Penerapan Edge Detection dan Object Recognition untuk Parsing State Catur

Christian Albert Hasiholan (13521078)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13521078@std.stei.itb.ac.id

***Abstract*— Permainan catur merupakan permainan strategi yang membutuhkan analisis posisi bidak secara akurat untuk mendukung berbagai aplikasi, seperti analisis permainan, pelatihan model kecerdasan buatan, dan dokumentasi langkah permainan. Makalah ini membahas implementasi sistem parsing state catur menggunakan metode edge detection dan object recognition. Edge detection digunakan untuk mendeteksi garis dan kontur pada papan catur guna mengidentifikasi batas dan sudut setiap kotak pada papan. Sementara itu, object recognition digunakan untuk mengenali jenis dan warna bidak yang berada pada tiap kotak dengan membandingkan hasil ekstraksi fitur citra bidak dengan template yang telah disediakan. Proses implementasi mencakup beberapa tahap, mulai dari preprocessing citra, segmentasi papan, identifikasi kotak kosong menggunakan citra biner, hingga pengenalan jenis bidak melalui perbandingan edge dengan template menggunakan metode perbedaan citra. Eksperimen dilakukan dengan berbagai parameter pada edge detection, threshold warna, dan ukuran kernel dilasi untuk meningkatkan akurasi pengenalan. Hasil pengujian menunjukkan tingkat akurasi yang tinggi dengan hanya satu kesalahan pengenalan dari tujuh pengujian yang dilakukan. Sistem ini diharapkan dapat digunakan sebagai dasar dalam pengembangan aplikasi analisis catur otomatis yang sederhana namun efektif.**

Keywords—catur; edge detection; object recognition;

I. PENDAHULUAN (*HEADING 1*)

Catur merupakan salah satu permainan strategi klasik yang telah dimainkan selama berabad-abad dan tetap populer hingga saat ini, baik dalam kompetisi profesional maupun aktivitas *refreshing*. Permainan catur melibatkan papan berukuran 8x8 dengan 64 kotak yang diisi oleh 32 bidak dengan gerakan yang unik. Untuk dapat menganalisis jalannya permainan catur dengan otomatis, diperlukan sistem yang harus mampu membaca posisi bidak pada papan dengan akurasi tinggi. Deteksi posisi bidak ini penting dalam berbagai aplikasi seperti sistem analisis permainan, data train kecerdasan buatan untuk pengambilan langkah permainan, hingga perekaman langkah permainan untuk kebutuhan pengarsipan dan pembelajaran.

Salah satu metode yang efektif untuk menyelesaikan tantangan ini adalah dengan menggunakan teknik *edge detection* dan *object recognition*. *Edge detection* memungkinkan deteksi garis dan kontur yang membentuk sisi dan grid pada papan catur, sehingga setiap kotak dapat diidentifikasi dan diproses secara terpisah. Sementara itu, *object recognition* digunakan untuk mengidentifikasi jenis bidak catur yang terdapat pada masing-masing kotak, seperti raja, ratu, benteng, kuda, menteri, dan pion, serta membedakan antara bidak berwarna putih dan hitam.

Makalah ini membahas secara mendalam bagaimana *edge detection* dan *object recognition* diimplementasikan dalam proses parsing state catur. Pembahasan meliputi tahapan *preprocessing* citra, identifikasi papan catur, segmentasi kotak pada papan catur, hingga pengenalan bidak catur yang terdeteksi. Hasil akhir dari makalah ini diharapkan dapat menghasilkan sistem yang mampu melakukan parsing kondisi papan catur secara otomatis dengan tingkat akurasi yang tinggi, serta dapat diaplikasikan ke dalam kebutuhan lainnya.

II. LANDASAN TEORI

A. *Catur*

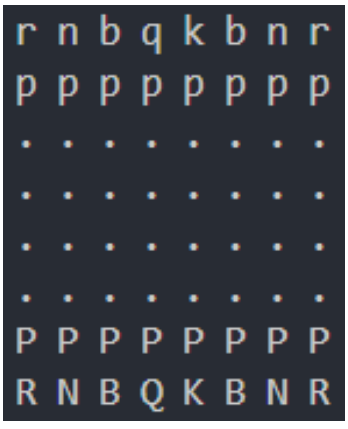
Catur merupakan salah satu permainan yang paling populer di dunia. Jutaan orang memainkannya baik untuk hobi atau hiburan, maupun bersaing secara kompetitif dalam kejuaraan. Catur adalah permainan strategi yang dimainkan secara bergantian, dan setiap pemain mendapatkan hak dan kesempatan yang sama. Permainan catur melibatkan pemindahan bidak dengan gerakan khusus untuk setiap jenis bidak. Tujuan utama permainan ini adalah mencapai skakmat, yaitu ketika raja lawan berada dalam kondisi terancam dan tidak memiliki langkah sah untuk menghindarinya.

