Web Service untuk Ekstraksi Informasi KTP Menggunakan Google Cloud Vision

Nandaffa Rizky Putra, Reza Fuad Rachmadi, dan Supeno Mardi Susiki Nugroho
Departemen Teknik Komputer, Institut Teknologi Sepuluh Nopember (ITS)
e-mail: fuad@its.ac.id

***Abstrak*—Penelitian ini mengembangkan sistem ekstraksi data otomatis dari citra Kartu Tanda Penduduk (KTP) menggunakan *Optical Character Recognition* (OCR) dari Google Cloud Vision API, yang diimplementasikan dalam bentuk *web service*. Metode yang digunakan melibatkan pengumpulan citra KTP, desain sistem ekstraksi data, perancangan dan pembuatan *web service*, dan pengujian akhir. *Web service* menggunakan arsitektur RESTful dan dikembangkan dengan Python, FastAPI, RabbitMQ, dan Celery. Hasil OCR diproses melalui *workflow post-processing* yang mengaplikasikan *regular expressions* dan *fuzzy matching* untuk menghilangkan informasi tidak relevan dan mengoreksi kesalahan urutan pada hasil OCR. Implementasi *web service* difasilitasi oleh Docker, yang memungkinkan *deployment* yang fleksibel. Hasil penelitian menunjukkan kondisi dan kemiringan citra KTP mempengaruhi hasil OCR. Namun, dengan alur kerja *post-processing* yang tepat, sistem mampu menangani berbagai kondisi citra KTP. *Web service* ini diharapkan dapat memudahkan ekstraksi data KTP secara otomatis untuk berbagai kebutuhan.**

***Kata Kunci*—Ekstraksi Informasi, Google Cloud Vision, Kartu Tanda Penduduk, *Optical Character Recognition*, *Web Service*.**

I. PENDAHULUAN

VISI komputer adalah salah satu cabang dari ranah ilmu kecerdasan buatan yang berfokus pada menganalisis dan memproses data dari sebuah citra atau gambar [1]. Meningkatnya popularitas layanan berbasis *Software-as-a-Service* yang menawarkan solusi siap pakai [2] dan kemunculan Web 3.0 yang menawarkan kecerdasan buatan sebagai salah satu gebrakannya [3] membuat pengaplikasian berbagai solu-si dan layanan visi komputer menjadi lebih mudah. Pengembang dapat dengan mudah mengintegrasikan berbagai layanan visi komputer dalam aplikasi mereka dan menjadikannya solusi dari sebuah permasalahan [4].

Digitalisasi berbagai aspek kehidupan di masa kini berlangsung dengan cepat, termasuk Kartu Tanda Penduduk (KTP) Indonesia yang telah bertransformasi menjadi e-KTP. Namun, pemanfaatan e-KTP masih belum maksimal, banyak layanan publik yang pendaftarannya masih membutuhkan fotokopi dan validasi KTP secara manual. Terlebih lagi, pengisian formulir dan data diri pada layanan publik di Indonesia sebagian besar masih menggunakan metode manual yang relatif lambat. Diperlukan upaya untuk lebih memaksimalkan penggunaan teknologi dalam penanganan KTP.

Untuk memaksimalkan penggunaan KTP dalam proses registrasi serta validasi pada berbagai macam layanan publik yang ada, maka dibuatlah sebuah sistem yang dapat melakukan ekstraksi informasi yang ada pada KTP melalui citra dari KTP tersebut dengan menggunakan metode *Optical Character Recognition* (OCR) [5]. Bagian OCR dari sistem ini menggunakan layanan Cloud Vision API dari Google yang memiliki akurasi tinggi [6] dengan biaya yang relatif terjangkau. Cloud Vision API sendiri mempunyai parameter-parameter yang dapat disesuaikan dengan kebutuhan penggunanya. Sistem ini akan direpresentasikan dalam bentuk *web service* yang juga disertakan dengan *front-end* sederhana.

Pembahasan pada penelitian ini dimulai dengan presentasi mengenai uraian dari penelitian ini (Bagian II). Kemudian dilanjutkan dengan penjelasan mengenai hasil yang berisikan arsitektur dari sistem yang dibuat dan hasil eksplorasi, serta diskusinya (Bagian III). Terakhir, didapatkan kesimpulan dari penelitian yang telah dilakukan (Bagian IV).

II. URAIAN PENELITIAN

A. Penelitian Terdahulu

Ada beberapa penelitian yang telah dilakukan terkait dengan penelitian ini. Penelitian oleh Toha dan Triayudi (2022) telah berhasil merancang sebuah sistem yang mampu mengekstrak 14 jenis atribut dari KTP menggunakan metode OCR berbasis *Tesseract* OCR versi 3.01, dengan tingkat akurasi 98,09% untuk gambar KTP yang baik [7]. Namun, akurasi sistem ini menurun menjadi 51,42% pada gambar KTP yang kurang baik atau rusak. Di sisi lain, penelitian Afdholudin *et al.* (2022) juga berhasil mengimplementasikan sistem ekstraksi dan validasi data e-KTP menggunakan Cloud Vision API, dengan akurasi 100% dalam pengujian dua gambar. Data dari sistem ini kemudian diverifikasi dengan *database* Disdukcapil untuk konfirmasi keaslian KTP [8].

