Aleksander LAMZA

Uniwersytet Jana Kochanowskiego w Kielcach, Wydział Nauk Ścisłych i Przyrodniczych
ORCID: 0000-0002-9763-2105

doi:10.15199/48.2023.04.32

Autonomiczna kamera do wykrywania wolnych miejsc parkingowych

Streszczenie. *W artykule została opisana architektura systemu autonomicznej kamery do wykrywania wolnych miejsc parkingowych. System został zrealizowany i przetestowany zarówno w warunkach laboratoryjnych, jak i rzeczywistych. Omówione zostały kluczowe elementy systemu, ich współdziałanie, a także możliwości modyfikacji i rozwoju.*

Abstract*. The article describes the architecture of an autonomous camera system to detect free parking spaces. The system was implemented and tested both in laboratory and real conditions. The key elements of the system, their interaction, as well as the possibilities of modification and development were discussed. (**An autonomous camera for detecting free parking spaces**)*

Słowa kluczowe: systemy monitorowania, miejsca parkingowe, kamera, sztuczne sieci neuronowe.
Keywords: surveillance systems, parking lots, camera, artificial neural networks.

Wstęp

Wraz ze wzrostem liczby samochodów w ruchu miejskim coraz pilniejsze staje się zapewnienie dostępu do miejsc parkingowych. Dotyczy to zwłaszcza dużych miast, a w szczególności lokalizacji o ponadprzeciętnym obciążeniu parkującymi pojazdami, takimi jak wydzielone miejskie parkingi, centra handlowe czy węzły komunikacyjne. Rośnie więc zapotrzebowanie na systemy informacyjne zapewniające dostęp do danych na temat lokalizacji parkingów i – co najważniejsze – pozwalające na znalezienie wolnego miejsca.

W wybranych lokalizacjach instalowane są systemy monitorujące przestrzenie parkingowe. Niektóre oferują możliwość wskazywania wolnych miejsc (dotyczy to zwłaszcza parkingów centrów handlowych). Są to jednak systemy lokalne, powiązane z konkretnym miejscem. Co więcej, tego typu rozwiązania można wprowadzić zazwyczaj tylko w ściśle określonych warunkach, co wynika ze sposobu działania. Najczęściej stosuje się czujniki powiązane z miejscami, takie jak np. czujnik pola magnetycznego, czujnik odległości [2]. Wiążą się z tym stosunkowo duże koszty instalacji oraz serwisowania. Innym rozwiązaniem jest skorzystanie z kamer i metod analizy obrazu [1]. Rozwiązania tego typu można stosować na otwartych przestrzeniach, jednak wymagają znacznie większych mocy obliczeniowych w związku z zastosowaniem sieci neuronowych i algorytmów analizy obrazu [3].

Głównym celem omawianego projektu jest wyjście naprzeciw zasygnalizowanym tu potrzebom i zaprojektowanie oraz zrealizowanie systemu udostępniającego informację o miejscach parkingowych w wygodny dla użytkownika, zagregowany sposób.

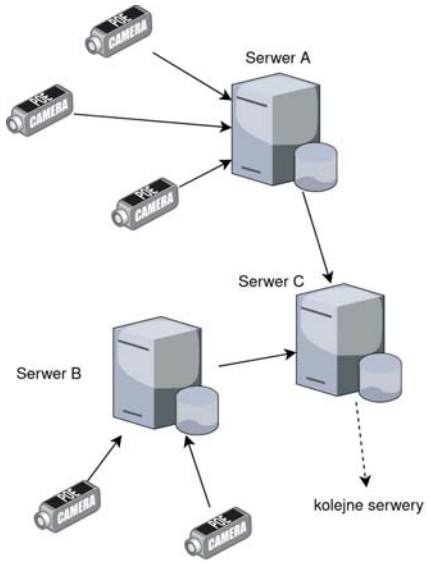
Na etapie koncepcyjnym przyjęto następujące założenia:

- głównym komponentem systemu są autonomiczne kamery,
- kamery dokonują analizy obrazu w celu wykrycia wolnych i zajętych miejsc parkingowych,
- kamery wysyłają dane o miejscach parkingowych do serwera lub serwerów,
- w systemie może się znajdować dowolna liczba kamer,
- wymagany jest co najmniej jeden serwer odpowiedzialny za integrowanie i składowanie danych z kamer; liczba serwerów nie jest ograniczona,
- dane o miejscach parkingowych mogą być udostępniane innym systemom i aplikacjom.