Satu set buah catur memiliki dua pasukan bidak yang berbeda, masing-masing memiliki delapan pion, dua kuda, dua menteri, dua benteng, menteri, ratu dan raja. Pemain dapat membedakan pasukan bidak mereka berdasarkan warna, dengan warna terang dan gelap. Papan catur berukuran 8x8 dengan total 64 kotak yang terdiri dari warna terang (putih) dan gelap (hitam).

Gambar 2.1 Catur

Dalam era digital, permainan catur juga banyak dimainkan secara daring melalui platform seperti Chess.com dan Lichess. Catur online memungkinkan pemain dari seluruh dunia untuk saling berkompetisi dalam waktu nyata, dengan berbagai format permainan seperti *blitz*, *rapid*, dan *classical*. Selain itu, banyak platform catur online menyediakan analisis permainan berbasis kecerdasan buatan yang dapat membantu pemain meningkatkan kemampuan dengan menganalisis langkah demi langkah permainan mereka.

Notasi state pada catur mengacu pada representasi seluruh keadaan papan, termasuk posisi bidak, giliran pemain, status rokade, dan kemungkinan en passant. Notasi ini merepresentasikan posisi dan pergerakan bidak dengan menggunakan huruf dan angka. Misalnya, "e4" menunjukkan pion digerakkan ke kolom e, baris ke-4. Notasi ini penting dalam mendokumentasikan langkah-langkah permainan, memfasilitasi analisis pertandingan, dan memungkinkan rekonstruksi posisi pada papan catur. Notasi ini dapat diperoleh melalui parsing pada state catur, kemudian diubah ke dalam format notasi.



Gambar 2.2 State catur

Gambar 2.2 merupakan bentuk state catur yang umumnya dipakai pada bot permainan catur, misalnya Stockfish. State tersebut digunakan khususnya pada bot yang bekerja melalui CLI, sehingga bot dapat mengingat statenya dan dapat menampilkannya ke pengguna juga. Penjelasan mengenai symbol-simbol pada state tersebut adalah sebagai berikut:

- R/r : Rook (Benteng)
- N/n : Knight (Kuda)
- B/b : Bishop (Menteri)
- Q/q : Queen (Ratu)
- K/k : King (Raja)
- P/p : Pawn (Pion)

Perbedaan huruf kapital dengan huruf kecil menunjukkan warna bidak tersebut. Huruf kapital menunjukkan bahwa bidak tersebut berwarna terang (putih), sedangkan huruf kecil menunjukkan bahwa bidak berwarna gelap (hitam).

B. Citra Biner

Citra biner adalah citra yang memiliki hanya dua nilai graylevel, 0 dan 1. Artinya citra tersebut hanya terdiri atas warna hitam dan putih, dengan hitam bernilai 1 dan putih bernilai 0 tergantung konvensi yang dipakai.

Alasan citra biner sering digunakan dalam memproses citra digital adalah karena kebutuhan memori yang rendah, komputasi kecil saat melakukan operasi logika, merepresentasikan citra hasil deteksi edge, untuk segmentasi objek, dan lebih focus pada analisis bentuk morfologi.

C. Edge Detection

Tepi (edge) adalah perubahan nilai intensitas nilai keabuan yang mendadak (besar) dalam jarak yang singkat. Tepi memiliki arah, dan arah ini berbeda-beda bergantung pada perubahan intensitas. Tepi biasanya terdapat pada batas antara dua daerah yang berbeda intensitas dengan perubahan yang sangat cepat di dalam citra.

