

Дигитално процесирање на слика

**Тема: Photo Sketching**

Ментор: Изработила:

проф. др. Ивица Димитровски Бојана Боцевска, 221227

2024

**Contents**

[**Абстракт** 3](#_Toc177257742)

[**Вовед** 4](#_Toc177257743)

[**Stroke-Based Rendering (SBR)** 5](#_Toc177257744)

[**Кохерентно цртање на линии (CLD)** 6](#_Toc177257745)

[Главни компоненти на CLD 7](#_Toc177257746)

[Примена на CLD 8](#_Toc177257747)

[**Имплементација** 11](#_Toc177257748)

[**Финален резултат** 13](#_Toc177257749)

[**Користена литература** 15](#_Toc177257750)

# **Абстракт**

Во денешно време областа на Photo Sketching станува порелевантна во не-фотореалистично рендерирање и стилизација на слики. Трансформирање на дигитални фотографии во слики налик на рачно скициран цртеж има голема примена во 2Д и 3Д грфика, како на пример стилизација на модели и позадински елементи во видео игри.

Овој проект има за цел да развие и имплементира алгоритам за скицирање врз основа на градиентите на сликата. Ќе се дискутираат математичките принципи кои служат за генерирање на скици, со фокус на детекција на рабови, а ќе се споменат и комплексни алгоритми како што се Stroke-Based Rendering (SBR) и Painterly Rendering.

# **Вовед**

Не-фотореалистично рендерирање е област од компјутерската графика која се фокусира на создавање на дигитално рендерирани скици, наместо реалистични слики. Со помош на техники од оваа област (како фото-скицирање), уметници и програмери можат да трансформираат дигитални слики во скици. Тоа понатаму има примена во разни домени на уметноста, од анимација до видео игри. Со зголемената побарувачка за стилизирана компјутерски графика, се зголемиле и потребите за поквалитетн софтвер. На почеток се користеле едноставни алгоритми за детекција на рабови, но со развој на технологјата се развивале и пософистицирани методи за детекција со поугледни резултати. Техниките на NPR користат алгоритми кои го имитираат однесувањето на традиционалните уметнички алатки, како што се четки, моливи и пенкала со мастило, овозможувајќи им на програмерите да постигнат ефекти кои би биле тешки или невозможни со традиционалните фотореалистичко рендерирање техники.

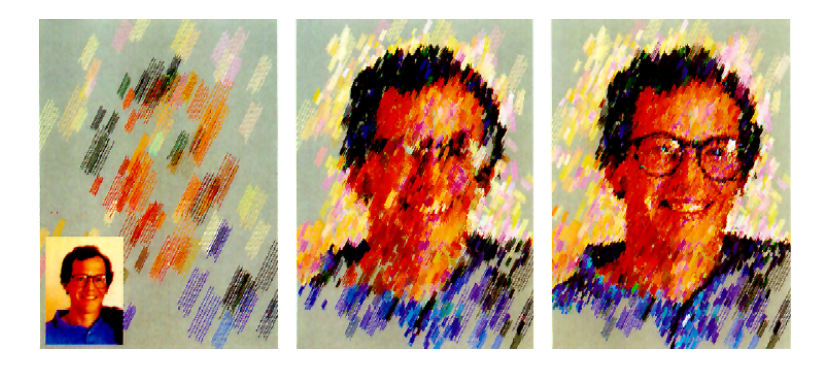
Фокусот на овој проект е Фото скицирање, пристап заснован на користење на градиент за генерирање на скици од дигитални слики. Дополнително, се дискутираат и други широко користени методи за скицирање на фотографии, како што се Stroke-Based Rendering (SBR) и Coherent line drawing (CLD).

# **Stroke-Based Rendering (SBR)**

Stroke-Based Rendering (SBR) го симулира процесот на цртање со поставување на линии на слика врз основа на нејзината основна структура. Позицијата на линиите се пресметува од интензитетот на боите или градиентот на сликата, додека пак должината, ширината и ориентацијата - се различни за да се создаде визуелно богата и детална скица. SBR е особено корисен за создавање уметнички ефекти во статички слики и во анимации, каде што поставувањето на линиите и стилот може да се менуваат за да имитираат различни медиуми.



Во SBR, секоја линија се гледа како самостојна, а финалната слика се добива со спојување на секој слој заедно. Методот опишан во кодот следи сличен принцип, користејќи испреплетени јадра како насоки за како треба да се ориентирани линиите и поставувајќи ги според насоката на доминантниот раб. Сепак, техниките на SBR често вклучуваат пософистицирани правила за поставување и способност за симулирање на различни уметнички алатки, како што се четки и пенкала.



