Politechnika Śląska

28 listopada 2024,
Gliwice

Laboratorium ZAOWR

Rok akademicki	Termin	Rodzaj studiów	Kierunek	Prowadzący	Grupa
2024/2025	Czwartek, 10:00-13:15	Stacjonarne	Informatyka	Dr inż. Marcin Paszkuta Mgr inż. Mateusz Płonka	IGT

Ćwiczenie nr 2

Temat: Kalibracja systemu kamer stereo

Informatyka, IGT

Maksymilian Kisiel

Spis treści

1 Zadanie 1 - kalibracja stereo

Do wykrywania wzorca wykorzystane zostały zdjęcia dostarczone przez prowadzącego, a dokładniej zdjęcia z katalogu "*ZAOWiR Image set - Calibration/Chessboard/Stereo 2/*". Kod programu wykonujący to zadanie widoczny jest poniżej (Kod 1).

```
if __name__ == "__main__":
    from zaowr_polsl_kisiel import stereo_calibration, stereo_rectify,
    are_params_valid

    # CALIBRATION IMAGES
    left_cam = r"./ZAOWiR Image set - Calibration/Chessboard/Stereo 2/cam2/"
    right_cam = r"./ZAOWiR Image set - Calibration/Chessboard/Stereo 2/cam3/"

    # SAVED CALIBRATION FILES
    left_cam_params = "./tests/calibration_params/calibration_params_left.json"
    right_cam_params = "./tests/calibration_params/calibration_params_right.json"
    stereo_cam_params = "./tests/calibration_params/stereo_calibration_params.json"

    # CHECK IF THE PROVIDED FILES CONTAIN VALID PARAMS
    left_valid, params_left = are_params_valid(left_cam_params)
    right_valid, params_right = are_params_valid(right_cam_params)
    stereo_valid, stereo_params = are_params_valid(stereo_cam_params)

    # RUN THE CALIBRATION IF THE FILES DON'T EXIST OR CONTAIN INVALID VALUES
    if not left_valid or not right_valid or not stereo_valid:
        stereo_calibration(
            chessBoardSize=(10, 7),
            squareRealDimensions=50.0,
            calibImgDirPath_left=left_cam,
            calibImgDirPath_right=right_cam,
            globImgExtension="png",
            saveCalibrationParams=True,
            calibrationParamsPath_left=left_cam_params,
            calibrationParamsPath_right=right_cam_params,
            saveStereoCalibrationParams=True,
            stereoCalibrationParamsPath=stereo_cam_params,
        )

        # Revalidate parameters after calibration
        left_valid, params_left = are_params_valid(left_cam_params)
        right_valid, params_right = are_params_valid(right_cam_params)
        stereo_valid, stereo_params = are_params_valid(stereo_cam_params)

        # Check again to ensure parameters are valid
        if not left_valid or not right_valid or not stereo_valid:
            raise RuntimeError("Calibration failed. Parameters are still invalid.")
```

Kod 1: Fragment kodu odpowiedzialny za kalibrację stereo

2 Zadanie 2 - wyznaczanie odległości bazowej

Za pomocą kodu przedstawionego poniżej (Kod 13), wyznaczana jest odległość bazowa wewnątrz funkcji **stereo_calibrate()**, która następnie zapisywana jest do pliku **JSON**. Dla kamer **cam2** i **cam3** jest to 15.31 cm, a dla kamer **cam1** i **cam4** jest to 45.26 cm.

```
image_size = grayImg_left.shape[::-1]  # (width, height)

ret, CM1, dist1, CM2, dist2, R, T, E, F = cv.stereoCalibrate(
    objPoints,  # Object points
    imgPoints_left,  # Image points from the left camera
    imgPoints_right,  # Image points from the right camera
    cameraMatrix_left,  # Camera matrix for the left camera
    distortionCoefficients_left,  # Distortion coefficients for the left camera
    cameraMatrix_right,  # Camera matrix for the right camera
    distortionCoefficients_right,  # Distortion coefficients for the right camera
    image_size,  # Image size (assumes both cameras have the same resolution)
    criteria=terminationCriteria,  # Termination criteria
    flags=stereoCalibrationFlags  # Stereo calibration flags
)

baseline = np.round((np.linalg.norm(T)) * 0.1, 2) # baseline in cm
```