Pendeteksian tepi bertujuan untuk meningkatkan penampakan garis batas atau objek di dalam citra. Pendeteksian tepi mengekstraksi representasi gambar garis-garis di dalam citra. Salah satu kegunaan deteksi tepi adalah untuk segmentasi objek, yaitu mendeteksi objek melalui bentuknya. Bentuk objek dapat diperoleh dari hasil pendeteksian tepi. Setelah tepi objek dideteksi, selanjutnya objek dipisahkan dari latar belakangnya, untuk kemudian digunakan dalam proses pengenalan objek (object recognition).

D. Object Recognition

Object recognition adalah teknik dalam computer vision yang digunakan untuk mengidentifikasi objek dalam gambar atau video. Saat manusia melihat gambar atau video, kita dapat dengan mudah mengenali objek, pemandangan, dan detail visual yang ada. Object recognition bertujuan untuk

mengajarkan komputer melakukan hal serupa, yaitu memahami isi dalam sebuah gambar. Teknologi ini menjadi kunci dalam berbagai aplikasi, seperti mobil tanpa pengemudi yang mampu mengenali rambu lalu lintas dan membedakan pejalan kaki dari tiang lampu, serta digunakan dalam bidang seperti identifikasi penyakit pada citra medis, inspeksi industri, dan penglihatan robotik.

Object detection dan object recognition sering disamakan, namun keduanya berbeda dalam eksekusi. Object detection berfokus pada menemukan lokasi objek dalam gambar, sementara object recognition tidak hanya mengidentifikasi tetapi juga mengklasifikasikan objek tersebut. Teknik-teknik dalam object recognition melibatkan pendekatan berbasis *machine learning* dan *deep learning*, yang kini menjadi metode dominan dalam menyelesaikan masalah pengenalan objek.

III. Implementasi

Implementasi dari parser state catur terdiri dibagi ke dalam beberapa tahap dengan tujuan untuk memaksimalkan hasilnya. Tahap pertama adalah untuk memperoleh edge dari gambar keseluruhan untuk menentukan papan catur. Edge detection dilakukan dengan operator Canny, serta dilakukan dilatasi agar edge menjadi lebih tebal.

Selanjutnya dicari kontur pada gambar edge, untuk memperoleh hanya bagian papan catur saja. Tujuannya adalah untuk menangani gambar yang memiliki padding, misalnya seperti hasil screenshot yang tidak tepat pada papan catur.

```
img = cv2.imread('board.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# cv2.imshow('gray', gray)

canny = cv2.Canny(gray,20,255)

kernel = np.ones((3, 3), np.uint8)
canny = cv2.dilate(canny, kernel, iterations=1)

_, biner = cv2.threshold(gray, 200, 255, cv2.THRESH_BINARY)

cv2.imshow('gray', biner)

# cv2.imshow('can', canny)

contours, _ = cv2.findContours(canny, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
largest_contour = max(contours, key=cv2.contourArea)

epsilon = 0.02 * cv2.arcLength(largest_contour, True)
approx = cv2.approxPolyDP(largest_contour, epsilon, True)

pieces_img = ['pawn.png', 'rook.png', 'night.png', 'bishop.png', 'queen.png', 'king.png']

x_corner = []
y_corner = []
```

Setelah papan caturnya ditemukan, maka sudut-sudutnya dapat diperkirakan. Sudut ini nantinya akan digunakan untuk memperoleh tiap kotak yang ada pada papan catur untuk selanjutnya dikenali masing-masing.

```
if len(approx) == 4:
    sorted_points = sorted(approx[:, 0], key=lambda x: (x[1], x[0]))
    top_left, bottom_left = sorted_points[:2]
    top_right, bottom_right = sorted_points[2:]

    x_corner = sorted([top_left[0], top_right[0], bottom_left[0], bottom_right[0]])
    y_corner = sorted([top_left[1], top_right[1], bottom_left[1], bottom_right[1]])
else:
    x_corner = [0, img.shape[1]]
    y_corner = [0, img.shape[0]]

step_x = (x_corner[-1] - x_corner[0]) // 8
step_y = (y_corner[-1] - y_corner[0]) // 8

result = [[None for _ in range(8)] for _ in range(8)]
for i in range(8):
    for j in range(8):
        x1, y1 = x_corner[0] + j * step_x, y_corner[0] + i * step_y
        x2, y2 = x1 + step_x, y1 + step_y

        normal_square = gray[y1+7:y2, x1+7:x2]
        edged_square = canny[y1+2:y2-2, x1+2:x2-2]

        kernel = np.ones((5, 5), np.uint8)
        dilated_square = cv2.dilate(edged_square, kernel, iterations=1)

        if recognize_blank(dilated_square):
            result[i][j] = "."
        else:
            color = recognize_color(normal_square)
            piece = recognize_piece(normal_square, color, pieces_img)
            result[i][j] = piece

display(result)
```