B. Metode Penelitian

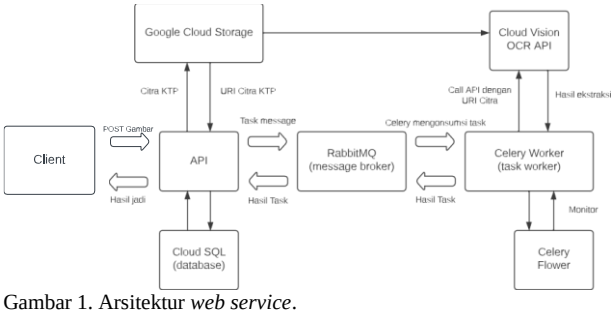
Metode yang digunakan pada penelitian ini adalah merancang sistem ekstraksi data KTP otomatis dan mengeksplorasi Cloud Vision API untuk menemukan parameter dan alur kerja terbaik untuk keperluan tersebut. Kemudian, sistem tersebut diintegrasikan ke dalam sebuah *web service* yang menjadi la- yer tambahan yang menghubungkan pengguna dengan sis-tem. Langkah-langkahnya adalah sebagai berikut:

1) Pengumpulan Citra KTP

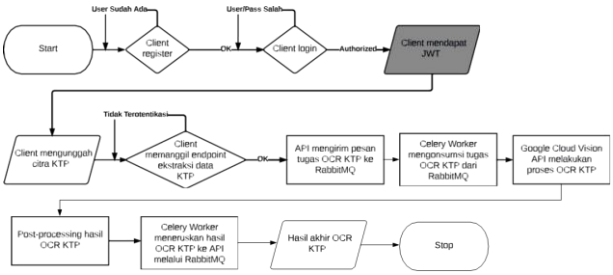
Pada tahap ini, penulis mengumpulkan beberapa citra KTP sebagai basis utama dalam penelitian ini. Citra KTP ini kemudian dikelompokkan berdasarkan kondisi dari KTP tersebut. Selain itu, dilakukan juga pengumpulan data atribut yang ada pada KTP, seperti format NIK, kota, dan lain-lain. Data awal ini adalah basis dari eksplorasi dan pengaturan parameter yang akan dilakukan pada langkah selanjutnya.

2) Perancangan Sistem Ekstraksi Data KTP

Setelah pengumpulan awal, dimulai proses perancangan sistem ekstraksi data KTP. Perancangan ini meliputi eksplorasi dan pengaturan parameter pada Google Cloud Vision untuk menemukan parameter dan *workflow* pra dan pasca ekstraksi terbaik yang dapat menghasilkan ekstraksi data pada KTP yang efisien dan minim kesalahan. Beberapa masalah yang akan dicoba untuk dimitigasi dalam proses peran-



Gambar 1. Arsitektur *web service*.



Gambar 2. Diagram alir *web service*.

cangan ini adalah hasil ekstraksi mentah yang tidak akurat, kesalahan pada hasil ekstraksi citra KTP yang kurang baik, dan kesalahan urutan atribut KTP.

Eksplorasi ini juga akan mengeksplor ekstraksi data pada beberapa jenis kondisi KTP, yaitu kondisi baik dan kondisi kurang baik. Selain itu, juga dilakukan penyusunan data agar lebih tertata dan mudah diproses lebih lanjut pada berbagai macam *front-end*.

3) Pembuatan Web service

Setelah sistem ekstraksi berfungsi dengan baik, sebuah *web service* dibangun sebagai fondasi utama untuk sistem ekstraksi data KTP. *Web service* ini dibuat dengan Python dan *framework* FastAPI, bertugas untuk memproses permintaan ekstraksi data KTP dari *front-end*. Selain itu, layanan dan *framework*, seperti RabbitMQ dan Celery dipertimbangkan untuk digunakan, dan *web service* ini ditunjang oleh Google *Cloud Platform* untuk kebutuhan *deployment* dan penyimpanan data. Untuk aspek keamanan, metode JSON *Web Token* (JWT) dan *Cross-Site Request Forgery* (CSRF) token digunakan.

4) Deploy Web service dan Pengujian Akhir

Web service yang sudah selesai kemudian di-*deploy* dan diuji dengan cara memanggil *endpoint* untuk proses ekstraksi data citra KTP pada *web service* tersebut. Tahap ini dilakukan untuk menemukan *bug* atau kesalahan pada sistem serta mengembangkan kemampuan sistem bilamana diperlukan.

III. HASIL DAN DISKUSI

A. Desain dan Implementasi Web service

Web service sistem ekstraksi data KTP ini berbasis pada empat komponen penting, yaitu sebuah REST API (*REpresentational State Transfer Application Programming Interface*, RabbitMQ, Celery, dan Google Cloud Vision API OCR). REST API pada sistem ini dibangun dengan *library* REST API bernama FastAPI, yang berbasis bahasa Python. API ini akan berperan sebagai *gateway* utama antara *client* dengan fungsi inti, yaitu ekstraksi data KTP. Diagram arsitektur dari sistem ini ada pada Gambar 1.

