна четири пара тачака да бисмо одредили матрицу пројективног пресликавања.

**Дефиниција 4.** *Четири тачке пројективне равни су* ***у општем положају*** *ако међу њима не постоји ни једна тројка колинеарних тачака.*

**Теорема 4.** *(Основна теорема пројективне геометрије) Постоји пројективно пресликавање $F$ које четири тачке $A, B, C, D$ које су у општем положају слика редом у тачке $A', B', C'$ и $D'$ (такође у општем положају).*

*Доказ.* Доказаћемо прво да постоји пројективно пресликавање које слика базне тачке $A_0(1:0:0), B_0(0:1:0), C_0(0:0:1), D_0(1:1:1)$ у четири дате тачке $A(a_1:a_2:a_3), B(b_1:b_2:b_3), C(c_1:c_2:c_3)$ и $D(d_1:d_2:d_3)$. Нека су $\alpha, \beta$ и $\gamma$ реални бројеви различити од нуле такви да важи

$$D = \alpha A + \beta B + \gamma C.$$

Овакви бројеви постоје јер су дате тачке у општем положају па су и свака три вектора представника међусобно линеарно независна. Тражено пресликавање $P$ има матрицу $P = \begin{pmatrix} \alpha A & \beta B & \gamma C \end{pmatrix}$.

Заиста,

$$PA_0 = \begin{pmatrix} \alpha a_1 & \beta b_1 & \gamma c_1 \\ \alpha a_2 & \beta b_2 & \gamma c_2 \\ \alpha a_3 & \beta b_3 & \gamma c_3 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \alpha a_1 \\ \alpha a_2 \\ \alpha a_3 \end{pmatrix} = \alpha \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}.$$

Дакле, $A$ јесте слика базне тачке $A_0$.

Слично се покаже и да су $B$ и $C$ слике тачака $B_0$ и $C_0$ редом.

Провјеримо сада шта је слика тачке $D_0$.

$$\begin{pmatrix} \alpha a_1 & \beta b_1 & \gamma c_1 \\ \alpha a_2 & \beta b_2 & \gamma c_2 \\ \alpha a_3 & \beta b_3 & \gamma c_3 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha a_1 + \beta b_1 + \gamma c_1 \\ \alpha a_2 + \beta b_2 + \gamma c_2 \\ \alpha a_3 + \beta b_3 + \gamma c_3 \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix}$$

На исти начин можемо да пронађемо пресликавање $Q$ које базне тачке слика у $A', B', C'$ и $D'$ редом.

Коначно, тражено пресликавање $F$ је композиција пресликавања $P^{-1}$ и $Q$, а његова матрица једнака је $QP^{-1}$. □

**Примјер 4.** *Одредити слику квадрата $ABCD$ при пројективном пресликавању задатом матрицом $P = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 1 \end{pmatrix}$. Тјемена квадрата имају следеће координате: $A(0,1)$, $B(0,0)$, $C(1,0)$, $D(1,1)$.*

*Рјешење.* Да бисмо одредили слику квадрата, довољно је пресликати његова тјемена. Зато их је потребно представи хомогеним координатама. Потражимо прво слику тачке $A$.

$$\begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \\ 2 \end{pmatrix}$$

Сада знамо да је њена слика тачка $A'(0:2:2) = (0:1:1)$. Значи, тачка $A$ се пресликала у себе. Лако се добије да исто важи за тачке $B$ и $D$. Њихове слике су оне саме: $B'(0:0:1)$, а $D'(1:1:1)$. Слика тачке $C$ биће $C'(2:0:1)$. Дакле, квадрат $ABCD$ се пресликао у правоугли трапез $A'B'C'D'$. △

Из датог примјера се види да пројективна пресликавања не чувају дужине нити углове (што је случај са еуклидским), па чак ни паралелност. Једине инваријанте пројективних пресликавања су колинеарност и конкурентност. Слика квадрата може бити трапез (или било који други четвороугао), а произвољан четвороугао се може пресликати у квадрат.

# 5 Афина пресликавања

Као што је већ напоменуто, пројективна пресликавања чине групу. Афина пресликавања су подгупа ове групе. Дакле, свако афино пресликавање може се представити као пројективно док обрнуто не важи.

У општем случају, афино пресликавање је облика:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, det(a_{ij}) \neq 0.$$

Њиме се тачка $M(x, y)$ слика у тачку $M'(x', y')$. Пресликавање можемо записати и на следећи начин (краће):

$$M' = A \cdot M + b.$$

Колоне матрице $A$ јесу слике базних вектора, а вектор колона $b$ је слика координатног почетка.

Приказано у хомогеним координатама, афино пресликавање изгледа овако:

$$\lambda \begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}. \tag{5}$$

**Теорема 5.** *Нека је пројективно пресликавање $P$ задато својом матрицом*

$$P = \begin{pmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{pmatrix}. \tag{6}$$

*Права $p$ одређена последњом врстом матрице $P$ ($p[p_{31} : p_{32} : p_{33}]$)се слика у бесконачно далеку праву $x'_3 = 0$.*

*Рјешење.* Једначина праве $p$ је $p_{31}x_1 + p_{32}x_2 + p_{33}x_3 = 0$, а из формуле пресликавања (4) имамо да важи:

$$x'_3 = p_{31}x_1 + p_{32}x_2 + p_{33}x_3.$$

Дакле, за све тачке $A(x_1 : x_2 : x_3)$ које припадају правој $p$ трећа координата слике $A'$ биће једнака нули, тј. $x'_3 = 0$. То значи да се свака тачка праве $p$ слика у бесконачно далеку тачку па се права $p$ слика у бесконачно далеко праву $x'_3 = 0$. $\triangle$

Аналогно се доказује и обрат овог тврђења: да права која се пројективним пресликавањем (6) слика у бесконачно далеку праву има координате $[p_{31} : p_{32} : p_{33}]$.

Из свега наведеног и једначине (5) закључујемо да су афина пресликавања заправо подгрупа пројективних пресликавања која бесконачно далеку праву $x_3 = 0$ сликају у њу саму.

Веза између пројективних и афиних пресликавања је важна јер подаци са којима радимо (нпр. пиксели приликом обраде дигиталних фотографија) ријетко кад имају хомогене координате.

Зато је добро што можемо да преводимо координате из афиних у хомогене и обрнуто као и да користимо пресликавања која су у том тренутку подеснија (афина или пројективна).

Афина пресликавања (за разлику од пројективних) чувају паралелност, тј. паралелне праве се њима сликају у паралелне праве. Тако је слика квадрата при афином пресликавању увијек паралелограм (али зато не мора бити правоугли). Важи и да се за свака два троугла може наћи афино пресликавање (тим прије пројективно) које слика један у други.