Tahap selanjutnya merupakan mengenali bidak pada tiap kotak yang telah diperoleh. Sebelum mengenali bidak, perlu diketahui dahulu apakah kotak tersebut berisi sebuah bidak. Untuk memastikannya maka digunakan citra biner yang diperoleh dari potongan gambar edge sesuai dengan kotak yang berkaitan. Kemudian rasio warna hitam pada kotak tersebut dihitung, apabila melewati threshold, maka dianggap sebagai kotak kosong.

```
def recognize_blank(square):
    black_pixels = np.sum(square == 0)
    total_pixels = square.size

    if black_pixels / total_pixels > 0.80:
        return True
```

Selanjutnya kotak yang tidak kosong diproses untuk mengenali warna dan jenis bidak pada kotak tersebut. Untuk mengenali warna bidak. Metode yang digunakan juga masih mirip dengan metode dalam mengenali kotak kosong, namun kali ini dibandingkan rasio mana yang lebih dominan antara pixel putih dan hitam.

```
def recognize_color(square):
    white_pixels = np.sum(square == 255)
    black_pixels = np.sum(square == 0)

    if white_pixels > black_pixels:
        return "white"
    else:
        return "black"
```

Terakhir, kotak diproses untuk mengenali jenis bidak yang ada pada kotak tersebut. Untuk membantu mengenali jenis bidak, digunakan gambar template untuk setiap jenis bidak.

Metode pengenalan bidak didasari oleh selisih antar citra, jadi citra template yang selisihnya paling kecil dengan citra yang dikenali dianggap sebagai jenis bidak yang sama. Untuk meningkatkan akurasinya, citra yang dipakai merupakan edgenya saja dan dilakukan dilatasi agar pixel-pixel edge semakin banyak. Metode ini dipilih diantara penggunaan *machine learning* atau *deep learning* dengan tujuan eksperimen, yaitu dengan cara yang simpel apakah dapat diperoleh performa akurasi yang cukup baik.

Setelah warna dan jenis bidak telah dikenali, notasi berdasarkan kedua hal tersebut kemudian disimpan dalam array 8x8 sesuai dengan ukuran papan catur. Untuk kotak yang kosong akan diisi dengan ".". Maka dengan ini, hasil parsing dari sebuah state papan catur telah berhasil diperoleh.

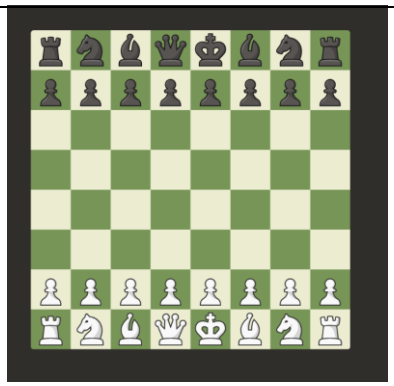
```python
def recognize_piece(square, color, pieces_img):
    piece_edges = cv2.Canny(square, 20,255)

    kernel = np.ones((5, 5), np.uint8)
    dilated_piece = cv2.dilate(piece_edges, kernel, iterations=1)

    cv2.imshow('dilated', dilated_piece)

    min_diff = float('inf')
    recognized_piece = "."

    for piece in pieces_img:
        template = cv2.imread(piece, cv2.IMREAD_GRAYSCALE)
        template_edges = cv2.Canny(template, 20,255)
        dilated_template = cv2.dilate(template_edges, kernel, iterations=1)

        if dilated_piece.shape != dilated_template.shape:
            dilated_template = cv2.resize(dilated_template, (dilated_piece.shape[1], dilated_piece.shape[0]))

        diff = cv2.absdiff(dilated_piece, dilated_template)

        diff_value = np.sum(diff)

        if diff_value < min_diff:
            min_diff = diff_value
            recognized_piece = piece.split('.')[0]

    if recognized_piece == ".":
        return recognized_piece

    if color == "white":
        piece_notation = recognized_piece[0].upper()
    elif color == "black":
        piece_notation = recognized_piece[0].lower()
    else:
        piece_notation = "."

    return piece_notation
```

