TUGAS INDIVIDU
IF5152 COMPUTER VISION
APLIKASI SEDERHANA INTEGRATIF

Dipersiapkan oleh:

Sa’ad Abdul Hakim - 13522092 - K01

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
JL. GANESA 10, BANDUNG 40132
2025

Daftar Isi

I Workflow Pipeline

Aplikasi ini tidak dibangun sebagai satu *pipeline* monolitik, melainkan sebagai kumpulan empat modul skrip Python yang independen, di mana setiap skrip menangani satu bagian fitur (Filtering, Edge Detection, Feature Points, dan Geometry). Setiap skrip (filtering.py, edge.py, featurepoints.py, geometry.py) dirancang untuk:

1. **Memuat dataset**: Secara otomatis memuat gambar standar dari skimage.data (cameraman, coins, checkerboard, astronaut) dan satu gambar pribadi (Candy.jpg).
2. **Memproses Gambar**: Menerapkan fungsi-fungsi yang relevan untuk modul tersebut (misal, Gaussian/Median, Sobel/Canny, Harris/FAST/SIFT, dan Transformasi Proyektif).
3. **Menyimpan Hasil**:
    - Menyimpan gambar *output* (hasil filter, *edge map*, *feature marking*, gambar transformasi) sebagai *file* .png di dalam folder modul yang sesuai.
    - Mengekspor data kuantitatif (parameter yang digunakan, statistik fitur, matriks homografi) ke dalam *file* .csv untuk analisis lebih lanjut.

II Proses dan Hasil Setiap Fitur

II.1. Fitur 01: Image Filtering

Penjelasan Singkat Teori: *Image filtering* adalah teknik memodifikasi atau menyempurnakan gambar dengan melewatkan *kernel* (filter) pada setiap piksel. **Gaussian Filter** adalah filter *low-pass* yang menghaluskan gambar dan mengurangi *noise* dengan mengganti nilai piksel dengan rata-rata terbobot dari piksel tetangganya. **Median Filter** adalah filter non-linear yang mengganti nilai piksel dengan nilai median dari piksel tetangganya, efektif untuk menghilangkan *noise salt-and-pepper* sambil menjaga ketajaman tepi.

Parameter yang Digunakan: parameter yang digunakan bersifat seragam untuk semua gambar.

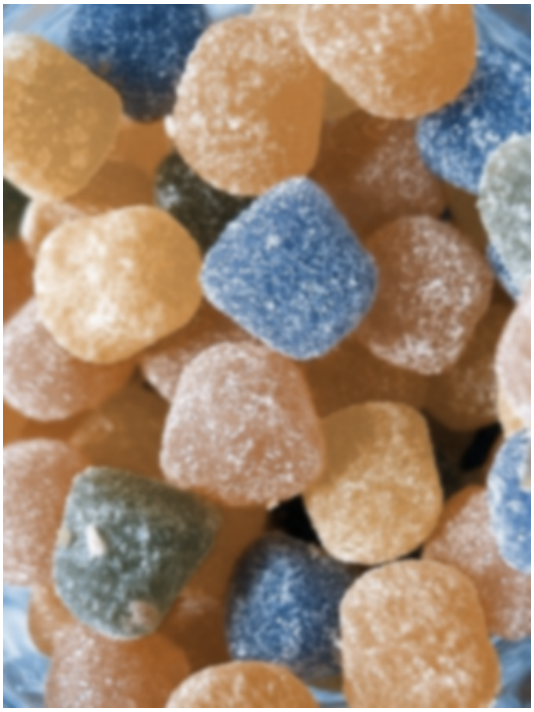
- **Gaussian**: `sigma=1.5`
- **Median**: `disk(5)` (radius 5 piksel)

Screenshot Hasil:

- Gambar Standar (`cameraman`):
    - Hasil Gaussian:

- Hasil Median:

- Gambar Pribadi (`Candy.jpg`):
  - Hasil Gaussian:



- Hasil Median:

Tabel Ringkas Output: Tabel berikut merangkum parameter yang diterapkan pada setiap gambar.