# 6 Алгоритми за одређивање пројективног пресликавања

Због саме теме овог рада (а и уопште у рачунарској графици) одређивање пројективног пресликавања заузима кључно мјесто. Постоји више алгоритама помоћу којих долазимо до матрице пројективног пресликавања. Пошто је пројективно пресликавање одређено са четири пара тачака, природно је да постоји алгоритам који се базира на овој чињеници.

## 6.1 Наивни алгоритам

Основни и најинтуитивнији алгоритам за одређивање пројективног пресликавања назива се наивни алгоритам.

Идеја је да функција прими четири пара тачака у општем положају (оригинали и слике), а као излаз врати матрицу пресликавања које је њима одређено. Како основна теорема пројективне геометрије тврди да постоји овакво пресликавање, њен доказ јесте алгоритам за проналажење истог.

Наивни алгоритам:

1. Сваку од унијетих тачака представити помоћу хомогених координата
2. За тачке $A, B, C$ и $D$ (оригинали) пронаћи реалне бројеве $\alpha, \beta$ и $\gamma$ тако да важи
$$D = \alpha A + \beta B + \gamma C \tag{7}$$
3. Формирати матрицу $P = \begin{pmatrix} \alpha A & \beta B & \gamma C \end{pmatrix}$
4. За тачке $A', B', C'$ и $D'$ (слике) пронаћи реалне бројеве $\alpha', \beta'$ и $\gamma'$ тако да важи $D' = \alpha' A' + \beta' B' + \gamma' C'$
5. Формирати матрицу $Q = \begin{pmatrix} \alpha' A' & \beta' B' & \gamma' C' \end{pmatrix}$
6. Излазна матрица има облик $QP^{-1}$.

Да овај алгоритам даје жељени резултат, гарантује нам доказ теореме 4. Једино остаје питање: како најлакше пронаћи бројеве $\alpha, \beta$ и $\gamma$ (на исти начин се проналазе и $\alpha', \beta'$ и $\gamma'$)? Наравно, можемо ријешити систем једначина (Гаусовом методом или помоћу детерминанти), али је у већини програмских језика далеко једноставније користити матрице и њихове особине. Из једначине (7) видимо да важи

$$D = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}.$$

Дакле, вектор колона која садржи тражене коефицијенте се добија на следећи начин:

$$\begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = M^{-1} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix}$$

где је $M = \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{pmatrix}$ матрица чије су колоне хомогене координате тачака $A, B$ и $C$.

## 6.2 *DLT* алгоритам

Некада, из практичних разлога, користимо више од четири пара тачака за одређивање пројективног пресликавања.

То суштински значи да треба да ријешимо предетерминисан систем (систем који има више једначина него непознатих). Мала је вјероватноћа да су тачке које користимо изабране довољно прецизно и да сви парови (оригинали и слике) дефинишу једно исто пројективно пресликавање. Заправо, у пракси најчешће и нису, тј. $\lambda A'_i \neq PA_i$ за свако $i = 1, \ldots, n$. Зато се

користе различите методе које приближно (најбоље могуће) дају рјешење траженог система.

Дакле, непознати су елементи матрице пројективног пресликавања (њих девет, од којих је осам независно), а имамо више од четири пара тачака што значи више од осам једначина које треба да буду задовољене. Једно од рјешења овог проблема даје $DLT$ алгоритам (енг. *direct linear transformation*) који се непосредно ослања на $SVD$ декомпозицију матрице. Следеће тврђење игра важну улогу у разумијевању овог алгоритма.

**Теорема 6.** *Нека су $A(x_1 : x_2 : x_3)$ и $A'(x_1' : x_2' : x_3')$ , $x_3' \neq 0$ одговарајуће тачке пројективног пресликавања равни чија је матрица $P = (p_{ij})$. Тада вектор $(p_{11}, p_{12}, p_{13}, p_{21}, p_{22}, p_{23}, p_{31}, p_{32}, p_{33})$ задовољава хомогени систем ранга 2 чија је матрица:*

$$\begin{pmatrix} 0 & 0 & 0 & -x_3'x_1 & -x_3'x_2 & -x_3'x_3 & x_2'x_1 & x_2'x_2 & x_2'x_3 \\ x_3'x_1 & x_3'x_2 & x_3'x_3 & 0 & 0 & 0 & -x_1'x_1 & -x_1'x_2 & -x_1'x_3 \end{pmatrix}. \tag{8}$$

Доказ теореме може се пронаћи у презентацији [1].

Матрицу (8) краће записујемо $\begin{pmatrix} 0^T & -x_3'A^T & x_2'A^T \\ x_3'A^T & 0^T & -x_1'A^T \end{pmatrix}$.

Када ово знамо, елементе матрице пресликавања добићемо као рјешење система одређеног паровима тачака које користимо. Они ће образовати матрицу

$$M = \begin{pmatrix} M_1 \\ M_2 \\ \vdots \\ M_n \end{pmatrix}. \tag{9}$$

При томе је

$$M_i = \begin{pmatrix} 0^T & -x_{i3}'A_i^T & x_{i2}'A_i^T \\ x_{i3}'A_i^T & 0^T & -x_{i1}'A_i^T \end{pmatrix}$$

матрица одређена једном кореспонденцијом $A_i \leftrightarrow A_i'$. Међутим, као што је већ наглашено, у пракси овај систем често нема рјешење (осим нула рјешења). Зато се у $DLT$ алгоритму користи $SVD$ декомпозиција матрице која алгебарским методама даје вектор најприближнији рјешењу.

### 6.2.1 $SVD$ декомпозиција матрице

$SVD$ декомпомпозиција (енг. *singular value decomposition*) је разлагање произвољне матрице на производ три матрице које имају специфична својства. Наиме, важи следеће:

Нека је $M$ матрица формата $n \times m$. Тада постоји јединствено представљање

$$M = UDV^T.$$

При томе је $U$ матрица формата $n \times m$, $V$ ортогонална матрица формата $m \times m$, а $D$ дијагонална матрица формата $m \times m$ и њене дијагоналне вриједности су позитивне и сортиране опадајуће. Додатно, $U$ је таква да важи $\|Ux\| = \|x\|$ за произвољан вектор $x$.

Ова декомпозиција има бројне примјене, а између осталог користи се и за одређивање приближног рјешења система линеарних једначина.

У овом конкретном случају, за одређивање матрице $P$ пројективног пресликавања, потребно је одредити рјешење система

$$M \begin{pmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{31} \\ p_{32} \\ p_{33} \end{pmatrix} = 0 \tag{10}$$

или краће $Mp = 0$, гдје је $M$ матрица задата једначином (9). Нула вектор јесте рјешење, али није од важности у овом случају (нула матрица не одређује ни једно пројективно пресликавање). Систем може да има јединствено рјешење, али (као што је већ речено) то углавном није случај. Тада покушавамо да нађемо приближно рјешење система. Питање остаје шта значи то „приближно"?

