Metody próbkowania sygnałów i obrazów - Aliasing 2D

ArtsyKimiko

23 października 2023

1 Cel ćwiczenia

Celem niniejszego zadania jest zrozumienie i ilustrowanie zjawiska aliasingu w kontekście obiektów ruchomych. Aliasowanie jest zjawiskiem, w którym informacje o obiekcie lub zjawisku są źle próbkowane, co prowadzi do zniekształceń w wynikowym obrazie lub sygnale. W ramach tego zadania zostanie wykorzystane obracające się śmigło jako przykład obiektu ruchomego oraz sensor o odczycie sekwencyjnym do rejestracji obrazów.

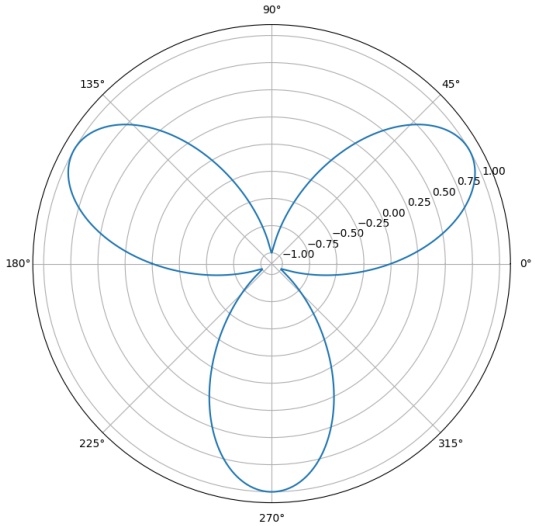
2 Analiza Generowania Sekwencji i Animacji Obracającego się Śmigła

2.1 Generowanie Sekwencji Obrazów Śmigła

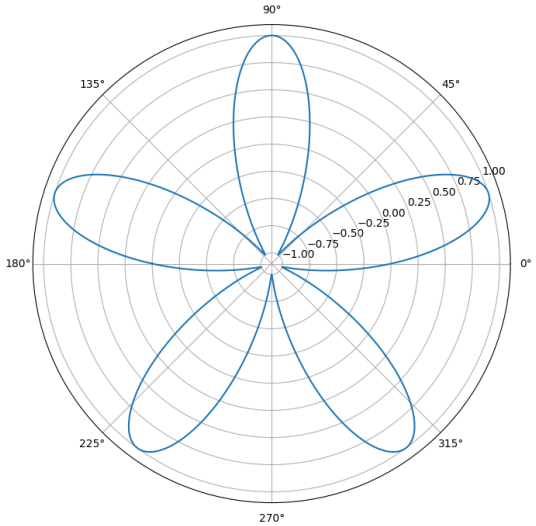
Do wygenerowania sekwencji obrazów przedstawiających obracające się śmigło zostanie użyta funkcja `propeller`. Ta funkcja generuje krzywą, która opisuje ruch śmigła na podstawie określonej liczby łopatek i fazy. W naszym przypadku liczba łopatek wynosi 3 i 5. Krzywa ta jest matematycznie opisana, określając, jak zmienia się kształt śmigła w zależności od kąta obrotu.

```
def propeller(theta, m):
    return np.sin(N*theta + m * np.pi / 10)
```

Następnie zostanie użyty wykres w współrzędnych polar (`plt.polar`), aby wizualizować tę krzywą i zobaczyć, jak zmienia się kształt śmigła w zależności od fazy obrotu.



Rysunek 1: Wizualizacja zmiany kształtu śmigła z 3 łopatkami w zależności od fazy obrotu.



Rysunek 2: Wizualizacja zmiany kształtu śmigła z 5 łopatkami w zależności od fazy obrotu.