Filter	Gambar	Parameter

Gaussian	cameraman	σ = 1.5
Median	cameraman	radius = 5
Gaussian	coins	σ = 1.5
Median	coins	radius = 5
Gaussian	checkerboard	σ = 1.5
Median	checkerboard	radius = 5
Gaussian	astronaut	σ = 1.5
Median	astronaut	radius = 5
Gaussian	personal	σ = 1.5
Median	personal	radius = 5

Analisis Efek Parameter:

- Gaussian (sigma=1.5): Memberikan efek halus (blur) yang konsisten di semua gambar. Pada cameraman_gaussian.png, ini berhasil mereduksi noise pada area rumput dan langit, namun mengorbankan sedikit ketajaman pada tripod dan jaket.
- Median (radius=5): Memberikan efek yang sangat berbeda. Pada cameraman_median.png, filter ini menghilangkan detail tekstur halus (rumput) dan menciptakan efek seperti "lukisan cat minyak" (painterly effect), namun garis-garis tepi yang kuat (siluet) tetap terjaga dengan baik. Pada gambar personal_median.png, efek ini sangat jelas terlihat di mana tekstur gula pada permen menjadi halus.

II.2. Fitur 02: Edge Detection & Sampling

Penjelasan Singkat Teori: Deteksi tepi bertujuan untuk mengidentifikasi titik-titik dalam gambar dengan perubahan intensitas yang tajam. **Sobel** adalah filter turunan pertama yang menghitung gradien intensitas, menghasilkan *edge map* yang tebal dan sensitif terhadap *noise*. **Canny** adalah metode multi-tahap yang lebih canggih, melibatkan *Gaussian blur*, deteksi gradien, *non-maximum suppression*, dan *hysteresis thresholding* untuk menghasilkan tepi yang tipis (1 piksel) dan bersih.

Parameter yang Digunakan: Skrip `edge.py` menerapkan dua set parameter Canny untuk perbandingan.

- **Sobel**: (Parameter *default* `skimage`)
- **Canny (Low)**: `sigma=1.0`, `low_threshold=0.05`, `high_threshold=0.15`
- **Canny (High)**: `sigma=3.0`, `low_threshold=0.1`, `high_threshold=0.3`

Screenshot Hasil:

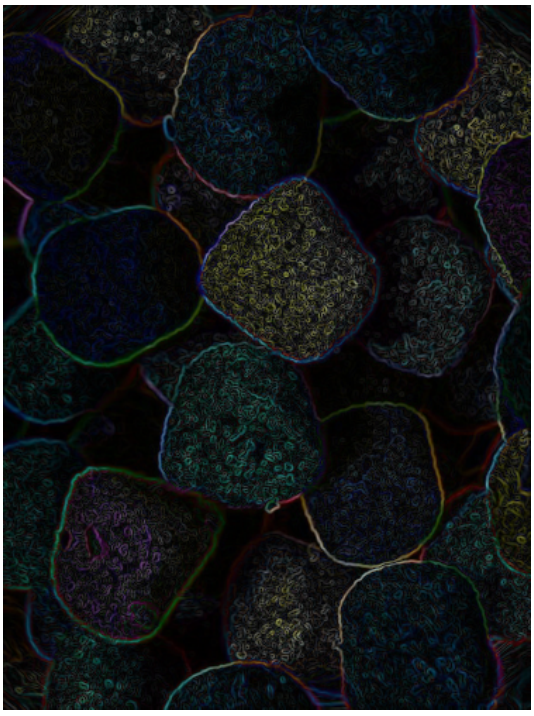
- Gambar Standar (`cameraman`):

- Hasil Sobel:

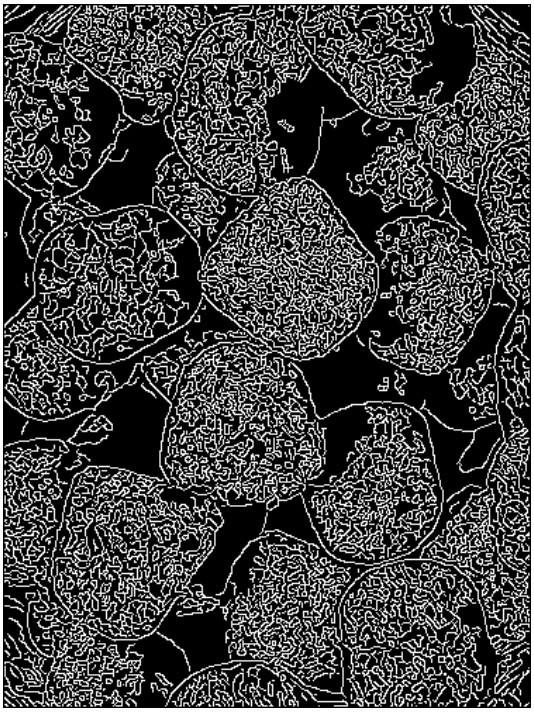
- Hasil Canny (Low):

- Hasil Canny (High):

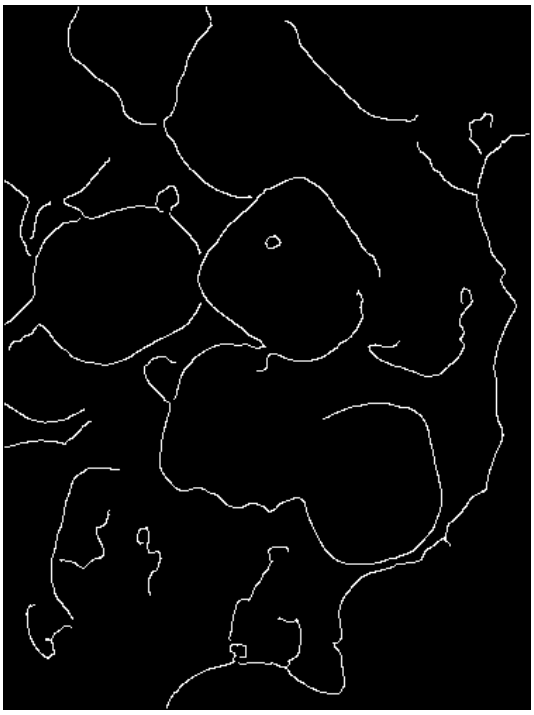
- Gambar Pribadi (`Candy.jpg`):
    - Hasil Sobel:



    - Hasil Canny (Low):



- Hasil Canny (High):



Tabel Ringkas Output: Tabel berikut merangkum efek perubahan parameter

Metode	Gambar	Low Threshold	High Threshold	Parameter / Sampling	Efek Hasil

Sobel	cameraman	-	-	Single channel	Gradien lokal, hasil halus tapi tepi tebal
Canny (Low)	cameraman	10	50	-	Menangkap lebih banyak detail, namun lebih noisy
Canny (High)	cameraman	50	150	-	Mengabaikan tepi lemah, hasil lebih bersih tapi kehilangan detail halus
Sobel	coins	-	-	Single channel	Gradien lokal, hasil halus tapi tepi tebal
Canny (Low)	coins	10	50	-	Menangkap lebih banyak detail, namun lebih noisy
Canny (High)	coins	50	150	-	Mengabaikan tepi lemah, hasil lebih bersih tapi kehilangan detail halus
Sobel	checkerboard	-	-	Single channel	Gradien lokal, hasil halus tapi tepi tebal
Canny (Low)	checkerboard	10	50	-	Menangkap lebih banyak detail, namun lebih noisy
Canny (High)	checkerboard	50	150	-	Mengabaikan tepi lemah, hasil lebih bersih tapi kehilangan detail halus
Sobel	astronaut	-	-	Sampling per channel	Gradien lokal, hasil halus tapi tepi tebal
Canny (Low)	astronaut	10	50	-	Menangkap lebih banyak detail, namun lebih noisy
Canny (High)	astronaut	50	150	-	Mengabaikan tepi lemah, hasil lebih bersih tapi kehilangan detail halus