Алгебарски посматрано, то би било рјешење за које је норма вектора $Mp$ најмања, тј. тражићемо $p$ за које функција $\|Mp\|$ достиже минимум.

Вектор колону $p$ чине елементи матрице пројективног пресликавања. Самим тим, скалирање не утиче на резултат ($p$ и $\lambda p$ дефинисаће исто пресликавање). Зато можемо сматрати да важи $\|p\| = 1$.

Захваљујући $SVD$ декомпозицији, матрицу $M$ можемо записати као $UDV^T$. Тада проблем одређивања минимума постаје знатно лакши. Пошто је $\|UDV^Tp\| = \|DV^Tp\|$, $\|Mp\| = \|DV^Tp\|$. Означимо $x = V^Tp$. Како важи $\|V^Tp\| = \|p\| = 1$ ($V$ је ортогонална), добијамо да је $\|x\| = 1$.

Остаје да нађемо $x$ за које се достиже минимум $\|Dx\|$. Међутим, $D$ је дијагонална матрица са опадајућим дијагоналним вриједностима, па тражени минимум задовољава вектор

$$x = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}^T.$$

Матрица $V$ је ортогонална и зато, да би важило $V^Tp = x$, $p$ мора да буде последња колона матрице $V$.

Дакле, рјешење $p$ које минимизује алгебарску грешку система (10) је последња колона матрице $V$ ($M = UDV^T$).

### 6.2.2 Алгоритам

*DLT* алгоритам прима $n$ парова тачака као улаз, а као излаз враћа матрицу пројективног пресликавања.

*DLT* алгоритам:

1. Сваку од унијетих тачака представити помоћу хомогених координата
2. За сваки пар (оригинал и слика) одредити $2 \times 9$ матрицу (8)
3. Од тако добијених матрица формирати матрицу $M$ формата $2n \times 9$. То је матрица система који рјешавамо
4. Одредити $SVD$ декомпозицију матрице $M$
5. Од последње колоне матрице $V$ направити матрицу формата $3 \times 3$. То је матрица $P$ траженог пројективног пресликавања.

### 6.2.3 Алгебарска и геометријска грешка

*DLT* алгоритам није геометријске природе. Он је направљен тако да смањи алгебарску грешку (тј. норму $\|Mp\|$) при рјешавању предетерминисаног система.

Свака од коресподенција $A_i \leftrightarrow A_i'$ које користимо за одређивање пројективног пресликавања (ако има више од четири пара тачака) доприноси порасту грешке рјешења које ћемо добити. Зато свакој кореспонденцији одговара вектор парцијалне грешке $\epsilon_i$. Норма овог вектора назива се **алгебарско растојање**.

$$d_{alg}(A_i', PA_i)^2 = \|\epsilon_i\|^2 = \left\| \begin{pmatrix} 0^T & -x_{i3}' A_i^T & x_{i2}' A_i^T \\ x_{i3}' A_i^T & 0^T & -x_{i1}' A_i^T \end{pmatrix} p \right\|^2$$

Парцијалне грешке доприносе **алгебарској грешци** система чији је вектор $\epsilon = Mp$.

$$\sum_i d_{alg}(A_i', PA_i)^2 = \sum_i \|\epsilon_i\|^2 = \|Mp\|^2 = \|\epsilon\|^2$$

Будући да се бавимо геометријским проблемом, може се поставити питање зашто је *DLT* алгоритам направљен тако да минимизује алгебарско, а не геометријско растојање. Ипак, испоставља се да методе које смањују алгебарску грешку, уз одговарајуће нормализације, дају веома добро рјешење. Осим тога, велика предност је што се њима добија јединствено рјешење, а сами алгоритми су ефикасни (мале сложености). На примјер, алгоритам који одређује $SVD$ декомпозицију матрице за потребе *DLT* алгоритма (тј. када није неопходно израчунати матрицу $U$) је линеарне сложености.

Додатно, алгебарска и геометријска грешка су на извијестан начин повезане што није тешко израчунати. Геометријска грешка представља збир квадрата растојања свих парова тачака $(A_i', PA_i)$. Дакле, она износи

$$\sum_i d(A'_i, PA_i)^2.$$

Означимо са $A'_i$ и $\tilde{A}'_i$ дату слику тачке и слику тачке добијене пресликавањем $P$ које је резултат $DLT$ алгоритма ($\tilde{A}'_i = PA_i$), редом.

Тада је

$$d_{alg}(A'_i, \tilde{A}_i) = \left\| \begin{pmatrix} 0^T & -x'_{i3}A_i^T & x'_{i2}A_i^T \\ x'_{i3}A_i^T & 0^T & -x'_{i1}A_i^T \end{pmatrix} p \right\| = \left\| \begin{pmatrix} -x'_{i3}\tilde{x'_{i2}} + x'_{i2}\tilde{x'_{i3}} \\ x'_{i3}\tilde{x'_{i1}} - x'_{i1}\tilde{x'_{i3}} \end{pmatrix} \right\| =$$

$$= \sqrt{(-x'_{i3}\tilde{x'_{i2}} + x'_{i2}\tilde{x'_{i3}})^2 + (x'_{i3}\tilde{x'_{i1}} - x'_{i1}\tilde{x'_{i3}})^2}. \tag{11}$$

Са друге стране, геометријско растојање дато је са:

$$d(A'_i, \tilde{A}'_i) = \sqrt{\left(\frac{x'_{i1}}{x'_{i3}} - \frac{\tilde{x'_{i1}}}{\tilde{x'_{i3}}}\right)^2 + \left(\frac{x'_{i2}}{x'_{i3}} - \frac{\tilde{x'_{i2}}}{\tilde{x'_{i3}}}\right)^2}. \tag{12}$$

Примјетимо да смо хомогене координате морали записати као еуклидске (афине) да бисмо израчунали еуклидско растојање.

Из једначина (11) и (12) добијамо везу геометријског и алгебарског растојања.

$$d(A'_i, \tilde{A}'_i) = d_{alg}(A'_i, \tilde{A}'_i)/(x'_{i3}\tilde{x'_{i3}})$$

Ова два растојања биће једнака ако је $x'_{i3} = \tilde{x'_{i3}} = 1$. Можемо претпоставити да је $x'_{i3} = 1$ јер је то уобичајено представљање тачке (која није бесконачно далека). Са друге стране, $\tilde{x'_{13}}$ биће једнако 1 ако је пресликавање облика

$$P = \begin{pmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ 0 & 0 & 1 \end{pmatrix}.$$

Дакле, у случају афиних пресликавања равни, алгебарска грешка једнака је геометријској јер је испуњен услов $x'_{i3} = \tilde{x'_{i3}} = 1$.

