ChatGPT

Jak działa techniczne skanowanie kart w aplikacji TCGplayer?

Aplikacja TCGplayer wykorzystuje zaawansowane algorytmy widzenia komputerowego do szybkiego rozpoznawania kart kolekcjonerskich za pomocą aparatu w smartfonie. Rozwiązanie to, nazwane **Roca Vision**, stanowi jedno z najskuteczniejszych narzędzi automatycznej identyfikacji kart w branży kolekcjonerskiej [1]. W skrócie – aplikacja potrafi wykryć fizyczną kartę w kadrze, przetworzyć jej obraz, zidentyfikować dokładnie *jaki to egzemplarz* (nazwa, seria, numer), a następnie powiązać go z odpowiednim wpisem w katalogu TCGplayer lub innym źródle danych. Poniżej znajduje się szczegółowe omówienie etapów takiego skanowania oraz wskazówki techniczne, jak zbudować podobną funkcjonalność we własnej aplikacji.

Mechanizm rozpoznawania kart w TCGplayer (Roca Vision)

Aplikacja TCGplayer opiera się na mechanizmie **Roca Vision**, który zasila zarówno funkcję skanowania mobilnego, jak i przemysłowe sortery kart TCGplayer. Jest to rozwiązanie oparte na widzeniu komputerowym i uczeniu maszynowym, zaprojektowane z myślą o równowadze między **dokładnością, szybkością** a **"samoświadomością"** (pewnością co do wyników) [1]. Roca Vision przetworzył już ponad **700 milionów kart** z **ponad 98% skutecznością** rozpoznania [2], co świadczy o ogromnej bazie danych i dopracowanych algorytmach stojących za tą technologią.

Aplikacja podczas skanowania **analizuje obraz karty** – jej grafikę, układ i tekst – i porównuje te cechy z wewnętrznym katalogiem kart. Każde rozpoznanie jest przypisane do konkretnej pozycji w katalogu TCGplayer (konkretnej gry, edycji i wersji karty) [3]. Dzięki temu nawet karty mające wiele wydań lub wariantów są identyfikowane jednoznacznie (np. konkretna edycja danej karty Pokémon czy Magic: The Gathering). Co istotne, system raportuje **poziom pewności (confidence)** dla każdego dopasowania – np. oznacza skany jako **GOOD, FAIR, POOR** zależnie od szacowanej dokładności [4] [5]. Jeśli aplikacja nie jest pewna rozpoznania (np. **POOR**, niski confidence), umieszcza taką kartę w kategorii „Unidentified" do ręcznej weryfikacji [5].

W przeciwieństwie do prostych metod opartych tylko na odczycie tekstu, Roca Vision stosuje **rozpoznawanie obrazu**. Oznacza to, że system uczy się wyglądu kart (grafiki, układu tekstu, symboli) i dzięki temu potrafi rozpoznać kartę nawet pod pewnym kątem czy przy różnym oświetleniu. Przykładowo, zauważono że karty w językach innych niż angielski mogą być trudniejsze do automatycznego rozpoznania – wynika to z faktu, że zmieniony tekst na karcie wpływa na jej wizualne cechy, które algorytm analizuje [6]. Innymi słowy, Roca Vision nie „rozumie" tekstu jako takiego, lecz traktuje całą kartę jako obraz do porównania – jeśli wersja karty ma inną szatę graficzną lub napisy, algorytm może mieć nieco niższą pewność identyfikacji. Mimo to, przy standardowych kartach anglojęzycznych osiągana jest bardzo wysoka trafność, a skanowanie jest **błyskawiczne**. Twórcy chwalą się, że aplikacja **skanuje karty z niezrównaną szybkością i dokładnością** – użytkownik nie musi ręcznie wpisywać nazw, wystarczy zbliżyć kamerę [7].

Uwaga: W praktyce system TCGplayer czasem może błędnie rozpoznać *edycję* (set) karty, zwłaszcza jeśli ta sama ilustracja była drukowana wielokrotnie. Dlatego aplikacja umożliwia użytkownikowi weryfikację i zmianę wersji przed ostatecznym zapisaniem skanu [8]. Na przykład ta sama karta Pokémon z

identyczną grafiką wydana w dwóch różnych dodatkach może zostać automatycznie przypisana do niewłaściwego – użytkownik powinien to skorygować. Mimo tych niuansów, rdzeń technologii – rozpoznanie, jaka to karta – jest zwykle trafny.

