POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY
INSTYTUT STEROWANIA I ELEKTRONIKI PRZEMYSŁOWEJ

PRACA MAGISTERSKA
na kierunku AUTOMATYKA I ROBOTYKA

Adam Srebro
Nr albumu 181889

Marcin Śladowski
Nr albumu 173747

Rok. akad. 2005/2006
Warszawa, 13.11.2006

„Autonomiczny system wizyjny robota mobilnego”

Zakres pracy:

1. Wprowadzenie
2. Budowa układu wizyjnego z zastosowaniem kamery cyfrowej
3. Algorytm analizy obrazu
4. Ocena zastosowanej metody przetwarzania obrazu.
5. Budowa programu
6. Rozwiązanie problemów związanych z realizacją
7. Przykładowe możliwości dalszej rozbudowy
8. Podsumowanie i wnioski.

Podpis i pieczątka
Kierownika Zakładu Dydaktycznego

Opiekun naukowy:
dr inż. Witold Czajewski

Termin wykonania: Grudzień 2006

1. Wstęp

Zmysł wzroku jest jednym z głównych ośrodków percepcji otoczenia u człowieka. Dzięki odbieraniu bodźców świetlnych człowiek jest w stanie klasyfikować obiekty uwzględniając ich kształt i kolor. Zmysł ten odpowiedzialny jest w dużym stopniu za orientację w otoczeniu. Pozwala określić pozycję obiektów względem obserwatora. Te i wiele innych zalet wzroku starano się wykorzystać w systemach komputerowych do wykonywania różnych zadań. O ile wykonywanie pojedynczego zadania polegającego np. na wykrywaniu obiektów o stałych cechach w niezmiennych warunkach otoczenia nie jest trudne dla systemu wyposażonego w komputer klasy PC to wykonanie takiego zadania przez zminiaturyzowany system autonomiczny nie jest już łatwe. Mimo wielu ograniczeń tj. wymiary, szybkość przetwarzania danych zużycie energii i wymagana niezawodność systemy te są obecnie intensywnie rozwijane. Spowodowane jest to faktem, zapotrzebowania na roboty inspekcyjne, które mogłyby pracować w warunkach środowiskowych gdzie występują duże zakłócenia transmisji sygnałów lub zasięg transmisji ogranicza pole działania robota. Eksploracja tuneli, bunkrów czy powierzchni nieznanych planet to tylko jeden z aspektów zastosowania autonomicznych robotów.

Rozwiązania wykorzystujące wizję mają przewagę nad innymi systemami percepcji otoczenia jak np. systemy ultradźwiękowe czy podczerwieni. Z obrazu można wywnioskować więcej szczegółów, a tym samym teoretycznie precyzyjniej zrealizować założony cel. Budowa urządzeń mogących samodzielnie i dokładnie wykonywać powierzone im zadania, a przy tym energooszczędnych (brak strat na przesył sygnałów) i niedrogich w produkcji to cel wielu ośrodków robotyki na całym świecie.

Autorzy tej pracy będący uczestnikami Ogólnopolskich Zawodów Sumo Robotów wybrali wykonanie konkretnego zadania - budowy autonomicznego systemu wizyjnego z przeznaczeniem dla robota sumo. Wybór ten był podyktowany chęcią zbudowania systemu wykrywania przeciwnika na ringu sumo, który byłby lepszy od dotychczas stosowanych rozwiązań.

1.1. Cel pracy

Podstawowym celem pracy jest stworzenie autonomicznego systemu wizyjnego dla robota sumo, który byłyby pozbawiony wad dotychczas stosowanych czujników. Celem pobocznym jest udowodnienie przydatności pojedynczego mikroprocesora do analizy obrazu na żywo.

1.2. Układ pracy

Niniejsza praca magisterska składa się część elektronicznej i informatycznej oraz badań i symulacji.

Rozdział 2 wprowadza w tematykę zawodów sumo robotów.

Rozdział 3 opisuje etapy tworzenia układów elektronicznych potrzebnych do obsługi kamery cyfrowej. W ich skład wchodzi układ zasilania, przetwarzania danych i układy komunikacyjne. Ponadto opisuje wykorzystywane interfejsy komunikacyjne.

Rozdział 4 zawiera charakterystykę układu optycznego. Począwszy od budowy i opisu działania kamery, aż po używane tryby zapisu zdjęcia.

Rozdział 5 jest opisem budowy algorytmu przetwarzania obrazu z wyszczególnieniem problemu optymalizacji szybkości działania.

Rozdział 6 traktuje o kalibracji parametrów obrazu z uwzględnieniem problemów związanych z niwelacją wpływu oświetlenia.

Rozdział 7 prezentuje programy, które wspomagały budowanie systemu wizyjnego. Należały do nich środowiska programowania mikroprocesorów, aplikacje testowe i własne programy symulacyjne.

Rozdział 8 przedstawia porównanie trzech różnych rodzajów systemów wykrywania przeciwników na ringu sumo.

2. Robot

2.1. Zawody sumo

2.1.1. Historia i specyfikacja zawodów

Zawody sumo robotów w założeniu miały nawiązywać do tradycyjnego japońskiego sportu walki, jakim jest „sumo". Zamiast zawodników sumo na ringu stają dwa autonomiczne roboty, których zadaniem jest wypchnięcie przeciwnika poza obszar ringu. Prekursorem i twórcą zasad pierwszych zawodów sumo dla robotów autonomicznych i zdalnie sterowanych był Pan Hiroshi Nozawa przewodniczący Fuji Software Inc. Pierwsze zawody sumo robotów odbyły się w 1989 r. w Japonii i brały w nich udział 33 roboty. Pierwsze oficjalne otwarte zawody miały miejsce rok później i zgromadziły 147 zawodników. Z każdym rokiem w Japonii rosła liczba uczestników i fanów tych zawodów, co doprowadziło do szybkiego upowszechnienia tej dyscypliny. W piątej edycji dorocznych japońskich zawodów wprowadzono podział na kategorię dla reprezentantów wyższych uczelni technicznych i pozostałych konstruktorów (kategoria otwarta). W zawodach tych zostało wystawionych 309 sumo robotów zaprojektowanych przez studentów i ponad 1200 przez pozostałych konstruktorów. Na początku lat 90-tych zawody sumo robotów zaczęto rozgrywać również w USA. Po kilku latach turnieje sumo robotów stały się tak popularne, że wprowadzono nowe kategorie m.in. „mini sumo robots class" i „micro sumo robot class". Obecnie zawody sumo robotów odbywają się w wielu krajach. Największym prestiżem i popularnością cieszą się wciąż w Japonii, gdzie stanowią niemalże narodowy sport. W Japonii ogólnokrajowe zawody razem z fazą kolejnych etapów eliminacji trwają 4 miesiące. Spośród 9 regionów na japońskiej mapie sumo i tysięcy wystawianych robotów (w 2001 r. startowało ponad 4 tys. robotów) tylko 128 występuje w ścisłym finale, który odbywa się rokrocznie na stadionie „Kokugikan" w Tokio. Zawody sumo robotów są nieodłączną częścią największych festiwali robotów na całym świecie. Do największych można zaliczyć „Robo Games" odbywające się w San Francisco, na których prowadzone się zmagania robotów sumo aż w pięciu kategoriach wagowych. Duże zawody odbywają się również w takich miastach jak: Portland, Seattle czy Los Angeles. W Polsce od 2004 odbywają się Ogólnopolskie Zawody Sumo Robotów (od 2006 Międzynarodowe) w kategorii wagowej 3 kg wg specyfikacji [23] zgodnej w większości punktów z specyfikacją japońską (FUJISOFT ABC Inc) [11].

2.1.2. Zasady meczu sumo

Podstawowym wymaganiem dla robota biorącego udział w zawodach robotów sumo organizowanych w Polsce jest to, że robot musi być autonomiczny. Oznacza to, że nie może być sterowany z zewnątrz przez operatora czy komputer. Robot podejmuje decyzje na podstawie bieżących wartości otrzymywanych z różnych czujników, w jakie jest wyposażony, zgodnie z algorytmem programu. Japońska edycja tych zawodów dopuszcza dodatkowo do udziału w zawodach roboty zdalnie sterowane (RC) spełniające określone wymogi, co do sterowania [11].

Przedstawiony w punkcie 2.2 robot został zaprojektowany i skonstruowany z przeznaczeniem do walk robotów w kategorii „sumo" nazywanej również japońską od kraju pochodzenia. W tej kategorii masa robota nie może przekroczyć 3 kg, a wymiary jego podstawy przed startem muszą się mieścić w kwadracie o boku 20 cm. Wysokość robota nie jest limitowana. Zgodnie z zasadami fair-play robot nie może używać urządzeń mogących bezpośrednio i celowo uszkodzić przeciwnika. Używanie urządzeń mających na celu zakłócanie pracy układów elektronicznych robota drużyny przeciwnej jest również zabronione. O zwycięstwie w założeniu ma decydować konstrukcja mechaniczna i elektroniczna robota oraz jego algorytm sterowania.

Zawody rozgrywane są na ringu w kształcie koła o średnicy 154 cm pokrytym gumą. Koniec ringu wyznacza biała linia o szerokości 5 cm. Na ringu znajdują się dwie równoległe brązowe linie „shikiri" o wymiarach 20x2 cm, za którymi należy ustawić roboty przed rozpoczęciem walki. W promieniu 1 metra od ringu wymagane jest zachowanie wolnej przestrzeni w celu uniknięcia zakłóceń pracy robota (w innym przypadku może on wykrywać znajdujące się tam obiekty, a nie przeciwnika). Po wydaniu komendy rozpoczęcia meczu przez sędziego robot może zmienić swoje położenie dopiero po 5 sekundach. Po tym czasie robot startuje, a jego zadaniem jest wykryć przeciwnika i wypchnąć go poza obszar ringu (białą linię). Mecz składa się z trzech rund trwających po około 1 minuty każda. Jeżeli w tym czasie walka nie została rozstrzygnięta, a roboty są w zwarciu (zaklinowane) lub nie mają ze sobą kontaktu (nie wykrywają przeciwnika) to sędzia przerywa mecz. Za każdą wygraną walkę zawodnik otrzymuję 1 punkt niezależnie od tego czy robot przeciwnika został wypchnięty poza ring, czy sam go opuścił np. w wyniku niewłaściwie działających czujników. Zawodnik sumo, który jako pierwszy zdobędzie 2 punkty wygrywa mecz.

2.2. Budowa robota sumo

2.2.1. Wymagania podstawowe i dodatkowe

Każdy robot sumo oprócz wymagań wagi wymiarów i sterowania określonych w regulaminie [23] musi posiadać system czujników, których zadaniem jest:

- Wykrywanie krańca ringu (czujniki białej linii)
- Wykrywanie przeciwnika

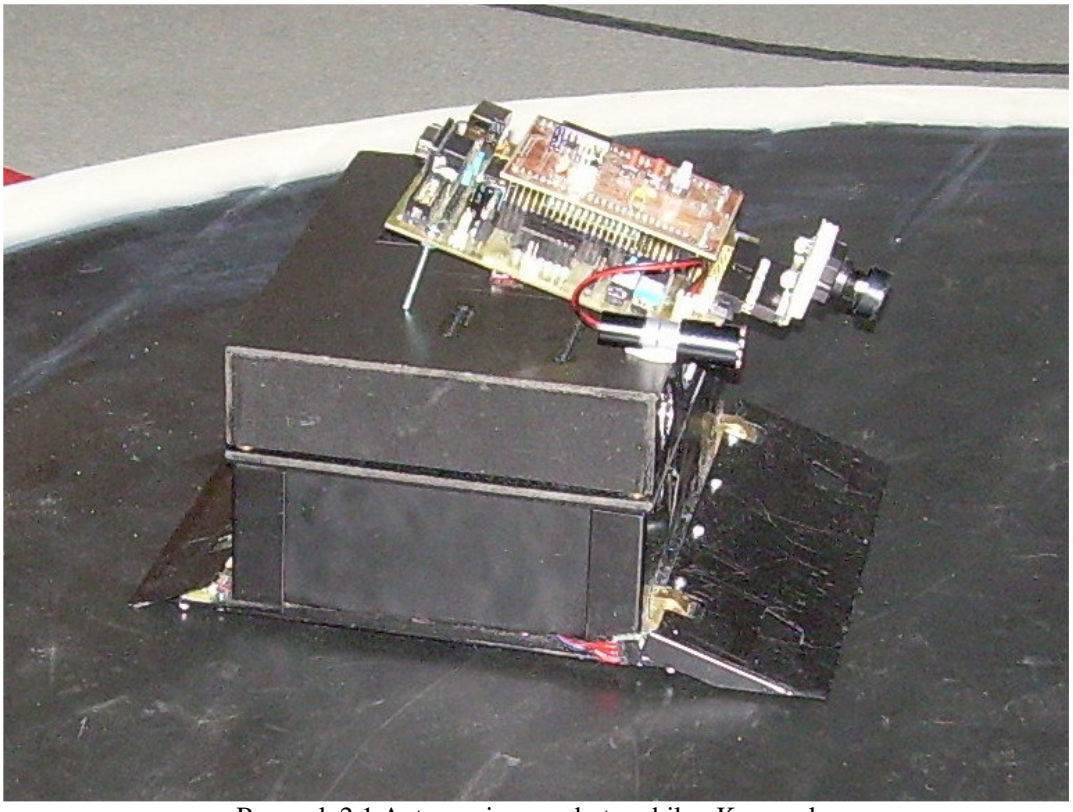
Do wykrywania białej linii wyznaczającej koniec ringu używane są najczęściej odbiciowe czujniki podczerwieni. Niezbędne jest żeby czujniki takie miały możliwość regulacji progu zadziałania tak, aby zawsze były w stanie wykryć teoretycznie białą linię niezależnie od ringu, na którym odbywają się zawody. Przeprowadzenie procesu kalibracji automatycznej lub ręcznej (potencjometr) czujników bezpośrednio przed rozpoczęciem zawodów na danym ringu daje gwarancję ich poprawnego działania. Większość konstrukcji robotów sumo ma podstawę w kształcie prostokąta stąd najczęściej umieszcza się czujniki białej linii w narożnikach. Takie ułożenie czujników pozwala w prosty sposób określić położenie robota względem linii i odpowiednio wysterować napędy. Czujniki krańca linii powinny być zamocowane na odpowiedniej wysokości i zabezpieczone przed uszkodzeniem mechanicznym.