*Интерактивен сликарски систем за рендерирање]. Корисникот кликнува на различна локации, а линиите се генерираат на овие локации; се земаат бојата и градиентите од изворна слика.*

Компоненти:

1. **Позиционирање на линии**:

Потезите се поставуваат врз основа на значајните карактеристики на сликата, како што се рабовите, градиентите на интензитетот или текстурите. Целта е да се постават линии во деловите на сликата каде што доминираат сенки и темни предели или каде што има поголем број на објекти или детали. Овој процес може да се автоматизира со анализа на контрастот и градиентите на дадена слика, или може да биде полуавтоматизиран, овозможувајќи им на уметниците рачно да го водат поставувањето на ударот.

1. **Атрибути на исцртаните линии**:

Секоја исцртана линија го има следните параметри:

* + **Должина и ширина**: Поголеми линии покриваат поголема површина, додека помали линии овозможуваат приказ на повеќе детали.
  + **Ориентација**: Линиите ја следат или насоката на градиентите или рабовите на сликата.
  + **Боја**: Во одредени SBR методи, обоени линии се користат за да се имитираат сликарски техники.

Прилагодувањето на овие атрибути создава широк опсег на визуелни ефекти. На пример, при цртање на линија, потезите може да се разликуваат по дебелина за да се имитира притисок врз моливот при рачното цртање, додека при симулација на сликарски техники потезите со различни големини и форми создаваат илузија дека се користеле четки.

1. **Оптимизација на линиските потези**:

Напредните SBR техники користат алгоритми за оптимизација за да се осигурат дека потезите се поставени на ефикасен и визуелно привлечен начин. Ова често вклучува балансирање помеѓу користење на што е можно помалку потези, а сепак зачувување на деталите и карактерот на сликата. Алгоритмите како што се **Voronoi-based stroke placement** и **trial-and-error** методи ги прилагодуваат позициите и атрибутите на линиите за да добијат што е можно послична слика со оригиналната.

# **Coherent line drawing (CLD)**

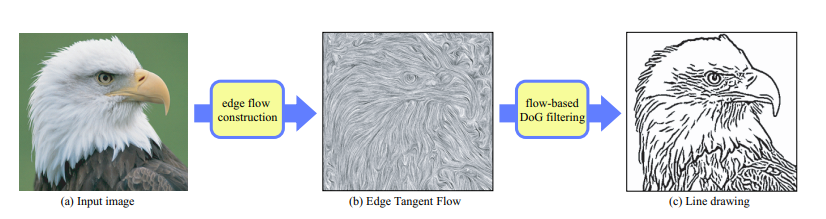
Кохерентно цртање на линии (CLD) е напредна не-фотореалистична техника на рендерирање, дизајнирана да произведува континуирани, мазни и визуелно привлечни линиски цртежи. За разлика од традиционалните методи за откривање рабови, кои можат да генерираат фрагментирани или нерамни линии, CLD се фокусира на зачувување на кохерентноста на потезите, обезбедувајќи линиите да се движат по контурите на објектите на сликата. Овој пристап создава скицирани цртежи со поприроден изглед кои се послични на рачно изработени.

## **Главни компоненти на CLD**

1. **Тек на тангента на рабовите (ETF)**:  
   Главен елемент на CLD е Edge Tangent Flow (ETF). Тоа претставува метод за усогласување на линиските потезите со природните контури на сликата. ETF пресметува векторско поле над сликата, каде што секој вектор се усогласува со тангентата на локалниот раб. Примарната цел на ETF е да ја израмни насоката на рабовите за да се избегнат нагли промени или дисконтинуитети во текот на линиите.

ETF се гради во три чекори:

* + **Првична пресметка на градиент**: Со користење на методи како Sobel методот најпрво се пресметуваат градиентите на сликата за да се пронајдат почетните насоки на линиите на рабовите.
  + **Измазнување на полето на проток**: Насоките на овие рабови се измазнуваат со ширење на векторското поле низ целата слика. Овој процес обезбедува усогласување на векторите во близина, создавајќи кохерентен проток низ објектот.
  + **Поставување на линии**: Финалниот вектор ја води програмата за тоа каде треба да биде поставена секоја линија, со што се осигура прецизност и ефикасност.