Sobel	personal	-	-	Sampling per channel	Gradien lokal, hasil halus tapi tepi tebal
Canny (Low)	personal	10	50	-	Menangkap lebih banyak detail, namun lebih noisy
Canny (High)	personal	50	150	-	Mengabaikan tepi lemah, hasil lebih bersih tapi kehilangan detail halus

Analisis Efek Perubahan Parameter: Perbandingan antara Canny (Low) dan Canny (High) sangat jelas menunjukkan *trade-off* antara detail dan *noise*.

- **Analisis Sobel**: Filter Sobel, sebagai operator gradien orde pertama, menghasilkan *edge map* yang menunjukkan intensitas perubahan gradien. Seperti yang terlihat pada `cameraman_sobel.png`, Sobel berhasil mendeteksi tepi, namun hasilnya tidak bersih. Tepi yang dihasilkan tebal dan rentan terhadap *noise* (misalnya, tekstur rumput di `cameraman_sobel.png` masih muncul sebagai gradien lemah). Karena tidak ada mekanisme *thresholding* ganda seperti Canny, Sobel menangkap semua gradien, baik yang kuat (tepi) maupun yang lemah (*noise*). Pada gambar RGB pribadi (`personal_sobel.png`), hasilnya sangat bising dan sulit diinterpretasi karena filter diterapkan per *channel* warna pada tekstur gula yang padat.
- **Analisis Canny (Low vs. High)**: Perbandingan antara Canny (Low) dan Canny (High) sangat jelas menunjukkan *trade-off* antara detail dan *noise*.
    - Pada `cameraman_canny_low.png`, `sigma=1.0` (blur minimal) dan *threshold* rendah menyebabkan area rumput di latar depan terdeteksi sebagai jutaan tepi kecil (*noise*).
    - Pada `cameraman_canny_high.png`, `sigma=3.0` (blur lebih banyak) menghaluskan tekstur rumput, dan *threshold* yang lebih tinggi memastikan hanya tepi yang benar-benar kuat (siluet, tripod, gedung) yang terdeteksi. Hasilnya jauh lebih bersih.
    - Efek ini bahkan lebih ekstrem pada gambar pribadi. `personal_canny_low.png` meledak dengan *noise* karena tekstur gula pada setiap permen, sementara `personal_canny_high.png` berhasil mengisolasinya dan hanya menampilkan garis luar permen. Ini membuktikan bahwa Canny jauh lebih superior daripada Sobel dalam mengisolasi tepi yang sebenarnya dari *noise* tekstur.

II.3. Fitur 03: Feature Points

Penjelasan Singkat Teori: Deteksi *feature point* adalah proses mengidentifikasi titik-titik menonjol (minat) dalam gambar. **Harris** mendeteksi "sudut" (*corner*) di mana terdapat

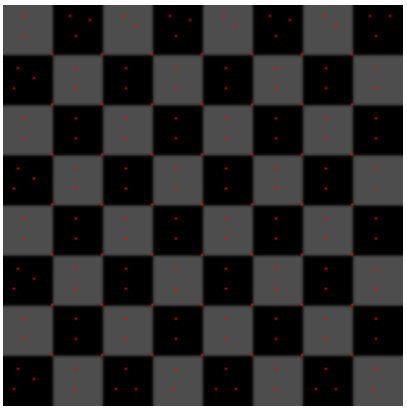
perubahan intensitas yang signifikan di semua arah. **FAST** (*Features from Accelerated Segment Test*) adalah pendeteksi sudut yang sangat cepat, ideal untuk aplikasi *real-time*. **SIFT** (*Scale-Invariant Feature Transform*) adalah algoritma canggih yang mendeteksi *keypoint* yang invarian terhadap penskalaan dan rotasi, membuatnya sangat kuat untuk pencocokan gambar.