Proses ekstraksi data pada KTP itu sendiri meliputi proses pengunggahan citra visual KTP, pengunggahan citra visual KTP ke Google *Cloud Storage*, proses OCR dengan Google Cloud Vision OCR API pada Celery Worker, *post-processing* hasil OCR, dan pengiriman hasil akhir OCR ke *client*.

Selain itu, API ini juga berfungsi untuk keperluan CRUD (*Create*, *Read*, *Update*, *Delete*) dan autentikasi pengguna. Komponen kedua pada sistem ini adalah RabbitMQ yang berfungsi sebagai *message* broker yang memfasilitasi komunikasi asinkron, menerima dan mengirim pesan tugas untuk ekstraksi data KTP untuk kemudian diteruskan pada Celery Worker. Komponen ketiga, Celery Worker, berfungsi sebagai antrian pekerjaan, yang akan mengonsumsi pesan tugas dari RabbitMQ dan menjalankan tugas ekstraksi data KTP menggunakan Google Cloud Vision OCR API secara asinkron dan paralel. Komponen keempat, Google Cloud Vision API, adalah layanan visi komputer Google yang dapat digunakan untuk otomatisasi analisis gambar dan deteksi teks. Pada kasus ini, Google Cloud Vision OCR API digunakan sebagai metode utama untuk ekstraksi data pada KTP.

Client juga merupakan bagian penting dari sistem ini, yang pada penelitian ini berfungsi sebagai pengguna *web service* melalui *front-end*. Dalam penelitian ini, *client* direpresentasikan sebagai halaman web sederhana yang berinteraksi dengan *web service* dengan memanggil REST API. Meski berfokus pada *front-end web*, *web service* ini bisa disesuaikan untuk *front-end* lain seperti aplikasi *mobile*.

Komponen tambahan seperti Google *Cloud Storage*, *Cloud* SQL, dan Celery Flower juga berperan penting. Google *Cloud Storage* digunakan untuk penyimpanan sementara citra visual KTP yang akan diproses, sementara Celery Flower berfungsi sebagai penunjang Celery Worker, memberikan panel kontrol dan *monitoring* pekerjaan pada Celery Worker. *Cloud* SQL digunakan sebagai solusi basis data pada penelitian ini.

Cara kerja dari sistem ini secara keseluruhan adalah sebagai berikut: (1) *Client* membuat akun. (2) *Client* melakukan proses *login.* Jika *login* sukses, maka sebuah JSON *Web Token* (JWT) akan diterima oleh *client* sebagai bentuk autentikasi. (3) *Client* memanggil *endpoint* untuk proses ekstraksi data KTP. Di saat yang sama, *client* menyertakan citra KTP dan JWT yang sudah diterima sebagai bentuk autentikasi. (4) *Post-processing* hasil OCR. (5) Pengiriman hasil akhir OCR ke *client*. Langkah-langkah ini dapat direpresentasikan dalam sebuah diagram alir seperti pada Gambar 2.

Berikut penjelasan-penjelasan dari *package* yang digunakan dalam *web service* ini:

fastapi[all] : FastAPI, *framework web service* utama.

uvicorn : Server ASGI (*Asynchronous Server Gateway Interface*) untuk menjalankan FastAPI.

celery : *Framework task queue* untuk menjalankan tugas secara asinkronus dan paralel.

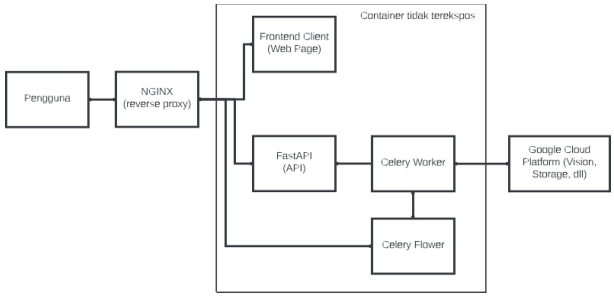
flower : *Framework* untuk memonitor Celery.

fastapi_csrf_protect : Untuk kebutuhan mitigasi serangan CSRF.

google-cloud-vision & google-cloud-storage: Untuk menggunakan fungsi-fungsi Google Cloud Vision API dan Google Cloud Storage.

PyJWT & passlib: Untuk kebutuhan JWT dan *handling* sandi.

SQLModel, pymysql: Untuk kebutuhan komunikasi dan ope-



Gambar 3. Diagram sistem dengan Docker.

Gambar 4. Citra KTP dengan kondisi baik dan tidak miring.

rasi dengan basis data.

regex, pandas, protobuf, python-multipart: Untuk kebutuhan *post-processing* hasil OCR awal.

python-Levenshtein, fuzzywuzzy: Untuk kebutuhan *fuzzy matching* dalam *post-processing*.

Pada *web service* ini, terdapat empat *endpoint* utama yang dapat dipanggil oleh *client*. *Endpoint* pada konteks *web service* adalah sebuah URL yang dapat digunakan untuk mengirim permintaan atau menerima sebuah respons dari sebuah *web service*. *Endpoint-endpoint* tersebut adalah sebagai berikut:

/csrftoken : *Endpoint* untuk mendapatkan token anti-CSRF.