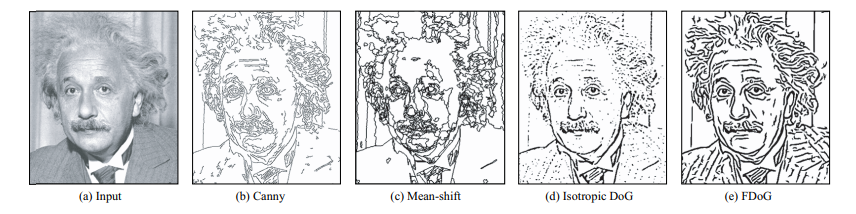


1. **Цртање на линии според ETF**:

Откако ќе се воспостави ETF, алгоритмот генерира линии што течат по ова тангентно поле. Овие линии ги следат контурите на предметите на сликата, создавајќи континуирани потези кои ги следат формите. Мазноста на линиите е една од главните предности на CLD, бидејќи потезите имаат тенденција да го имитираат начинот на кој уметникот природно би ги следел контурите на објектот. Ова е во контраст со традиционалното откривање на рабовите, кое има тенденција да ги третира рабовите како изолирани точки или кратки сегменти, што резултира со испрекинати или нерамни линии.

1. **Стилизирање**:

Во CLD, тежината или дебелината на линиите може да се прилагодат врз основа на карактеристиките на сликата. Подебелите линии можат да нагласат важни контури, додека потенките линии можат да претставуваат ситни детали. Ова овозможува селективно нагласување, слично на тоа како уметникот може да ја менува линијата за да го зголеми визуелниот интерес или да истакне одредени аспекти на цртежот.



1. **Намалување на шумот**:

Голем предизвик во исцртање на линии е справувањето со непотребните детали. CLD вклучува техники за намалување на шумот на сликата за да се отстранат мали и неприметливи детали од сликата кои би можеле потенцијално да го нарушат изгледот на финалниот продукт. Со фокусирање само на доминантните текови на рабовите, CLD го поедноставува цртежот додека ја одржува основната структура на сликата

## **Примена на CLD**

Кохерентното цртање на линии има широка примена во различни области на уметноста и дизајнот, особено онаму каде што се потребни јасност и едноставност. Некои значајни примени вклучуваат:

1. **Илустрирање на стрипови**:

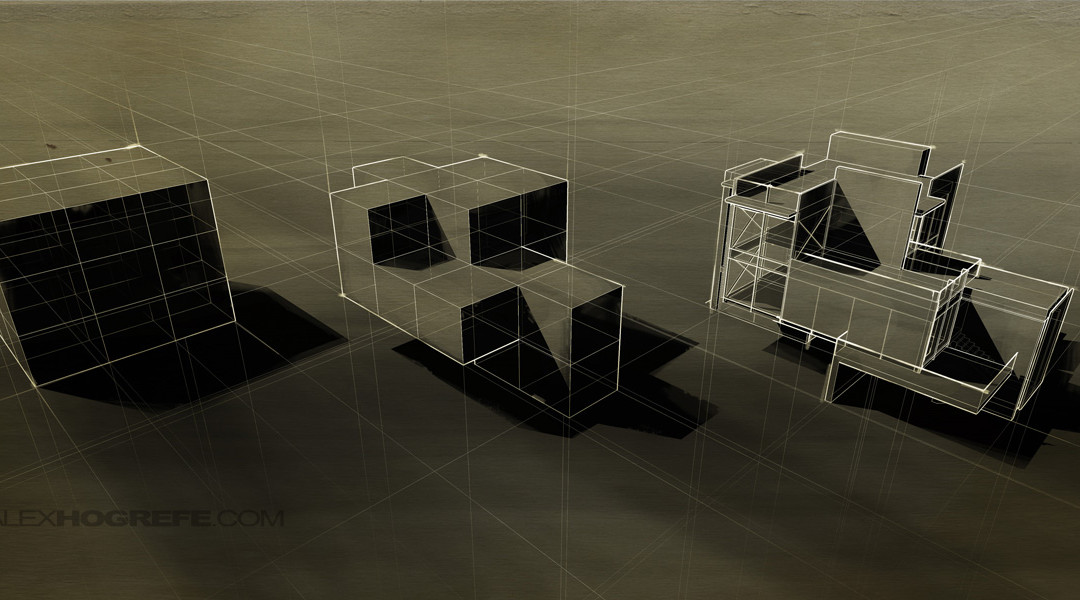
Способноста на CLD да произведува мазни линии го прави идеален за цртани филмови и стрипови, каде што јасните, стилизирани контури се белег на медиумот. Уметниците можат да користат CLD за да ја генерираат основната линија, која потоа може дополнително да се стилизира или пополни со бои.