Tak zdefiniowane ramy systemu są szerokie i pozwalają na elastyczne dostosowanie do konkretnej sytuacji. Omawiane rozwiązanie sprawdzi się dzięki temu zarówno w przypadku małego parkingu z jedną kamerą, jak i dużego, wielopoziomowego parkingu, bądź sieci parkingów.

Architektura systemu

Na podstawie przedstawionych wyżej założeń zaprojektowano stosowną architekturę (rys. 1). Daje ona pełną swobodę w doborze liczby zarówno samych kamer, jak i serwerów składujących i udostępniających dane.



Rys. 1. Ogólna architektura systemu sieci kamer

Poniżej zostaną omówione główne komponenty omawianego systemu wraz z uwagami dotyczącymi dokonanych wyborów na potrzeby zrealizowania działającego prototypu.

Autonomiczna kamera

Jak wspomniano, kamery są autonomiczne. Oznacza to, że do prawidłowej pracy nie wymagają innych elementów systemu. Każda kamera dokonuje akwizycji obrazu, rozpoznaje miejsca parkingowe (na podstawie dostarczonej konfiguracji), analizuje na bieżąco obraz i przygotowuje

paczkę danych z informacjami o zajętych i wolnych miejscach.

Pierwszą decyzją, którą należało podjąć, było wytypowanie zastosowanej matrycy kamery. W przypadku opracowywanego rozwiązania przeanalizowano dostępne na rynku materiały i określono jaki rodzaj przetwornika zostanie użyty. W tym celu przeglądnięto dostępne kamery USB oraz przeanalizowano rozwiązania embedded czujników obrazu. Na podstawie testów, wstępnych implementacji, projektu PCB stwierdzono, że warto skoncentrować się na rozwiązaniach przedstawionych poniżej. W tabeli 1 znajduje się zestawienie cech typów przetworników.

Tabela 1. Parametry czujnika

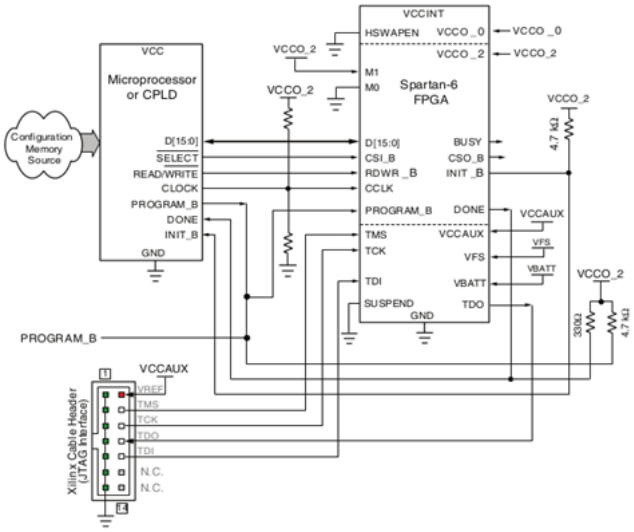
CCD	CMOS
Nie można odczytać zawartości pojedynczego piksela. Trzeba odczytać zawartość całej matrycy i potem dopiero wybrać interesujący nas piksel. To powoduje, że ich działanie jest dość powolne	Można odczytywać zawartość dowolnej liczby pikseli i w dowolnej kolejności, tak jak odczytuje się zawartość pamięci komputerowych. Z tego względu działają znacznie szybciej
Matryca ma jeden przetwornik ładunku na napięcie i jeden przetwornik A/D. Zawartość wszystkich pikseli jest odczytywana po kolei przez ten układ	Każdy piksel matrycy CMOS ma swój przetwornik ładunek na napięcie i układ odczytujący zawartość pikseli odczytuje już napięcie wytworzone padającym na ten piksel światłem. W bardzo zaawansowanych matrycach CMOS każdy piksel ma swój przetwornik A/D, co ułatwia i przyspiesza dalszą obróbkę obrazu
Ze względu na swą budowę matryce CCD pobierają więcej mocy w czasie pracy, bardziej się więc grzeją	Zużywają mniej mocy elektrycznej
Większy współczynnik wypełnienia, czyli stosunek powierzchni pikseli do powierzchni całej matrycy	Mniejszy współczynnik wypełnienia, gdyż część powierzchni matryc zajmują obwody przetwarzające ładunek na napięcie
Mniejsze szumy	Większe szumy