Wykrywanie karty w kadrze (detekcja obrazu)

Pierwszym zadaniem aplikacji skanującej jest **odnalezienie fizycznej karty na obrazie z kamery**. Karta może pojawić się na różnych tłach, pod kątem, w zmiennym oświetleniu – algorytm musi więc zlokalizować jej kontury i obszar, aby dalej móc przeprowadzić identyfikację. TCGplayer i podobne aplikacje stosują tu klasyczne metody wizji komputerowej wspomagane kilkoma założeniami co do wyglądu karty:

- **Kontrastowe tło:** Zaleca się umieszczanie kart na jednolitym, gładkim tle o kolorze kontrastującym z obramowaniem karty [9]. Dzięki temu krawędzie karty łatwiej odróżniają się od tła. Przykładowo, biała kartka papieru jako podkład świetnie uwidoczni ciemne krawędzie kart z czarną ramką, a czarne tło – karty o jasnych brzegach.
- **Pełny widok karty:** Należy upewnić się, że cała karta mieści się w kadrze i nie jest zasłonięta [10]. Algorytmy detekcji zazwyczaj szukają **pełnego prostokątnego konturu** – jeśli karta wychodzi poza kadr lub jest zasłonięta, program może jej nie rozpoznać. Nawet etui (protektor) powinno być prosto założone – karta powinna być wyśrodkowana w koszulce, by krawędzie były wyraźnie widoczne [10].
- **Ograniczenie odbić światła:** Powierzchnie kart (zwłaszcza z połyskiem, np. karty foliowane) potrafią odbijać światło i tworzyć odblaski, które „wypalają" fragmenty obrazu. TCGplayer rekomenduje trzymać telefon pod kątem ~45° względem karty lub skorzystać z lampy błyskowej w aplikacji, aby zminimalizować odblaski [11]. Skanowanie pod kątem jest techniką zmniejszającą odbicie – karta nadal może zostać wykryta i poprawnie rozpoznana mimo perspektywy, a unikamy dużej plamy światła na środku.

Proces detekcji: W praktyce wykrywanie karty często przebiega podobnie do wykrywania dokumentów lub wizytówek w aplikacjach skanujących. Typowy pipeline wygląda następująco:

1. **Wstępne przetworzenie obrazu:** Zdjęcie z kamery może być przeskalowane i przefiltrowane dla łatwiejszej analizy. Przykładowo, zmniejszenie rozdzielczości przyspiesza obliczenia, a wyrównanie kontrastu (np. metodą adaptacyjnej equalizacji histogramu) może uwydatnić krawędzie karty w różnych warunkach oświetleniowych [12] [13]. Często obraz konwertuje się do odcieni szarości, a następnie zwiększa kontrast lub binaryzuje (progowanie do czerni i bieli) – karta ma zazwyczaj wyraźny, prostokątny obrys, który po takiej operacji staje się dobrze widoczny jako biały kształt na czarnym tle lub odwrotnie [14].
2. **Wykrycie konturów lub prostokątów:** Kolejnym krokiem jest znalezienie konturów obiektów na obrazie. Wykorzystuje się do tego algorytmy takie jak **Canny (detekcja krawędzi)** oraz **findContours** z biblioteki OpenCV, aby wyłuskać obszary obramowane liniami [14]. Spośród znalezionych konturów filtruje się te, które wielkością i kształtem odpowiadają karcie – zazwyczaj odrzuca się bardzo małe kształty, nieregularne lub ekstremalnie wydłużone linie itp. [15]. Szukamy konturu zbliżonego do czworokąta o proporcjach zbliżonych do proporcji karty (typowa karta ma proporcje ok. 63×88 mm, czyli ~1:1.4).
3. **Przybliżenie do kształtu karty:** Gdy uda się znaleźć kontur obejmujący potencjalną kartę, algorytm stara się wyznaczyć dokładne narożniki tego prostokąta. Wykorzystuje się tu metody aproksymacji kształtu – np. obliczenie **otoczki wypukłej** (convex hull) dla konturu i uproszczenie jej do 4 narożnych punktów [16] [17]. Innymi słowy, algorytm zamienia wykryty kształt na idealny czworokąt (cztery wierzchołki), który powinien odpowiadać krawędziom karty. Przydatne bywa

„prostowanie" zaokrąglonych rogów – karty mają lekko zaokrąglone rogi, ale dla uproszczenia kontur traktuje się jakby miał ostre narożniki, co ułatwia obliczenia [16].

4. **Korekcja perspektywy:** Jeśli karta była zeskanowana pod kątem (co, jak wspomniano, pomaga przy odblaskach), to uzyskany czworokąt będzie trapezem z perspektywy kamery. Na potrzeby rozpoznawania dobrze jest przekształcić obraz tak, aby „spionować" kartę – wykonuje się **transformację perspektywiczną** (np. funkcją `warpPerspective` z OpenCV) wykorzystując współrzędne czterech narożników. Wynikiem jest obraz karty widzianej od frontu, wyrównany i odpowiednio skalibrowany (np. do ustalonej wielkości). Tak przygotowany wycinek – prostokątny obraz samej karty – trafia następnie do etapu rozpoznawania właściwego.