Umiejętność określenia położenia robota przeciwnika względem własnego robota jest niezbędna żeby można było wykonać manewr ataku. Stosowane są tu różne metody począwszy od najprostszych jak czujniki zderzakowe (np. przełączniki krańcowe zamocowane do zderzaków robota) po bardziej skomplikowane jak skanery podczerwieni czy ultradźwiękowe. Porównanie tych systemów z zaprojektowanym i wykonanym systemem wizyjnym znajduje się w rozdziale 8. Optymalny system to taki, który z każdego miejsca w ringu jest w stanie bardzo szybko wykryć przeciwnika i określić jak daleko on się znajduje. W praktyce rzadko udaje się zrealizować wszystkie te cele ze względu na ograniczenia poszczególnych systemów. Dodatkowym utrudnieniem jest to, że zarówno obiekt dokonujący pomiaru jak i mierzony (przeciwnik) są w ciągłym ruchu. Istotną rzeczą jest też to, że warunki pomiarowe mogą ulegać zmianie np. w sytuacji, gdy przeciwnik używa takich samych sensorów i metody transmisji to pomiar odległości może być zakłócony.

2.2.2. Krótka charakterystyka robota

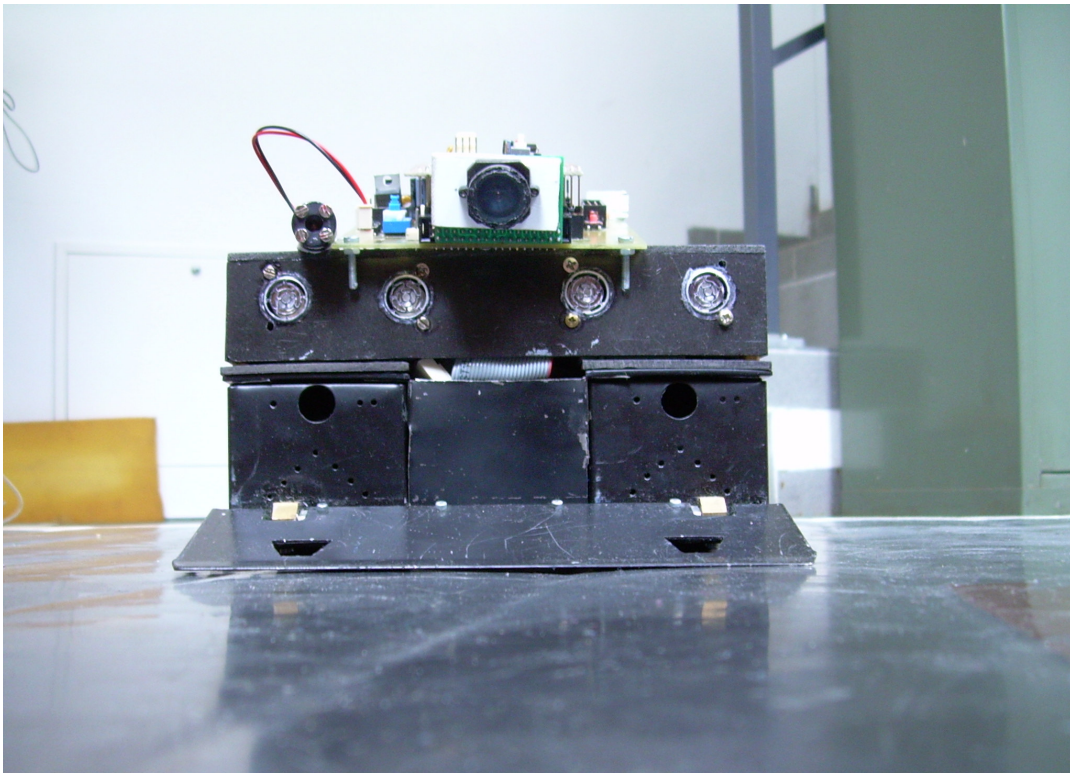
Autorzy zaprojektowali od postaw wszystkie elementy mechaniczne i elektroniczne robota mobilnego o nazwie Kruszynka (rysunek 2.1), dla którego powstał dedykowany system wizyjny opisany w niniejszej pracy.

Zbudowany robot sumo jest robotem 4 kołowym z napędem różnicowym. Zastosowano silniki prądu stałego o prędkości znamionowej 7500 obr/min z przekładniami redukcyjnymi (56:1), dzięki czemu uzyskano maksymalny moment obrotowy równy 0,7 Nm. Dodatkowo silniki wyposażone są w enkodery hallotronowe o rozdzielczości 16 impulsów na obrót wału silnika. Robot porusza się z prędkością liniową około 0,7 m/s przy napięciu zasilania 12 V. Poruszanie się z taką prędkością daje możliwość robienia uników podczas walki i powoduje, że przeciwnikowi trudno jest określić dokładne położenie robota będącego w ruchu. Układ sterowania silników został zbudowany w oparciu o dwa pojedyncze zintegrowane mostki H i mikrokontroler Atmega 16. Mikrokontroler steruje pracą dwóch mostków H, przy czym pojedynczy mostek jest połączony z parą silników nieleżących w jednej osi. Takie połączenie umożliwia realizację napędu różnicowego stosowanego powszechnie w pojazdach gąsienicowych.



Rysunek 2.1 Autonomiczny robot mobilny Kruszynka

Robot wyposażony jest w czujniki optyczne wykrywające białą linię oraz czujniki ultradźwiękowe do lokalizacji położenia przeciwnika. Informacje z enkoderów hallotronowych są wykorzystywane przez mikrokontroler do ustalania kątów obrotu robota przy manewrach w miejscu. Pozwalają również wykryć kontakt z przeciwnikiem. Robot wyposażony jest w samonastawne kliny z przodu i z tyłu. Takie rozwiązanie mechaniczne pozwala na atak przeciwnika będącego w dowolnej pozycji wykonując manewr obrotu maksymalnie o 90°. Rozwiązanie to sprawdziło się również w praktyce podczas walk robota. Pewną niedoskonałością tej konstrukcji okazały się jedynie opony, które nie zapewniły optymalnej przyczepności do podłoża.



Rysunek 2.2 Autonomiczny robot mobilny Kruszynka – widok z przodu

3. Płyty prototypowe

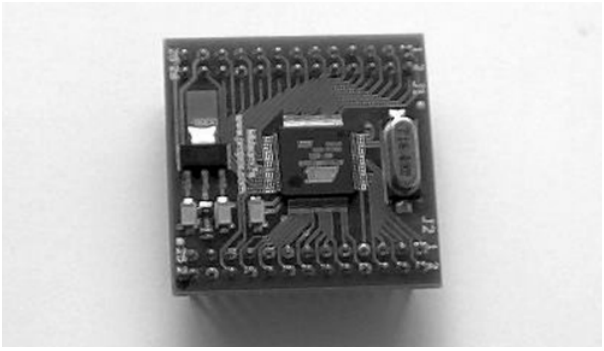
3.1. Mikrokontroler

W prototypowym systemie przetwarzania zastosowano mikrokontroler AT91SAM7S64 firmy Atmel. Procesor ten został wyposażony w 32-bitowy rdzeń ARM7TDMI. Zaletą procesorów z rdzeniami typu ARM jest to, że mogą pracować z większą częstotliwością niż popularne mikrokontrolery z rodziny 8051, AVR czy PIC. Dla użytego mikrokontrolera maksymalna częstotliwość pracy wynosi 55 MHz. Procesory ARM mają architekturę typu RISC (Reduced Instruction Set Computers). Dzięki zredukowanej liczbie instrukcji i zwiększonej liczbie rejestrów ogólnego przeznaczenia uzyskano prostsze kody rozkazów a tym samym szybsze ich wykonywanie. W odróżnieniu od mikrokontrolerów typu CISC (Complex Instruction Set Computers), które posiadają wiele instrukcji operujących bezpośrednio na pamięci, mikrokontrolery z rdzeniem ARM posiadają tylko 2 takie rozkazy – Load i Store. Pozostałe instrukcję wykonywane są na rejestrach. Dodatkowo dzięki potokowemu przetwarzani instrukcji duża część z nich jest wykonywana w jednym cyklu zegarowym. Potokowe przetwarzanie instrukcji polega na tym, że jednostka centralna w tym samym czasie, gdy wykonuje bieżącą instrukcję dekoduje już następną i dodatkowo pobiera jeszcze kolejny rozkaz (znajdujący się za instrukcją dekodowaną). Przydatną cechą niespotykaną we wcześniejszych architekturach jest możliwość wykonywania większości instrukcji warunkowo. Odbywa się to przez ustawianie flag (ustawienia tego dokonują instrukcje, które wcześniej przetwarzały dane). Szczegółowe informacje na temat architektury rdzenia i samego mikrokontrolera znajdują się w dokumentacji [1] i [2]. Do ważniejszych parametrów mikrokontrolera AT91SAM7S64 należą:

- Pamięci typu: FLASH 64kB, RAM 16kB
- 2 interfejsy USART (+ dodatkowy kanał UART do debuggowania)
- Interfejsy: USB 2.0, SPI, I2C, JTAG, 1-Wire
- 4-kanałowy 16-bitowy kontroler PWM
- 3 timery (input caputre, output compare)
- 8-kanałowy 10-bitowy przetwornik A/C
- Licznik czasu rzeczywistego (RTT)
- Do 32 linii I/O z tolerancją 5 V
- Napięcie zasilania 3,3 V i tryb obniżonego poboru mocy

3.2. Minimoduł MMsam7s

W celu możliwości łatwej wymiany mikrokontrolera w przypadku jego uszkodzenia, lub potrzeby użycia jednostki o większej pamięci zdecydowano się na zakup mikrokontrolera zamontowanego na mini płytce o standardowym rastrze wyprowadzeń 2,54 mm (rysunek 3.1). Dzięki temu układ AT91SAM7s64 stał się oddzielnym minimodułem, który można łatwo dołączyć i odłączyć od płyty głównej. Dodatkowym atutem minimodułu jest to, że posiada wbudowany stabilizator 3,3 V (400 mA) potrzebny do stabilizacji napięcia mikrokontrolera oraz rezonator kwarcowy 18,432 MHz. Model minimodułu zakupiony przez nas (MMsam7s64-2) posiada dodatkowo zamontowaną pamięć typu FLASH o pojemności 4 MB obsługiwaną przez interfejs SPI. Schemat minimodułu znajduję się w instrukcji [25].