2.2 Animacja Obracającego się Śmigła

Do stworzenia animacji obracającego się śmigła wykorzystywana jest biblioteka 'FuncAnimation', która umożliwia generowanie animacji na podstawie sekwencji klatek. Funkcja 'animate' jest odpowiedzialna za aktualizację wykresu z obrazem śmigła dla każdej klatki animacji. W miarę zmiany zmiennej 'frame', animacja przedstawia obracające się śmigło.

```
figure = plt.figure(figsize=[8, 8])
plot = plt.polar([])[0]

def animate(frame):
    propeller_curve = propeller(thetas, m=frame)
    plot.set_data((thetas, propeller_curve))

animation = FuncAnimation(figure, animate, frames=100, interval=25)
```

Ten krok jest ważny, ponieważ pozwala na wizualizację obiektu ruchomego, w tym przypadku obracającego się śmigła.

Jest to również istotne z punktu widzenia zrozumienia, w jaki sposób obiekt jest próbkowany i jak różne fazy obrotu wpływają na jego wygląd.

3 Zamiana Współrzędnych Biegunowych na Kartezjańskie

W eksperymencie konieczna jest zamiana współrzędnych biegunowych opisujących obracające się śmigło na współrzędne kartezjańskie, aby umożliwić rejestrację obrazów na matrycy sensora. Funkcja `polar_to_cartesian` przekształca współrzędne biegunowe (kąt i promień) na współrzędne kartezjańskie (x i y). To kluczowe działanie pozwala na dokładne określenie położenia punktów w dwuwymiarowej przestrzeni, która jest reprezentowana przez matrycę sensora.

```
def polar_to_cartesian(theta, r):
    x = r * \cos(\theta)
    y = r * \sin(\theta)
    return \column_stack([x, y])
```

4 Funkcja Symulacji Sensora `capture`

Funkcja `capture` symuluje proces rejestracji obrazów przez sensor CMOS. Przyjmuje ona jako argumenty:

- `func` - tablicę zawierającą dane punktów funkcji matematycznej opisującej obiekt (w naszym przypadku, śmigło).
- `resolution` - rozdzielczość matrycy sensora.
- `threshold` - próg odległości, który określa, czy punkt na obrazie jest rejestrowany przez sensor.
- `low` i `high` - granice zakresu przestrzeni, w której dokonywane są pomiary.

Wewnętrznie, funkcja tworzy siatkę punktów o zadanej rozdzielczości, reprezentujących obszar matrycy sensora. Następnie oblicza odległości między punktami tej siatki a punktami obiektu `func` za pomocą funkcji `distance_matrix`. Jeśli odległość pomiędzy punktem na matrycy a najbliższym punktem na obiekcie jest mniejsza lub równa zadanemu progu (`threshold`), punkt na matrycy jest uznawany za rejestrowany.

Wynikiem funkcji `capture` jest macierz reprezentująca obraz rejestrowany przez sensor. Wartości w tej macierzy określają, które punkty na matrycy są rejestrowane (1) i które nie (0).

```
def capture(func: np.array, resolution: int, threshold: float = 0.1, low: float = -1,
    high: float = 1) -> np.array:
    grid_x, grid_y = np.meshgrid(np.linspace(low, high, resolution), np.linspace(low,
        high, resolution))
    grid = np.column_stack([grid_x.flatten(), grid_y.flatten()])

    distances = distance_matrix(grid, func)
    capture = (np.min(distances, axis=1) <= threshold).astype(int).reshape(resolution,
        resolution)

    return capture
```

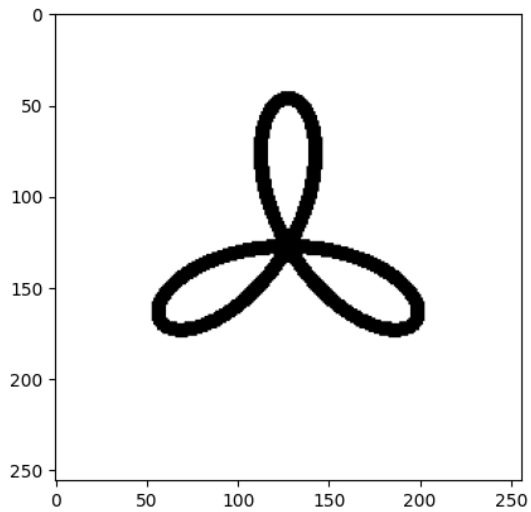
Ostatecznie, funkcja ta umożliwiła symulację procesu rejestracji obrazów przez sensor i zilustrowała, w jaki sposób rozdzielczość i próg odległości wpływają na jakość zarejestrowanych obrazów. Ta analiza odgrywała kluczową rolę w zrozumieniu zjawiska aliasingu w kontekście obiektów ruchomych.