### 6.2.4 Нормализовани $DLT$ алгоритам

Важна карактеристика сваког алгоритма је његово понашање при малој промјени улазних величина. У случају одређивања пројективног пресликавања поставља се питање: шта ће се догодити ако се на неки начин промијене тачке које користимо? На примјер, ако посматрамо исте тачке али у другом координатном систему. Или ако примјенимо неку пројективну трансформацију на улазне парове тачака. Било би добро када би матрице добијене $DLT$ алгоритмом прије и након неке од наведених промјена биле исте или бар на неки начин повезане. Међутим, то није случај. $DLT$ алгоритам биће инваријантан у односу на избор тачака једино ако се претходно

уради нормализација истих.

Алгоритам за нормализацију тачака је следећи:

1. Израчунати тежиште система тачака (афино)
2. Транслирати тежиште у координатни почетак (матрицу транслације означимо са $G$)
3. Примјенити хомотетију тако да просјечна удаљеност тачке од координатног почетка буде $\sqrt{2}$ (матрицу хомотетије означимо са $S$)
4. Матрица $T = SG$ је матица нормализације тачака.

Осим што чини $DLT$ алгоритам стабилнијим, нормализација тачака смањује и нумеричку грешку која се јавља приликом рачуна.

Дакле, умјесто $DLT$ алгоритма, можемо користити и нормализовани $DLT$ алгоритам који има извјесне предности:

1. Одредити матрицу $T$ нормализације тачака (оригинали)
2. Одредити матрицу $T'$ нормализације тачака (слике)
3. Нормализовати оригиналне тачке $\overline{A_i} = TA_i$, $i = 1, \ldots, n$
4. Нормализовати слике тачака $\overline{A'_i} = T'A'_i$, $i = 1, \ldots, n$
5. Одредити $DLT$ алгоритмом матрицу $\overline{P}$ користећи нормализоване тачке $\overline{A_i}$ и $\overline{A'_i}$
6. Матрица траженог пресликавања добија се као $P = T'^{-1}\overline{P}T$.

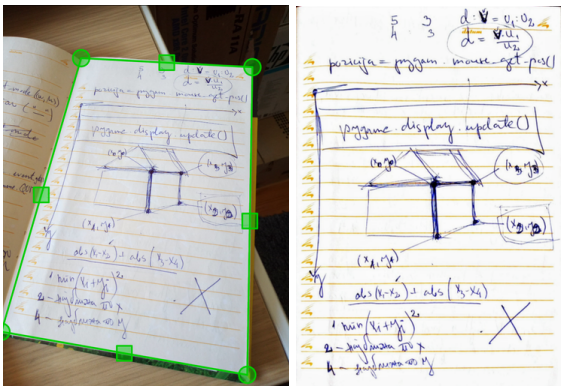
Важно је напоменути да, уколико су улаз тачно четири пара тачака, $DLT$ и нормализовани $DLT$ алгоритам дају исти резултат као наивни алгоритам. У том случају је матрица пројективног пресликавања $P$ једнозначно одређена тим паровима тачака. То јест, важи $\lambda A'_i = PA_i$ за свако $i \in \{1, 2, 3, 4\}$.

# 7 Отклањање пројективне дисторзије

Као што је већ речено, пројективна дисторзија настаје када оса фотоапарата није нормална на раван коју фотографишемо. То се дешава због непрецизности фотографа, али и из оправданих разлога. На примјер, да не би дошло до пројективне дисторзије када фотографишемо високу грађевину, било би неопходно да се значајно удаљимо од ње, што некада није могуће (слика 10). Неопходно је мало искосити фотоапарат да би цијела грађевина била на фотографији и тада изгледа да се она сужава од дна ка врху (жабља перспектива). У архитектури ово често представља незанемарљив проблем и неопходно је отклонити пројективну дисторзију са фотографије (ако већ није могуће направити фотографију без дисторзије).

Слика 10: Примјер пројективне дисторзије (*Оливера Живановић*)

Отклањање пројективне дисторзије користе и апликације на мобилним телефонима које служе за „скенирање" документа, тј. превођење фотографија направљених мобилним телефоном у pdf фајл (слика 11). Између осталих модификација које изврше на фотографији, обавезно је и отклањање дисторзије да би документ заиста имао облик правоугаоника (као у стварности).



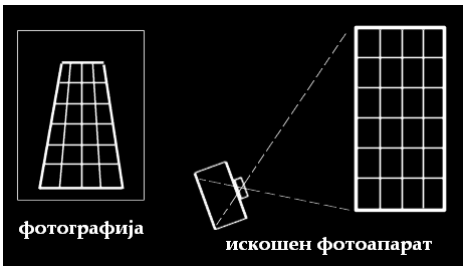
(а) Прије обраде (б) Након обраде

Слика 11: Мобилни скенери

Ове мобилне апликације отклањају пројективну дисторзију софтверски. Међутим, могуће је и механички отклонити ову врсту дисторзије. Тачније, постоји посебна фото-опрема која служи да до пројективне дисторзије уопште и не дође.

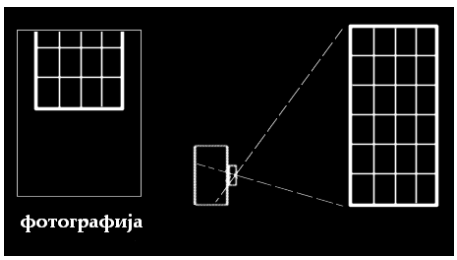
## 7.1 Отклањање дисторзије механички

Вратимо се на проблем због кога је и осмишљен начин за механичко отклањање пројективне дисторзије.

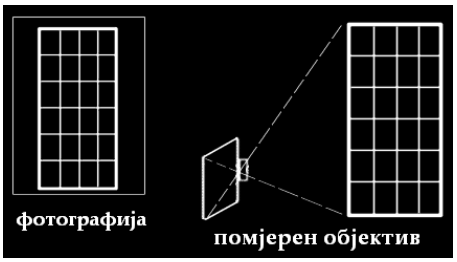



Слика 12: Како настаје пројективна дисторзија ($www.f-stop.com$)

Потребно је фотографисати грађевину значајно вишу од нас. Када бисмо се довољно удаљили од ње, све би било добро и фотографија би одговарала стварности, али то најчеше није могуће. Широкоугаони објективи су од велике помоћи. Захваљујући њима, могуће је фотографисати цијелу грађевину иако стојимо јако близу ње. Ипак, углавно је потребно мало нагнути фотоапарат уназад. Тада оса апарата више неће бити нормална на предњу фасаду и грађевина ће на фотографији бити мање или више деформисана, јавиће се пројективна дисторзија (слика 12).