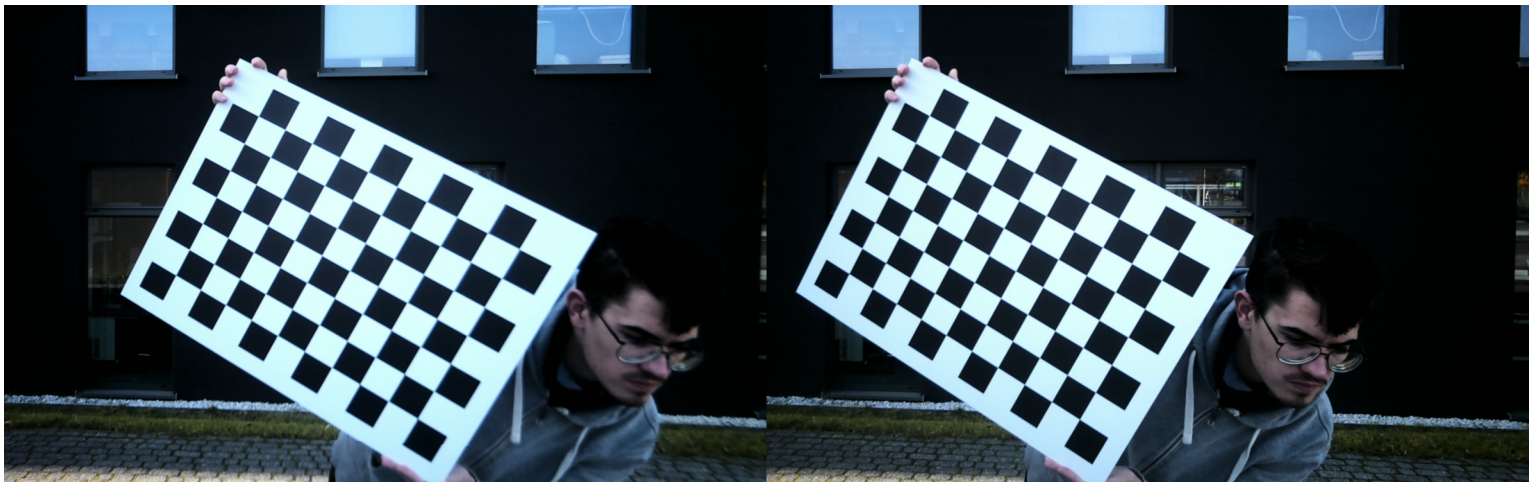
Kod 2: Fragment funkcji **stereo_calibrate()** odpowiedzialny za wyznaczanie odległości bazowej

3 Zadanie 3 - rektyfikacja obrazów

W tym zadaniu wykonano rektyfikację obrazu **28.png** z kamery **cam2** oraz **cam3** za pomocą funkcji **stereo_rectify()**, której wywołanie znajduje się poniżej (Kod 3). Wyniki działania programu widoczne są na poniższym zdjęciu (Rys. 1).

```
if __name__ == "__main__":
    from zaowr_polsl_kisiel import stereo_rectify, are_params_valid

    left_cam = r"./ZAOWiR Image set - Calibration/Chessboard/Stereo 2/cam2/"
    right_cam = r"./ZAOWiR Image set - Calibration/Chessboard/Stereo 2/cam3/"

    # SAVED CALIBRATION FILES
    left_cam_params = "./tests/calibration_params/calibration_params_left.json"
    right_cam_params = "./tests/calibration_params/calibration_params_right.json"
    stereo_cam_params = "./tests/calibration_params/stereo_calibration_params.json"

    # CHECK IF THE PROVIDED FILES CONTAIN VALID PARAMS
    left_valid, params_left = are_params_valid(left_cam_params)
    right_valid, params_right = are_params_valid(right_cam_params)
    stereo_valid, stereo_params = are_params_valid(stereo_cam_params)

    rectified_images_dir = "./tests/rectified_images"

    stereo_rectify(
        calibImgDirPath_left=left_cam,
        calibImgDirPath_right=right_cam,
        imgPoints_left=params_left["imgPoints"],
        imgPoints_right=params_right["imgPoints"],
        loadStereoCalibrationParams=True,
        stereoCalibrationParamsPath=stereo_cam_params,
        saveRectifiedImages=True,
        rectifiedImagesDirPath=rectified_images_dir,
        whichImage=0,
        drawEpipolarLinesParams=(20, 3, 2),
    )
```

Kod 3: Fragment kodu odpowiedzialny za rektyfikację



Rys. 1: Zrektyfikowana para obrazów **28.png** z kamery **cam2** oraz **cam3**

4 Zadanie 4 - różne metody interpolacji

W tym zadaniu wykorzystano różne metody interpolacji do rektyfikacji obrazów. W kolejnych sekcjach przedstawione zostaną otrzymane wyniki czasowe (w sekundach), jak i same obrazy. Aby wykonać to zadanie, zmieniono sposób wywołania funkcji **stereo_rectify()**, a dokładniej dodano flagę **testInterpolationMethods=True**. Zmodyfikowane wywołanie funkcji znajduje się poniżej (Kod 4).

Wykorzystane metody interpolacji:

- INTER_NEAREST (Sekcja 4.1),
- INTER_LINEAR (Sekcja 4.2),
- INTER_CUBIC (Sekcja 4.3),
- INTER_AREA (Sekcja 4.4),
- INTER_LANCZOS4 (Sekcja 4.5).