W ramach wstępnych badań przetestowano i wytypowano kamerę Logitech C920. Kamera ta idealnie nadaje się w sytuacjach, gdy potrzebna jest jej integracja z własnym rozwiązaniem i przystosowaniem przechwytywania obrazu na inną niż standardowe platformę obliczeniową. Podstawowe parametry kamery:

- matryca CMOS, 2 megapiksele,
- rozdzielczość maksymalna: 1920 x 1080 pikseli,
- kąt widzenia: 78 stopni,
- interfejs: USB.

W celu wyznaczenia wolnych i zajętych miejsc na obrazie pobranym z kamery należy zaimplementować odpowiedni mechanizm analizy obrazu. Na etapie wstępnych badań wybór padł na sieć neuronową. W związku z tym konieczne było zapewnienie platformy sprzętowej pozwalającej na implementację sieci neuronowej działającej w czasie rzeczywistym. W projekcie zdecydowano się na zastosowanie połączenia trzech komponentów: układu FPGA, mikroprocesora oraz komputer PC. Analiza przeprowadzona na wstępnym etapie wykazała że zastosowane zostaną układy firmy Xilinx, rodzina Spartan [4]. Przeszukano sieć w celu określenia, które rozwiązanie będzie najwłaściwsze, przeanalizowano schematy i odniesiono funkcjonalność płyt do ich ceny. Wybór padł na płytę Spartan-6 SP601 (rys. 2).

Rys. 2. Płytka prototypowa Spartan-6 SP601.



Rys. 3. Ogólny schemat modułu układu FPGA.

Płyta prototypowa została wykorzystana w początkowych, badawczych etapach pracy nad systemem. Docelowym rozwiązaniem jest układ zaprojektowany na potrzeby kamery, przy czym podstawa – czyli dobór FPGA, mikroprocesora i peryferiów pozostaje bez zmian względem pierwszego prototypu. Po wyborze układów z serii Spartan 6 pobrano ze strony producenta najistotniejsze dokumenty, które są niezbędne do wykonania projektu PCB oraz projektu wsadu do układu FPGA. Z najistotniejszych dokumentów to:

- podręcznik konfiguracji (Spartan-6 FPGA Configuration User Guide) [5],
- parametry zasilania (Spartan-6 FPGA Data Sheet: DC and Switching Characteristics) [6],
- opis obudowy i wyprowadzeń (Spartan-6 FPGA Packaging and Pinouts) [7].

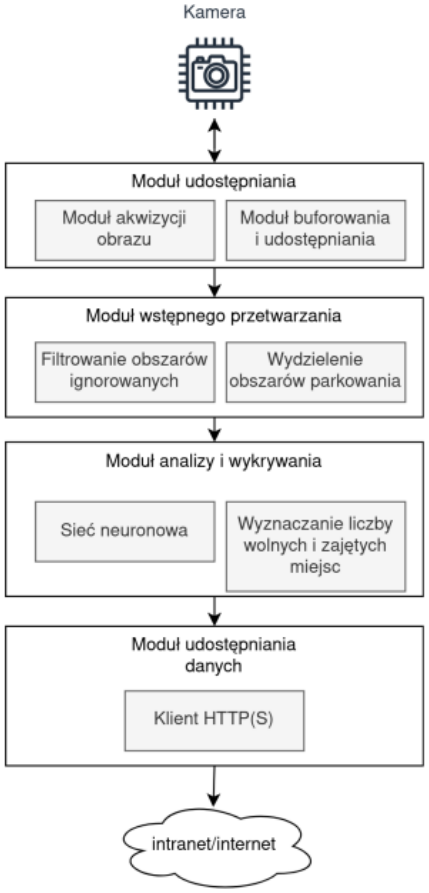
Dokumenty te są potrzebne do konfiguracji układ FPGA za pomocą zewnętrznego procesora, a także do określenia wymagań prądowo-napięciowych dla toru zasilania dla układu FPGA oraz do przygotowania symbolu i footprintu na potrzeby wykonania projektu PCB. W omawianym przypadku do programowania układu FPGA używamy interfejsu SelectMap którego schemat ogólny przedstawiono na rysunku 3.