(а) Непотпуна фотографија




(б) Фотографија без дисторзије

Слика 13: Отклањање дисторзије механички ($www.f-stop.com$)

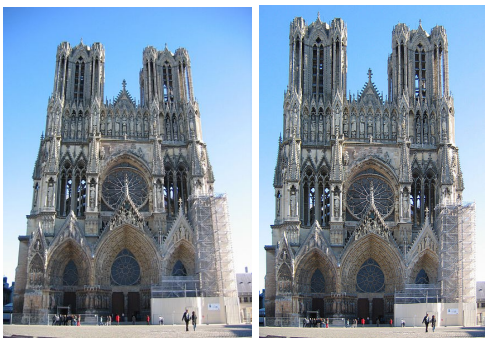
Ако покушамо да избјегнемо појаву дисторзије, на фотографији ће бити видљив само приземни дио зграде (слика 13 (а)).

Било би идеално, када бисмо могли да подигнемо објектив, а да раван сензора остане паралелна равни грађевине коју фотографишемо. Баш то и јесте механичко рјешење за ову врсту дисторзије (слика 13 (б)).

Још од средине прошлог вијека, произвођачи фото-опреме су креирали објективе који су могли да се помјерају на одређене начине. Најприје је *Nikon* пласирао објектив који може да отклони пројективну дисторзију јер има могућност помјерања горе-доље (енгл. *shift lens*). Већ 1973. године *Canon* прави објектив који може и да се помјера и ротира (енг. $tilt - shift$ *lenses*). Ротација објектива помаже код истовременог фокусирања објеката који су на различитом одстојању од фотоапарата. Ови објективи називају

Слика 14: $Shift$ објектив и *view* фотоапарат ($www.f - stop.com$)

се још и *PC* објективи од енглеског *perspective correction*. Осим њих, постоје фотоапарати (*large format view camera*) којима посебан објектив није потребан јер имају могућност помјерања и ротирања равни у којој се налази сензор (енг. *film plane*). *PC* објектив и *view* фотоапарат приказани су на слици 14.



(а) Обичан објектив (б) *PC* објектив

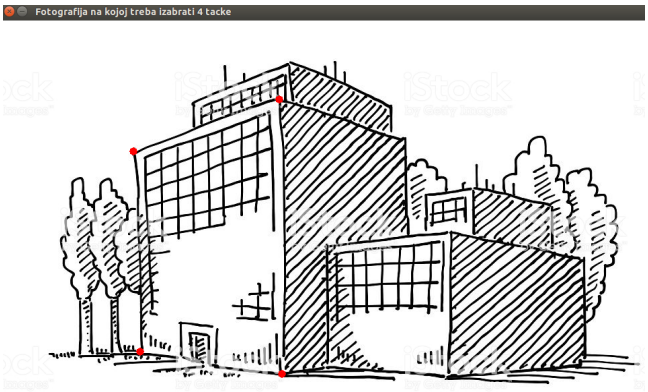
Слика 15: Фотографија са и без дисторзије (*www.wikipedia.org*)

Примјер фотографије снимљене без *PC* објектива и истог призора фотографисаног уз коришћење овог објектива може се видјети на слици 15.

## 7.2 Отклањање дисторзије софтверски

Пројективна дисторзија са дигиталних фотографија може се уклонити и софтверски. Програм се заснива на пројективној геометрији или прецизније, пројективном пресликавању.

Фотографија и одређена раван простора који смо фотографисали су повезани јединственим пројективним пресликавањем. То је последица чињенице да је централна пројекција једне равни пројективно пресликавање (слика 5, глава 2). Дакле, проблем се своди на следеће: потребно је одредити ово пресликавање и онда њиме пиксел по пиксел пресликати дигиталну фотографију. Тако добијамо фотографију без дисторзије.




(а) Улаз


(б) Излаз

Слика 16: Отклањање дисторзије софтверски (коришћена слика је преузета са *www.istockphoto.com*)

Остаје још само питање како одредити потребно пројективно пресликавање.

Довољно је на фотографији изабрати четири тачке које у стварности представљају правоугаоник. На фотографији оне одређују неки произвољан четвороугао. Тих осам тачака (тјемена четвороугла на фотографији и правоугаоника у стварности) једнозначно одређују пројективно пресликавање. Наивни алгоритам описује поступак како од ових тачака добијамо матрицу пресликавања. Такође, потребна је и одговарајућа функција која одређује координате тјемена правоугаоника. Ту можемо искористити и однос дужине и ширине истог у стварности (нпр. ако се ради о некој грађевини, овај однос је познат).

Уколико се из неког разлога користи више од четири пара тачака, пројективно пресликавање се може одредити помоћу $DLT$ алгоритма или, још боље, нормализованог $DLT$ алгоритма.

Остаје још пресликати сваки пиксел (у хомогеним координатама јер је матрица пресликавања увијек формата $3 \times 3$) и тако формирати нову фотографију. Међутим, у пракси се не ради тако већ управо супротно. За сваки пиксел нове фотографије (коју креирамо) тражимо пиксел који му одговара на улазној фотографији. То се постиже уз помоћ инверза добијеног пројективног пресликавања. Инверз ће увијек постојати јер су пројективна пресликавања бијекције. Дакле, користи се такозвано обрнуто мапирање (енг. *reverse pixel mapping*). Оно је погодно јер има одређене предности у односу на директно мапирање. Наиме, приликом директног пресликавања пиксела, добијене координате не морају увијек бити цјелобројне вриједности (а координате пиксела јесу цјелобројне). Због тога на излазној фотографији може бити необојених пиксела, разних преклапања и неправилности. Међутим, када се користи обрунуто мапирање, до тога не долази. Боји се један по један пиксел излазне фотографије. Ако одређеном пикселу одговара неки са координатама у децималном запису (који заправо не постоји) на улазној фотографији, врши се интерполација боја сусједних пиксела и тако добија довољно добар резултат.

Јасно је да се приликом отклањања пројективне дисторзије на овај начин губи на квалитету фотографије, али је та мана занемарљива у односу на велику примјену и корист коју софтверско отклањање дисторзије пружа. Са друге стране, механичко отклањање дисторзије не утиче на квалитет фотографије јер се дисторзија отклони прије фотографисања.

# 8 Програм за отклањање дисторзије

У овом поглављу дат је код програма који отклања пројективну дисторзију. Код је писан у програмском језику Python.

Када се програм покрене, отвара се Pygame прозор са фотографијом са које откањамо дисторзију. Корисник само треба да изабере четири тачке у општем положају које у стварности образују правоугаоник (а на фотографији, због пројективне дисторзије, неки произвољан четвороугао). Програм затим изгенерише и прикаже фотографију на којој је пројективна дисторзија отклоњена, тј. изабране тачке на излазној фотографији су заиста

тјемена правоугаоника.