*Слика превземена од компјутерската игра XIII. Црни рабови на силуета симулираат естетски комичен изглед.*

1. **Технички и архитектонски цртежи**:

Во техничките илустрации и архитектонските рендери, јасноста и прецизноста се најважни. Способноста на CLD да ги следи геометриските контури на структурите помага да се генерираат чисти, кохерентни линии кои јасно ги дефинираат објектите и просторите. Ова може да биде особено корисно при креирање шеми, нацрти и технички дијаграми.

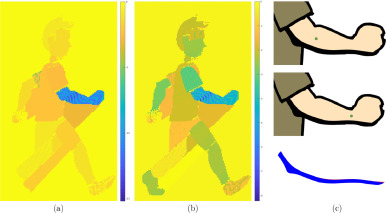


Нацрт на геометриски елементи

1. **Cel Shading и анимација**:

Во анимацијата, особено во стилови како cel shading, континуираните линии се од суштинско значење за да се осигура дека предметите остануваат визуелно кохерентни додека се движат и трансформираат. CLD помага да се одржи конзистентноста на линијата низ кадрите, осигурувајќи дека ликовите и предметите ја задржуваат својата форма и јасност во текот на процесот на анимација.





Мерење на слични сенки создадени од различни почетни локации



Слика од игра која користи техники CLD за да постигне cel shaded ефект врз 3Д моделите

# **Имплементација**

Во следниот дел ќе се разгледува конкретната имплементација од проектот. Кодот со кој е изграден алгоритмот е базиран на поденоставна верзија од SBR како што беше предходно нагласено. За ефикасно имплементирање на кодот користени се библиотеките cv2, numpy, mathplotlub, како и пакетите skimage и scipy.

Детален преглед на кодот:

1. **Вчитување на слика**

Најпрво сликата се вчитува. Потоа за да се осигура дека сликата правилно ќе се конвертира се претвара од rgb alpha во rgb..

img = io.imread('flower.png')

if img.shape[2] == 4:

img = color.rgba2rgb(img)

im = color.rgb2gray(img)

Конверзијата на сликата во сиви тонови му помага на алгоритмот да се фокусира исклучиво на разликите во интензитетот, Сепак тоа може да се прескокне ако крајниот резултат треба да биде во боја.

1. **Измазнување на сликата**

Измазнување на сликата е суштински значајно при припрема на сликата за процесирање. Овај чекор го намалува шумот во сликата за да се олесни издвојување на рабовите во истата. Се користат два методи за измазнување:

* + **Гаусонов метод**: Применува Гаусоново заматување на сликата, со што се намалува високо фреквенцискиот шум и овозможува креирање на помазни рабови. Математички приказ:



Каде G(x,y;σ) е Гаусонова функција со стандардна девијација σ.

* + **Филтрирање на медиана**: Овој филтер ги зачувува рабовите на сликата преку замена на секој пиксел со медианата од соседните два пиксели.

if smooth\_kernel == "gauss":

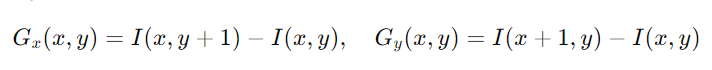
smooth\_im = filters.gaussian(img, sigma=np.sqrt(2))

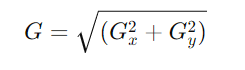
else:

smooth\_im = filters.median(img)

1. **Пресметка на градиентот**

Откако сликата ќе се измазне, алгоритмот пресметува градиенти за да ги идентификува областите со поинтенсивни промени на интензитетот, поточно рабовите

* + **Преден градиент**: Едноставен метод кој ја споредува разликата меѓу соседни пиксели.
  + **Собел метод**: Понапреден метод кој поефикасно го пресметува градиентот користејќи конволуција со Sobel кернели. Магнитудата на градиентот се добива со:



Каде Gx и Gy се градиенти филтрирани во насоките х и у.

if not gradient\_method:

diff\_x = img[:, 1:width] - img[:, 0:width - 1]

diff\_y = img[1:height, :] - img[0:height - 1, :]

g = np.sqrt(np.square(diff\_x) + np.square(diff\_y))

else:

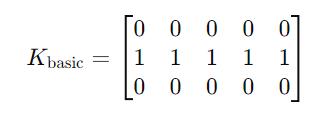
sobelx = cv2.Sobel(img, cv2.CV\_64F, 1, 0, ksize=5)

sobely = cv2.Sobel(img, cv2.CV\_64F, 0, 1, ksize=5)

g = np.sqrt(np.square(sobelx) + np.square(sobely))

1. **Насоки на преплетување**

За да се симулираат линии нацртани со молив, кодот користи јадро со хоризонтална линија. Ова јадро се ротира во повеќе насоки (на пр. хоризонтална, вертикална, дијагонала) за да се доловат рабовите од различни агли. Обично јадро со големна (2×kernel size+1) се иницијализира на следниот начин:



Операцијата потоа генерира мапа од потези која имитира линии од молив усогласени со контурите на сликата. Таква мапа се генерира за секоја насока

basic\_ker = np.zeros((kernel\_size \* 2 + 1, kernel\_size \* 2 + 1))

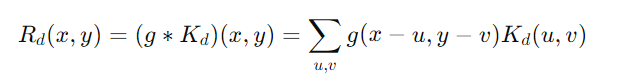
basic\_ker[kernel\_size + 1, :] = 1

for d in range(num\_of\_directions):

ker = transform.rotate(basic\_ker, (d \* 180) / num\_of\_directions)

res\_map[:, :, d] = signal.convolve2d(g, ker, mode='same')

Овој чекор идентификува која насока е најсоодветна на рабовите за секој пиксел.



1. **Готова мапа на потези**

По одредувањето на најсоодветната насока за секој пиксел, алгоритмот генерира конечна мапа со методата np.argmax. Мапата потоа се подесува со прилагодување на ширината и интензитетот на линиите. Конечната скица е направена со превртување на мапата со потези, создавајќи црно-бела скица со молив:



max\_pixel\_indices\_map = np.argmax(res\_map, axis=2)

s\_tag\_normalized = (s\_tag - np.min(s\_tag.ravel())) / (np.max(s\_tag.ravel()) - np.min(s\_tag.ravel()))

s = 1 - s\_tag\_normalized

Со тоа завршува главниот дел од програмата. Останува само да се врати мапата до каде што е повикана функцијата и да се прокаже крајниот резултат.

# **Финален резултат**

За приказ на производот добиен од демо – кодот искористена е библиотеката на matplotlib.pyplot. Во продолжение има повеќе примери кои се добиени при тестирање и извршување.

Во првиот пример (Слика 1) ги користам вредностите на атрибутите кои почетно се хардкодирани.

kernel\_size=8, stroke\_width=1, num\_of\_directions=8, smooth\_kernel="gauss",  
 gradient\_method=1, stroke\_darkness=2



Слика 1

Следно, со зголемување на бројот на насоки и темнината на линијата, може да се примети дека линиите се потемни и појасни (Слика 2). Тоа е бидејќи зголемениот број на насоки овозможува појасно да се дефинираат рабовите, а со тоа и крајните линии.

kernel\_size=8, stroke\_width=6, num\_of\_directions=16, smooth\_kernel="gauss",

gradient\_method=1, stroke\_darkness=4



Слика 2

Од друга страна пак, ако го намалиме бројот на насоки на само четири, се добива покрут изглед на лиините (Слика 3). Тоа е бидејќи програмата е лимитирана само да се движи по х и у оските, без можност да создава криви линии.

kernel\_size=8, stroke\_width=6, num\_of\_directions=4, smooth\_kernel="gauss",

gradient\_method=1, stroke\_darkness=2



Слика 3

# **Користена литература**

* <https://3dstereophoto.blogspot.com/2018/05/non-photorealistic-rendering-edge.html>
* [Cewu Lu, Li Xu and Jiaya Jia. Combining Sketch and Tone for Pencil Drawing Production.](https://www.cse.cuhk.edu.hk/leojia/projects/pencilsketch/npar12_pencil.pdf)
* [Mengtian Li, Zhe Lin, Radom´ır Mech, Ersin Yumer and Deva Ramanan. Photo-Sketching: Inferring Contour Drawings from Images](https://www.ri.cmu.edu/app/uploads/2019/01/Li-Mengtian-WACV-2019-Photo-Sketching.pdf)
* [Aaron Hertzmann. Stroke-Based Rendering](https://www.cs.ucdavis.edu/~ma/SIGGRAPH02/course23/notes/S02c23_3.pdf)
* [Henry Kang, Seungyong Lee, and Charles K. Chui. Flow-Based Image Abstraction](https://www.umsl.edu/~kangh/Papers/kang_tvcg09.pdf)
* [Henry Kang, Seungyong Lee and Charles K. Chui. Coherent Line Drawing](https://umsl.edu/cmpsci/about/People/Faculty/HenryKang/coon.pdf)
* [Zhengyan Tong, Xuanhong Chen, Bingbing Ni and Xiaohang Wang. Sketch Generation with Drawing Process Guided by Vector Flow and Grayscale](https://ojs.aaai.org/index.php/AAAI/article/view/16140)
* [Simon M. Danner and Christoph J. Winklhofer. Cartoon Style Rendering](https://www.cg.tuwien.ac.at/courses/Seminar/WS2007/comicstyle.pdf)