Rysunek 3.1 Minimoduł MMsam7s

3.3. Płyty główne przetwarzania obrazu

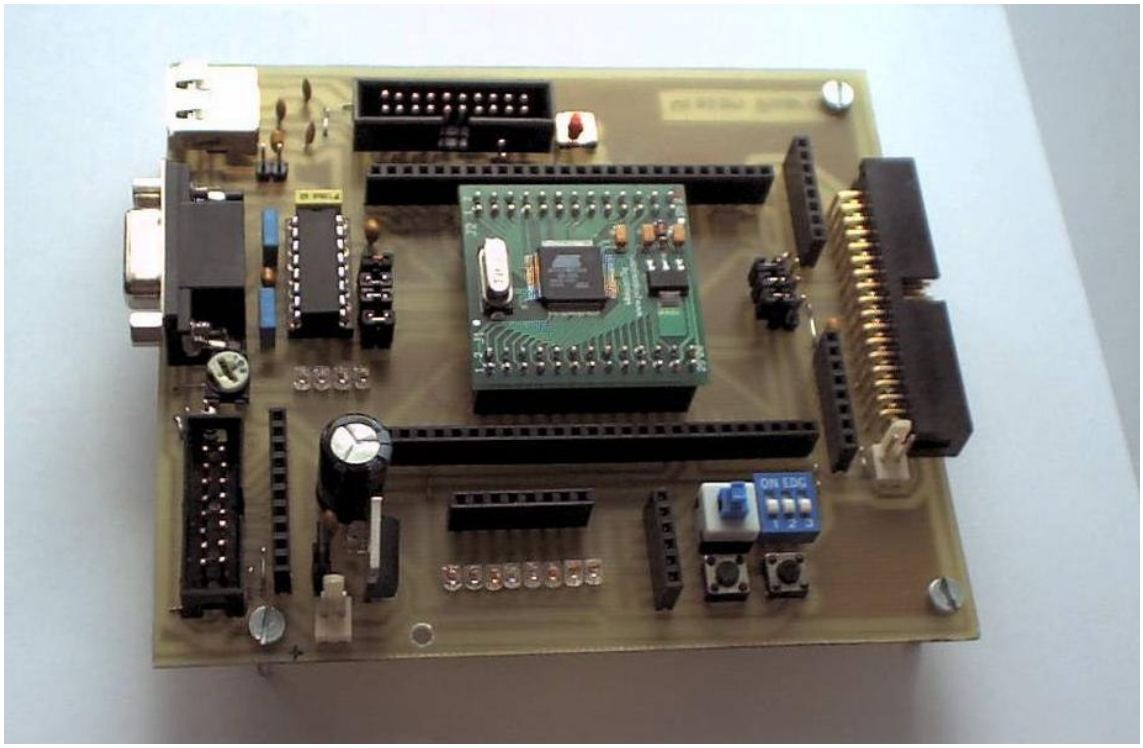
Nieodłączną częścią każdego systemu przetwarzania jest płyta główna, która zawiera niezbędne moduły służące do realizacji założonych celów. W przypadku autonomicznych systemów wizyjnych płyta ta ma największe znaczenie, gdyż stanowi jednostkę przetwarzania i identyfikacji obrazu. W odróżnieniu od systemów wizyjnych nie autonomicznych, gdzie płyta taka stanowi jedynie blok pośredni między kamerą a urządzeniem przetwarzania (komputerem), ważną rzeczą jest dobór mikrokontrolera. Układ taki musi charakteryzować się takimi cechami, aby mógł wykonywać określone przez projektanta zadania przy spełnieniu kryteriów bezpieczeństwa i szybkości (szczególnie w systemach czasu rzeczywistego). Zaprojektowane płyty główne są płytami dedykowanymi do przetwarzania i identyfikacji obrazów. Ze względu jednak na ich modułową budowę mogą być wykorzystywane w innych projektach (po zaprojektowaniu i dołączeniu potrzebnych modułów). Jako realizacja sprzętowa tej części pracy zostały zaprojektowane i wykonane

dwie płyty przetwarzania obrazu oraz płyta z buforem pamięci i płyta kontrolna. Druga płyta przetwarzania obrazu powstała w wyniku dodania kilku przydatnych komponentów do projektu pierwszej. Dziedziczy ona jednak wszystkie funkcje, które posiadała pierwsza płyta i posiada ten sam układ wyprowadzeń.

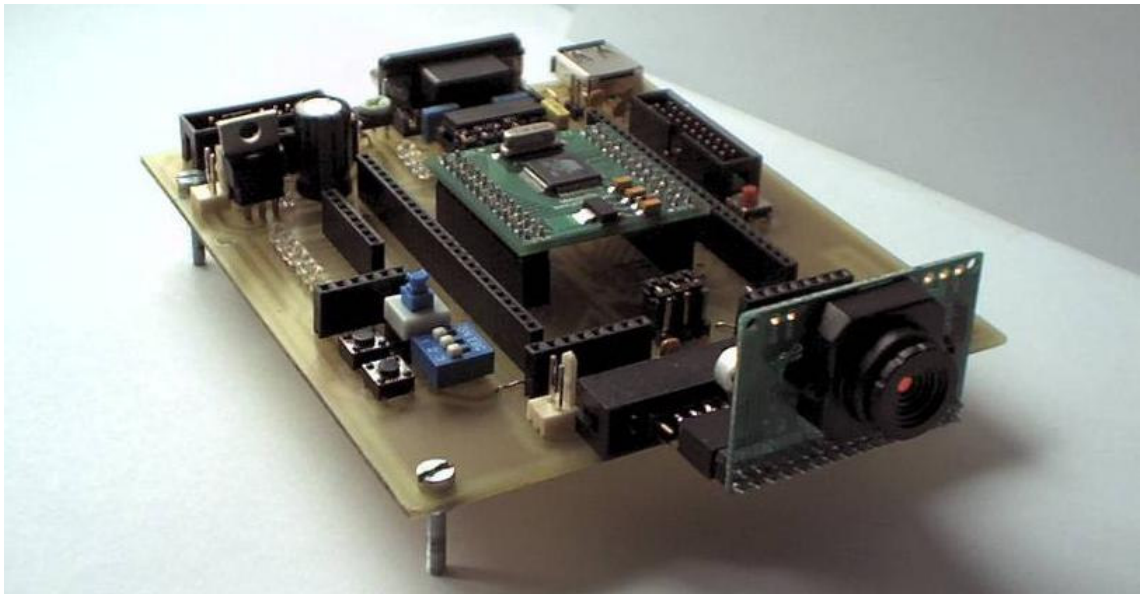
W celu szybkiego sprawdzenia kamery i uzyskania zdjęć potrzebnych do pracy nad algorytmem programu przetwarzania obrazu wykonano układ elektroniczny na płytce wielostykowej w oparciu o projekt [18]. Wykonane na tym etapie testy regulacji były pomocne przy tworzeniu programu docelowego na mikrokontrolerze AT91SAM7S64. W wykonanym testowym układzie elektronicznym użyto mikrokontrolera Atmega32 i programu (z pewnymi zmianami) autorstwa osób wymienionych w pracy [18]. Układ ten pozwalał w bardzo niewielkim stopniu sprawdzić możliwości kamery, a uruchomiony program umożliwiał jedynie bardzo wolne skanowanie obrazu w pionie i tylko w jednym trybie (16-bitowy YUV, gdzie UV nie jest odczytywane).

3.3.1. Płyta główna przetwarzania obrazu – projekt1 (RVB1)

Pierwszy prototyp płyty głównej przetwarzania obrazu przedstawiony na rysunku 3.2 został wykonany na laminacie jednostronnym. Odpowiednie ułożenie elementów i ścieżek pozwoliło zminimalizować liczbę zworek na płytce. Płyta została wyposażona w złącza interfejsów USB, RS232 i JTAG. Złącze kamery pozwala zamontować kamerę bezpośrednio do płytki. Równolegle do minimodułu MMsam7s poprowadzono dwie szyny gniazd 16-pinowych połączonych bezpośrednio z minimodułem. Rozwiązanie takie pozwala na dowolne konfigurowanie wolnych wyprowadzeń mikrokontrolera poprzez podłączenie ich do wybranych elementów wejścia/wyjścia znajdujących się na płytce (przyciski, przełączniki, diody kontrolne, wyświetlacz LCD). W gniazdach tych montowana jest płytka kontrolna, która łączy dwa kanały wyjściowe (Y, UV) i trzy piny synchronizacji kamery z odpowiednimi pinami mikrokontrolera (tab. 3.1). Płyta ma możliwość zasilania z komputera poprzez złącze USB (ograniczony pobór prądu do 100 mA) oraz z innego źródła napięcia stabilizowanego o wartości 5 V. Na płytce wyprowadzone zostało również analogowe wyjście kamery (ANALOG_OUT) dające możliwość bezpośredniego połączenia kamery z telewizorem lub kartą telewizyjną komputera.



Rysunek 3.2 Płyta główna przetwarzania obrazu RVB1 z płytką kontrolną



Rysunek 3.3 Płyta główna przetwarzania obrazu RVB1 z dołączoną kamerą

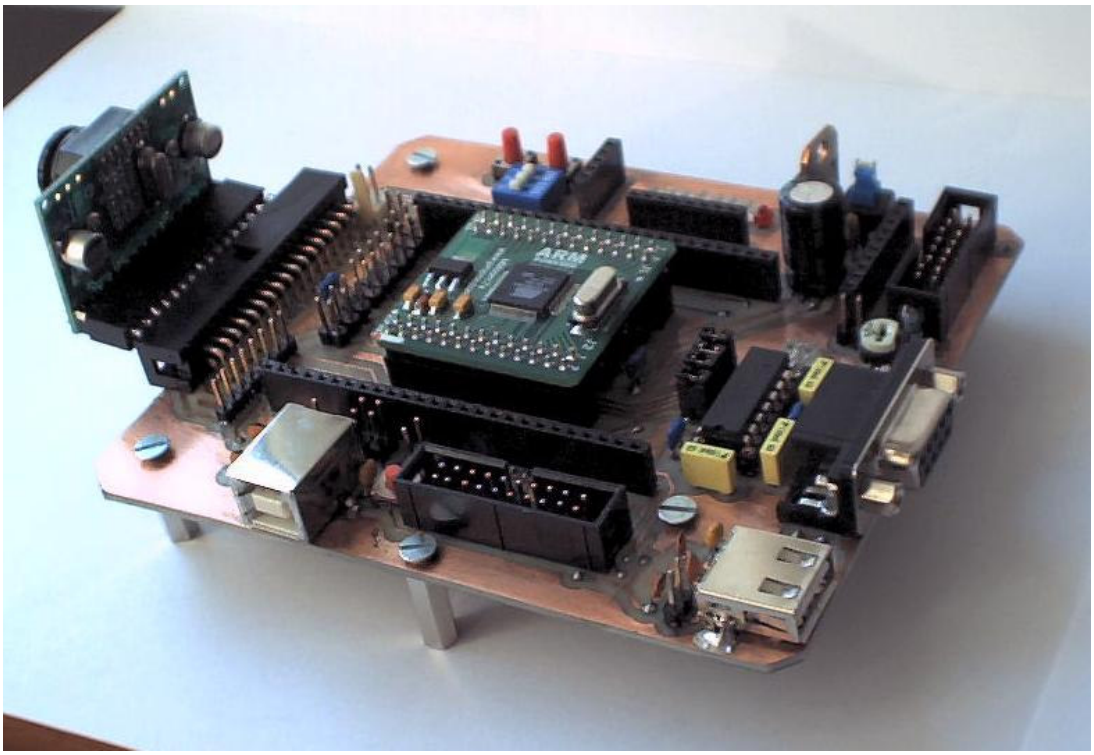
Porty mikrokontrolera	Piny zewnętrzne połączone z mikrokontrolerem	Pełniona funkcja w układzie
PA0	PCLK	Synchronizacja kamery
PA1	HREF	
PA2	Q1(VSYN)	
PA3	SDA	Interfejs SCCB
PA4	SCL	
PA5	RXD	Interfejs RS232
PA6	TXD	
PA7	RTS	
PA8	CTS	
PA9	LED1	Elementy zadające i kontrolujące pracę kamery (opcjonalne)
PA10	LED2	
PA11	LED3	
PA12	Laser	
PA13	Przełącznik P1	
PA14	Przycisk S1	
PA15	Przycisk S2	
PA16	UV0	8 bitowy port UV kamery
PA17	UV1	
PA18	UV2	
PA19	UV3	
PA20	UV4	
PA21	UV5	
PA22	UV6	
PA23	UV7	
PA24	Y0	8 bitowy port Y kamery
PA25	Y1	
PA26	Y2	
PA27	Y3	
PA28	Y4	
PA29	Y5	
PA30	Y6	
PA31	Y7	

Tabela 3.1 Opis połączeń i przeznaczenia pinów mikrokontrolera na płycie głównej przetwarzania obrazu.

3.3.2. Płyta główna przetwarzania obrazu – projekt2 (RVB2)

Drugi prototyp płyty głównej przetwarzania obrazu został tak zaprojektowany, żeby nie wymagał instalowania dodatkowego modułu płytki kontrolnej (rysunek 3.4). Na tym etapie prac autorzy zdecydowali o funkcjach poszczególnych portów i pinów mikrokontrolera. Porty UV i Y kamery zostały podłączone bezpośrednio na płycie bazowej. Uwzględniono jednak możliwość odpięcia portu UV kamery w przypadku, gdy użytkownik nie wykorzystuję tej składowej (tryb RGB 8-bitowy, tylko składowa Y obrazu). Daje to możliwość podłączenia dodatkowych elementów wejścia/wyjścia np.: wyświetlacza LCD (niepołączonego bezpośrednio) i dodatkowych czujników dla robota. Na płycie został zamontowany przerzutnik JK wydłużający czas wysokiego stanu na VSYN do momentu nadejścia HREF. Pełny opis procesu synchronizacji mikrokontrolera z kamerą znajduję się w podpunkcie 5.2.1 Dodano drugi port USB typu A z ograniczeniem prądowym (bezpiecznik polimerowy). Zakupiono dwie wersje minimodułów różniące się tym, że nowsza wersja

posiadała wyprowadzone dodatkowe piny mikrokontrolera oraz pewne zmiany układowe pozwalające na konfigurację minimodułu. Aby w pełni wykorzystać zalety nowszego modułu w projekcie płyty głównej RB02 podłączono wyprowadzone piny TST i ERASE do zworek z goldpinów. Rozmieszczenie elementów na płycie głównej oraz opis konfiguracji zworek znajduję się w dodatku A.