/register : *Endpoint* untuk keperluan pendaftaran pengguna.

/login : *Endpoint* untuk keperluan *login*. Setelah pengguna berhasil *login*, *endpoint* mengembalikan sebuah nilai JWT sebagai bentuk autentikasi.

/text-detection : *Endpoint* utama untuk melakukan proses OCR yang ditunjang oleh RabbitMQ dan Celery Worker. *Endpoint* ini menggunakan cara *post-processing* secara konvensional dengan *regular expressions* (regex).

Pada penelitian ini, *client* yang digunakan adalah sebuah halaman web sederhana sebagai bentuk dari implementasi *client* dari sistem *web service* penelitian ini. Halaman web ini akan berkomunikasi dengan *web service* yang sudah dijelaskan pada subbab sebelumnya. Halaman web ini memiliki fungsi sederhana untuk *login*, *register*, dan proses ekstraksi data dari citra KTP itu sendiri. Sebelum *client* bisa melakukan proses ekstraksi data dari citra KTP, *client* harus mendaftarkan diri dengan fungsi register, dan apabila sudah memiliki akun pengguna bisa langsung *login*. *Client* pada penelitian ini dibuat dengan HTTP, JavaScript, dan CSS sederhana dan di-*host* pada sebuah web server, seperti NGINX atau Apache.

Sistem *web service* ini dapat diimplementasikan dengan menjalankan tiap komponen *web service* secara manual, atau dengan menggunakan Docker. Komponen *web service* yang


```
[
    {
        "description": "PROVINSI DAERAH ISTIMEWA YOGYAKARTA
        \nKABUPATEN SLEMAN\n: 6474[DIPOTONG UNTUK ALASAN PRIVASI]",
        "bounding_poly": {
            "vertices": [
                {
                    "x": 472,
                    "y": 359
                },
                {
                    "x": 3597,
                    "y": 359
                },
                {
                    "x": 3597,
                    "y": 2409
                },
                {
                    "x": 472,
                    "y": 2409
                }
            ]
        }
    },
    {
        "description": "PROVINSI", "bounding_poly": {
        "vertices": [
            {
                "x": 3161,
                "y": 2410
            },
# dan seterusnya
```


Gambar 5. Hasil OCR citra KTP dengan kondisi baik dan tidak miring.

harus dijalankan adalah FastAPI (dengan bantuan uvicorn), Celery Worker, dan Celery Flower. FastAPI dijalankan dengan uvicorn, sebuah web server ASGI (*Asynchronous Server Gateway Interface*) berbasis bahasa Python.

Penggunaan Docker dalam pengimplementasian sistem ini memudahkan proses karena fitur "*docker-compose*", yang memungkinkan sistem untuk di-*deploy* di mana saja tanpa khawatir akan isu inkompatibilitas. Docker menjalankan setiap komponen sistem dalam "*container*", semacam komputer virtual yang menjalankan tiap komponen secara individu. Kontainer ini dapat dikonfigurasi untuk setiap komponen sistem, menjadikannya fleksibel dan mudah disesuaikan. Docker juga memungkinkan untuk *client* dan *web service* untuk di-*deploy* dalam satu lingkungan yang sama secara bersamaan. Dalam konteks ini, NGINX digunakan sebagai *reverse proxy* untuk menghubungkan *client* dengan *web service*. Diagram implementasi sistem dengan menggunakan Docker disajikan pada Gambar 3.

B. Hasil OCR Citra KTP dengan Google Cloud Vision

Pada penelitian ini, digunakan beberapa jenis citra KTP, yaitu dengan kondisi baik dan kurang baik, serta citra yang tidak miring dan miring. Citra KTP dengan kondisi baik dan tidak miring akan digunakan sebagai citra untuk merepresentasikan kasus yang tergolong sempurna, seperti disajikan pada Gambar 4.

1) Citra KTP dengan Kondisi Baik dan Tidak Miring

Terlihat pada citra ini semua atribut pada KTP terlihat dengan jelas dan tidak ada kemiringan yang terjadi pada sumbu mana pun. Ketika citra tersebut di-OCR dengan Google Cloud Vision OCR, hasil yang didapat seperti pada Gambar 5.

Hasil OCR dikumpulkan dalam bentuk *list*, dengan setiap item memiliki dua properti utama: "description" yang berisi teks terbaca dan "bounding_poly" yang mencakup properti "vertices". "Vertices" adalah koordinat yang mencirikan kotak batas area di mana teks berada dalam gambar. Koordinat tersebut diambil dari empat pojok area teks: kiri atas, kanan atas, kanan bawah, dan kiri bawah. Format ini adalah *output*

```
['PROVINSI [PROVINSI]', 'KABUPATEN [KABUPATEN]', ': [NIK]',
  '[JENIS KELAMIN]', 'Nama', '[NAMA]', 'Tempat/Tgl Lahir [TTL]',
  'Jenis kelamin', 'Alamat', 'Gol. Darah [GOL. DARAH]',
  '[ALAMAT BARIS 1]', '[ALAMAT BARIS 2]', '[RT/RW]',
  '[KELURAHAN]', 'RT/RW', 'Kel/Desa', 'Kecamatan [KECAMATAN]',
  '[PEKERJAAN]', 'Agama', '[AGAMA]', 'Status Perkawinan: [STATUS PERKAWINAN]',
  'Pekerjaan', ' Kewarganegaraan: WNI', 'Berlaku Hingga SEUMUR HIDUP',
  ' [TEMPAT PENERBITAN]', '[TANGGAL PENERBITAN]', 'ANDALOUD ']
```