5 Analiza Działania Sensora i Rejestracji Obrazów

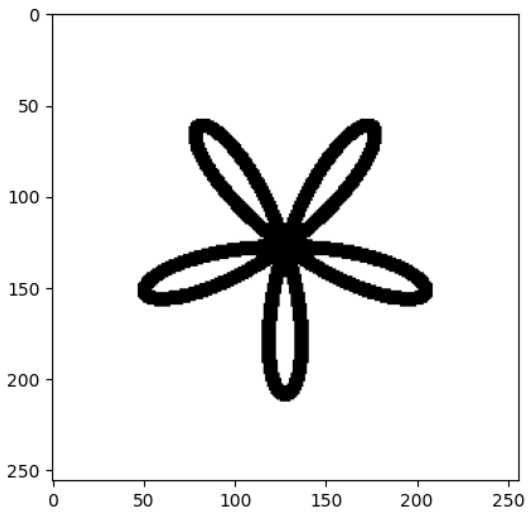
Kolejnym kluczowym etapem eksperymentu jest analiza działania sensora i procesu rejestracji obrazów śmigła w różnych fazach obrotu. W analizowanym fragmencie kodu dokładnie rozważany jest proces akwizycji danych oraz tworzenia sekwencji obrazów.

Pierwszym krokiem jest wywołanie funkcji `capture`, która symuluje akwizycję danych przez sensor. Parametry tej funkcji, takie jak rozdzielczość (`resolution`), próg odległości (`threshold`) oraz zakres przestrzeni (`low` i `high`), mają istotne znaczenie dla procesu akwizycji. Wynik tej funkcji to obraz, który ilustruje, jak sensor rejestruje obraz obiektu w zadanym środowisku.

```
_ = plt.imshow(capture(polar_to_cartesian(thetas, rs), resolution=256, threshold=0.05,
    low=-\pi / 2, high=\pi / 2), cmap="Greys")
```



Rysunek 3: Sekwencja obrazów przedstawiających obracające się śmigło z 3 łopatkami.



Rysunek 4: Sekwencja obrazów przedstawiających obracające się śmigło z 5 łopatkami.

Następnie zostają przygotowane sekwencje obrazów śmigła w różnych fazach obrotu. Dostępne dane dotyczące faz (`ms`) oraz funkcja `propeller` są wykorzystywane do generowania odpowiednich obrazów śmigła dla każdej fazy. Zauważalne jest, że zmiany faz obrotu wpływają na kształt i wygląd obrazów śmigła.

```
thetas = np.linspace(-\pi, \pi, 1000)
ms = np.arange(-M // 2, M // 2)
funcs = [ ]

for m in ms.tolist():
    r = propeller(thetas, m=m)
    func = polar_to_cartesian(thetas, r)
    funcs.append(func)
```

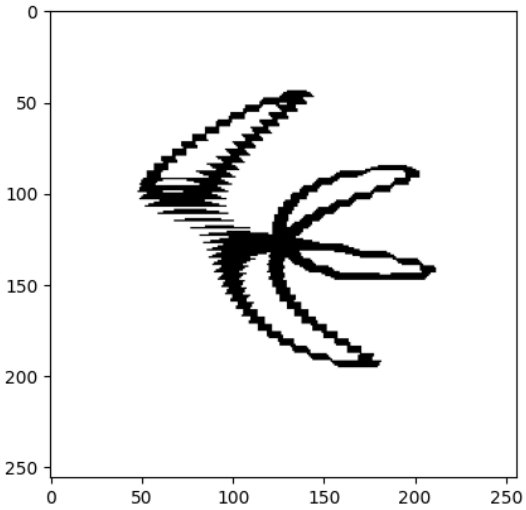
Ostatecznie, zebrane dane reprezentujące sekwencje obrazów są przekształcane w tablicę NumPy w celu dalszej analizy. To pozwala na efektywne przechowywanie i manipulowanie danymi.

```
    funcs = np.asarray(funcs)
```