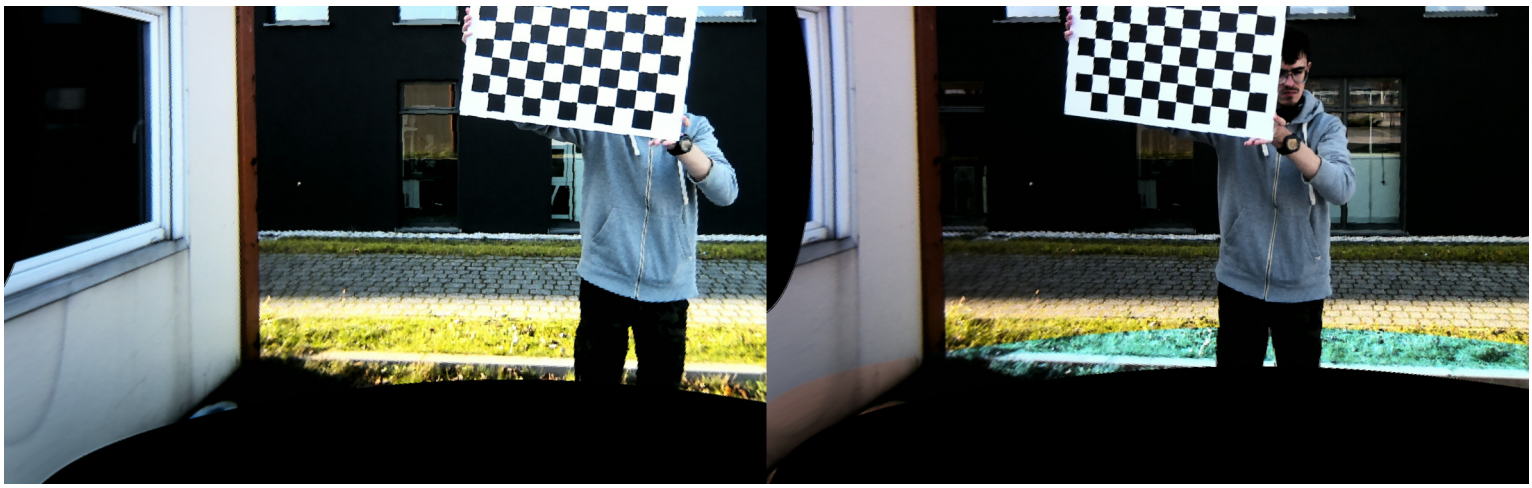
```
if __name__ == "__main__":
    from zaowr_polsl_kisiel import stereo_rectify, are_params_valid

    left_cam = r"./ZAOWiR Image set - Calibration/Chessboard/Stereo 2/cam1/"
    right_cam = r"./ZAOWiR Image set - Calibration/Chessboard/Stereo 2/cam4/"

    # SAVED CALIBRATION FILES
    left_cam_params = "./tests/calibration_params/calibration_params_left.json"
    right_cam_params = "./tests/calibration_params/calibration_params_right.json"
    stereo_cam_params = "./tests/calibration_params/stereo_calibration_params.json"

    # CHECK IF THE PROVIDED FILES CONTAIN VALID PARAMS
    left_valid, params_left = are_params_valid(left_cam_params)
    right_valid, params_right = are_params_valid(right_cam_params)
    stereo_valid, stereo_params = are_params_valid(stereo_cam_params)

    rectified_images_dir = "./tests/rectified_images"

    stereo_rectify(
        calibImgDirPath_left=left_cam,
        calibImgDirPath_right=right_cam,
        imgPoints_left=params_left["imgPoints"],
        imgPoints_right=params_right["imgPoints"],
        loadStereoCalibrationParams=True,
        stereoCalibrationParamsPath=stereo_cam_params,
        testInterpolationMethods=True, # NEW FLAG
        saveRectifiedImages=True,
        rectifiedImagesDirPath=rectified_images_dir,
        whichImage=0,
        drawEpipolarLinesParams=(20, 3, 2),
    )
```

Kod 4: Fragment kodu odpowiedzialny za rektyfikację

4.1 Metoda INTER_NEAREST

```
Interpolation type INTER_NEAREST:
  left_image: 0.0027368090004529 22
  right_image: 0.0054708329998902 63
  total: 0.0082076420003431 85
```