Parameter yang Digunakan:

- **Harris**: `method='k'`, `k=0.05`
- **FAST**: (Parameter *default* `skimage`)
- **SIFT**: (Parameter *default* `skimage`)

Screenshot Hasil:

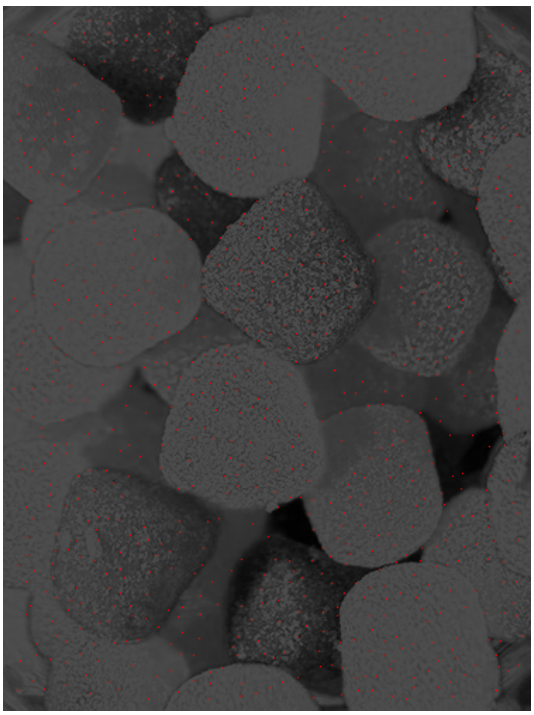
- Gambar Standar (`checkerboard` & `cameraman`):
    - Hasil Harris:



    - Hasil FAST:

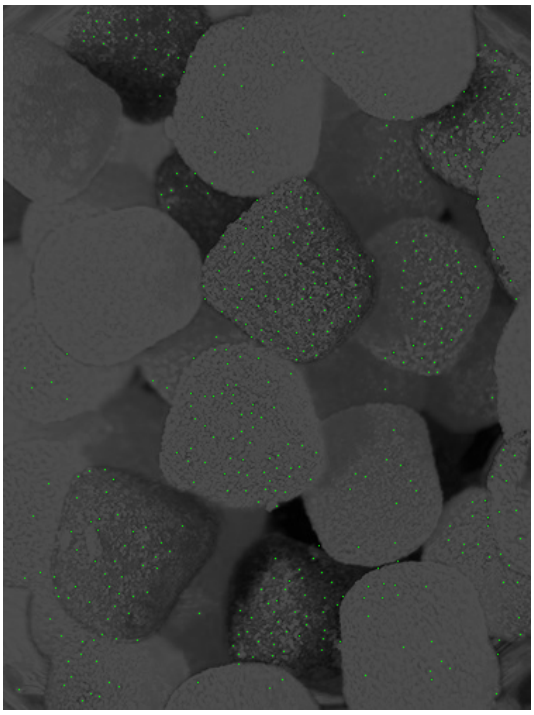
- Hasil SIFT:

- Gambar Pribadi (`Candy.jpg`):
  - Hasil Harris:

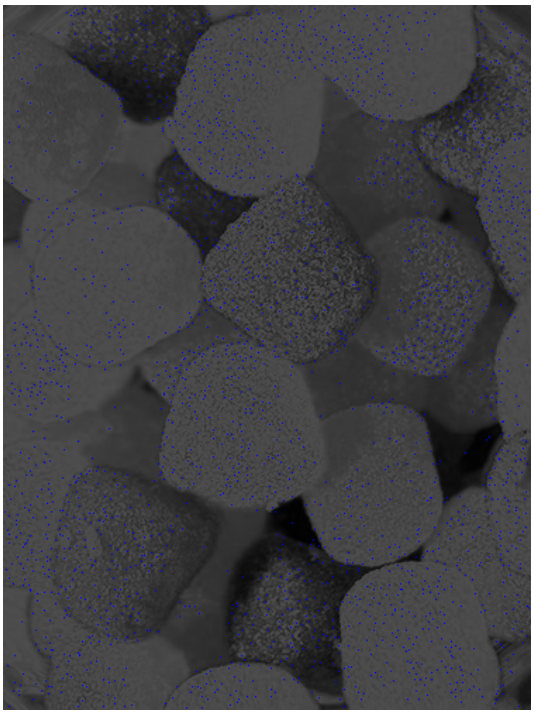


  -

- Hasil FAST:



- Hasil SIFT:



Tabel Ringkas Output: Tabel berikut merangkum statistik fitur yang diekstraksi dari semua gambar.