Warto zaznaczyć, że współczesne rozwiązania mogą też korzystać z sieci neuronowych do detekcji obiektów. Alternatywnie więc do ręcznego znajdowania konturów można wytrenować model detekcji (np. sieć typu YOLO) wykrywający pozycje kart na obrazie. Jednak w przypadku pojedynczych kart na jednolitym tle podejście konturowe działa wystarczająco dobrze i szybko, stąd jest powszechnie stosowane. Co więcej, aplikacja TCGplayer umożliwia nawet **skanowanie wielu kart jednocześnie** – jeśli użytkownik rozłoży kilka kart obok siebie, algorytm potrafi znaleźć kilka konturów i rozpoznać wszystkie naraz [18]. Własna aplikacja również może zaimplementować pętlę po wykrytych konturach – każdy zidentyfikowany prostokąt karty przetwarzamy osobno w kolejnych krokach.

Rozpoznawanie karty: analiza obrazu vs. OCR tekstu

Gdy mamy już wyodrębniony obraz karty (przycięty do jej krawędzi i wyrównany), następuje kluczowy etap **identyfikacji**, czyli ustalenia, co to za karta. Istnieją dwa główne podejścia techniczne do rozpoznawania karty ze zdjecia:

1. Rozpoznawanie na podstawie obrazu (sztuczna inteligencja / porównywanie wzorców):
Jest to metoda, którą stosuje m.in. TCGplayer w swoim Roca Vision. Polega na tym, że aplikacja *nie czyta bezpośrednio napisów* na karcie, ale traktuje całą grafikę karty jako unikalny wzorzec wizualny. Dzięki technikom **machine learning** można nauczyć model, aby na podstawie wyglądu rozpoznawał konkretny egzemplarz. W praktyce mogą być tu wykorzystane różne techniki:

- **Metody klasyczne (feature matching):** Wcześniejsze projekty próbowały używać algorytmów wykrywania cech charakterystycznych obrazu, takich jak SIFT, SURF czy ORB, aby pobrać z każdej karty zestaw **punktów charakterystycznych** (np. krawędzie rysunku, układ symboli itp.). Następnie porównuje się te cechy między obrazem skanowanym a bazą danych kart w katalogu [19]. Takie podejście bywa skuteczne dla małych zbiorów, ale skaluje się gorzej, gdy liczba możliwych kart rośnie – duża liczba punktów i ich porównań spowalnia wyszukiwanie, a karty o bardzo podobnej grafice lub z dużą ilością drobnego tekstu mogą dawać mylące wyniki [20].
- **Porównywanie poprzez „odcisk" (hash) obrazu:** Inną techniką jest obliczenie tzw. *hashy percepcyjnych* (pHash) obrazów kart. Taki hash to krótka sygnatura, która podobnym obrazom przypisuje podobne wartości. Można wcześniej wyliczyć hash dla każdej karty referencyjnej, a następnie dla skanowanej karty i znaleźć najbliższy w bazie [21]. To podejście jest szybkie dla znajdowania bardzo podobnych obrazów (np. identycznych kart), jednak bywa wrażliwe na zmiany jasności, kąta czy przycięcia obrazu [22]. W efekcie samo hashowanie często nie wystarcza w trudniejszych warunkach.
- **Sieci neuronowe (CNN i uczenie głębokie):** Współcześnie najskuteczniejszą metodą jest użycie **konwolucyjnych sieci neuronowych** do wygenerowania tzw. *wektorów cech* lub bezpośredniej klasyfikacji kart. Twórcy systemów rozpoznawania kart (np. zespół firmy Collectors) opisują, że lepsze wyniki uzyskali wykorzystując pretrenowane sieci CNN do ekstrakcji cech z obrazów kart, a następnie porównując te cechy w przestrzeni wielowymiarowej [23] [24]. Mówiąc prościej, sieć zamienia obraz karty na unikalny ciąg liczb (wektor), który *zbiera istotne informacje o obrazie*.

Wektory dla tego samego wzoru karty będą do siebie bardzo zbliżone, nawet jeśli zdjęcie było zrobione pod kątem czy w innym oświetleniu. Pozwala to szukać **najbardziej podobnej karty** wśród tysięcy innych poprzez szybkie wyszukiwanie sąsiadów w bazie tych wektorów (np. za pomocą struktur ANN – approximate nearest neighbors). Taka metoda jest zarówno dokładna, jak i skalowalna – dodanie nowych kart sprowadza się do wygenerowania ich wektorów i dołączenia do bazy, bez trenowania od zera modelu na tysiące klas.

- **Klasyfikacja jednym modelem:** Alternatywnie można próbować wytrenować sieć neuronową, która od razu klasyfikuje obraz na konkretną kartę (wyjściem sieci jest np. identyfikator karty). Jednak ze względu na ogromną liczbę unikalnych kart (dziesiątki tysięcy, biorąc pod uwagę wszystkie gry i sety), podejście to jest mniej praktyczne – wymagałoby niezmiernie wielu klas i ciągłej aktualizacji modelu o nowe karty. Dlatego w praktyce TCGplayer i podobne systemy wolą podejście z wektorami cech i wyszukiwaniem podobieństwa.