Kod 5: Czas obliczeń metody **INTER_NEAREST**

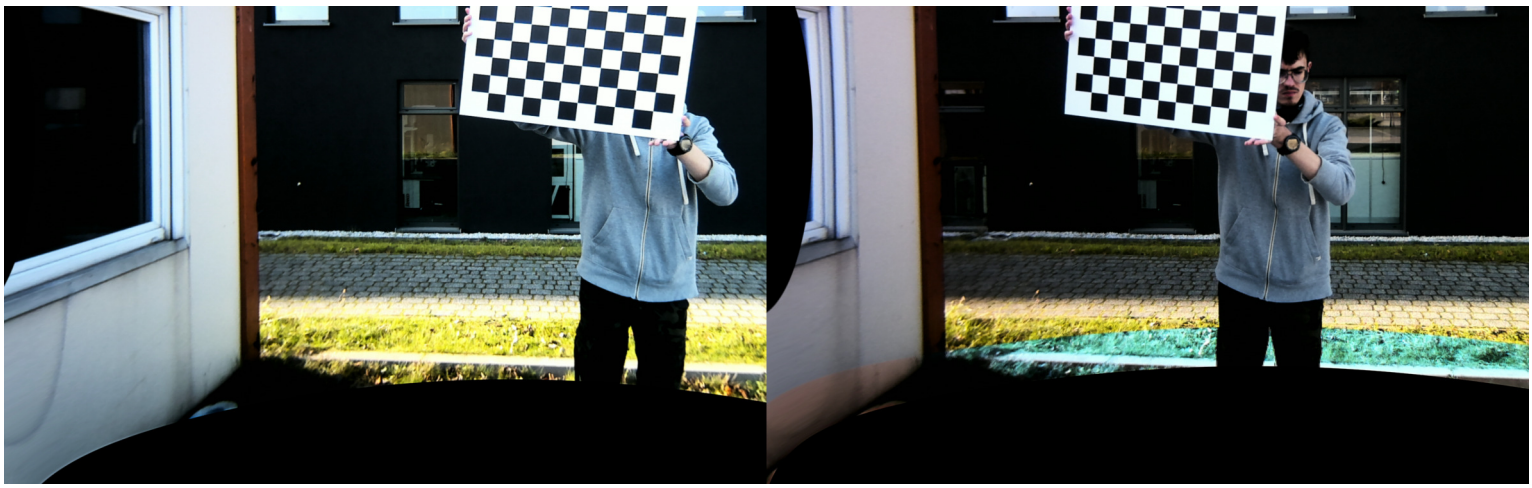


Rys. 2: Zrektyfikowana para obrazów **28.png** z kamery **cam1** oraz **cam4** metodą **INTER_NEAREST**

4.2 Metoda INTER_LINEAR

```
Interpolation type INTER_LINEAR:
  left_image: 0.006343457000184571
  right_image: 0.0028934239999216516
  total: 0.009236881000106223
```

Kod 6: Czas obliczeń metody **INTER_LINEAR**

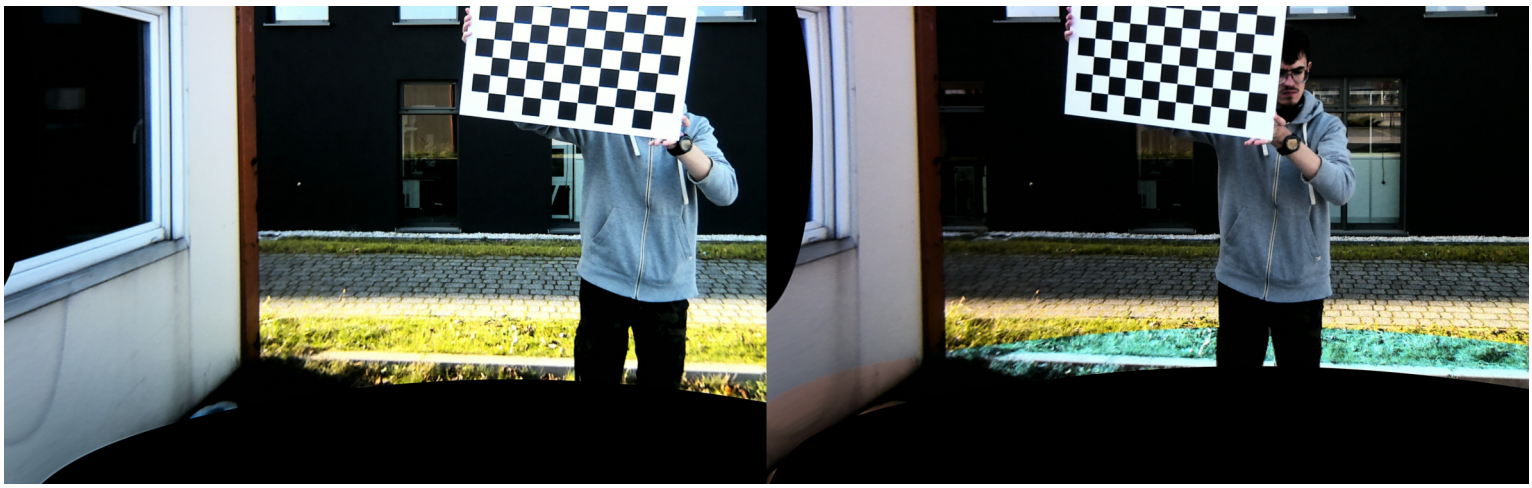


Rys. 3: Zrektyfikowana para obrazów **28.png** z kamery **cam1** oraz **cam4** metodą **INTER_LINEAR**