У коду је коришћен наивни алгоритам за одређивање пројективног пресликавања. Иако је имплементиран и *DLT* алгоритам, наивни је илустративнији због своје геометријске природе. Програм пореди резултате ова два алоритма и може се видјети да они дају исту матрицу пројективног пресликавања. Разлика може да се појави тек у шестој децимали (слика 17).

```
user@Makina:~/Desktop/Master$ python otklanjanjeDistorzije.py
[[  2.49067236e+00  -2.62100896e-02  -1.97841694e+02]
 [  1.00383777e+00   1.71136237e+00  -3.05805887e+02]
 [  2.44324191e-03  -6.31568424e-05   1.00000000e+00]]


[[  2.49067235e+00  -2.62100883e-02  -1.97841705e+02]
 [  1.00383782e+00   1.71136248e+00  -3.05805878e+02]
 [  2.44324189e-03  -6.31568400e-05   1.00000000e+00]]
user@Makina:~/Desktop/Master$
```

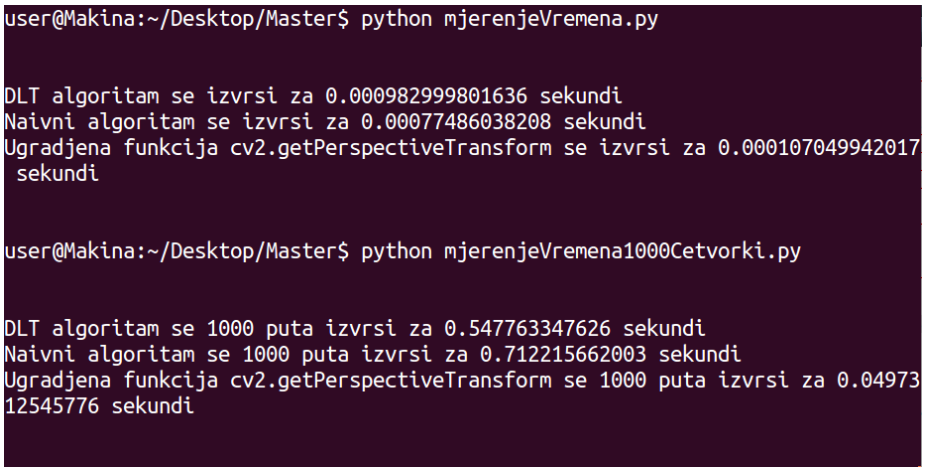
Слика 17: Резултат наивног и *DLT* алгоритма (за четири пара тачака)

У библиотеци OpenCV-Python, која служи специјално за проблеме компјутерског вида, постоји функција cv2.getPerspectiveTransform() која рачуна матрицу пројективног пресликавања за унијете координате четири пара тачака. Ова функција даје исту матрицу пресликавања као и функција naivniAlgoritam() која је коришћена у овом програму.

OpenCV нуди и функцију cv2.warpPerspective() која за фотографију и матрицу (улаз) прави фотографију која је слика улазне при датом пресликавању. Ова функција није коришћена у програму да би се нагласила важност обрнутог мапирања. Међутим, ни процес пресликавања пиксела није обављен у потпуности. То би подразумијевало и одабир и имплементацију интерполације које ће најбоље одредити боју пиксела који немају конкретан оригинал. Зато је, након одређивања координата пиксела помоћу инверза матрице пресликавања, употријебљена уграђена функција cv2.remap() која трансформише улазну фотографију помоћу мапе пресликаних пиксела. Унутар те функције обављају се све потребне интерполације и апроксимације.

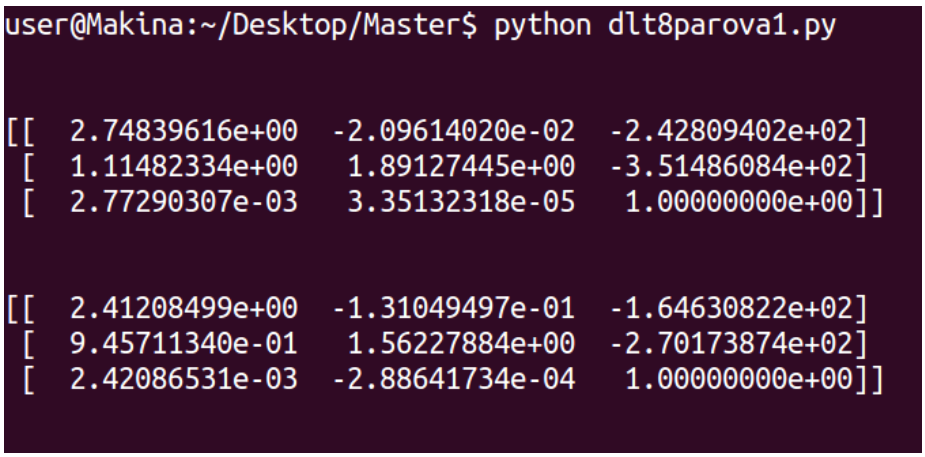
Резултат рада овог програма може се видјети на слици 16.

Што се тиче времена потребног за рачунање матрице пројективног пресликавања, *DLT* је унеколико бржи од наивног алгоритма, док најбољи резултат даје функција cv2.getPerspectiveTransform(). Али све док се ради са четири пара тачака, о тој разлици нема потребе водити рачуна. На слици 18 се види да се и после примјене алгоритама 1000 пута, не уштеди више од $0,67$ секунди.




Слика 18: Поређење времена извршавања алгоритама

Може се показати и колика је грешка *DLT* алгоритма примјењеног на више од четири пара тачака. На слици 19 су приказане матрице добијене наивним алгоритмом (који прима четири пара тачака) и *DLT* алгоритмом (који је радио са још додатна четири пара).




Слика 19: Матрице добијене наивним и *DLT* алгоритмом

Иако *DLT* алгоритам прави одређену грешку, она не утиче драстично на отклањање пројективне дисторзије што се може видјети на слици 20.


Слика 20: Излазна фотографија при употреби наивног (лијево) и *DLT* алгоритма који прима осам парова тачака (десно)

У наставку слиједи поменути код.