Aplikacja TCGplayer prawdopodobnie wykorzystuje model głębokiego uczenia na urządzeniu lub na serwerze, który otrzymuje obraz karty (np. już wykadrowany) i zwraca listę najlepszych dopasowań z bazy katalogowej, wraz z poziomem pewności. Jeśli pewność jest wysoka, aplikacja automatycznie wybiera ten wynik, jeśli niższa – może poprosić użytkownika o potwierdzenie lub poprawkę (np. wskazanie właściwej edycji, jak wspomniano) [25]. Dzięki takiemu podejściu, **użytkownik widzi od razu nazwę karty, jej numer, edycję oraz np. cenę rynkową**, bo aplikacja wiąże obraz z konkretnym rekordem w swojej bazie. To podejście jest szybkie – nie wymaga żadnego dodatkowego inputu od użytkownika – ale jego implementacja wymaga zbudowania lub użycia gotowego modelu CV wyszkolonego na obrazach kart.

Przykład wykrycia i rozpoznania karty na podstawie obrazu. Powyżej widzimy wynik działania algorytmu rozpoznawania: zielona ramka oznacza wykryty obszar karty na oryginalnym zdjęciu, a nałożony opis („Dragon whelp”) to nazwa rozpoznanej karty. Taki efekt uzyskuje m.in. system **Magic Card Detector** wykorzystujący OpenCV – po wykryciu konturu karty następuje identyfikacja karty poprzez porównanie obrazu z bazą (tu akurat rozpoznano kartę *Dragon Whelp* z gry **Magic: The Gathering**). W aplikacji TCGplayer analogicznie po znalezieniu karty w kadrze następuje automatyczne określenie, jaka to karta, na podstawie cech wizualnych.

2. Rozpoznawanie oparte o odczyt tekstu (OCR):
Drugie podejście, które bywa stosowane zwłaszcza w autorskich projektach lub gdy chcemy uniknąć

trenowania modeli, to wykorzystanie **OCR (Optical Character Recognition)** – czyli próby *odczytania drukowanego tekstu z obrazu karty* – aby uzyskać kluczowe informacje identyfikujące kartę. Karty kolekcjonerskie zazwyczaj mają na sobie unikalne oznaczenia tekstowe: nazwę, numer katalogowy, kod edycji, itp. Jeśli uda się je z obrazu wyciągnąć w formie tekstowej, można następnie zapytać o taką kartę w bazie danych (np. poprzez API).

W przypadku Twojej aplikacji (skanującej karty Pokémon) sensowne może być właśnie podejście OCR: **wykryj kartę, a następnie odczytaj nazwę karty oraz numer** wydruku. Te dwie informacje w połączeniu zwykle pozwalają jednoznacznie zidentyfikować kartę lub zawęzić wyniki wyszukiwania do właściwej karty. Przykładowo, karta Pokémon ma nazwę (np. *Pikachu*) oraz numer i denominację setu, np. *25/102*. Sama nazwa „Pikachu" może dotyczyć dziesiątek różnych kart (z różnych dodatków), ale już Pikachu o numerze 25/102 wskazuje na konkretną kartę (w tym wypadku np. Pikachu z bazy **Jungle** albo innego setu, trzeba to sprawdzić – to tylko ilustracja).

Techniczna realizacja OCR: Aby skutecznie odczytać tekst z karty, warto wykonać następujące kroki:

- **Przygotowanie obrazu do OCR:** Zakładamy, że mamy już wycięty obraz samej karty (po detekcji i ewentualnym wypłaszczeniu perspektywy). Dobrze jest dodatkowo obrócić go w poprawną orientację, tak aby tekst na karcie był poziomy. Jeśli karta została zeskanowana do góry nogami lub bokiem, należałoby wykryć orientację – można np. spróbować odczytać jakiś tekst i jeśli wyjdzie mocno nieczytelny, obrócić o 90° itd. (Na szczęście karty mają zazwyczaj wyraźną górę/dół, więc wystarczy pilnować trzymania telefonu prosto). Można też ręcznie określić orientację po konturze (np. założyć, że w Pokemonach górna krawędź jest szersza u dołu obrazu kamery – to jednak może być zawodne przy różnych ułożeniach).
- **Lokalizacja pól tekstowych:** W karcie zazwyczaj konkretne informacje są nadrukowane w określonych miejscach. Dla kart Pokémon: nazwa znajduje się na górze karty, wyróżniona większą czcionką; numer karty wraz z symbolem setu jest na dole, zwykle prawy dolny róg w okienku z licencją. Można podejść do OCR na dwa sposoby:
- Uruchomić algorytm OCR na **całym obrazie karty** i potem spróbować wśród wykrytego tekstu znaleźć te właściwe fragmenty (np. nazwa będzie prawdopodobnie najkrótszym i największym tekstem u góry, numer formatu „xx/yyy" rozpoznamy po znaku „/" itp.).
- Albo **wykroić z obrazu tylko interesujące nas regiony** – np. z góry karty pas o określonej wysokości, gdzie powinna być nazwa, oraz z dołu niewielki obszar gdzie szukamy numeru. To wymaga jednak znać z grubsza układ karty. W przypadku oficjalnych kart Pokémon układ jest dość stały, więc można przyjąć proporcjonalnie, że np. ~10% od górnej krawędzi w dół to obszar nazwy, a ~10% od dołu to obszar numeru, i tak je wyciąć do analizy.
- **Odczyt OCR:** Do rozpoznawania tekstu można wykorzystać np. biblioteki typu **Tesseract OCR** (open-source) lub wbudowane mechanizmy platform (na iOS – Vision z rozpoznawaniem tekstu, na Androidzie – ML Kit Text Recognition itp.). Warto upewnić się, że obraz jest o wysokiej rozdzielczości i kontrastowy w miejscach tekstu: czasem pomaga konwersja do czerni i bieli z odpowiednim thresholdem dla samego regionu tekstowego, aby litery odcinały się od tła. Na kartach tło bywa kolorowe lub z grafiką, co może utrudniać OCR – w nazwach Pokémonów zazwyczaj jednak tekst jest ciemny na jasnym tle (biała ramka z nazwą), a numer bywa czarny na jasnym tle w prawym dolnym rogu (choć obok jest symbol setu jako obrazek). Dobre oświetlenie i ostrość są kluczowe – należy wymusić autofokus na karcie przed zrobieniem zdjęcia/skanu. Jeśli użytkownik ruszy aparatem, wyniki OCR drastycznie się pogorszą, więc najlepiej *zamrozić* ujęcie w momencie gdy karta jest wyraźna (TCGplayer robi to automatycznie – wyzwala „skan" gdy obraz jest odpowiednio ostry).
- **Interpretacja wyników OCR:** Po otrzymaniu tekstu z OCR trzeba go oczyścić i wykorzystać. Np. zdarza się, że OCR odczyta coś błędnie – cyfry mogą być przekłamane (1 vs I, 0 vs O), litery w nazwie też (np. „M" vs „N" obok siebie). Można zaimplementować pewne korekty: jeśli wiemy, że

oczekujemy formatu „number/number”, możemy odfiltrować inne znaki. Dla nazwy karty można pokusić się o słownik nazw (np. listę Pokemonów) i sprawdzić najbliższe dopasowanie jeśli OCR zwrócił coś nieidealnego. Jednak często wystarczy po prostu przekazać wynik do wyszukiwarki API – jeśli jest lekko niedokładny, API może nic nie znaleźć lub zwrócić złe wyniki, wtedy użytkownik i tak będzie musiał wybrać ręcznie. W miarę możliwości warto więc poprawiać OCR, ewentualnie **wykorzystać wiele pól**: np. jeśli odczytamy zarówno nazwę, jak i numer, można razem użyć ich do zawężenia wyników (o tym poniżej).

Co ważne, podejście OCR jest relatywnie prostsze do wdrożenia od trenowania własnej sieci rozpoznającej obrazy – korzysta z gotowych silników rozpoznawania tekstu. Redditowi użytkownicy również sugerowali, że do zbudowania skanera kart **nie zawsze potrzeba skomplikowanego AI**, jeśli można oprzeć się na stałych cechach jak nadrukowany numer – np. *„zakładając, że identyfikator (ID) karty zawsze jest w tym samym miejscu, użyłbym po prostu OCR (np. Tesseract) do odczytania go”* [26] . Wiele karcianek ma takie stałe oznaczenia: - W **Magic: The Gathering** nowsze karty mają w lewym dolnym rogu unikalny kod (numer karty oraz skrót edycji i języka, np. *175/280* oraz kod setu) – to pozwala jednoznacznie zidentyfikować kartę [27] . - W **Yu-Gi-Oh!** każda karta ma unikalny kod (np. „LOB-EN000”) nadrukowany w górnej części – również możliwy do odczytania maszynowo. - W **Pokémon TCG** standardem jest numer karty oraz liczba kart w secie (np. *55/108*) oraz mały symbol dodatku. Jeśli zna się mapowanie symbolu na nazwę dodatku lub posiada się bazę, to numer + nazwa dodatku jest unikalny. Bez rozpoznawania symbolu można użyć nazwy karty + numeru + ewentualnie roku wydania (wydrukowany gdzieś w przypisie) do zgadywania wersji.