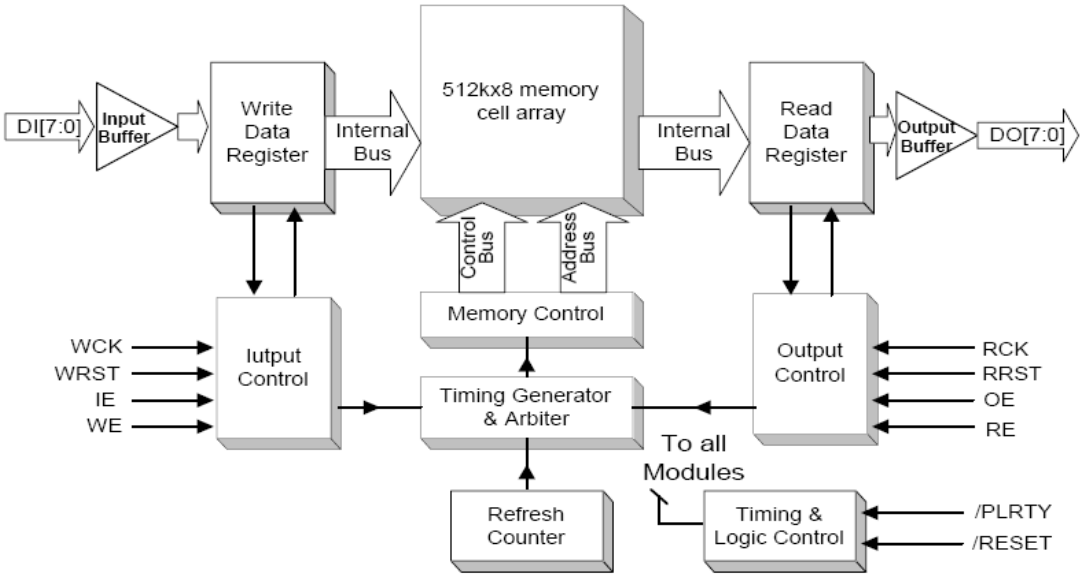


Rysunek 3.4 Płyta główna przetwarzania obrazu RVB2

3.3.3. Płyta pamięci obrazu FIFO (RMB1)

W celu przechwytywania obrazu z kamery z maksymalną szybkością zaprojektowano układ wg schematu (dodatek A). Układ składa się z dwóch buforów synchronicznej pamięci FIFO typu AL440C-20 połączonych równolegle. Jeden bufor obsługuje jeden kanał kamery, przy czym sterowanie obu odbywa się synchronicznie. Pamięć AL440C-2 ma pojemność 512 KB i organizację 8-bitową. Pozwala na pracę z maksymalną szybkością 50 MHz, która jest również graniczną częstotliwością pracy kamery. Schemat blokowy pamięci przedstawia rysunek 3.5. Jest ona zorganizowana tak, że umożliwia niezależną kontrolę wejścia i wyjścia poprzez piny kontrolne. WCK i RCK są wejściami zegarowymi odpowiednio dla danych zapisywanych i odczytywanych z pamięci. Częstotliwość zapisu może być inna niż częstotliwość odczytu (w praktyce zależy nam na szybszym zapisie i wolniejszym odczycie). Aby zapewnić warunki poprawnej pracy układu należy oba wejścia zegarowe taktować ze stałą częstotliwością równą, co najmniej 1MHz. Wskaźnik do bieżącej pozycji zapisu w

pamięci jest inkrementowany w takt sygnału zegarowego WCK. Kiedy chcemy zresetować ten wskaźnik (ustawić na 0) np. przed rozpoczęciem nowego cyku zapisu obrazu z kamery należy ustawić odpowiedni stan na wejściu WRST (stan zależny od przyjętej logiki sterującej). Analogicznie przed rozpoczęciem odczytu danych z pamięci należy zresetować wskaźnik odczytu poprzez pin RRST.



Rysunek 3.5 Schemat blokowy pamięci AL440C z instrukcji Averlogic [3]

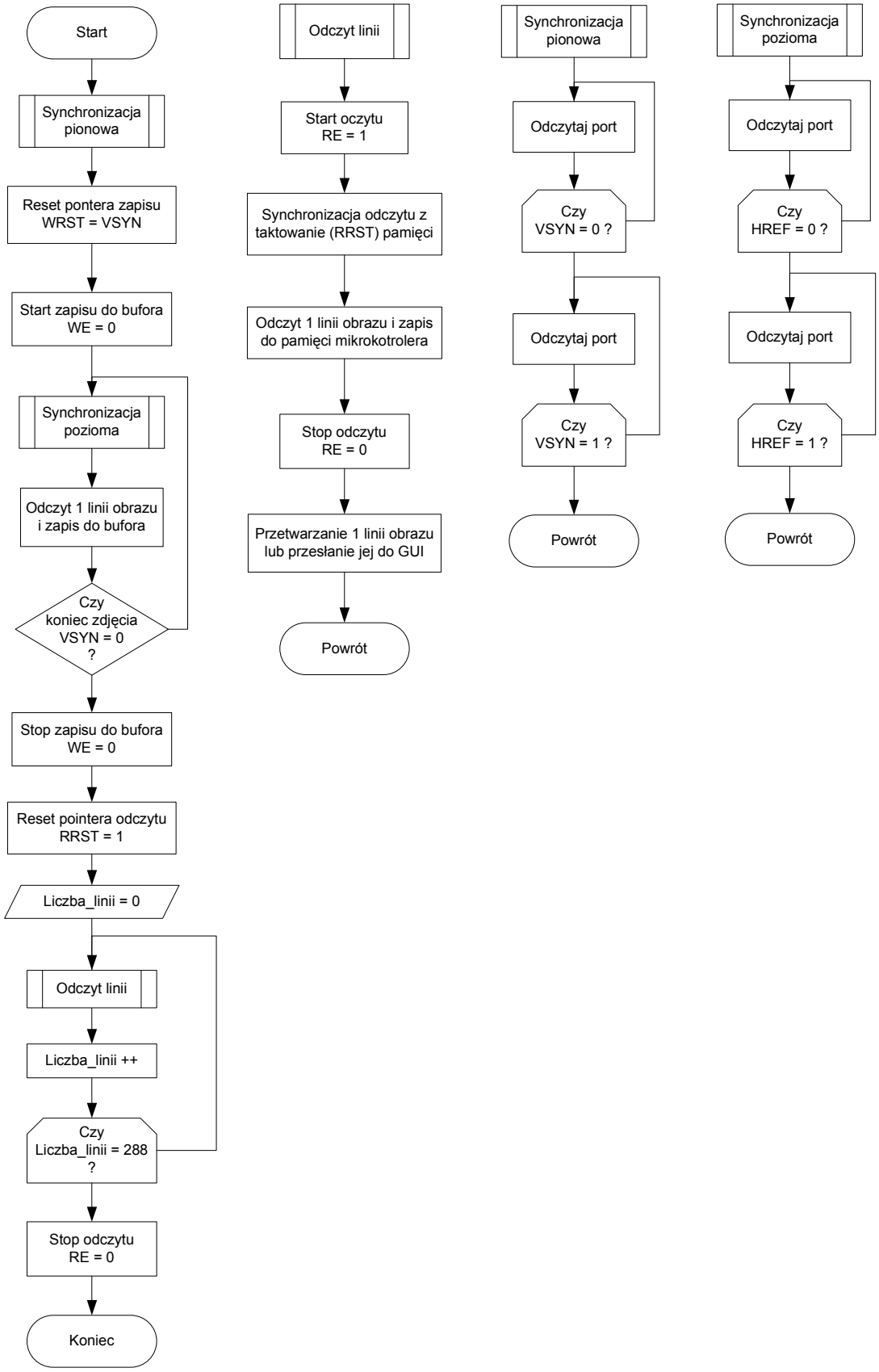
Wejścia WE i RE sterują odpowiednio zapisem i odczytem do pamięci. Jeżeli są nieaktywne to operacje zapisu i odczytu są przerywane a wartości wskaźników pamięci nie ulegają zmianie. Po uaktywnieniu tych wejść odczyt i zapis zostają wznowione od miejsc, w których zostały zatrzymane. Wejścia IE i OE umożliwiają przerywanie zapisu i odczytu, ale nie zatrzymują inkrementacji wskaźników pamięci. W programie testowym nie były wykorzystywane i ustawione na stałe w stan aktywny (Input Enable, Output Enable). Stan ustawiony na wejściu PLRTY decyduje o logice sygnałów sterujących. Jeżeli np. PLRTY = L to stanem aktywnym wszystkich wejść sterujących jest stan wysoki. Tabela 3.2 przedstawia ustawienia wejść sterujących pamięci FIFO dla operacji zapisu i odczytu a tabela 3.3 połączenia wyprowadzeń modułów: pamięci, kamery i mikrokontrolera.

Zapis obrazu do bufora (PLRTY=GND), stan aktywny H				
WRST	**WE**	**IE**	**WCK**	**Opis**
H	-	-	PCLK	Reset wskaźnika zapisu
L	H	H	PCLK	Zapis danych
L	L	-	PCLK	Zatrzymanie zapisu
Odczyt obrazu z bufora przez AT91SAM7S64				
RRST	**RE**	**OE**	**RCK**	**Opis**
H	H	H	-	Reset wskaźnika odczytu
L	H	H	Taktowanie z uP	Odczyt danych
L	L	H	-	Zatrzymanie odczytu

Tabela 3.2 Sterowanie pracą bufora FIFO

AL440C-20	**KAMERA (OV6620)**	**AT91SAM7S64**
U1) DI [1:7] Wejścia (8 bitów)	Kanał Y [1:7]	
U2) DI [1:7] Wejścia (8 bitów)	Kanał UV [1:7]	
U1) DO [1:7] Wyjścia (8 bitów) Y		PA[24:30]
U2) DO[1:7] Wyjścia (8 bitów) UV		PA[16:22]
		PA[31] = GND
		PA[23] = GND
U1,U2) WCK [13]	PCLK	
PLRTY = GND (stan aktywny H)	-	-
U1,U2) WRST[14]	VSYN	PA15 (Wejście)
U1,U2) WE[10]	HREF	PA2 (Wyjście)
*U1,U2) IE[11]		PA1 (Wyjście) lub VCC
U1,U2) RCK[32]		PA0 (Taktowanie z uP)
U1,U2) RRST[31]		PA9 (Wyjście)
U1,U2) RE[35]		PA10 (Wyjście)
*U1,U2) OE[34]		PA11 (Wyjście) lub VCC

Tabela 3.3 Połączenia wyprowadzeń modułów systemu



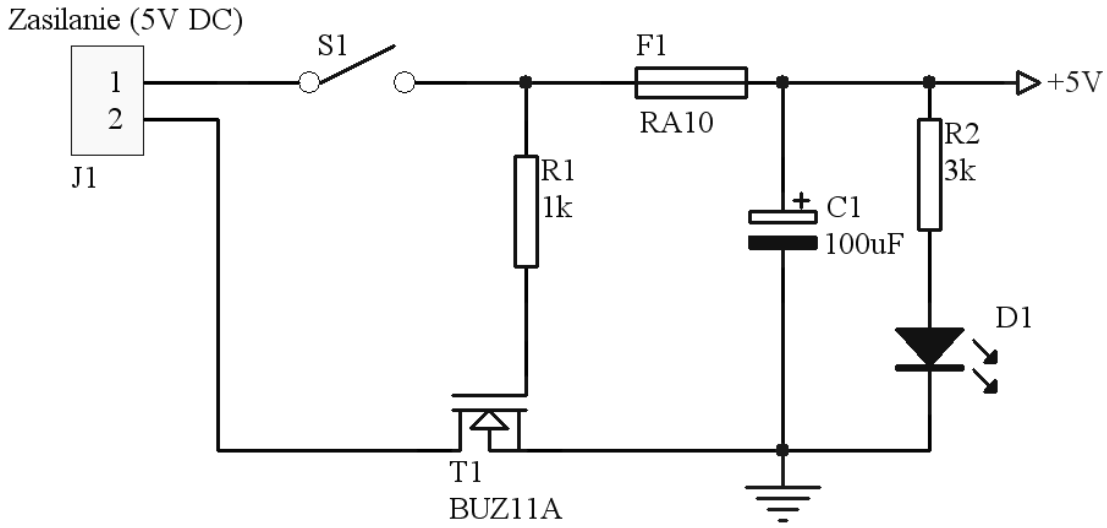
Rysunek 3.6 Schemat algorytmu przesyłu obrazu z wykorzystaniem pamięci FIFO