Na podstawie wytypowanego układu stworzono projekt w środowisku ISE oraz określono jaka wielkość układu jest potrzebna by zaimplementować kod przetwarzający ramkę obrazu wg uproszczonego algorytmu. W tym celu zdecydowano się stworzyć projekt zawierający pełny interfejs do procesora. Interfejs ten zawiera magistralę

danych, adresową, sygnały kontrolne. Drugą stronę interfejsu stanowi magistrala do pamięci RAM. Po stronie procesora przygotowano oprogramowanie, które było odpowiedzialne za przetestowanie opracowanej implementacji dla układu FPGA. Całość sprawdzono w rzeczywistym układzie. Na podstawie opracowanego kodu uruchomiono implementację i otrzymano wyniki jednoznacznie wskazujące, że tego typu układy mają wystarczająco dużo zasobów, aby obsłużyć przetwarzanie obrazu pobieranego z matrycy kamery. Dla układu Spartan-3 otrzymano 72% zużytych zasobów, a dla Spartan-6 58%. Należy podkreślić, że mowa tu o podstawowej implementacji i obsłudze kluczowych rejestrów.

Kolejnym ważnym zagadnieniem jest dobór interfejsu między układem FPGA a matrycą kamery. W wyniku testów, wstępnych implementacji i badań w środowisku symulacyjnym jednoznacznie wyszło, że interfejs komunikacyjny opracowanej kamery, w części pomiędzy kamerą a procesorem powinien być oparty o magistralę USB. Co ważne, nie można mylić interfejsu całej kamery od modułu wewnątrz niej, który odpowiada za dostarczenie nieskompresowanych ramek obrazu do głównego procesora. To właśnie w tym drugim przypadku wskazane jest, by dane były przesyłane za pomocą interfejsu USB. Cała natomiast kamera będzie miała interfejs sieciowy, taki jak Ethernet lub WiFi.

Na rysunku 4 został przedstawiony przepływ sterowania w autonomicznej kamerze. Jak widać, zostały wyszczególnione cztery główne moduły: udostępniania, wstępnego przetwarzania, analizy i wykrywania oraz udostępniania danych. Oprogramowanie zostało w większości napisane w języku Python, ze względu na jego wszechstronność, przenośność i dostępność bibliotek. Kluczowe i wymagające obliczeniowo mechanizmy, takie jak np. akwizycja danych, zostały zrealizowane sprzętowo.



Rys. 4. Schemat przepływu sterowania autonomicznej kamery.

Sieć serwerów

Autonomiczne kamery będące głównym elementem realizowanego projektu udostępniają dane m.in. o zajętych i wolnych miejscach parkingowych. Aby ta informacja była użyteczna, konieczne jest stworzenie systemu integrującego te dane. Został zaprojektowany rozproszony system, który ma nieograniczoną technicznie możliwość skalowania. Poglądowy schemat systemu został przedstawiony już wcześniej na rysunku 1.

Każdy z przedstawionych tu serwerów pełni przede wszystkim rolę serwera składującego dane. Kamery wysyłają do nich dane o zajętości monitorowanych miejsc. Dane te są zapisywane w lokalnych bazach danych serwerów (np. serwer A i B z rysunku), ale mogą być dalej przesyłane do kolejnych serwerów (serwer C z rysunku). Serwer ten integruje dane z wielu kamer, dając możliwość udostępnienia ich w formie zestawień, map itp. Za udostępnianie danych użytkownikom końcowym odpowiadają inne serwery (np. webowe).

W zależności od lokalnej konfiguracji na serwerach, propagowanie danych do kolejnych węzłów sieci może przebiegać na bieżąco, ale również w ustalonych interwałach. Istnieje tu pełna dowolność konfiguracji. Tak duża elastyczność zaprojektowanego systemu pozwala na zastosowanie go w wielu przypadkach i dostosowania do potrzeb wielu klientów.