Gambar 6. Hasil OCR citra KTP dengan kondisi baik dan tidak miring.

```
['[NIK]', 'PROVINSI [PROVINSI]', 'KABUPATEN [KABUPATEN]',
    '[JENIS KELAMIN]', '[NAMA]', '[TTL]', 'Gol. Darah [GOL. DARAH]',
    '[ALAMAT BARIS 1]', '[ALAMAT BARIS 2]', '[RT/RW]', '[KELURAHAN]',
    '[KECAMATAN]', '[PEKERJAAN]', '[ AGAMA]', '[STATUS PERKAWINAN]',
    'WNI', 'SEUMUR HIDUP', '[TEMPAT PENERBITAN]', '[TANGGAL PENERBITAN]',
    'ANDALOUD']
```

Gambar 7. Hasil OCR citra KTP setelah melalui *post-processing* tahap *filtering*.

```
final_data = {
    "nik": [NIK],
    "provinsi": PROVINSI [PROVINSI],
    "kotakab": KABUPATEN [KABUPATEN/KOTA],
    "nama": [NAMA],
    "ttl": [TTL],
    "kelamin": [KELAMIN],
    "gol. darah": [GOLDAR],
    "alamat": [ALAMAT],
    "rt_rw": [RTRW],
    "kelurahan": [KELURAHAN],
    "kecamatan": [KECAMATAN],
    "agama": [AGAMA],
    "status": [STATUS],
    "pekerjaan": [PEKERJAAN],
    "kewarganegaraan": [KEWARGANEGARAAN],
    "masa_berlaku": [MASA BERLAKU]
    }
```

Gambar 8. Susunan *dict* akhir hasil OCR citra KTP kondisi baik dan tidak miring.

standar dari Google Cloud Vision OCR API.

Hasil OCR pada citra dengan kondisi baik dan tidak miring ini diawali dengan elemen *list* yang berisikan hasil OCR secara keseluruhan yang setiap barisnya dipisahkan dengan karakter *whitespace* "\n". Elemen *list* kedua dan seterusnya adalah teks dari tiap-tiap kata yang ada pada citra KTP, secara urut dari atas sampai bawah beserta dengan variabel "vertices" yang menunjukkan posisi dari kata tersebut. Jika isi "description" elemen *list* paling pertama diubah menjadi sebuah *list* Python sesuai dengan apa yang terjadi pada tahap ekstraksi di *endpoint* "/text-detection" akan didapat sebuah *list* sebagaimana dalam Gambar 6 dengan informasi sensitif akan diganti dengan format "[ATRIBUT KTP]".

Terlihat pada *list* di Gambar 6 masih ada data yang tidak relevan, salah satunya adalah tulisan keterangan atribut KTP seperti "Agama", "Nama", dan lain-lain. Hasil tersebut kemudian masuk ke tahap *post-processing filtering* yang berguna untuk menghilangkan informasi yang tidak relevan dengan menggunakan *regular expressions* (regex) dan *fuzzy matching*. *Fuzzy matching* digunakan agar sekiranya terdapat typo pada hasil OCR, hasil tersebut masih dapat diproses oleh regex. Setiap elemen list dicek, dan apabila ada informasi yang tidak relevan seperti keterangan atribut (seperti tulisan "Nama"), simbol, dan elemen yang kurang dari tiga huruf. *List* hasil setelah melewati tahap *post-processing filtering* disajikan pada Gambar 7.

Terlihat pada *list* bahwa sebagian besar atribut yang tidak relevan sudah hilang, kecuali tulisan "ANDALOUD" yang merupakan data *noise* dari hasil OCR. *List* ini kemudian masuk melalui tahap *post-processing* pengurutan yang berguna untuk mengurutkan data-data hasil OCR tersebut karena ada kemungkinan data-data ini tidak dalam urutan yang tepat. *Post-processing* tahap ini juga menggunakan regex dan *fuzzy matching* untuk mengurutkan setiap atribut yang ada, kecuali untuk atribut Nama, Alamat, Kelurahan,

(a) (b)