Metode	Gambar	Jumlah Fitur	Rata-rata Response	Maksimum Response
Harris	cameraman	1321	-0.008114	5.208771
FAST	cameraman	189	0.506120	-
SIFT	cameraman	882	-	-
Harris	coins	477	-0.010627	2.687926
FAST	coins	177	0.379826	-
SIFT	coins	755	-	-
Harris	checkerboard	188	-0.194911	8.094433
FAST	checkerboard	0	0.500000	-
SIFT	checkerboard	150	-	-
Harris	astronaut	1272	-0.009647	7.337651
FAST	astronaut	183	0.449408	-
SIFT	astronaut	1229	-	-
Harris	personal	854	0.011971	2.974127
FAST	personal	469	0.517832	-
SIFT	personal	2737	-	-

Analisis Efek Perubahan Parameter/Metode:

- **Harris** pada `checkerboard_harris.png` bekerja dengan sempurna, mendeteksi persimpangan kotak-kotak, sesuai dengan tujuannya sebagai *corner detector*.
- **FAST**, sebagai detektor yang dioptimalkan untuk kecepatan, menunjukkan performa yang baik pada gambar `cameraman`. Ia berhasil mendeteksi **189 fitur**, yang cenderung terkonsentrasi pada sudut-sudut struktural yang jelas, seperti pada tripod, kamera, fitur wajah, dan beberapa bangunan di latar belakang. Ini kontras dengan `checkerboard` di mana ia gagal mendeteksi fitur apa pun (Jumlah_Fitur: 0).
- **SIFT** secara konsisten mendeteksi banyak fitur. Poin paling menarik adalah pada gambar `personal`. Karena tekstur gula yang kaya dan acak, SIFT mendeteksi **2737 fitur**, jumlah tertinggi sejauh ini, menunjukkan kekuatannya dalam menemukan *keypoint* unik di area bertekstur tinggi. Sebaliknya, `cameraman` yang

memiliki banyak area halus (langit, rumput) menghasilkan fitur SIFT yang lebih sedikit (882).

II.4. Fitur 04: Geometry

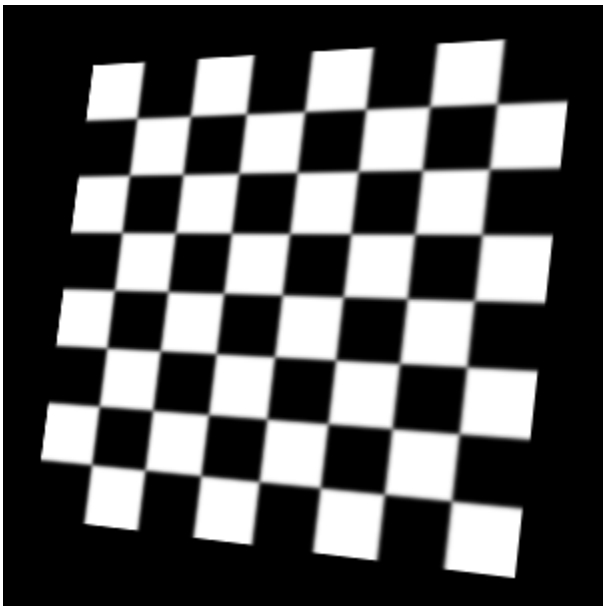
Penjelasan Singkat Teori: Geometri kamera mempelajari bagaimana dunia 3D diproyeksikan ke gambar 2D. **Homografi** (atau *Projective Transform*) adalah matriks 3x3 yang memetakan titik-titik dari satu bidang ke bidang lain. Ini sering digunakan untuk "meluruskan" gambar yang diambil dari sudut miring atau untuk mensimulasikan perubahan perspektif.