Przyjęto założenie, że każdy serwer ma własną bazę danych. Wybór był podyktowany przede wszystkim tym, że z założenia system miał być rozproszony, zdecentralizowany. Dzięki temu podnosi się poziom bezpieczeństwa danych, ponieważ uszkodzenie jednego serwera nie powoduje utraty danych przesłanych przez kamery połączone z innym serwerem. Ponadto sieć rozproszona umożliwia dowolne skalowanie, co w przypadku systemów scentralizowanych nie jest możliwe, ponieważ granica skalowalności jest ustalana przez możliwości serwera centralnego. Przy odpowiednio zaprojektowanej sieci serwerów możliwe jest zapewnienie 100-procentowego bezpieczeństwa danych, które uzyskuje się dzięki celowej redundancji, czyli przechowywania danych z poszczególnych kamer na wielu (w szczególnym przypadku na wszystkich) serwerach.

Konfiguracja kamery

Ponieważ kamera jest częścią sieci, należy ją odpowiednio skonfigurować, by mogła komunikować się z serwerem. Poza tym konieczne jest nadanie kamerze identyfikatora, dzięki któremu może być jednoznacznie określana.

Rys. 5. Zrzut ekranu aplikacji do konfigurowania obszarów parkowania.

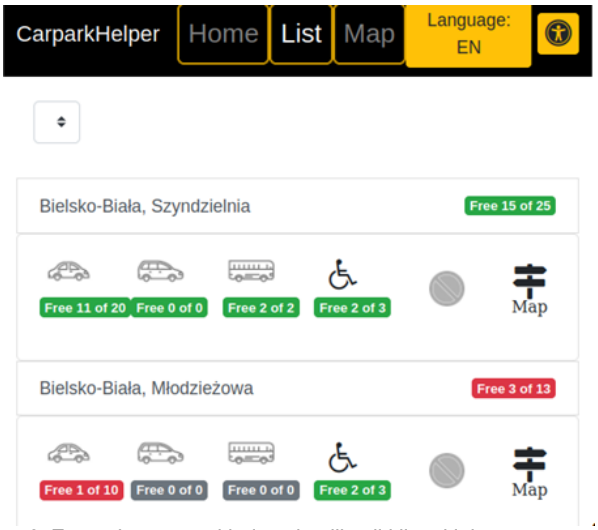
Nieodłącznym etapem konfiguracji kamery jest również zdefiniowanie w obserwowanym obrazie miejsc uznawanych za prawidłowe miejsca parkingowe, miejsca ignorowane przez system oraz miejsca, w których parkowanie jest zabronione. Dzięki temu kamera może

wysyłać do serwerów prawidłowe dane o zajętości miejsc. Do konfiguracji służy wygodne narzędzie, w którym graficznie można zaznaczać określone obszary. Przykładowy zrzut ekranu okna aplikacji został przedstawiony na rysunku 5.

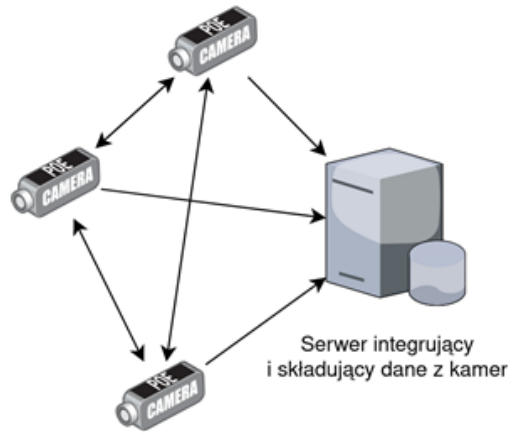
Aplikacje dla użytkowników końcowych

Dane przesyłane przez kamery są składowane w bazach danych na serwerach. Każdy serwer udostępnia API zgodne z architekturą REST. Interfejs ten umożliwia dostarczanie danych do dowolnej aplikacji bądź innego systemu. Jedynym warunkiem jest obsługa protokołu HTTP(S). Została opracowana szczegółowa dokumentacja tego interfejsu, dzięki czemu rozwój możliwych aplikacji klienckich jest otwarty.