4.3 Metoda INTER_CUBIC

```
Interpolation type INTER_CUBIC:
  left_image: 0.009495449000496592
  right_image: 0.006943047999811824
  total: 0.016438497000308416
```

Kod 7: Czas obliczeń metody **INTER_CUBIC**

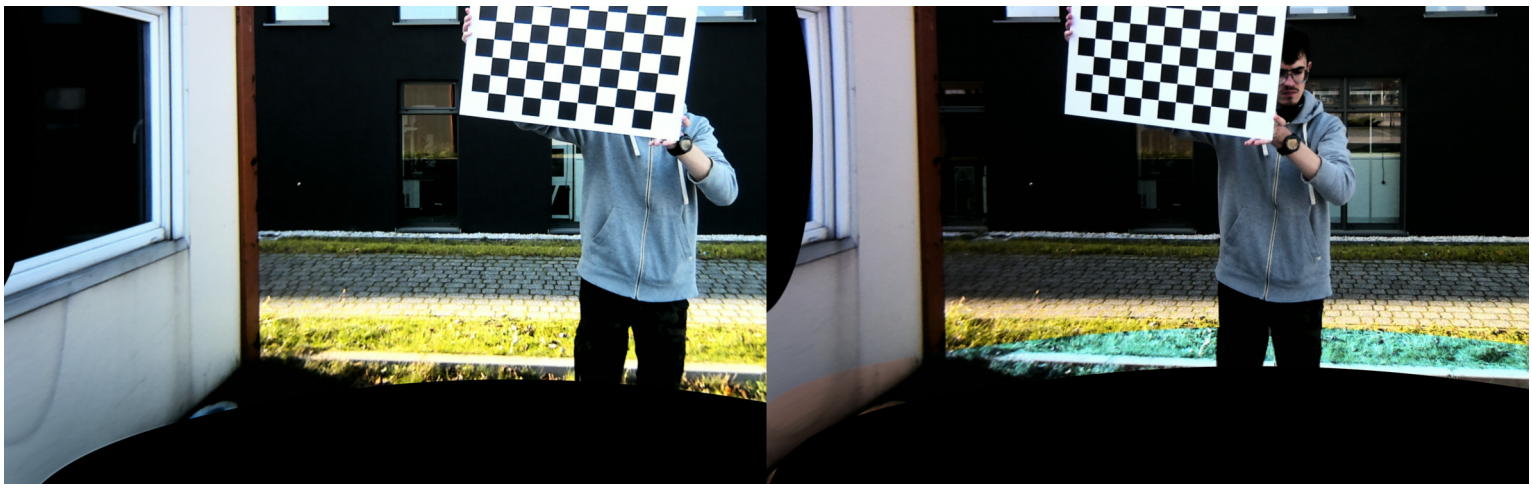


Rys. 4: Zrektyfikowana para obrazów **28.png** z kamery **cam1** oraz **cam4** metodą **INTER_CUBIC**

4.4 Metoda INTER_AREA

```
Interpolation type INTER_AREA:
  left_image: 0.002629888000228675
  right_image: 0.0018820269997377181
  total: 0.004511914999966393
```