Eksperimen lain yang dilakukan dalam implementasi adalah dalam hal *edge detection*, threshold untuk recognition, serta parameter dilatasi. Edge detection memiliki banyak jenis operator sehingga dapat dilakukan eksperimen untuk menentukan operator yang paling baik. Threshold untuk recognition misalnya warna dilakukan secara *trial and error* untuk melihat apakah threshold yang dipakai sudah memberikan hasil yang sesuai. Untuk parameter dilatasi, eksperimen dilakukan untuk menentukan ukuran kernelnya, ukuran kernel yang terlalu besar atau kecil dapat menyebabkan pengenalan jenis bidak menghasilkan prediksi yang salah, oleh karena itu perlu dicari ukuran yang optimal.

IV. HASIL

Dilakukan pengetesan menggunakan beberapa citra state catur yang berbeda. Citra juga ada yang terpusat pada papan catur dan ada juga yang tidak. Berikut merupakan citra serta hasil parsingnya.

Citra	Hasil Parsing
	r n b q k b n r p p p p p p p p . . . . P P P P P P P P R N B Q K B N R
	r n b . k b n r . p p p . p p p . . . . . q . . p . . . p . . . . P . . P . . . . . N . . P P P P . P P . R N B Q K B . R
	r n b . k b n r . p p p . p p p . . . . . q . . p . . . p . . . . P . . P . . . . . N . . P P P P . P P . R N B Q K B . R
	R . B K Q B N R P P P P P P P P N . . . . n . . p p p p p p p p r n b k q b . r

	B . B . R K . . P P . P . P P P . . . B . . . . . . P . P . . . p p . . p . . p r . . k N p . r . b . p . . p . . n . . q b n .
	r . b q k . n r p . p p p p . p p . . . . . p b . . . . P P . . . . N . . P P P P . . P P R N B Q K . . R
	r n b q k b n r p p p p p p p p . . . . P P P P P P P P R N B Q K B N R
	. r b q . k n r p . p . p p . p p . N . . . p b . . . p . . . . P P . . . . P P P P K . P P R N B Q R . . .

	R N B K Q B N R P P P P P P P P . . . . p p p p p p p p r n b k q b n r

Hasil parsing yang diperoleh sudah sangat akurat. Dari 9 pengujian yang dilakukan, kesalahan hanya ditemukan pada 1 pengujian saja. Kesalahan tersebut terjadi pada pengujian kelima, dimana ada kesalahan pengenalan jenis bidak dalam suatu kotak. Sedangkan untuk pengenalan warna bidak tidak ditemukan kesalahan di satu pengujian pun.

V. Kesimpulan

Dari hasil pengujian, dapat ditarik sebuah kesimpulan bahwa parsing untuk state catur berhasil dilakukan meskipun masih terdapat sedikit kesalahan. *Edge detection* dan *object recogniton* berhasil diimplementasikan untuk memenuhi kebutuhan segmentasi kotak, identifikasi warna, serta jenis-jenis bidak catur. Parsing yang dilakukan pada sebuah state catur menghasilkan keluaran berupa state dalam bentuk notasi catur yang umum digunakan.

Ucapan Terima Kasih

Dengan selesainya makalah ini, penulis ingin mengucapkan syukur kepada Tuhan Yang Maha Esa atas berkat yang diberikan-Nya sehingga penulis dapat menyelesaikan makalah ini dengan baik dan tepat waktu. Penulis juga ingin berterima kasih kepada Bapak Dr. Ir. Rinaldi Munir, M.T., selaku dosen dari Mata Kuliah Pemrosesan Citra Digital IF4073 yang telah memberikan materi dan pengetahuan untuk menyelesaikan makalah ini. Penulis juga berterima kasih kepada orang tua dan teman-teman yang telah membantu dan membuat penulis bersemangat untuk menyelesaikan makalah ini.

Referensi

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/18-Pendeteksian-Tepi-Bagian1-2024.pdf

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/19-Pendeteksian-Tepi-Bagian2-2024.pdf

[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/20-Pendeteksian-Tepi-Bagian3-2024.pdf

[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/24-Citra-Biner-2024.pdf

[5] https://www.chess.com/id/terms/catur

[6] https://python-chess.readthedocs.io/en/latest/engine.html

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 15 Januari 2024

Christian Albert Hasiholan