Podsumowując: **TCGplayer wykorzystuje metodę obrazową**, która jest bardziej zaawansowana, natomiast w aplikacji pisanej własnoręcznie **można z sukcesem zastosować OCR** kluczowych elementów. Obie drogi mają plusy i minusy. Rozpoznawanie obrazem działa nawet, gdy tekst jest mało czytelny (np. rozmazany) – bo opiera się na całości wyglądu – ale wymaga trenowania modelu lub posiadania bazy wzorców. OCR z kolei jest łatwiejsze do wdrożenia na starcie, ale wymaga wyraźnego tekstu na karcie i może być utrudnione przez ozdobne czcionki czy tła (choć nazwy kart zazwyczaj drukowane są dość czytelnie).

Wykorzystanie danych karty i integracja z API

Gdy aplikacja uzyska już informacje o karcie – albo poprzez bezpośrednie rozpoznanie (jak TCGplayer, który już wie jaką kartę znalazł), albo poprzez odczyt nazwy/numeru – kolejnym krokiem jest **zapytnie do bazy danych** w celu uzyskania szczegółów i przedstawienia wyniku użytkownikowi. W przypadku TCGplayer, aplikacja jest powiązana z ich wewnętrznym katalogiem kart i cen. Po rozpoznaniu ID karty, aplikacja natychmiast wyświetla użytkownikowi np. nazwę, rozszerzenie (set), aktualną cenę rynkową itp., pobrane z katalogu TCGplayer. Użytkownik może dodać kartę do swojej kolekcji, listy sprzedaży czy gdzie tam potrzebuje. W Twoim przypadku chcesz użyć zewnętrznego API – np. **Pokémon TCG API** (dostępnego przez RapidAPI) – do znalezienia informacji o kartach Pokémon.

Scenariusz może wyglądać tak: po zeskanowaniu karty aplikacja odczytała, że to np. *„Pikachu”* nr *25/102*. Konstruujesz więc zapytanie HTTP do API kart Pokémon, przekazując te dane. Według podanego przykładu, API akceptuje parametry zapytania jak `search=pikachu` oraz ewentualnie sortowanie [28] . Być może można też zawęzić po numerze lub edycji – to zależy od możliwości API. Jeśli nie, najprostsze podejście to wysłać zapytanie z samą nazwą karty (np. `search=Pikachu`) i ewentualnie posortować wyniki tak, by najbardziej prawdopodobny był pierwszy (choć parametr `sort=episode_newest` w przytoczonym URL wygląda nietypowo, może chodziło o sortowanie po najnowszym wydaniu).

Otrzymanie i obsługa wyników: API zwróci zapewne listę pasujących kart (w przypadku samej nazwy *Pikachu* może być kilkadziesiąt wyników różnych Pikachu z różnych zestawów). Dlatego aplikacja

powinna obsłużyć kilka scenariuszy: - Jeśli wynik jest jednoznaczny (np. użytkownik podał na tyle szczegółów, że API zwróci dokładnie jedną kartę), można automatycznie ją wyświetlić jako rozpoznaną. - Jeśli jest wiele wyników, warto **wyświetlić najlepsze dopasowanie** na górze, ale dać opcję użytkownikowi, by rozwinął listę i wybrał inny. TCGplayer robi podobnie przy niższej pewności – pokazuje przypisanie karty, ale umożliwia zmianę zestawu/wersji [8]. U Ciebie np. możesz pokazać pierwszą kartę z wyników API (zakładając, że jest to właściwa w większości przypadków), wraz z jakąś informacją (np. nazwą zestawu, numerem) i zapytać „Czy ta karta się zgadza?". Jeśli tak – użytkownik zatwierdza i przechodzicie dalej. Jeśli nie – użytkownik wybiera z listy inny wariant (np. znalazło Pikachu z innego setu – wtedy wybiera właściwy). - Jeśli OCR odczytało zarówno nazwę, jak i numer, można spróbować łącznie sformułować zapytanie. Np. niektóre API pozwalają zapytać o *name+number* jednocześnie. Jeśli nie bezpośrednio, można pobrać wszystkie karty o danej nazwie i po stronie aplikacji przefiltrować po numerze. Na przykład: wyszukaj „Pikachu", dostajesz listę Pikachu z różnymi numerami i setami; wtedy aplikacja sama może znaleźć w tych wynikach taki, który ma `cardNumber == 25` (zakładając numer przed ukośnikiem) lub `cardNumber == "25/102"` jeśli API tak podaje. Taki wynik wtedy prezentujesz jako domyślny.

Ważnym elementem jest zapewnienie przyjaznego **UX (doświadczenia użytkownika)** – czyli płynne przejście od zeskanowania do wyniku. W idealnym przypadku skanowanie jednego obrazu karty powinno skutkować natychmiastowym pokazaniem co to za karta i np. dodaniem jej do listy. Gdy jednak pojawią się wątpliwości (wiele wyników, niski confidence), lepiej zatrzymać się i poprosić użytkownika o wybór. Zbyt automatyczne przypisywanie złej karty byłoby mylące. TCGplayer dla pewności pokazuje zeskanowaną kartę obok wizerunku katalogowego dopasowanej karty, by użytkownik mógł sam porównać czy wszystko się zgadza [8].