Gambar 9. (a) Citra KTP kondisi baik yang sangat miring (A) dan (b) citra KTP kondisi baik yang miring pada dua sumbu (B).

```
#Hasil citra A
['[NIK]', 'PROVINSI [PROVINSI]', 'KABUPATEN [KABUPATEN]',
    '[JENIS KELAMIN]', '[NAMA]', '[TTL]', '[ALAMAT BARIS 1]',
    '[ALAMAT BARIS 2]', '[RT/RW]', '[KELURAHAN]', '[KECAMATAN]',
    '[AGAMA]', '[STATUS PERKAWINAN]', '2000', 'Gol. Darah [GOL. DARAH]',
    '[PEKERJAAN]', 'WNI', ' SEUMUR HIDUP', 'NDA PER', 'KART',
    '[TEMPAT PENERBITAN]', '[TANGGAL PENERBITAN]','RDA Bramy',
    'PENDUDUK KARTU', 'KARTU TAN']

#Hasil citra B
['[NIK]', 'PROVINSI [PROVINSI]', 'KABUPATEN [KABUPATEN]',
    '[NAMA]', '[TTL]', '[JENIS KELAMIN]', 'Gol. Darah,
    '[ALAMAT BARIS 1]', '[ALAMAT BARIS 2]', '[RT/RW]',
    '[KELURAHAN]', '[KECAMATAN]', '[AGAMA]', '[STATUS PERKAWINAN]',
    '[PEKERJAAN] KART', 'WNI', 'SEUMUR HIDUP ', 'NDA PEN',
    '[TEMPAT PENERBITAN]', '[TANGGAL PENERBITAN]', 'BAK',
    'PENDUDUK KARTU', 'KARTU TAN']
```

Gambar 10. Hasil OCR citra baik yang miring setelah melalui *post-processing* tahap *filtering*.

Kecamatan, Agama, Status, Pekerjaan, Kewarganegaraan, dan Masa Berlaku. Atribut tersebut, selain Nama dan alamat, dicek dengan mencocokkan atribut yang ada dengan daftar masing-masing, contohnya seperti Kelurahan dan Kecamatan yang ada di Indonesia, serta Pekerjaan, Status dan Agama yang resmi berlaku pada KTP. Proses pencocokan tersebut juga melibatkan *fuzzy matching*. Atribut Nama dan Alamat tidak melalui proses pencocokan maupun regex, sehingga kedua atribut tersebut disendirikan dalam *list*, lalu kemudian diurutkan pada hasil akhir berdasarkan posisi mereka pada *list*. Urutan yang tepat adalah "NIK, Provinsi, Kabupaten, Nama, TTL, Jenis Kelamin, Gol. Darah, Alamat, RT/RW, Kelurahan, Kecamatan, Pekerjaan, Agama, Status Perkawinan, Kewarganegaraan, Masa Berlaku". Pada citra ini atribut "Jenis Kelamin" berada di atas "Nama", yang di mana tidak sesuai dengan urutan. Berikut adalah hasil akhir data KTP setelah melalui *post-processing* pengurutan dalam Gambar 8.

Untuk kasus KTP dengan kondisi bagus dan tidak miring, hasil yang didapat sangat bagus dengan hasil yang hampir tidak ada kesalahan kecuali data *noise* yang ada pada akhir list. Hasil dari citra dengan kondisi bagus dan tidak miring akan dijadikan patokan untuk hasil-hasil citra dengan kondisi yang berbeda.

2) Citra KTP dengan Kondisi Baik dan Miring

Pada penelitian ini, digunakan beberapa citra KTP dengan kondisi baik namun dengan beberapa tingkat kemiringan. Hal ini dilakukan untuk mengeksplor kemungkinan pengguna menggunakan citra yang miring dan seberapa efektif metode *post-processing* yang digunakan dapat menghasilkan hasil akhir yang sesuai. Citra digunakan ada pada Gambar 9.

Kemiringan pada citra dapat mempengaruhi hasil mentah dari Google Cloud Vision OCR API karena teks yang ada pada dokumen menjadi terdistorsi. Posisi dari tiap atribut juga dapat berubah karena KTP pada citra tidak lagi berbentuk persegi panjang normal yang datar. Setelah melalui *post-processing* tahap *filtering*, berikut adalah *list* hasil OCR citra A dan B dalam Gambar 10.

Tabel 1.
Keterangan Citra Kondisi Baik A

Atribut	Kesalahan (Karakter)	Keterangan
NIK	0	-
Provinsi	0	-
Kabupaten	0	-
Nama	0	-
TTL	0	-
Jenis Kelamin	0	-
Gol. Darah	0	Memang tidak ada Golongan Darah pada citra ini
Alamat	0	-
RT/RW	0	-
Kelurahan	0	-
Kecamatan	0	-
Agama	0	-
Status	0	-
Pekerjaan	4	Terdapat kata *noise* "KART"
Kewarganegaraan	0	-
Masa Berlaku	0	-
Data *noise*	-	Ada 8 kata *noise*

Tabel 2.
Keterangan Citra Kondisi Baik B

Atribut	Kesalahan (Karakter)	Keterangan
NIK	0	-
Provinsi	0	-
Kabupaten	0	-
Nama	0	-
TTL	0	-
Jenis Kelamin	2	Kehilangan tanda "-", "LAKI" menjadi "LAKE"
Gol. Darah	0	-
Alamat	0	-
RT/RW	0	-
Kelurahan	0	-
Kecamatan	0	-
Agama	0	-
Status	0	-
Pekerjaan	0	-
Kewarganegaraan	0	-
Masa Berlaku	0	-
Data *noise*	-	Ada 6 data *noise*