Rysunek 3.6 przedstawia algorytm programu, który został napisany w celu przesyłu obrazu z wykorzystaniem pośredniczącego bufora FIFO. Ostatecznie jednak algorytm ten nie został zaimplementowany z powodu uszkodzenia układu AL440C podczas testów. Pracę zakończono na etapie poprawnego zapisu danych do pamięci FIFO. Podczas uruchamiania układu natknięto się na wiele nieścisłości i niezgodności w instrukcji układu AL440C [3]. Płytka drukowana została wykonana w oparciu o schemat zamieszczony w tej instrukcji, na którym pin TST jest nie podłączony. Po uruchomieniu układu okazało się, że pobiera on prąd o około 40 mA większy od prądu znamionowego przy danej częstotliwości taktowania wg instrukcji. Po dokładnym sprawdzeniu układu i wyeliminowaniu jako przyczyn błędów nieprawidłowego wykonania obwodu drukowanego porównano opisy układów pokrewnych tej samej firmy. Jak się okazało pin TST musi być zawsze podłączony do masy podczas normalnej pracy (jest wykorzystywany do przeprowadzania pewnych testów na buforze i w typowych układach pracy jest niewykorzystywany). Po zwarciu tej końcówki do masy pobór prądu spadł do wartości w granicach podawanych przez producenta. Mimo, że układ ten jest układem nowym (status Sample) to producent nie umieścił w jego instrukcji bardzo ważnych informacji o występowaniu pewnych stanów zabronionych (zabronione kombinacje stanów wejść sterujących), w których układ nie działa poprawnie. Informację te znaleziono w instrukcji innego układu z tej samej rodziny (AL422B revision V1.5). Brak tych informacji powodował problemy w początkowej fazie uruchamia układu. Instrukcja od zastosowanej pamięci nie określała też jednoznacznie czy układ może pracować przy poziomie napięć wyższym od 3,3 V. Informacje zawarte w instrukcjach układów z tego samego segmentu mówiły o przystosowaniu układów z tej rodziny do pracy przy napięciu 5 V. Przeprowadzone testy potwierdziły to. Po zakończeniu i przetestowaniu części programu odpowiedzialnego za zapis do pamięci przełączano układy wejścia i wyjścia. Omyłkowo wykonano błędne połączenia, które to najprawdopodobniej doprowadziły do uszkodzenia układu. Okazało się, że porty układu są bardzo wrażliwe na błędne połączenia czy chwilowe ich zwarcia. Jest to niewątpliwa wada tego układu porównując np. do mikrokontrolera AT91SAM7S, w którym porty zostały zabezpieczone przed takimi skutkami.

Pomimo wymienionych wad i nieścisłości oraz faktu, że układ ten pobiera dużo więcej energii (ponad 2-krotnie większy prąd pobierany niż cały układ elektroniczny z kamerą przy tym samy napięciu zasilania) jego zastosowanie jest celowe. Pozwala on w pełni wykorzystać szybkość robienia zdjęć przez kamerę, co jest ważne przy systemach wizyjnych, w których następuje szybkie przemieszczanie obiektów i (lub) kamery.

3.4. Układ kontroli i zabezpieczenia

Układ kontroli i zabezpieczenia przedstawiony na rysunku 3.7 jest blokiem odpowiedzialnym za zapewnienie właściwej polaryzacji napięcia zasilania. Kolejną funkcją tego bloku jest ochrona układów elektronicznych zasilanych pośrednio przez niego, przed skutkami zwarć i przeciążeń. Dodatkowo zastosowano filtrację napięcia zasilania poprzez kondensator C1 oraz kontrolę stanu zasilania stosując szeregowe połączenie diody D1 i rezystora R2. Płyta główna przetwarzania obrazu, w skład, której wchodzi przedstawiony układ kontroli i zabezpieczeń, jest zasilana z płyty sterowania robota napięciem stabilizowanym 5 V DC (złącze J1).



Rysunek 3.7 Układ kontroli i zabezpieczenia

3.4.1. Ochrona przed odwrotną polaryzacją napięcia zasilania

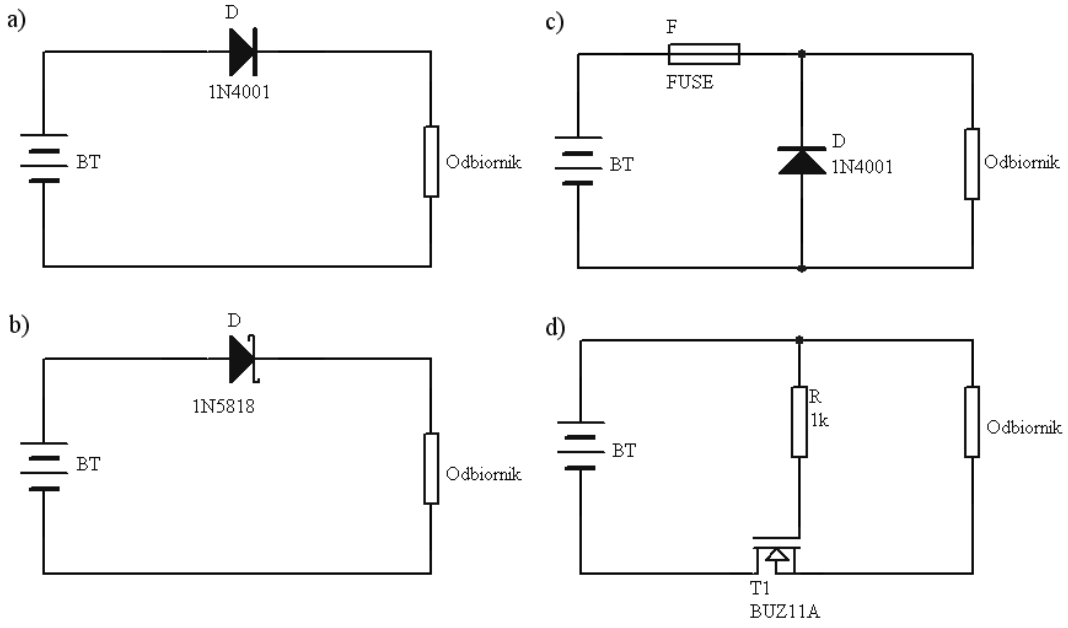
Spośród kilku znanych metod zabezpieczenia układu elektronicznego przed odwrotną polaryzacją napięcia przedstawionych na rysunku 3.8 układ z tranzystorem MOSFET z kanałem typ P został wybrany i zastosowany. Układ zabezpieczenia z diodą prostowniczą szeregowo włączoną z odbiornikiem (rysunek 3.8a) został odrzucony ze względu na wnoszony spadek napięcia na diodzie podczas przewodzenia wynoszący około 0,7 V (zależnie od typu i mocy diody). Zastosowanie diody szybkiej (Schottky'ego) pozwala na zmniejszenie tego spadku do około 0,35 V, co nadal było w naszym przypadku niewystarczające. Ponieważ robot jest zasilany z baterii akumulatorów zależało nam na minimalizacji strat energii.

Układem niewprowadzającym spadku napięcia przy przewodzeniu jest układ diody równolegle włączonej do odbiornika (rysunek 3.8c). Układ taki wymaga bezpiecznika o

wartości niższej od maksymalnego prądu przewodzenia diody (Io), a zarazem wyższej od maksymalnego prądu obciążenia w normalnych warunkach pracy. W praktyce dobierana jest dioda o Io kilkakrotnie większym od maksymalnego prądu obciążenia i szybki bezpiecznik.

Przy odwrotnej polaryzacji napięcia zasilania przez diodę popłynie prąd zwarciowy ograniczony jedynie przez rezystancję diody i wydajność źródła (rezystancję wewnętrzną) powodujący przepalenie bezpiecznika. Widoczną wadą tego układu z punktu widzenia zastosowań w układach zasilanych bateryjnie jest przepływ w pewnym odcinku czasu dużego prądu powodującego częściowe rozładowanie baterii akumulatorów.

Powyższych wad jest pozbawiony układ z tranzystorem MOSFET, który nie powoduje rozładowania ogniwa zasilającego przy odwrotnej polaryzacji napięcia zasilania, a spadek napięcia pomiędzy drenem a źródłem w stanie przewodzenia wynosi zaledwie kilkanaście mV (zależnie od typu tranzystora). W projekcie zastosowano tranzystor MOSFET typu BUZ11A charakteryzujący się bardzo małą rezystancją dren-źródło w stanie całkowitego otwarcia (0,045 Ω) a co za tym idzie małym spadkiem napięcia na złączu Uds wynoszącym 15 mV.

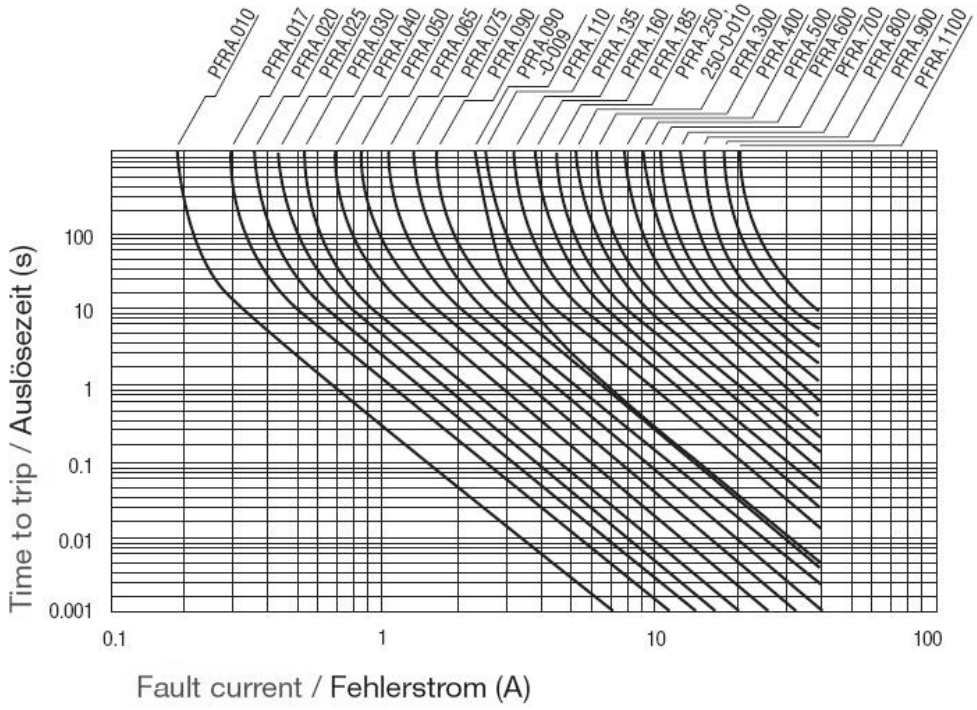


Rysunek 3.8 Schematy układów zabezpieczających przed odwrotną polaryzacją

3.4.2. Ochrona przed przeciążeniami i zwarciami w układzie odbiornika

W układzie z rysunku 3.7 zastosowano również ochronę przed zwarciami i przeciążeniami w obwodzie odbiornika w postaci bezpiecznika. Jest to bezpiecznik polimerowy (resetowalny) o charakterystyce przedstawionej na rysunku 3.9 [28]. W projekcie

został zastosowany bezpiecznik PFRA.010 firmy Schurter. W przypadku wystąpienia przeciążenia o wartości 1 A bezpiecznik wyłączy układ w czasie 0,4 s, a przy zwarciu w czasie od kilku do kilkunastu ms (zależnie od rezystancji źródła). Zasada działania tego bezpiecznika jest podobna do zasady działania pozytora (ceramicznego rezystora PTC) i opiera się na zależności rezystancji struktury od jej temperatury, a ta od płynącego przez bezpiecznik prądu. Bezpieczniki polimerowe odznaczają się jednak o rząd mniejszą rezystancją w normalnych warunkach pracy (przewodzenia) i o rząd mniejszymi czasami wyłączeń. Zastosowany bezpiecznik ma w normalnych warunkach pracy rezystancję 3,5 Ω co przy napięciu zasilania płytki przetwarzania obrazu równym 5 V ograniczy prąd zwarciowy do wartości 1,4 A. Jedynym niewielkim minusem tego rozwiązania jest spadek napięcia występujący na bezpieczniku. W zaprojektowanym układzie w warunkach normalnej pracy spadek ten wynosi 0,3 V (przy prądzie obciążenia 85 mA i napięciu zasilania 5 V).