Parameter yang Digunakan: program tidak menggunakan parameter tetap, melainkan **mengestimasi** parameter.

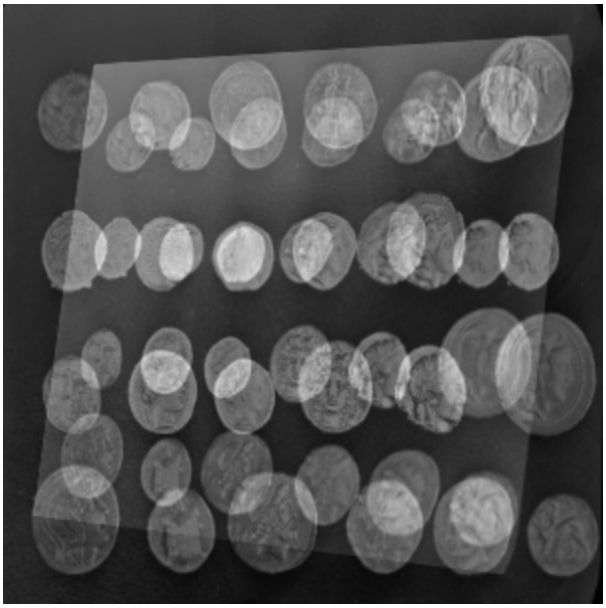
1. Empat titik sudut dari `data.checkerboard()` didefinisikan sebagai titik sumber (`src_points`).
2. Empat titik tujuan (`dst_points`) didefinisikan secara manual untuk mensimulasikan efek miring.
3. Sebuah matriks homografi (`tform`) diestimasi dari pasangan titik ini.
4. **Matriks yang sama ini kemudian diterapkan ke semua gambar uji.**

Screenshot Hasil:

- Gambar Standar (`checkerboard dan coins`):
    - Hasil Transformasi:



    - Hasil Overlay:



- Gambar Pribadi (`Candy.jpg`):
    - Hasil Transformasi:

    - Hasil Overlay:

Tabel Ringkas Output: Tabel berikut menunjukkan matriks homografi yang diestimasi.

Gambar	h00	h01	h02	h10	h11	h12	h20	h21	h22
cameraman	0.6460	-0.1014	45	-0.0584	0.7350	30	-0.0006	-0.0001	1
coins	0.6460	-0.1014	45	-0.0584	0.7350	30	-0.0006	-0.0001	1
astronaut	0.6460	-0.1014	45	-0.0584	0.7350	30	-0.0006	-0.0001	1
personal	0.6460	-0.1014	45	-0.0584	0.7350	30	-0.0006	-0.0001	1

Analisis Efek:

- Simulasi ini berhasil. `checker_transformed.png` menunjukkan warp perspektif yang diharapkan.
- Dengan menerapkan matriks yang sama ke gambar lain (misal `cameraman_transformed.png` dan `personal_transformed.png`), skrip ini secara efektif mensimulasikan pengambilan gambar yang berbeda (cameraman, permen) dari sudut kamera miring yang sama persis.
- Gambar *overlay* (misal `cameraman_overlay.png` dan `personal_overlay.png`) secara visual mengonfirmasi transformasi yang terjadi, menunjukkan gambar asli dan gambar yang telah di-*warp* secara transparan di atas satu sama lain.

III Komparasi dan Refleksi Pribadi

III.1. Komparasi Hasil (Gambar Standar vs. Gambar Bonus/Tambahan)

Perbandingan antara gambar standar (`cameraman`, `coins`, `checkerboard`, `astronaut`) dan gambar pribadi (`Candy.jpg`) memberikan wawasan penting tentang bagaimana karakteristik gambar memengaruhi setiap algoritma.