Twoja aplikacja po wyświetleniu wyniku powinna następnie **oczekiwać na kolejną kartę** pojawiającą się w kadrze. Najlepiej, by cały proces się zapętlał: użytkownik przykłada następną kartę przed aparat, aplikacja automatycznie ją wykrywa i rozpoznaje, znów wyświetla wynik do akceptacji itd., aż do zakończenia skanowania kolekcji. Pomiędzy kolejnymi kartami warto dać krótką informację (np. „Umieść następną kartę przed obiektywem...") i ewentualnie możliwość przerwania/skonczenia skanowania.

Optymalizacja procesu skanowania – wskazówki i potencjalne problemy

Budując własną aplikację do skanowania kart, napotkasz szereg wyzwań technicznych. Wiele z nich to rzeczy, które twórcy TCGplayer już rozwiązali lub zminimalizowali. Poniżej kilka wskazówek i typowych problemów wraz z rozwiązaniami:

- **Oświetlenie i ostrość:** Jak wspomniano, dobre oświetlenie jest kluczowe. Unikaj silnych odbić – lepsze jest jasne, rozproszone światło. W ciemnych warunkach rozważ zaświecenie latarki telefonu. Upewnij się, że autofocus ostrzy na powierzchnię karty – czasem aparat w telefonie może „szukać" ostrości. Można skorzystać z ciągłego autofocusa (typowo tryb `continuous-picture` / `continuous-video`) i podpowiedzieć użytkownikowi, by trzymał aparat ok. 15 cm od karty [18] (TCGplayer rekomenduje ~6–8 cali, czyli ~15–20 cm odległości od karty [18]). Zbyt blisko – może nie złapać ostrości; zbyt daleko – karta zajmie mały obszar obrazu i tekst może być nieczytelny.
- **Stabilność ujęcia:** Jeśli planujesz automatyczne wyzwalanie skanu (bez klikania przycisku przez użytkownika), zaimplementuj warunek stabilności obrazu. Możesz np. sprawdzać, czy kontur karty jest wykrywany przez pewną liczbę kolejnych klatek i nie zmienia znacząco położenia – to oznaka, że użytkownik trzyma telefon nieruchomo nad kartą. Wtedy możesz automatycznie

„zamrozić” klatkę do rozpoznania (albo zrobić zdjęcie). To zapobiegnie sytuacji, gdzie OCR czy rozpoznawanie działa na rozmytym kadrze w ruchu.

- **Różnorodność kart:** Jeśli aplikacja ma wspierać wiele gier (nie tylko Pokémon), musisz przygotować się na różne rozmiary, kolory obramowań, położenie tekstu itd. TCGplayer radzi sobie z kartami **Magic, Pokémon, Yu-Gi-Oh!, One Piece, Dragon Ball, itd.** – to możliwe dzięki trenowaniu modelu na wielu danych oraz dodatkowym ustawieniom skanowania (np. wybór gry w aplikacji). Własną aplikację możesz na początek zawęzić do jednej gry dla uproszczenia. Jeśli jednak planujesz obsłużyć np. też Magic czy Yu-Gi-Oh, rozważ pozwolenie użytkownikowi wybrania rodzaju gry przed skanowaniem albo automatyczne rozpoznanie po wymiarach/aspekcie (np. karty Yu-Gi-Oh są trochę mniejsze i mają inny układ graficzny). Wtedy będziesz mógł dostosować algorytm OCR (inne miejsce numeru) lub bazę wyszukiwania w zależności od gry.
- **Błędy OCR i walidacja:** Przy odczytywaniu tekstu z kart często zdarzają się drobne pomyłki. Warto wprowadzić pewne walidacje – np. nazwy kart z reguły składają się z liter (czasem cyfr, np. „No. 7”), ale raczej nie zawierają dziwnych znaków. Numer karty to liczby i ewentualnie ukośnik „/” i litery oznaczające set (czasem). Jeśli OCR zwróci coś, co nie pasuje do oczekiwanego formatu, możesz spróbować ponownie z innymi parametrami (np. innym thresholdem) lub oznaczyć wynik jako niepewny i poprosić użytkownika o ręczne wpisanie nazwy. Lepsze to niż automatyczne wyszukanie złej nazwy. Podobnie, jeśli API zwróci kompletnie nieskoordynowane wyniki, warto użytkownika o tym poinformować („Nie znaleziono karty – spróbuj ponownie zeskanować” albo dać opcję poprawy ręcznej).
- **Czas przetwarzania:** Postaraj się, by większość ciężkich obliczeń (detekcja konturów, OCR) działo się w czasie rzeczywistym (na żywo w podglądzie kamery), *zanim* user interface pokaże wynik. Ideałem jest doświadczenie „real-time” – użytkownik przesuwa kamerę nad kartami, a aplikacja od razu rozpoznaje kolejne karty niemal płynnie. TCGplayer mocno zoptymalizował szybkość – skanowanie jest natychmiastowe nawet na telefonie [7]. Ty możesz na początek zrobić tryb pojedynczego zdjęcia (naciśnij, zrób foto i wtedy wykryj+OCR), ale docelowo dąż do optymalizacji: np. używaj strumienia z kamery w niskiej rozdzielczości do wykrycia konturu (szybciej), a dopiero gdy wykryjesz kartę – zrób zdjęcie w wyższej jakości do OCR/rozpoznania. Takie podejście daje responsywność i dokładność tam, gdzie potrzebna.
- **Testowanie i uczenie się na błędach:** Rozpoznawanie kart to proces podatny na różne rzeczy (nowe typy kart, zmiany designu, uszkodzenia kart, itp.). Warto testować aplikację na wielu przykładach. TCGplayer np. napotkał trudności z pewnymi kartami: w materiałach wspominają, że **karty z unikalnymi foliami, podstawowe lądy (basic lands) w Magicu, tokeny, czy bardzo stare wydania mogą być trudniejsze do rozpoznania automatycznego** [29] [30]. Powodem bywa albo nietypowy wygląd, albo brak unikalnych cech (np. wiele podstawowych lądów ma tę samą ilustrację z drobnymi różnicami). Twoja aplikacja może mieć analogiczne wyzwania – np. podstawowe karty Energii w Pokémon (które mają niemal ten sam obraz i różnią się tylko symbolem – tu OCR „Nazwa” by pomógł, bo energia ma napis „Energy – Fire” itp.). Przygotuj się, że 100% automatyzacji jest trudne – czasem potrzebna będzie interwencja użytkownika lub dopracowanie algorytmu dla specyficznych przypadków.