Rysunek 3.9 Charakterystyka bezpieczników polimerowych [28]

3.4.3. Działanie układu kontroli i zabezpieczenia

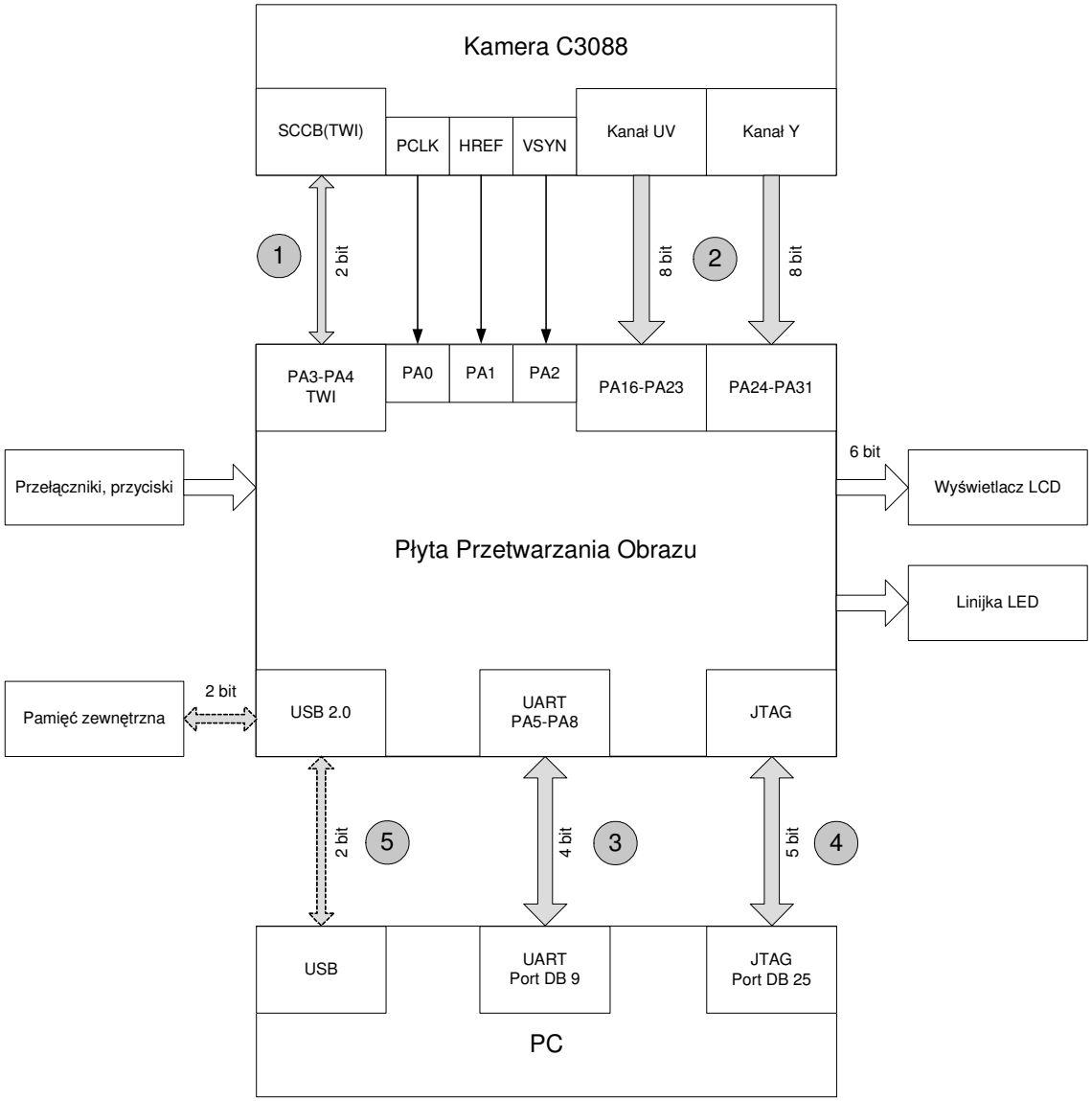
W normalnych warunkach pracy, gdy nie występuję przeciążenie lub zwarcie a biegunowość napięcia zasilania jest zachowana tranzystor T1 przewodzi pod warunkiem, że $U_{GS} > U_{GSth}$. Dla zastosowanego typu tranzystora napięcie progowe bramka-źródło powodujące przejście w stan przewodzenia wynosi 4 V i jest niższe od napięcia zasilania wynoszącego 5 V, zatem założenia są spełnione. W przypadku odwrotnej polaryzacji napięcia

zasilania potencjał bramki (G) w stosunku do potencjału źródła (S) wyniesie -5 V i tranzystor zostanie natychmiast wyłączony powodując przerwę w obwodzie masy między ogniwem zasilania a pozostałą częścią układu. Dla tranzystorów MOSFET czasy reakcji na zmiany napięcia U_{GS} są rzędu nanosekund. Do momentu przywrócenia prawidłowej polaryzacji napięcia zasilania tranzystor będzie w stanie odcięcia. W stanie odcięcia rezystancja R_{DS} dla tranzystorów MOSFET jest bardzo duża (dziesiątki MΩ), co zapobiegnie w naszym przypadku rozładowaniu się akumulatora przy odwrotnym połączeniu przewodów zasilania.

O poprawnym działaniu układu z punktu widzenia ciągłości obwodu i prawidłowej polaryzacji napięcia od strony zasilania informuję nas dioda kontrolna D1. W przypadku wystąpienia jednego z wyżej wymienionych stanów awaryjnych dioda będzie zgaszona do czasu jego ustąpienia.

3.5. Interfejsy komunikacyjne

W projekcie wykorzystano kilka interfejsów komunikacyjnych, które okazały się niezbędne do programowania, kalibracji i przesyłu danych pomiędzy poszczególnymi blokami układu. Rysunek 3.10 przedstawia schemat blokowy z zaznaczonymi interfejsami.



9. Interfejs szeregowy synchroniczny SCCB(TWI)
10. Interfejs równoległy
11. Interfejs szeregowy asynchroniczny RS-232
4. Interfejs JTAG
5. Interfejs USB

Rysunek 3.10 Schemat blokowy systemu przetwarzania obrazu z zaznaczeniem interfejsów

3.5.1. Interfejs SCCB

Zastosowana w systemie kamera posiada interfejs SCCB (Serial Camera Control), poprzez który dokonuję się nastawy parametrów kamery. Interfejs ten umożliwia również odczyt wszystkich rejestrów kamery. Istnieje możliwość regulacji wybranych parametrów kamery poprzez ustawienie określonych w instrukcji [20] portów, ale regulacja ta jest bardzo ograniczona. Dodatkowo wartości z portów odczytywane są tylko w momencie startu systemu (po włączeniu zasilania) lub po jego resecie. Z powodu powyższych ograniczeń ta prosta metoda ustawień parametrów kamery została odrzucona i napisany został program pozwalający na komunikację kamery z mikroprocesorem poprzez interfejs SCCB. Firmy zajmujące się dystrybucją tego modelu kamery często nadużywają w swoich ofertach reklamowych nazwy I2C - standardu szeregowej transmisji synchronicznej opracowanego i opatentowanego przez firmę Philips [21]. Mimo, że oba standardy opierają się na tych samych rozwiązaniach technicznych i metodzie transmisji danych (szeregowa 8-bitowa) jednak występują pewne różnice w budowie tych systemów. Aby można było w dalszej części tego podpunktu odwoływać się do magistrali I2C zostaną przedstawione cechy wspólne i różnice interfejsów I2C, TWI i SCCB (tab. 3.4).

	Firma	Philips	Atmel	OmniVision
1	Nazwa Interfejsu	**I2C** (Inter Integrated Circuit)	**TWI** (Two-Wire Interface)	**SCCB** (Serial Camera Control)
2	Maksymalny transfer	400 kbps (Fast Mode) 3,4 Mbps(Hs Mode)	400 kbps	400 kbps
3	Adresowanie	7, 10 bitowe	7, 10 bitowe	7 bitowe
4	Linia danych	SDA	TWD	SIO_D
5	Linia zegarowa	SCL	TWCK	SIO_C
6	Linia sterownia	-	-	SIO_E
7	Transmisja	szeregowa 8 bitowa	szeregowa 8 bitowa	szeregowa 8 bitowa
8	Bit potwierdzenia	ACK	ACK (A)	Don't care bit
9	Bit bez potwierdzenia	NACK	NACK (N)	NA

Tabela 3.4 Porównanie interfejsów I2C, TWI i SCCB

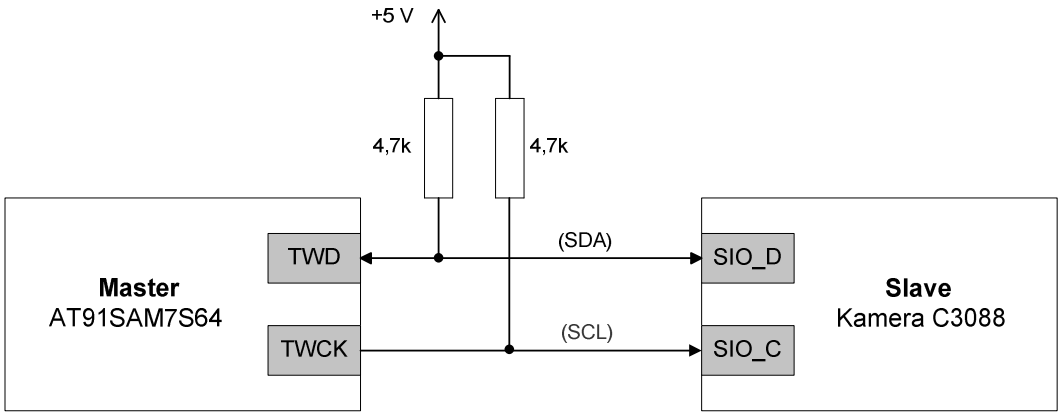
Z porównania wynika, że interfejsy I2C i TWI dla przedstawionych kryteriów nie różnią się poza nazewnictwem. (porównując interfejs Fast Mode I2C z TWI). Interfejs SCCB do trybu transmisji w większości przypadków też jest taki sam.

Istnieje jednak kilka różnic, które są istotne z punktu widzenia tworzenia i testowania programu jak i możliwości rozbudowy magistrali. Pierwszą różnicą jest dodanie w interfejsie

SCCB trzeciej linii SIO_E, która jest odpowiedzialna za wygenerowanie warunków początku i końca transmisji (start i stop). System może jednak pracować bez tego przewodu i wówczas warunki te są generowane przez mastera na linii SDA. W tym przypadku, wg specyfikacji interfejsu SCCB [19], master może być połączony tylko z jednym urządzeniem podrzędnym. Drugą różnicą jest bit potwierdzenia (don't care bit) odbioru danej 1-bajtowej, który w systemie SCCB może być pominięty przez mastera (master nie sprawdza po każdym wysłanym bajcie czy slave potwierdzi jego odbiór).

W przypadku błędów podczas odbioru bajtu przez slave'a zmienia on wartość specjalnie zdefiniowanego w tym celu rejestru w jego pamięci (don't care status register). W projekcie wykorzystywano mastera z zaimplementowanym sprzętowo interfejsem TWI, dlatego wymagał on każdorazowo potwierdzenia od slave'a po wysłaniu jednego bajtu (co realizowała kamera). Przy braku takiego potwierdzenia transmisja zostawała natychmiast przerywana.