## 8.1 Код

```
#Zbog polozaja fotoaparata u odnosu na ono sto fotografisemo
   nekada dolazi do nastajanja projektivne distorzije. Npr. list
   papira pravougaonog oblika na fotografiji moze da ima oblik
   trapeza. Ova distorzija moze da se otkloni softverski. Sledeci
   kod definise program koji radi upravo to. Na pocetku se ucitava
    proizvoljna fotografija. Korisnik bira tjemena cetvorougla
   koji je u stvarnosti pravougaonik, ali je na fotografiji
   deformisan. Program pronalazi matricu projektivnog
   preslikavanja koje ce izabrani cetvorougao vratiti u
   pravougaonik. Zatim se pronadjeno preslikavanje primjenjuje na
   pocetnu fotografiju. Tacnije, koristi se inverzno preslikavanje
    (slika pravougaonik u cetvorougao) da za svaki piksel nove
   fotografije pronadje onaj koji joj odgovara na fotografiji koji
    obradjujemo. Krajnji izlaz je nova fotografija na kojoj je
   uklonjena projektivna distorzija.


import pygame
from pygame.locals import *
import array
import cv2
import numpy
import numpy as np
from numpy import matrix
from numpy.linalg import inv
```

```
#Funkcija koja mijenja mjesta dva clana i u x i u y-nizu. Tj.
    mijenja mjesta dvije tacke.
def zamijeni(x,y,i,j):

  pom1 = x[i]
  pom2 = y[i]
  x[i] = x[j]
  y[i] = y[j]
  x[j] = pom1
  y[j] = pom2


#Korisnik moze da izabere tjemena redom kojim on zeli. Ova funkcija
     ih sortira: gornje lijevo tjeme je prvo (x_0,y_0), a ostala
    idu redom u direktnom smjeru. Ovdje je bitno kako izgleda
    koordinatni sistem pygame prozora. Pocetak (0,0) je gornji
    lijevi ugao. Desno je pozitivan smjer x-ose, a dolje pozitivan
    smjer y-ose.
def poredjajTacke(x,y):

  if x[1] + y[1] < x[0] + y[0]:
    zamijeni(x,y,0,1)
  if x[2] + y[2] < x[0] + y[0]:
    zamijeni(x,y,0,2)
  if x[3] + y[3] < x[0] + y[0]:
    zamijeni(x,y,0,3)
  if x[2] < x[1]:
    zamijeni(x,y,2,1)
  if x[3] < x[1]:
    zamijeni(x,y,3,1)
  if y[3] > y[2]:
    zamijeni(x,y,2,3)


#Ova funkcija odredjuje dimenzije pravougaonika u koji ce biti
    preslikan izabrani cetvorougao.
def dimenzije(x,y):

  duzina = (abs(x[0]-x[3]) + abs(x[1]-x[2]))/2
  visina = (abs(y[0]-y[1]) + abs(y[3]-y[2]))/2


  return duzina, visina


#Funkcija koja odredjuje koordinate tjemena pravougaonika.
def pronadjiSlike(x,y):

  d, v = dimenzije(x,y)


  pom = (x[1] + x[0])/2
  x[0] = pom

  pom = (y[3] + y[0])/2
  y[0] = pom
```

```
  x[1] = x[0]
  y[1] = y[0] + v
  x[2] = x[0] + d
  y[2] = y[1]
  x[3] = x[2]
  y[3] = y[0]


#Algoritam koji pronalazi matricu projektivnog preslikavanja (Data
    su cetiri para tacaka). Radi isto sto i ugradjena funkcija cv2.
    getPerspectiveTransform(). Ideja je pronaci preslikavanje
    baznih tacaka u "originale" i preslikavanje baznih tacaka u
    "slike". Preslikavanje "originala" u "slike" je neka vrsta
    kompozicije ova dva preslikavanja.
def naivniAlgoritam(originali, slike):

  P1 = preslikavanjeBaznihTacaka(originali)
  P2 = preslikavanjeBaznihTacaka(slike)

  M = P2*inv(P1)
  M = M/M[2,2]

  return M


#Funkcija koja projektivno preslikava cetiri bazne tacke u cetiri
    zadate tacke.
def preslikavanjeBaznihTacaka(tacke):

  #pravljenje homogonih koordinata za zadate tacke
  a1 = numpy.append(tacke[0], 1)
  b1 = numpy.append(tacke[1], 1)
  c1 = numpy.append(tacke[2], 1)
  d1 = numpy.append(tacke[3], 1)

  #bazne tacke
  a = np.array([1, 0, 0])
  b = np.array([0, 1, 0])
  c = np.array([0, 0, 1])
  d = np.array([1, 1, 1])

  #Odredjivanje preslikavanja. Trazimo matricu P za koju vazi: Pa =
     k_1a_1, Pb = k_2b_1, Pc = k_3c_1 i Pd = k_4d_1 pri cemu su k_i
     realni koeficijenti. Odakle slijedi da (bazne tacke su tako i
    izabrane) prva kolona matrice P ima oblik k_1a_1, druga k_2b_1,
     a treca k_3c_1. Sada samo treba naci k_1, k_2 i k_3. To nije
    problem jer imamo i cetvrtu tacku - d_1. Posto je Pd = d_1
    (mozemo uzeti da je k_4=1) mora da vazi da je d_1 = k_1a_1 +
    k_2b_1 + k_3c_1. Tj. d_1 = koef*A gdje je koef vektor kolona
    (program je vidi kao matricu) koja sadrzi koeficijente, a A
    matrica kojoj su kolone koordinate tacaka a_1, b_1 i c_1. sada
    je lako naci koeficijente, a samim tim i matricu preslikavanja.
  A = np.matrix([a1, b1, c1]).transpose()

  koef = np.dot(inv(A), d1)
```

```
  P = np.matrix([koef[0,0]*a1, koef[0,1]*b1, koef[0,2]*c1]).
    transpose()

  return P


def dltAlgoritam(originali, slike):

  #pravljenje homogonih koordinata za zadate tacke
  a = numpy.append(originali[0], 1)
  b = numpy.append(originali[1], 1)
  c = numpy.append(originali[2], 1)
  d = numpy.append(originali[3], 1)


  a1 = numpy.append(slike[0], 1)
  b1 = numpy.append(slike[1], 1)
  c1 = numpy.append(slike[2], 1)
  d1 = numpy.append(slike[3], 1)


  M = np.matrix([
    [0, 0, 0, -a[0], -a[1], -a[2], a1[1]*a[0], a1[1]*a[1], a1[1]*a
    [2]],
    [a[0], a[1], a[2], 0, 0, 0, -a1[0]*a[0], -a1[0]*a[1], -a1[0]*a
    [2]],
    [0, 0, 0, -b[0], -b[1], -b[2], b1[1]*b[0], b1[1]*b[1], b1[1]*b
    [2]],
    [b[0], b[1], b[2], 0, 0, 0, -b1[0]*b[0], -b1[0]*b[1], -b1[0]*b
    [2]],
    [0, 0, 0, -c[0], -c[1], -c[2], c1[1]*c[0], c1[1]*c[1], c1[1]*c
    [2]],
    [c[0], c[1], c[2], 0, 0, 0, -c1[0]*c[0], -c1[0]*c[1], -c1[0]*c
    [2]],
    [0, 0, 0, -d[0], -d[1], -d[2], d1[1]*d[0], d1[1]*d[1], d1[1]*d
    [2]],
    [d[0], d[1], d[2], 0, 0, 0, -d1[0]*d[0], -d1[0]*d[1], -d1[0]*d
    [2]],
  ])

  #SVD dekompozicija matrice M
  U, D, V = np.linalg.svd(M)

  #Poslednja kolona matrice V definishe trazheno projektivno
    preslikavanje
  P = V[-1].reshape(3, 3).astype(np.float32)
  P = P/P[2,2]


  return P



pygame.init()

# ucitavanje slike za pygame (najlaksi nacin da korisnik izabere
    tacke)
```

```
slika = pygame.image.load('zgrada2.jpeg')


#ucitavanje slike za OpenCV. Da bi se koristile funkcije cv2.remap
    () ili cv2.warpPerspective(), slika mora biti predstavljena kao
     numpy array
slika1 = cv2.imread('zgrada2.jpeg')


width = slika.get_width()
height = slika.get_height()


#pravljenje prozora koji ima sliku kao pozadini tako da korisnik
    moze klikom da izabere tjemena cetvorougla koji ce biti
    "ispravljen"
prozor = pygame.display.set_mode((width, height))
pygame.display.set_caption("Fotografija na kojoj treba izabrati 4
    tacke")
prozor.blit(slika, (0,0))


#pravljenje nizova u koji ce biti smjestene koordinate tacaka koje
    korisnik bira
run = True
i = 0
x = array.array('i', (0 for i in range(0,4)))
y = array.array('i', (0 for i in range(0,4)))



while run:

  #biranje i pamcenje koordinata piksela
  for dogadjaj in pygame.event.get():
    if dogadjaj.type == pygame.QUIT:
      run = False

    if dogadjaj.type == pygame.MOUSEBUTTONDOWN:
      pozicija = pygame.mouse.get_pos()

      pygame.draw.circle(prozor, [255, 0, 0], pozicija, 6, 0)
      pygame.display.update()


      x[i], y[i] = pozicija

      i = i+1


    if i == 4:   #ako su izabrane sve cetiri tacke:

      poredjajTacke(x,y)

      originali = np.float32([
        (x[0], y[0]),
        (x[1], y[1]),
```

```
        (x[2], y[2]),
        (x[3], y[3]),
      ])


      pronadjiSlike(x,y)


      slike = np.float32([
        (x[0], y[0]),
        (x[1], y[1]),
        (x[2], y[2]),
        (x[3], y[3]),
      ])

      M = naivniAlgoritam(originali, slike)

#Poredjenje matrica koje su rezultat dlt i naivnog algoritma. Za
    cetiri para tacaka, matrice su identicne
      print M
      print dltAlgoritam(originali, slike)


      M1 = inv(M) #ovo je matrica koja slika piksele nove
    fotografije u piksele fotografije koju smo ucitali



      #Treba jos samo izgenerisati novu fotografiju uz pomoc
    dobijene matrice. To se moze uraditi pomocu ugradjene funkcije
      #novaFotografija = cv2.warpPerspective(slika1, M, (width,
    height), flags=cv2.INTER_LINEAR)
      #cv2.imshow('novaFotografija', novaFotografija)


      #Medjutim, nije tesko ni "rucno" preslikati sve piksele

      #pravimo "mrezu" za matricu piksela
      indY, indX = np.indices((height, width), dtype=np.float32)

      #svaki piksel predstavimo homogenim koordinatama i stavimo ih
     u matricu tako da je jedan piksel jedna kolona
      matricaPiksela = np.array([indX.ravel(), indY.ravel(), np.
    ones_like(indX).ravel()])

      preslikaniPikseli = M1.dot(matricaPiksela)

      # dijelimo poslednjom koordinatom, tj. vracamo homogene
    koordinate u euklidske i pravimo dva niza piksela: x i y
    koordinata
      mapX, mapY = preslikaniPikseli[:-1]/preslikaniPikseli[-1]

      #ponovo vracamo u matricni oblik
      mapX = mapX.reshape(height, width).astype(np.float32)
      mapY = mapY.reshape(height, width).astype(np.float32)

      novaFotografija = cv2.remap(slika1, mapX, mapY, cv2.
```

```
    INTER_LINEAR)


      cv2.circle(novaFotografija, (x[0],y[0]), 6, (0,0,255), -1)
      cv2.circle(novaFotografija, (x[1],y[1]), 6, (0,0,255), -1)
      cv2.circle(novaFotografija, (x[2],y[2]), 6, (0,0,255), -1)
      cv2.circle(novaFotografija, (x[3],y[3]), 6, (0,0,255), -1)
      cv2.imshow('novaFotografija.png', novaFotografija)


      cv2.waitKey(0)
      cv2.destroyAllWindows()
      run = False

  pygame.display.update()

pygame.quit()
```