Kod 8: Czas obliczeń metody **INTER_AREA**

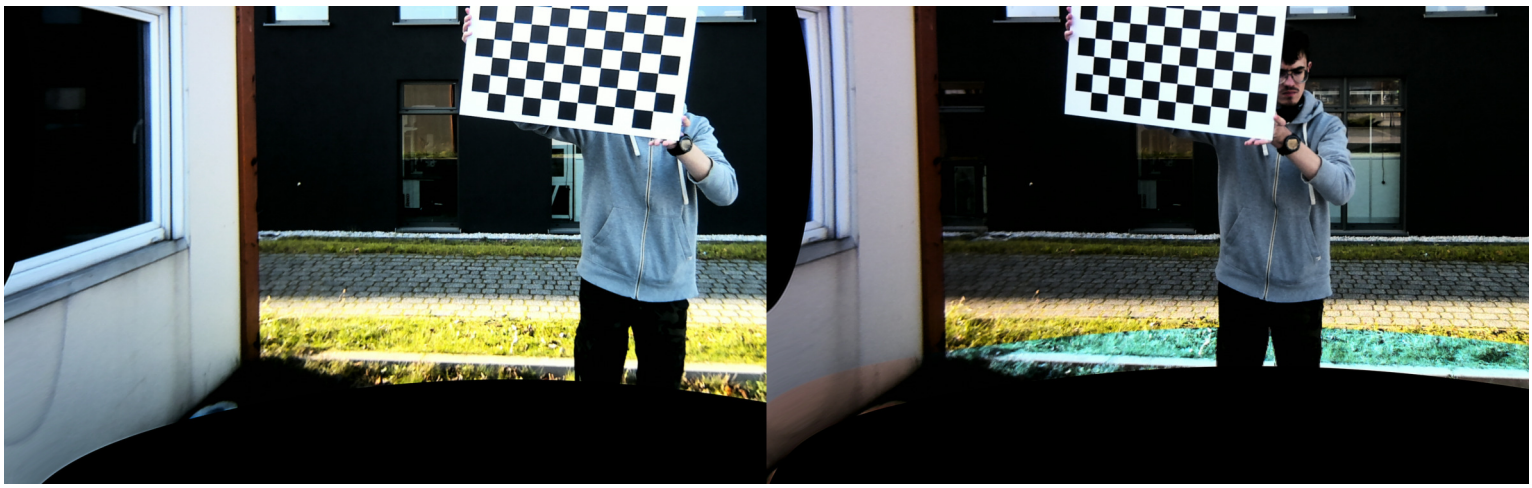


Rys. 5: Zrektyfikowana para obrazów **28.png** z kamery **cam1** oraz **cam4** metodą **INTER_AREA**

4.5 Metoda INTER_LANCZOS4

```
Interpolation type INTER_LANCZOS4:
  left_image: 0.02193242300018028
  right_image: 0.020166133000202535
  total: 0.042098556000382814
```

Kod 9: Czas obliczeń metody **INTER_LANCZOS4**



Rys. 6: Zrektyfikowana para obrazów **28.png** z kamery **cam1** oraz **cam4** metodą **INTER_LANCZOS4**

4.6 Wnioski

4.6.1 Analiza czasów

Metoda **INTER_AREA** osiągnęła najkrótszy czas (0.0045 s) - bardzo szybka metoda, która może być wykorzystywana w aplikacjach wymagających ciągłego przetwarzania obrazów.

Metoda **INTER_LANCZOS4** była najwolniejsza (0.0421 s) - najdokładniejsza z metod interpolacji, ale wymaga największego zużycia zasobów. Może być wykorzystywana w aplikacjach, które nie wymagają ciągłego przetwarzania obrazów, za to wymagają wysokiej precyzji.

4.6.2 Subiektywna analiza jakości

Obrazy przetworzone różnymi metodami interpolacji mogą się różnić pod względem ostrości krawędzi, artefaktów wizualnych oraz płynności przejść tonalnych.

- INTER_NEAREST: Najszybsza metoda, ale może powodować "blokowy" efekt na krawędziach.
- INTER_LINEAR: Dobre kompromisowe rozwiązanie, mniej ostrych krawędzi, lepsze płynne przejścia tonalne.
- INTER_CUBIC: Lepsze płynne przejścia i mniej artefaktów niż INTER_LINEAR, ale kosztem dłuższego czasu obliczeń.
- INTER_AREA: Idealna do zmniejszania rozmiaru obrazu, dobrze zachowuje szczegóły i jest szybka.
- INTER_LANCZOS4: Najlepsza jakość wizualna, ale największe wymagania czasowe. Ostre i precyzyjne krawędzie.

Analizując otrzymane obrazy łatwo można zauważyć wspomniane wcześniej poszarpane, "blokowe" krawędzie metody INTER_NEAREST, która według mnie wypadła najgorzej. Kolejne metody nie posiadają już tak widocznych artefaktów i różnią się nieznacząco. Drobne różnice mogą być zauważone w okolicach zamka błyskawicznego bluzy, którą założyła osoba wykonująca zdjęcia kalibracyjne (szczególnej analizie poddano prawe zdjęcie z każdej pary). Tutaj metody INTER_LINEAR i INTER_AREA zdają się dawać niemal identyczne wyniki, jednak są one (subiektywnie) gorsze od tych, które zostały otrzymane z metod INTER_CUBIC oraz INTER_LANCZOS4. Ostatnie dwie metody zdają się działać najlepiej (i dają podobne wyniki), ponieważ zamek posiada więcej szczegółów, a odcień bluzy wokół zamka, jak i sam zamek wydają się bardziej zbliżone do rzeczywistych.