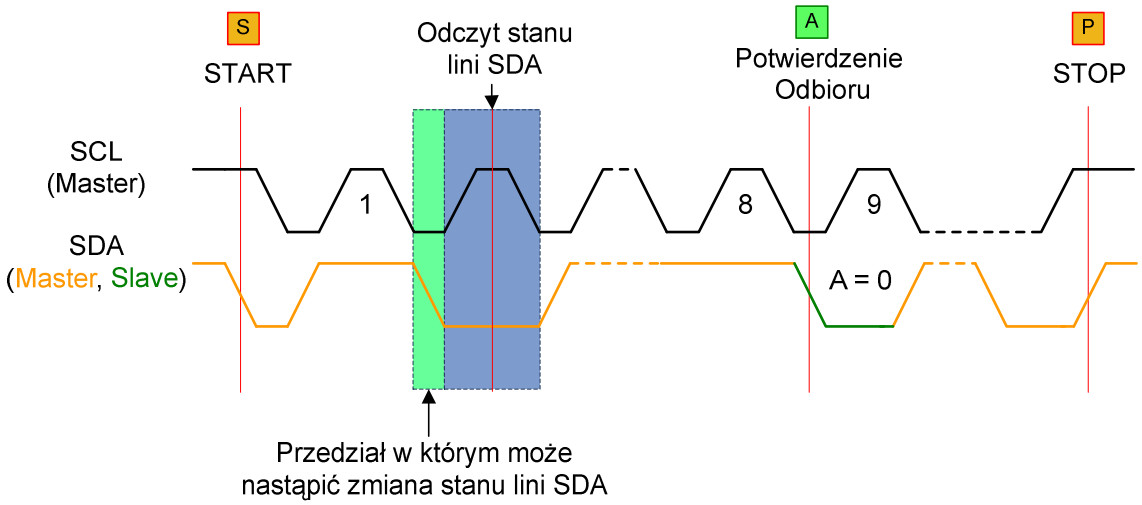
W tej części pracy zostanie przytoczony opis tego interfejsu w odniesieniu do interfejsu I2C. W szczególności zostaną pokazane zmiany, jakie należało zrobić w programie, aby dostosować sprzętowo zaimplementowany w mikroprocesorze interfejs TWI do interfejsu SCCB kamery. Interfejs I2C (Inter Integrated Circuit) jest to standard, w którym dane są przesyłane szeregowo linią SDA (Serial Data Line) w postaci paczek 8-bitowych synchronicznie z sygnałem taktującym podawanym na linię SCL (Serial Clock Line). Urządzeniem, które zarządza i steruje transmisją jest master. W naszym przypadku jest to mikroprocesor AT91SAM7S. Układ ten jest odpowiedzialny za taktowanie, wysyłanie sekwencji rozpoczynającej (start) i kończącej (stop) transmisję oraz przełączanie między trybem nadawania i odbierania danych na linii SDA. Kamera jest urządzeniem typu slave, które posiada unikalny adres i dzięki któremu jest rozpoznawalna w sieci przez mastera. W naszym projekcie kamera ma adres domyślny (single slave ID) ID = 0x60. Istnieje możliwość zmiany tego adresu po wcześniejszym ustawieniu pinu MULT(47) kamery w stan wysoki, a następnie ustawieniu 3 bitów (34, 35, 37), które dają możliwość zakodowania do 8 różnych adresów. Domyślny adres nie został jednak zmieniony, gdyż wymagałoby to ingerencji w płytkę kamery. Do prawidłowego działania transmisji przez I2C wymagane jest podciągnięcie obu linii interfejsu do plusa zasilania przez rezystory (rysunek 3.11), co zostało zrealizowane na płycie przetwarzania obrazu.



Rysunek 3.11 Schemat połączeń układów w sieci SCCB

Wartość zastosowanej rezystancji 4,7 kΩ nie jest krytyczna a jedynie zalecana i dobrana z zapasem. Tylko w większych systemach liczących powyżej kilkunastu elementów w sieci zachodzi realna potrzeba obniżenia tej wartości tak, aby utrzymać parametry transmisji. Związane jest to z pojemnością szyny, która jest przeładowywana przy zmianach stanów i przy zwiększającej się pojemności obniża się szybkość narastania napięcia, analogicznie zwiększanie rezystancji powoduję ten sam efekt. Praktycznie jedynym ograniczeniem w ilości elementów, jakie można połączyć w taką sieć jest właśnie pojemność szyny, której wartość maksymalna została ustalona na 400 pF [21]. Przyjmuje się, że typowy układ elektroniczny obsługujący interfejs I2C wprowadza pojemność około 10 pF (pojemność we/wy). Dodatkowo przy projektowani bardziej rozległego systemu należy wziąć pod uwagę pojemność przewodów i ścieżek szyny.

W zaprojektowanym układzie do szyny I2C dołączone są tylko 2 układy, a zastosowana rezystancja pozwala na prawidłową pracę układu przy pojemności szyny do 200 pF. Dane przesyłane na linii SDA mogą ulec zmianie jedynie wtedy, gdy linia zegarowa SCL jest w stanie niskim (rysunek 3.12). W odcinku czasu, gdy linia zegarowa jest w stanie wysokim, musi nastąpić odczyt stanu z linii SDA i przez ten odcinek czasu dane nie mogą ulec zmianie (w przypadku zmiany stanu na linii SDA w momencie odczytu wystąpią błędy transmisji). Dodatkowo taka zmiana stanu na linii SDA przy wysokim stanie SCL jest wykorzystywana przez interfejs do generacji warunków początku i końca transmisji (start i stop), co zostało pokazane na rysunku 3.12. W tym standardzie transmisji ustalono, że stanem spoczynkowym obu linii jest stan wysoki, dlatego sygnał startu jest generowany przez mastera w wyniku zmiany stanu linii danych z wysokiego na niski, a sygnał stopu w odwrotnej kolejności ze stanu niskiego na wysoki.



Rysunek 3.12 Przebiegi sygnałów na linii danych (SDA) i linii zegarowej(SCL).

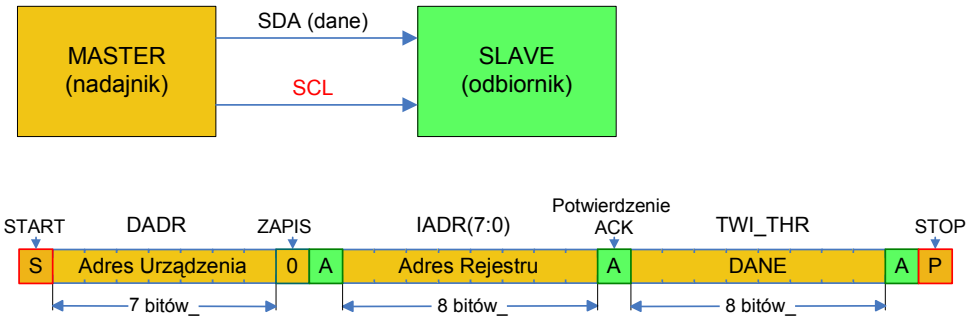
Komunikację mikroprocesora z kamerą poprzez interfejs SCCB możemy podzielić na etapy zapisu i odczytu danych z rejestrów kamery. Dokładny opis i przeznaczenie poszczególnych rejestrów znajduję się w rozdziale 6. Proces zapisu wartości do rejestru został przedstawiony na rysunku 3.13a.

Transmisję rozpoczyna master (mikroprocesor) generując sygnał startu, następnie wysyłany jest 7-bitowy adres urządzenia podrzędnego (slave), z którym master chcę się skomunikować (adres kamery) oraz ósmy bit informujący o tym, że master chce dokonać zapisu danych (0-zapis). Bity wysyłane są szeregowo od najstarszego (MSB) do najmłodszego (LSB) w takt sygnału zegarowego. Jeżeli ten 1 bajt łączności zostanie odebrany przez kamerę i przesłany adres jest zgodny z adresem kamery, wówczas kamera w odpowiedzi ustawi stan linii SDA na 0 (ACK). Jeśli tak się nie stanie, oznacza to, że wysłany adres jest nieprawidłowy lub wystąpiły błędy podczas transmisji. Po odbiorze sygnału ACK następuję wysłanie przez mastera adresu komórki pamięci (rejestru), do którego będą zapisane dane. Jeśli taka komórka istnieje to urządzenie podrzędne (kamera) ponownie dokonuje potwierdzenia (ACK). Trzeci z kolei wysyłany bajt przez mastera to wartość, jaką chcemy zapisać pod podany wcześniej adres rejestru kamery. Jeżeli operacja zapisu do rejestru się powiedzie to master otrzyma potwierdzenie (ACK) i zakończy transmisję wygenerowaniem sygnału stop. Odczyt wartości z kamery jest już nieco bardziej skomplikowany. Związane jest to z istnieniem wewnętrznego licznika wskazującego na adres rejestru, na którym została wykonana ostatnia operacja zapisu lub odczytu. Tylko operacja zapisu jest w stanie zmieniać dowolnie w podanym zakresie adresów rejestrów ten licznik (wskaźnik).

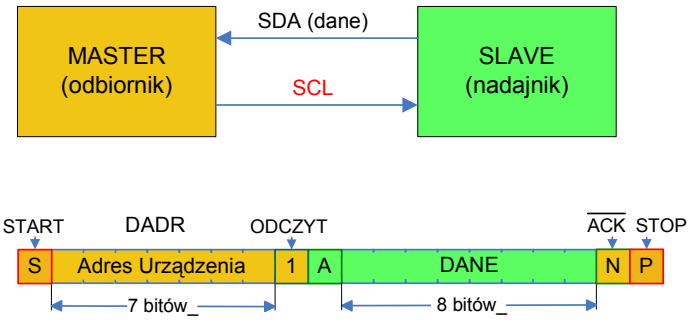
Dokonując tylko operacji odczytu odczytamy wartość spod ostatnio zapamiętanego przez licznik adresu (adres rejestru + 1). Funkcja ta w pewnych sytuacja jest wykorzystywana, np. w celu sprawdzenia czy ustawiona w rejestrze wartość faktycznie tam jest i czy się nie zmieniła. Funkcja odczytu wartości z bieżącego rejestru została z tego powodu zaimplementowana w programie, a sposób odczytu przedstawia rysunek 3.13b. Podczas odczytu to master staję się odbiornikiem, a kamera nadajnikiem danych. Aby master mógł odczytać dane z kamery należy ustawić 8 bit pierwszego bajtu na 0 (odczyt). Drugą istotną różnicą jest to, że master nie potwierdzi odbioru bajtu, jeśli jest to ostatni odczytany bajt (analogicznie nie potwierdza odczytu jednego bajtu). Jeżeli po przesłaniu bajtu slave nie otrzyma od mastera potwierdzenia (ACK) to znaczy, że więcej bajtów nie będzie wysyłanych i należy zakończyć transmisję. Sygnał zegarowy jest przerywany (przechodzi w stan wysoki), linia SDA zostaje odblokowana przez kamerę i również przechodzi w stan wysoki(podciągnięcie przez rezystor do plusa zasilania) generując warunek stop. Rysunek 3.13c przedstawia metodę odczytu dowolnego z dostępnych rejestrów w dowolnej kolejności (z ang. „random read”). Wymaga ona jednak przed dokonaniem odczytu wykonania „pustego zapisu” tzn. zapisu w celu uaktualnienia licznika (wskaźnika) bieżącego rejestru w pamięci. Zapis ten składa się jedynie z adresu urządzenia i adresu rejestru i nie zawiera bajtu danych (z ang. „dummy” byte write sequence), więc wartość w zapisywanym rejestrze nie ulegnie zmianie. Metoda ta okazała się bardzo przydatna szczególnie przy odczytywani wybranych rejestrów kamery.

Ostatnią zastosowaną metodą odczytu jest odczyt sekwencyjny (rysunek 3.13d). Pozwala on w jednym cyklu odczytać całą tablicę rejestrów kamery lub wybrany przez nas przedział rejestrów. Po każdym sygnale potwierdzenia odbioru danych od mastera zwiększana jest o 1 bieżąca wartość licznika (wskaźnika) położenia w pamięci kamery i tym samym sekwencyjnie przesyłane są rejestry o coraz większych adresach.

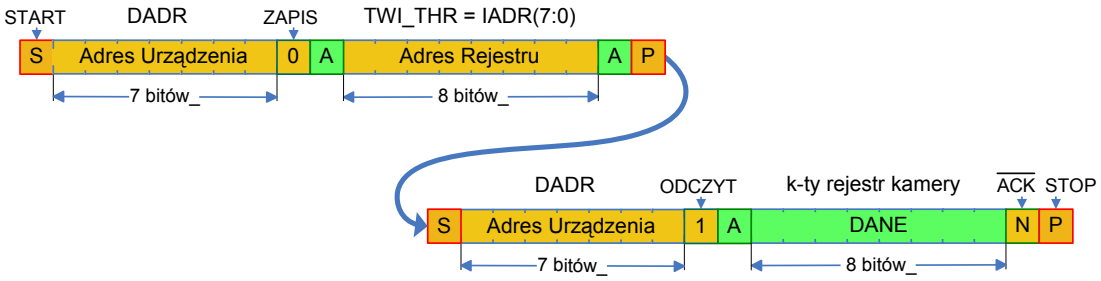
a) Zapis wartości (danej) do rejestru kamery