Na koniec, warto podkreślić, że **połączenie różnych technik daje najlepsze rezultaty**. TCGplayer osiąga wysoką skuteczność, bo wykorzystuje potężny model CV (*computer vision*) wsparty gigantyczną bazą danych kart oraz sprytnymi mechanizmami oceny pewności. W Twojej aplikacji możesz zacząć od prostszego OCR połączonego z wyszukiwaniem przez API – jest to wykonalne i wiele projektów hobbystycznych poszło tą drogą [31] [27]. Z czasem, jeśli baza kart urośnie lub OCR okaże się zawodny w pewnych sytuacjach, możesz spróbować wprowadzić elementy „uczące się” (np. trenować sieć do rozpoznawania kart na obrazach). Już teraz jednak dysponujesz planem: **wykryj kartę w kamerze, wyodrębnij obraz, odczytaj nazwę i numer, znajdź kartę w API, pokaż wynik i potwierdź – powtórz dla kolejnych kart**. Powodzenia w implementacji!

Źródła:

- Oficjalne materiały pomocy TCGplayer opisujące działanie i wskazówki do skanowania kart [1] [9] [11] [8].
- Blogi techniczne i dyskusje developerów o rozpoznawaniu kart za pomocą CV i OCR [23] [24] [26].
- Projekt open-source *Magic Card Detector* ilustrujący proces wykrywania konturów i identyfikacji kart z użyciem metod klasycznych (OpenCV, hash obrazu) [14] [17] [21].

---

[1] [3] [4] [5] [6] [29] [30] How Scan & Identify Technology Works – TCGplayer.com
https://help.tcgplayer.com/hc/en-us/articles/27303183354007-How-Scan-Identify-Technology-Works

[2] Announcing Scan & Identify: Revolutionize Your Inventory Listing
https://seller.tcgplayer.com/blog/announcing-scan-identify-revolutionize-your-inventory-listing

[7] TCGplayer on the App Store
https://apps.apple.com/us/app/tcgplayer/id1247645833

[8] [9] [10] [11] [25] Tips for Accurate Scanning – TCGplayer.com
https://help.tcgplayer.com/hc/en-us/articles/115009674788-Tips-for-Accurate-Scanning

[12] [13] [14] [15] [16] [17] [21] Magic Card Detector
http://tmikonen.github.io/quantitatively/2020-01-01-magic-card-detector/

[18] [28] How to Setup and Use the TCGplayer App for Card Scanning – TCGplayer.com
https://help.tcgplayer.com/hc/en-us/articles/23531246396183-How-to-Setup-and-Use-the-TCGplayer-App-for-Card-Scanning

[19] [20] [22] [23] [24] Automating Card Identification Using Computer Vision - Collectors Tech Blog
https://blog.collectors.com/image-search/

[26] [27] [31] TCG Card Scanner : r/howdidtheycodeit
https://www.reddit.com/r/howdidtheycodeit/comments/173h70t/tcg_card_scanner/