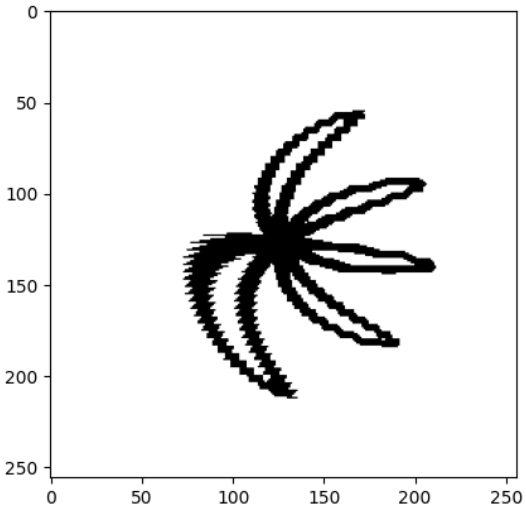
Funkcja `record` jest następnie wykorzystywana do symulacji procesu rejestracji obrazów. Funkcja ta pobiera listę sekwencji obrazów (`funcs`) oraz argumenty związane z działaniem sensora (przekazywane jako `capture_kwargs`). Wewnątrz funkcji `record` każda sekwencja obrazów jest przetwarzana przy użyciu funkcji `capture` z wcześniej zdefiniowanymi parametrami.

```
def record(funcs: list, capture_kwargs) -> np.array:
    """Simulate recording by applying capture in loop"""
    return np.asarray([capture(func, **capture_kwargs) for func in progress_bar(funcs)
        ])
```

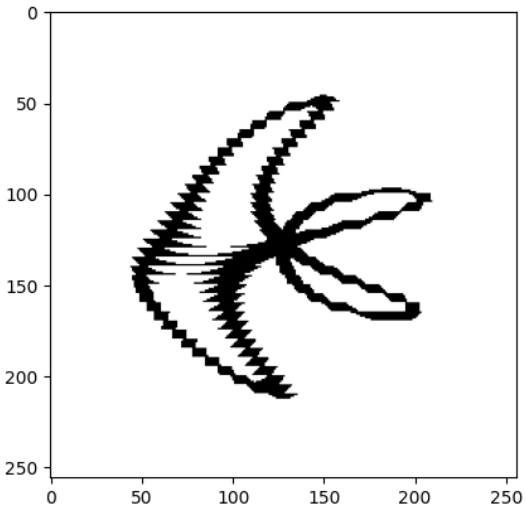
Dzięki temu procesowi uzyskuje się sekwencję obrazów, która reprezentuje to, co sensor widziałby w trakcie różnych faz obrotu śmigła.



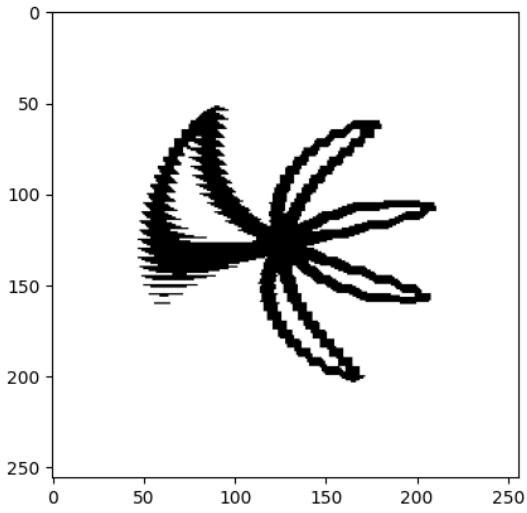
Rysunek 5: Pierwsza sekwencja obrazów przedstawiających obracające się śmigło z 3 łopatkami.