# 9 Закључак

Као што се у раду могло прочитати, пројективна дисторзија настаје као последица централног пројектовања тродимензионалних објеката на раван фотографије. За туристе и фотографе аматере, она не представља никакав проблем. Међутим постоје ситуације када ју је неопходно отколонити.

То је могуће учинити на два начина: механички и софтверски.

Механичко отклањање као резултат има фотографију већег квалитета, али се постиже захваљујући посебним фотоапаратима или објективима.

Ако је фотографија већ направљена са дисторзијом, може да се софтверски обради и уклони дисторзија, али се тада губи на квалитету фотографије.

Софтверско отклањане дисторзије темељи се на резултатима пројективне геометрије. Идеја је пронаћи пројективно пресликавање које повезује објекат какав видимо на фотографији (често мало искривљен) и исти тај објекат у стварности.

Постоји више алгоритама за проналажење пројективног пресликавања. У раду су изложени наивни и *DLT* алгоритам. Наивни алгоритам је чисто геометријске природе. Прима четири пара тачака (јер тачно толико парова јединствено одређује пројективно пресликавање) и даје потпуно тачан резултат.

За разлику од наивног, *DLT* алгоритам је алгебарске природе. Његова предност је у томе што може да одреди пресликавање и ако су дата више од четири пара тачака, али самим тим, јавља се одређена грешка. За отклањање дисторзије, ова грешка не прави примјетан проблем (што је и приказано у раду), али ради веће прецизности, може се извршити нормализација *DLT* алгоритма.

Да би се јасно разумјела изложена тема, потребно је познавање основа пројективне геометрије. Заинтересовани за ову област највише детаља могу пронаћи у књизи [3].

# Литература


[1] Вукмировић С., Примена пројективне геометрије у рачунарству, Презентација са предавања на Математичком факултету, Београд, 2018.

[2] Freifeld O., Methods in computer vision, Презентација, Ben-Gurion University of the Negev, Биршеба, 2017.

[3] Hartley R., Zisserman A., Multiple View Geometry in Computer Vision Second Edition, Cambridge University Press, 2003.

[4] Birchfield S., An Introduction to Projective Geometry (for computer vision)

[5] Андрејић В., Пројективна геометрија равни, Математички факултет, Београд, 2016.

Електронски извори:

[6] *www.f − stop.com*

[7] *www.wikipedia.org*

[8] *steemit.com*

[9] *www.nikon.rs*