Tabel 3.
Keterangan Citra Kondisi Kurang Baik A

Atribut	Kesalahan (Karakter)	Keterangan
NIK	1	Satu angka berubah
Provinsi	0	-
Kabupaten	0	-
Nama	0	-
TTL	0	-
Jenis Kelamin	0	-
Gol. Darah	0	Tidak ada Gol. Darah pada KTP
Alamat	0	-
RT/RW	0	-
Kelurahan	0	-
Kecamatan	0	-
Agama	0	-
Status	0	-
Pekerjaan	0	-
Kewarganegaraan	0	-
Masa Berlaku	0	-
Data *noise*	-	Ada 3 kata *noise*

Keterangan dari hasil citra A dan B setelah di-OCR dan melalui tahap *post-processing filtering* pada Tabel 1 dan Tabel 2. Hasil OCR disajikan dalam bentuk tabel keterangan karena alasan privasi dan data sensitif. Ketika hasil citra A dan B dibandingkan, terlihat bahwa citra B memiliki lebih banyak kesalahan pembacaan dan data *noise* dibandingkan citra A. Meskipun begitu, hasil dari kedua citra masih menghasilkan informasi yang sangat bisa digunakan dan urutan

```
#Hasil citra A
['[NIK]', 'PROVINSI [PROVINSI]', 'KOTA [KOTA]', '[NAMA]',
    '[TTL]', '[JENIS KELAMIN]', '[ALAMAT]', '[RT/RW]',
    '[KELURAHAN]', '[KECAMATAN]', '[AGAMA]', '[STATUS]',
    'Gol. Darah', '[PEKERJAAN]', 'WNI', 'SEUMUR HIDUP',
    'APEND YOUPA', '[KOTA PENERBITAN]', '[TANGGAL PENERBITAN]',
    'laws']

#Hasil citra B
['PROVINSI [PROVINSI]', 'KOTA [KOTA]', '[NIK]', '[NAMA]',
    '[TTL]', '[JENIS KELAMIN]', '[ALAMAT]', '[RTRW]',
    '[KELURAHAN]', '[KECAMATAN]', '[AGAMA]', '[STATUS]',
    '[ PEKERJAAN]', 'WNI', 'SEUMUR HIDUP', 'RTU TANS',
    'Gol. Darah', 'ENDUDUK KAR', 'A PENDU', '[KOTA PENERBITAN]',
    '[TANGGAL PENERBITAN]', 'Laws', 'NDA', 'SSS']

#Hasil citra C
[[NIK], 'PROVINSI [PROVINSI]', 'KOTA [KOTA]', '[NAMA]',
    '[TTL]', 'Gol. Darah', '[JENIS KELAMIN]', '[ALAMAT]',
    '[RTRW]', '[KELURAHAN]', '[KECAMATAN]', '[AGAMA]',
    '[STATUS]', '[PEKERJAAN]', 'WNI', 'SEUMUR HIDUP',
    '[KOTA PENERBITAN]', '[TANGGAL PENERBITAN]', 'Lans']
```

Gambar 11. Hasil OCR citra kurang baik *post-processing* tahap *filtering*.

(a) (b) (c)

Gambar 12. Citra KTP kondisi kurang baik: (a) citra KTP kondisi baik yang tidak miring (A), (b) citra KTP kondisi baik yang sangat miring (B), dan (c) citra KTP kondisi baik yang miring pada dua sumbu (C).

Tabel 4.
Keterangan Citra Kondisi Kurang Baik B

Atribut	Kesalahan (Karakter)	Keterangan
NIK	1	Satu angka berubah
Provinsi	0	-
Kabupaten	0	-
Nama	0	-
TTL	0	-
Jenis Kelamin	1	Kurang satu karakter
Gol. Darah	0	Tidak ada Gol. Darah pada KTP
Alamat	0	-
RT/RW	0	-
Kelurahan	0	-
Kecamatan	0	-
Agama	0	-
Status	0	-
Pekerjaan	0	-
Kewarganegaraan	0	-
Masa Berlaku	0	-
Data *noise*	-	Ada 9 kata *noise*

dari atribut-atribut KTP sebagian besar masih urut, kecuali pada hasil citra B yang urutannya terganggu banyaknya data *noise*. Selain jumlah data *noise*, yang perlu disorot adalah atribut "Jenis Kelamin" yang mengalami degradasi hasil, yang pada citra A hanya mengalami kehilangan satu simbol, sementara pada citra B ada dua karakter yang mengalami kesalahan. Hasil akhir kedua citra setelah *post-processing* pengurutan serupa dengan Gambar 8 yang berarti proses pengurutan berhasil mengurutkan hasil dari KTP.

3) Citra KTP dengan Kondisi Kurang Baik

Pada penelitian ini, juga digunakan beberapa citra KTP dengan kondisi kurang baik. Hal ini dilakukan untuk mengeksplor kemungkinan pengguna menggunakan citra yang sudah mengalami kerusakan fisik, contohnya seperti tulisan yang sudah mulai hilang. Citra yang digunakan ada pada Gambar 11. Hasil OCR ketiga citra tersebut setelah *post-processing filtering* dalam Gambar 12.