W ramach działającego prototypu powstały przykładowe aplikacje. Zrzut ekranu jednej z nich został przedstawiony na rysunku 6.



Rys. 6. Zrzut ekranu przykładowej aplikacji klienckiej.



Rys. 7. Bezpośrednie połączenie kamer między sobą.

Możliwości rozwoju

Poza opisanym wcześniej sposobem komunikacji między kamerami i serwerami istnieje możliwość wprowadzenia dodatkowo połączenia bezpośredniego między kamerami. Poglądowy schemat tego podejścia został przedstawiony na rysunku 7.

Jest to wycinek schematu przedstawionego wyżej. Jak widać, kamery mają możliwość wymiany informacji bezpośrednio, dzięki czemu mogą tworzyć sieć niezależną od sieci serwerów. To rozwiązanie mogłoby się okazać przydatne, gdy wymaganiem byłoby sterowanie np. obszarem obserwowanym z poziomu kamer. W przypadku dużego monitorowanego obszaru kamery mogłyby przesyłać sobie informacje o potrzebie monitorowania wybranych, np. jeszcze mało zajętych miejsc. Rozwiązanie to należy jednak traktować jako opcję, ponieważ podstawowy zakres funkcjonalny systemu jest realizowany w standardowej formie opisanej wcześniej. W wersji prototypowej kamery nie przechowują historycznych danych dotyczących zajętości miejsc, ponieważ taki mechanizm nie znalazł się w puli wymagań. Założeniem jest przesyłanie danych na bieżąco do serwera. Można sobie jednak wyobrazić scenariusz, w którym kamera, jako urządzenie autonomiczne, działa bez serwera i ma za zadanie rejestrować zajętość miejsc. Taki mechanizm można bez większych trudności zaimplementować. Ograniczeniem jest jedynie dostępna przestrzeń na lokalne składowanie danych, co można zniwelować poprzez dołożenie zewnętrznego nośnika danych.

Projekt pt. "Innowacyjna autonomiczna kamera do wykrywania wolnych miejsc parkingowych i zajętość pasów ruchu w oparciu o sieci neuronowe" (nr RPSL.01.02.00-24-065D/17) współfinansowany ze środków Europejskiego Funduszu Rozwoju Regionalnego w ramach Regionalnego Programu Operacyjnego Województwa Śląskiego na lata 2014-2020.

***Autor**: dr Aleksander Lamża, Uniwersytet Jana Kochanowskiego w Kielcach, Wydział Nauk Ścisłych i Przyrodniczych, Instytut Fizyki, ul. Uniwersytecka 7, 25-406 Kielce, E-mail: aleksander.lamza@us.edu.pl.*

LITERATURA

[1] Tiabur-Rahman T., Sk.Shariful A., Amanullah A., Zwiększenie dokładności wykrywania pojazdów na parkingach dzięki funkcjom podobnym do algorytmu Haar i AdaBoost, *Przegląd Elektrotechniczny*, 9/2021, 135-138.

[2] Szarata A., Rola pozyskiwania danych w kontekście funkcjonowania stref płatnego parkowania, Zeszyty Naukowo-Techniczne SITK RP, 1/2015 (105), Kraków, 107-117.

[3] Debaditya A., Weilin Y., Kourosh K., Real-time image-based parking occupancy detection using deep learning, Proceedings of the 5th Annual Research@Locate Conference, Adelaide, 2018. AustraliaVolume: 2087.

[4] Strona z dokumentacją płytki prototypowej Spartan-6 SP601, https://support.xilinx.com/s/topic/0TO2E000000YKg9WAG/spartan6-fpga-sp601-evaluation-kit.

[5] Podręcznik konfiguracji układu FPGA Spartan-6, https://docs.xilinx.com/v/u/en-US/ug380.

[6] Parametry zasilania układów FPGA Spartan-6, https://docs.xilinx.com/v/u/en-US/ds162.

[7] Opis obudowy I wyprowadzeń układów FPGA Spartan-6, https://docs.xilinx.com/v/u/en-US/ug385.