5 Zadanie 5 - wizualizacja linii epipolarnych i obszaru bez zniekształceń

Aby wykonać to zadanie, dodano funkcję **draw_epilines_aligned()** do pliku z funkcja **stereo_rectify()**, której zadaniem jest narysowanie linii epipolarnych i obszaru bez zniekształceń na obu zdjęciach. Fragment kodu wywołujący tę funkcję oraz sama funkcja znajdują się poniżej (Kod 10), a otrzymane zdjęcie widoczne jest na kolejnej stronie (Rys 7).

```
def draw_epilines_aligned( img_left: np.ndarray, img_right: np.ndarray, num_lines:
    int = 15, roi_left: tuple = None, roi_right: tuple = None, line_thickness: int =
    2, roi_thickness: int = 2, ) -> tuple[np.ndarray, np.ndarray]:
    """
    Draw uniformly spaced horizontal epipolar lines and optional ROI boxes on
    rectified images.

    :param img_left: Left rectified image.
    :param img_right: Right rectified image.
    :param num_lines: Number of horizontal epipolar lines to draw.
    :param roi_left: ROI tuple for the left image (x, y, width, height).
    :param roi_right: ROI tuple for the right image (x, y, width, height).
    :param line_thickness: Thickness of the epipolar lines.
    :param roi_thickness: Thickness of the ROI rectangle lines.
    :return: Tuple of images with horizontal epipolar lines and optional ROIs.
    """
    img_left_with_lines = img_left.copy()
    img_right_with_lines = img_right.copy()
    # Use image dimensions or ROI if available
    height, width = img_left.shape[:2]
    roi_left = roi_left if roi_left else (0, 0, width, height)
    roi_right = roi_right if roi_right else (0, 0, width, height)
    # Determine vertical range for lines based on ROI or full image height
    y_start, y_end = ( (roi_left[1], roi_left[1] + roi_left[3]) if roi_left else (0,
    height) )
    y_coords = np.linspace(y_start, y_end - 1, num_lines).astype(int)
    # Draw horizontal epipolar lines across full width
    for y in y_coords:
        color = (0, 0, 255)  # Red for lines
        cv.line(img_left_with_lines, (0, y), (width, y), color, line_thickness)
        cv.line(img_right_with_lines, (0, y), (width, y), color, line_thickness)
    # Draw ROI rectangles
    if roi_left:
        cv.rectangle(
            img_left_with_lines, (roi_left[0], roi_left[1]),
            (roi_left[0] + roi_left[2], roi_left[1] + roi_left[3]),
            (0, 255, 0), roi_thickness,
        )
    if roi_right:
        cv.rectangle(
            img_right_with_lines, (roi_right[0], roi_right[1]),
            (roi_right[0] + roi_right[2], roi_right[1] + roi_right[3]),
            (0, 255, 0), roi_thickness,
        )
    return img_left_with_lines, img_right_with_lines

# Load an example pair of images for rectification
img_left = cv.imread(images_left[whichImage])
img_right = cv.imread(images_right[whichImage])
# Apply rectification to both images
rectified_left = cv.remap(img_left, map1_left, map2_left, cv.INTER_LINEAR)
rectified_right = cv.remap(img_right, map1_right, map2_right, cv.INTER_LINEAR)
# Draw epilines using the fundamental matrix F
rectified_left_with_lines, rectified_right_with_lines = draw_epilines_aligned(
    rectified_left, rectified_right,
    num_lines=drawEpipolarLinesParams[0],
    roi_left=roi1, roi_right=roi2,
    line_thickness=drawEpipolarLinesParams[1],
    roi_thickness=drawEpipolarLinesParams[2],
)
# Combine the images side-by-side for visualization
rectified_pair = np.hstack((rectified_left_with_lines, rectified_right_with_lines))
```

Kod 10: Fragment kodu wywołujący funkcję rysującą linie epipolarne

Przedstawiona wcześniej funkcja **draw_epilines_aligned()** pozwala na określenie parametrów rysowania - parametr **drawEpipolarLinesParams** jest krotką (*ang. tuple*) składającą się z 3 elementów:

- liczba linii, które chcemy narysować,
- grubość rysowanych linii epipolarnych,
- grubość rysowanego obszaru ROI.

Na otrzymanym zdjęciu wykorzystano krotkę o parametrach **(20, 3, 2)**, które w mojej subiektywnej ocenie sprawdzają się najlepiej.

Rys. 7: Obraz po rektyfikacji z narysowanymi liniami epipolarnymi (na czerwono) i zaznaczonym obszarem bez zniekształceń (na zielono)

6 Zadanie 6 - eksport obrazów po rektyfikacji

W celu wykonania tego zadania, dodano flagę **saveRectifiedImages** i zmienną określającą w jakim folderze chcemy zapisać zdjęcia (**rectifiedImagesDirPath**) do funkcji **stereo_rectify()**. Po ustawieniu tej flagi na wartość **True** w wywołaniu funkcji i uzupełnieniu ścieżki zapisu, tworzony jest folder (o ile nie istniał wcześniej), a w nim zapisywane są odpowiednio:

- zrektyfikowane zdjęcie lewe,
- zrektyfikowane zdjęcie prawe,
- połączone zdjęcia z obu kamer z narysowanymi liniami i obszarem ROI,
- zdjęcia rektyfikowane różnymi metodami interpolacji (o ile wybrano tę opcję w wywołaniu funkcji).