Keterangan dari hasil ketiga citra tersebut dalam Tabel 3, Tabel 4, dan Tabel 5. Ketika dibandingkan dengan citra KTP dengan kondisi baik yang tidak miring, citra A mengalami

Tabel 5.
Keterangan Citra Kondisi Kurang Baik C

Atribut	Kesalahan (Karakter)	Keterangan
NIK	3	Dua angka berubah, tertambah satu angka
Provinsi	0	-
Kabupaten	0	-
Nama	2	Satu huruf berubah, kurang satu spasi
TTL	0	-
Jenis Kelamin	1	Satu karakter hilang
Gol. Darah	0	Tidak ada Gol. Darah pada KTP
Alamat	0	-
RT/RW	0	-
Kelurahan	1	Satu huruf berubah
Kecamatan	0	-
Agama	0	-
Status	0	-
Pekerjaan	0	-
Kewarganegaraan	0	-
Masa Berlaku	0	-
Data *noise*	-	Ada 1 kata *noise*

kesalahan baca dikarenakan ada bagian atribut NIK yang sudah kurang terlihat. Sementara, terlihat pada citra B dan C yang mengalami kesalahan baca yang lebih banyak, dengan citra C mengalami kesalahan baca yang paling banyak dari semua citra yang diujikan pada penelitian ini, yaitu 7 kesalahan dalam 4 atribut KTP. Hal ini menunjukkan bahwa kondisi dan posisi citra KTP mempengaruhi hasil akhir dari pembacaan. Hasil akhir ketiga citra setelah *post-processing* pengurutan juga serupa dengan Gambar 8.

IV. KESIMPULAN

Berdasarkan hasil dari penelitian ini, dapat ditarik tiga poin penting. Pertama, kehadiran *web service* yang telah dirancang dalam penelitian ini memfasilitasi ekstraksi informasi KTP secara otomatis, yang diharapkan dapat memenuhi berbagai kebutuhan seperti dalam administrasi layanan publik. Kedua, sistem ekstraksi informasi otomatis pada citra KTP yang dirancang dengan menggunakan Google Cloud Vision OCR API dan alur kerja *post-processing* yang menggunakan *regular expressions* dan *fuzzy matching* mampu mengakomodasi berbagai jenis citra KTP dalam berbagai kondisi, termasuk yang berkualitas baik maupun kurang baik, dan yang dalam posisi miring maupun tidak miring. Terakhir, penelitian ini menemukan bahwa kondisi dan posisi citra KTP berpengaruh signifikan terhadap hasil ekstraksi informasi. Citra KTP dengan kondisi baik dan tidak miring memberikan hasil terbaik tanpa kesalahan, sedangkan citra KTP dalam kondisi kurang baik dan miring pada dua sumbu memberikan hasil terburuk dengan tujuh kesalahan pembacaan pada empat atribut KTP.

DAFTAR PUSTAKA

[1] X. Li and Y. Shi, "Computer Vision Imaging Based on Artificial Intelligence," in *Proc. International Conference on Virtual Reality and Intelligent Systems (ICVRIS)*, Hunan, 2018, pp. 22–25. doi: 10.1109/ICVRIS.2018.00014.

[2] S. Satyanarayana, "Cloud computing: Saas," *Comput. Sci. Telecommun.*, vol. 4, no. 36, pp. 76–79, 2012.

[3] J. Hendler, "Web 3.0 emerging," *Computer*, vol. 42, no. 01, pp. 111–113, 2009, doi: 10.1109/MC.2009.30.

[4] R. G. Guntara, "Aplikasi pengenalan citra wajah di KTP menggunakan Google Cloud Vision API dan Kairos API berbasis android," *J. Comput. Sci. Appl. Informatics*, vol. 4, no. 2, pp. 198–207, 2022, doi: 10.28926/ilkomnika.v4i2.504.

[5] N. A. M. Isheawy and H. Hasan, "Optical character recognition (ocr) system," *IOSR J. Comput. Eng. Ver. II*, vol. 17, no. 2, pp. 2278–661, 2015, doi: 10.9790/0661-17222226.

[6] J. Goncalves and S. Paiva, "Inclusive Mobility Solution for Visually Impaired People using Google Cloud Vision," in *Proc. IEEE International Smart Cities Conference (ISC2)*, Manchester, 2021. doi: 10.1109/ISC253183.2021.9562892.

[7] M. Rizal Toha and A. Triayudi, "Penerapan membaca tulisan di dalam gambar menggunakan metode ocr berbasis website (studi kasus: e-ktp)," *J. Sains dan Teknol.*, vol. 11, no. 1, 2022, doi: 10.23887/jstundiksha.v11i1.42279.

[8] Afdholudin, Nehru, and Y. R. Hais, "Implementasi Sistem Ekstraksi dan Validasi Data E-KTP Sebagai Solusi Alternatif Otomatisasi Sistem Administrasi Data untuk Organisasi Kecil Non-Pemerintah," in *Prosiding Seminar Nasional Aplikasi Sains & Teknologi (SNAST)*, Yogyakarta, 2021, pp. A46–A54.