- **Karakteristik Gambar**:
    1. **Standar**: Umumnya *grayscale* (kecuali `astronaut`), memiliki kontras yang baik, dan subjek yang jelas (orang, koin, pola). Tekstur bervariasi, dari area halus (langit) hingga area bertekstur (rumput).
    2. **Pribadi (`Candy.jpg`)**: Berwarna (RGB), memiliki **tekstur frekuensi tinggi** yang sangat padat (kristal gula), dan terdiri dari banyak objek kecil yang tumpang tindih.
- **Perbandingan Fitur**:
    1. **Filtering**: Efek *blur* dari **Gaussian** terlihat serupa di kedua jenis gambar, hanya menghaluskan. Namun, **Median Filter** memiliki efek yang jauh lebih dramatis pada gambar pribadi. Tekstur gula yang halus "luntur" menjadi bercak-bercak warna (`personal_median.png`), menunjukkan bagaimana filter median menghilangkan detail halus yang dianggapnya *noise*.
    2. **Edge Detection**: Di sinilah perbedaan paling mencolok. Pada `cameraman`, *noise* Canny (Low) terbatas pada area rumput. Pada `personal_canny_low.png`, seluruh gambar meledak dalam deteksi tepi karena tekstur gula. Ini membuktikan bahwa parameter `sigma` dan *threshold* Canny **mutlak harus** disesuaikan secara drastis untuk gambar bertekstur tinggi agar dapat menghasilkan *output* yang bermakna (seperti `personal_canny_high.png`).
    3. **Feature Points**: Gambar `personal` adalah "tambang emas" untuk detektor fitur berbasis tekstur seperti **SIFT**. Ia menghasilkan **2737** fitur, hampir 3 kali lipat dari `cameraman` (882). Sebaliknya, gambar `checkerboard` yang sangat terstruktur dan bersih menghasilkan fitur SIFT yang relatif sedikit (150) dan 0 fitur FAST. Ini menunjukkan bahwa `checkerboard` baik untuk *corner detector* (Harris), tetapi `personal` lebih baik untuk *feature descriptor* (SIFT).

III.2. Refleksi Pribadi

Dalam pengerjaan tugas ini, beberapa pilihan desain implementasi diambil untuk memastikan kode yang modular, hasil yang komprehensif, dan analisis yang kuat.

- **Solusi Penanganan Gambar RGB dan Grayscale**: Gambar standar bervariasi (ada *grayscale* dan RGB), dan gambar pribadi juga RGB. Skrip saya dirancang untuk menangani kedua kasus ini secara otomatis:

    - Pada `filtering.py` (Median) dan `edge.py` (Sobel), saya mendeteksi `img.ndim == 3`. Jika ya, saya mengulangi (*looping*) filter pada setiap *channel* warna secara terpisah.
    - Pada `edge.py` (Canny) dan `featurepoints.py`, saya mengkonversi gambar RGB menjadi *grayscale* menggunakan `np.mean(img, axis=2)` sebelum diproses, karena algoritma ini beroperasi pada gambar *single-channel*.
- **Kendala yang Dihadapi**:
    - Saat implementasi SIFT, saya menemukan bahwa API `skimage.feature.SIFT` mungkin tidak selalu mengembalikan atribut `.responses`. Kode di `featurepoints.py` menyertakan logika untuk menangani ini (`hasattr(sift, "responses")`) dan mengisi `np.nan` di CSV jika respons tidak tersedia, yang terlihat pada *output* `statistik_feature.csv`.
    - Hasil 0 fitur untuk FAST pada `checkerboard` adalah kendala yang menarik, yang menunjukkan bahwa parameter *default* tidak selalu berfungsi untuk setiap jenis gambar (terutama yang sangat bersih dan sintetik).

Link Repository Github

https://github.com/saadabha/SaadAbdulHakim_13522092_IF5152_TugasIndividuCV

Link Sumber Gambar Bonus/Tambahan

https://www.pinterest.com/pin/458382068302004099/