Kod zmodyfikowanego wywołania oraz fragment kodu zapisujący zdjęcia znajdują się poniżej (Kod 11). Fragment ten pokazuje również w jaki sposób zapisywane są obrazy, gdy wywołamy funkcję do testowania różnych metod interpolacji.

```
# Save the rectified image to a file
if saveRectifiedImages:
    if (not rectifiedImagesDirPath) or (len(rectifiedImagesDirPath) == 0):
        raise RectifiedImgPathNotProvided

    if not os.path.exists(rectifiedImagesDirPath):
        os.makedirs(rectifiedImagesDirPath)

    if testInterpolationMethods and (len(rectifiedImagesDifferentInterpolations) > 0):
        for i, rectified_pair in enumerate(rectifiedImagesDifferentInterpolations):
            cv.imwrite(os.path.join(rectifiedImagesDirPath,
    f"rectified_pair_{interpolationTypesNames[i]}.png"), rectified_pair)

    else:
        cv.imwrite(os.path.join(rectifiedImagesDirPath, "rectified_left.png"),
    rectified_left)
        cv.imwrite(os.path.join(rectifiedImagesDirPath, "rectified_right.png"),
    rectified_right)
        cv.imwrite(os.path.join(rectifiedImagesDirPath, "rectified_stereo_pair.png"),
    rectified_pair)


stereo_rectify(
    calibImgDirPath_left=left_cam,
    calibImgDirPath_right=right_cam,
    imgPoints_left=params_left["imgPoints"],
    imgPoints_right=params_right["imgPoints"],
    loadStereoCalibrationParams=True,
    stereoCalibrationParamsPath=stereo_cam_params,
    testInterpolationMethods=True, # TEST DIFFERENT INTERPOLATION METHODS
    saveRectifiedImages=True, # SAVE RECTIFIED IMAGES
    rectifiedImagesDirPath=rectified_images_dir, # WHERE WE WANT TO SAVE
    whichImage=0,
    drawEpipolarLinesParams=(20, 3, 2)
)
```

Kod 11: Fragment kodu pozwalający zapisać obrazy po rektyfikacji

7 FOV

Zgodnie z poleceniem prowadzącego, wykonano również obliczenia, dzięki którym wyznaczono FOV (kąt widzenia kamery, *ang. Field Of View*). Obliczenia te znajdują się w funkcji **stereo_calibrate()** i są zapisywane do pliku zawierającego parametry kalibracji stereo (między innymi razem ze wspomnianą wcześniej odległością bazową - sekcja 2). Wynikiem kalkulacji jest FOV zarówno horyzontalne, jak i wertykalne. Aby wyznaczyć te wartości potrzebujemy znać macierze obu kamer po kalibracji. Fragment kodu przedstawiający moment wywołania oraz samą funkcję znajdują się poniżej (Kod 14). Dla kamer **cam2** i **cam3** otrzymano odpowiednio:

```
{
    "fov_left": [
        47.28002567642402, # HORIZONTAL FOVx
        30.445701515560305 # VERTICAL FOVy
    ],
    "fov_right": [
        48.18381757021475, # HORIZONTAL FOVx
        31.07180784431195 # VERTICAL FOVy
    ],
}
```

Kod 12: FOV pierwszego zestawu kamer

a dla kamer **cam1** i **cam4**:

```
{
    "fov_left": [
        69.60246068508005, # HORIZONTAL FOVx
        46.726255406265835 # VERTICAL FOVy
    ],
    "fov_right": [
        71.16479034680879, # HORIZONTAL FOVx
        47.95783925008398 # VERTICAL FOVy
    ]
}
```

Kod 13: FOV drugiego zestawu kamer

```
def calculate_fov(cameraMatrix: np.ndarray, imageSize: tuple[float, float]):
    """
    Calculate the horizontal and vertical FOV for a given camera.

    :param np.ndarray cameraMatrix: Intrinsic camera matrix (3x3).
    :param tuple[float, float] imageSize: Tuple containing image width and height
    (width, height).

    :return: tuple[float, float] Horizontal FOV and vertical FOV in degrees.
    """
    fx = cameraMatrix[0, 0]  # Focal length in x-axis
    fy = cameraMatrix[1, 1]  # Focal length in y-axis
    width, height = imageSize

    fov_horizontal = 2 * np.arctan2(width, (2 * fx)) * (180 / np.pi)
    fov_vertical = 2 * np.arctan2(height, (2 * fy)) * (180 / np.pi)

    return fov_horizontal, fov_vertical

fov_left = calculate_fov(cameraMatrix_left, image_size)
fov_right = calculate_fov(cameraMatrix_right, image_size)
```

Kod 14: Fragment kodu wyznaczający FOV kamery

8 Kod

Przygotowana na potrzeby tego laboratorium paczka, została zaktualizowana, aby wykonywać wszystkie wymagane zadania z tego laboratorium:

- kalibracja stereo,
- wyznaczanie odległości bazowej,
- wyznaczanie FOV horyzontalnego i wertykalnego,
- rektyfikacja obrazów,
- porównanie metod interpolacji,
- wizualizacja linii epipolarnych i obszaru bez zniekształceń,
- zapis zdjęć po rektyfikacji.

Kod paczki jest dostępny w serwisie GitHub, jednak kopia kodu źródłowego została załączona do sprawozdania w celu łatwiejszej weryfikacji wyników.