Rysunek 6: Pierwsza sekwencja obrazów przedstawiających obracające się śmigło z 5 łopatkami.



Rysunek 7: Druga sekwencja obrazów z 3 łopatkami.



Rysunek 8: Druga sekwencja obrazów z 5 łopatkami.

6 Wnioski

6.1 Przyczyna Powstania Zniekształceń

Zniekształcenia w obrazach rejestrowanych przez sensor wynikają z aliasingu, wynikającego z ograniczeń w próbkowaniu obiektów ruchomych. Główną przyczyną aliasingu jest ograniczona rozdzielczość sensora oraz dynamiczny ruch obiektów, jak obracające się śmigło.

Ograniczona rozdzielczość sensora oznacza, że nie jest w stanie precyzyjnie uchwycić każdego szczegółu obiektu. W przypadku obiektów ruchomych, takich jak obracające się śmigło, próbkowanie jest wykonywane w sposób dyskretny, co prowadzi do utraty ważnych informacji o dokładnym kształcie obiektu w ruchu. Ostatecznie, obraz może zawierać błędne informacje o fazie ruchu i liczbie łopatek.

W rezultacie obraz rejestrowany przez sensor może wykazywać zniekształcenia, takie jak migotanie, zniekształcone łopatki śmigła lub nawet utratę fragmentów obrazu. Zrozumienie tego zjawiska jest kluczowe przy projektowaniu sensorów oraz analizie obrazów obiektów ruchomych. Rozwiązania, takie jak zwiększanie rozdzielczości sensora, stosowanie antyaliasingu oraz techniki przetwarzania obrazów, pomagają minimalizować te zniekształcenia.

6.2 Propozycje Rozwiązania Problemu Aliasingu

Aby rozwiązać problem aliasingu w rejestrowaniu obiektów ruchomych, można podjąć następujące kroki:

- **Zwiększenie Rozdzielczości Sensora:** Jednym z najskuteczniejszych sposobów jest zwiększenie rozdzielczości sensora. Wyższa rozdzielczość pozwala na dokładniejsze próbkowanie obiektu i redukuje zniekształcenia. Jednak to rozwiązanie może wiązać się z większym zapotrzebowaniem na zasoby obliczeniowe.
- **Antyaliasing:** Stosowanie technik antyaliasingu podczas akwizycji danych może pomóc w redukcji aliasingu. Techniki te polegają na wygładzaniu próbkowanego sygnału, co pozwala na zachowanie dokładniejszych informacji o obiekcie w ruchu.
- **Ulepszona Optyka Sensora:** Poprawienie optyki sensora, takie jak zwiększenie ostrości obiektywu, może pomóc w redukcji zniekształceń w wynikowym obrazie.
- **Filtracja Post-Processing:** Po rejestrowaniu obrazów, można zastosować filtrację i techniki przetwarzania obrazu w celu usunięcia zniekształceń aliasingu.

6.3 Funkcja Dla Dowolnej Liczby Śmigieł

Funkcja, dla której liczba śmigieł n może być dowolna, ma postać:

$$f(x) = \sin(nx + m \cdot \pi/10)$$

Gdzie:

- n to liczba łopatek śmigła. Ta wartość może być dostosowywana w zależności od potrzeb, co pozwala na symulowanie różnych rodzajów śmigieł.
- m to przesunięcie fazowe, które również może być dostosowywane, co pozwala na kontrolowanie fazy ruchu, co jest szczególnie ważne w przypadku rejestrowania obiektów o zmiennym ruchu.

Podsumowując, zrozumienie przyczyny aliasingu i proponowane rozwiązania pozwalają na efektywne zarządzanie zniekształceniami w obrazach obiektów ruchomych. Elastyczna funkcja dla dowolnej liczby śmigieł umożliwia eksperymentowanie i dostosowanie do różnych scenariuszy symulacji.