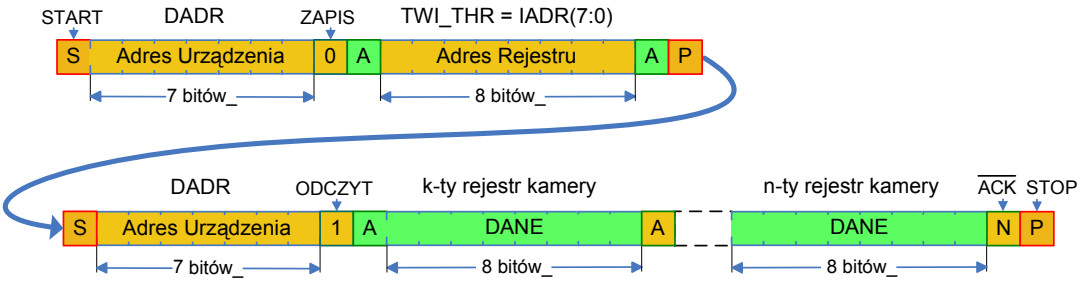
b) Odczyt wartości z bieżącego rejestru kamery



c) Odczyt wartości z rejestru kamery



d) Odczyt sekwencyjny rejestrów kamery



Rysunek 3.13 Operacje zapisu i odczytu w standardzie SCCB

W części programu odpowiedzialnej za komunikację między kamerą a mikrokontrolerem zaimplementowano wszystkie wymienione wcześniej metody zapisu i odczytu. Wykorzystano w ten sposób wszystkie możliwe tryby komunikacji w tym standardzie poza zapisem sekwencyjnym. Zapis sekwencyjny nie został uwzględniony w programie ze względu na brak przydatności w tym systemie. Regulacja parametrów kamery wymaga uaktualniania wybranych rejestrów w kolejności innej niż sekwencyjna. Szczególnie jest to ważne przy ustawianiu układów automatycznej regulacji kamery jak i częstotliwości taktowania zegara systemowego. Listing programu wraz z biblioteką obsługi sccb („sccb.c") znajduje się na płycie CD. W programie wykorzystywane są trzy funkcje:

- void write_register(int reg_address, char reg_value)
- void read_current(void)
- void read_register(int reg_address, int number_reg)

Pierwsza funkcja służy do zapisu wartości do rejestru. Jako parametry podajemy adres rejestru i wartość, jaka ma być zapisana pod tym adresem. Druga funkcja służy do odczytu rejestru bieżącego (na który wskazuje licznik w pamięci kamery). Trzeci funkcja pozwala zarówno na odczyt pojedynczego rejestru jak i odczyt grupy rejestrów sekwencyjnie. Jako parametry należy podać adres początkowy odczytu oraz liczbę rejestrów do odczytania. Wszystkie wartości odczytywane z rejestrów kamery są zapisywane w tablicy „tab_cam_reg" w pamięci SRAM mikrokontrolera.

3.5.2. Interfejs szeregowy RS-232 (V.24, TIA/EIA-232)

Do komunikacji kamery z komputerem wykorzystany został interfejs szeregowy asynchroniczny RS-232 [30]. Mikrokontroler AT91sam7s64 został wyposażony w dwa porty USART (Universal Synchronous/Asynchronous Receiver Transmitters) i dodatkowo jeden port UART do debuggingu (DBGU unit). Moduły USART poza sprzętowym wspieraniem interfejsu RS-232 wspierają też interfejsy RS485, ISO7816 (Smart Card) oraz modulację i demodulację IrDA.

W projekcie wykorzystany został jeden port USART0, poprzez który odbywa się komunikacja mikrokontrolera z komputerem w celu ustawiania wstępnych parametrów kamery. Poprzez interfejs RS-232 wysyłane są komendy sterujące z graficznego interfejsu użytkownika, a także przesyłany jest obraz z kamery do komputera. Na etapie tworzenia i testowania algorytmu przetwarzania obrazu interfejs ten był wykorzystywany do odczytu

pamięci SRAM mikrokontrolera pełniąc rolę debuggera. Do powyższych zastosowań transmisja asynchroniczna była wystarczająca. Pomimo tego do portu DB9 na płytce głównej przetwarzania obrazu zostały doprowadzone dwie dodatkowe linie RTS i CTS umożliwiające realizację transmisji synchronicznej. W trybie asynchronicznej transmisji szeregowej do komunikacji wymagane są tylko 3 linie: RxD (Receive Data), TxD (Transit Data) i linia odniesienia (masa). Przy łączeniu komputera z innym urządzeniem DTE (np. mikrokontrolerem) należy pamiętać, że port transmisyjny (TxD) jednego urządzenia musi być połączony z portem odbiorczym (RxD) drugiego. Tego typu połączenie nosi nazwę „null modem". W celu zwiększenia zasięgu transmisji przyjęto dopuszczalne poziomy napięć dla tego standardu z przedziału (-15 V do +15 V). Aby mikrokontroler mógł komunikować się z komputerem na tym poziomie napięć niezbędne było zastosowanie pośredniczącego układu konwertera sygnałów ze standardu TTL/CMOS do RS-232. W tym celu zastosowano konwerter ST3232, który gwarantuje poprawny transfer danych do prędkości 250 Kbps (prędkość dwukrotnie większa od wykorzystywanej w układzie). Układ ten zasilany jest pojedynczym napięciem 3 V, które zostaje przetworzone na poziomy napięć wyjściowych w zakresie ±13,2 V. Dodatkowo posiada wbudowane zabezpieczenie przeciw przeciążeniowe, które chroni go przed skutkami zwarć przewodów sygnałowych do masy. Aby zmniejszyć wrażliwość interfejsu RS-232 na zakłócenia (szumy) przyjęto w standardzie minimalny margines różnicy napięć między wyjściami a wejściami interfejsu wynoszący 2 V. Stąd ustalono poziomy napięć dla wejść i wyjść w stanie wysokim (1) i niskim (0) (tab. 3.5).

Piny	Stan wysoki (1)	Stan niski (0)
Wyjściowy(TxD)	-5 ÷ -15	+5 ÷ +15
Wejściowy(RxD)	-3 ÷ -15	+3 ÷ +15

Tabela 3.5 Poziomy napięć dla standardu RS-232

W zbudowanym układzie parametry transmisji dla obu urządzeń połączonych tym interfejsem zostały ustawione wg tabeli 3.6.

Lp.	Parametry transmisji	
1	Prędkość transmisji	115200 bps(bodów)
2	Liczba bitów danych	8
3	Liczba bitów stopu	1
4	Kontrola parzystości	Brak (no parity)
5	Tryb transmisji	Half Duplex

Tabela 3.6 Ustawienia parametrów transmisji

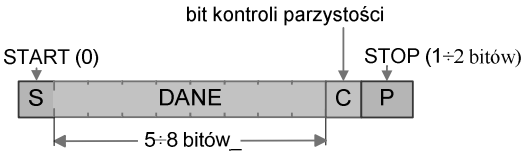
Dla standardu RS-232 przyjęto, że stanem spoczynkowym jest stan wysoki (-15 V do -5 V). Transmisję zawsze rozpoczyna bit startu ustawiający na linii stan niski (+5 V do +15 V). Dalej przesyłane są dane (od 5 do 8 bitów) bit po bicie od najmłodszego (LSB) do najstarszego (MSB). Kolejnym przesyłanym bitem jest bit kontroli parzystości (który został w naszym programie pominięty). Ramkę kończy bit stopu (stan niski), który może wynosić odpowiednio 1, 1,5 lub 2 (rysunek 3.14a). Bit stopu jest w rzeczywistości minimalnym czasem przerwy w nadawaniu, jaki jest wymagany, aby można było wysłać kolejną ramkę (paczkę) danych. Jeżeli przesyłane dane mają większy rozmiar od jednej ramki to zostają podzielone zgodnie z zadeklarowaną liczbą bitów danych w ramce i wysyłane szeregowo z odstępem czasowym, który wybraliśmy (bit stopu). Taki sposób transmisji, w którym sygnał startu musi pojawiać się na początku każdej ramki i długość jednej paczki danych nie może przekroczyć 8 bitów jest podyktowany względami bezpieczeństwa.

W odróżnieniu od interfejsu I2C, gdzie występuje linia zegarowa taktująca oba urządzenia, tutaj każde z urządzeń ma swój własny zegar. Bardzo ważną rzeczą jest to, żeby te zegary pracowały z możliwie tą samą częstotliwością, a im więcej bitów danych przesyłamy tym nawet mała różnica częstotliwości przekłada się na dużą różnicę w fazie, co powoduje błędy transmisji. Zbocze malejące, które jest wywołane przez bit startu pełni funkcję resetu i startu zegara odbiornika. W praktyce różnica częstotliwości między tymi zegarami nie powinna przekraczać 5%.

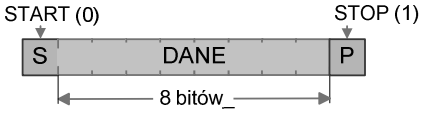
Budowę ramki danych, która została wykorzystana w programie do komunikacji przez interfejs RS-232 przedstawia rysunek 3.14b. Ustalono, że dane będą przesyłane w paczkach po 8 bitów, co odpowiada formatowi danych obrazu (wartość każdego piksela obrazu jest reprezentowana przez liczbę 8-bitową). Bit parzystości został pominięty gdyż testy wykazały, że nie występują zakłócenia w transmisji. Gdyby takie zakłócenia występowały to zastosowanie kontroli parzystości i tak nie dałoby nam gwarancji prawidłowej transmisji (np. nie wykryje błędu, jeżeli przekłamanie wystąpiło na 2*n bitach, gdzie n=1,2,3,4). Aby przyspieszyć stosunkowo wolną transmisję prze interfejs RS-232 skrócono wymagany odstęp miedzy kolejnymi ramkami danych (1 bit stopu). Podjęto także próbę zwiększenia prędkości transmisji do 230400 bps jednak ograniczenia sprzętowe nie pozwoliły na poprawny przesył danych przy tej prędkości. Podczas testów ustalono, że blok interfejsu RS-232 wbudowany w płytę główną komputera pozwala na transmisję z maksymalną prędkością 115200 bps (transfer rzeczywisty 92160 bps wg wzoru poniżej). Teoretycznie mikrokontroler i zastosowany układ pośredniczący (przetwornica pojemnościowa) pozwalają na transfer z prędkością 230400 bps, ale niewiele standardowych płyt głównych komputerów wspiera transmisję z taką prędkością.

$$T_R = \frac{T_S * 8}{L}, \quad \text{gdzie} \quad \begin{cases} T_R - Transwer \quad rzeczywisty \\ T_S - Transwer \quad maksyma\ln y \quad (T_S = 115200 \quad bps) \\ L - Liczba \quad bitów \quad w \quad ramce \quad (L = 10) \end{cases}$$

a) Ramka danych dla transmisji szeregowej asynchronicznej z kontrolą parzystości



b) Ramka danych wykorzystywana w programie (bez kontroli parzystości)



Rysunek 3.14 Ramki danych w transmisji szeregowej asynchronicznej

3.5.3. Interfejs równoległy

Porty Y i UV kamery są bezpośrednio połączone z wyprowadzeniami mikrokontrolera wg tab. 3.1 (rozdział 3). Pomimo, że mikrokontroler jest zasilany napięciem 3,3 V jego wyprowadzenia są przystosowane do pracy z sygnałami o amplitudzie 5 V. Eliminuje to konieczność stosowania konwerterów napięcia. Transmisja równoległa jest bardzo szybka, gdyż cały bajt (wartość 1 piksela) jest odbierany w jednym cyklu odczytu stanu portu (PDSR) Niewielką wadą tego rozwiązania jest tylko to, że wymaga ono dużej liczby pinów mikrokontrolera (liczba pinów musi być równa formatowi przesyłanych danych). W naszym przypadku przy odczycie obu składowych obrazu (Y i CrCb) magistrala równoległa zajmowała 16 pinów plus dodatkowe 3 wymagane przy synchronizacji z kamerą.

3.5.4. Interfejs JTAG (IEEE 1149.1)

Interfejs JTAG (Joint Test Action Group) jest szeregowym interfejsem umożliwiającym odczyt i programowanie pamięci FLASH i SRAM mikrokontrolera. Szczególnie przydatny jest do debuggowania programu oferując m.in.:

- Pracę krokową
- Pułapki programowe
- Podgląd pamięci danych oraz rejestrów

Interfejs ten jest wspierany przez mikrokontroler poprzez rejestr BSR (Boundary-Scan Register), który zawiera 96 komórek jednobitowych powiązanych z pinami mikrokontrolera. Każdy pin PA mikrokontrolera (we/wy) posiada przyporządkowane 3 bity w rejestrze BSR. Są to bity: input, output i control, których programowe ustawianie i odczyt pozwala na realizację wymienionych wcześniej zadań. W naszym przypadku korzystaliśmy z graficznego interfejsu użytkownika programu IAR, który automatycznie ustawiał poszczególne bity rejestru BSR po dokonaniu wyboru danej opcji w debuggerze. JTAG jest interfejsem czteroprzewodowym posiadającym dwie linie danych TDO (Test Data Output – sygnał danych z urządzenia docelowego do PC) i TDI (Test Data Input – sygnał danych z PC do układu docelowego). Trzecią linią TCK jest przesyłany sygnał zegarowy z komputera do mikrokontrolera (układu docelowego), a czwarta linia TMS (Test Mode Select) jest wykorzystywana przez komputer do przełączania trybu pracy układu docelowego. Prędkość transmisji podobnie jak dla standardu RS-232 wynosi 115200 bps. Programowanie mikrokontrolera odbywa się poprzez programator „ARMcable 1” [24] wyposażony w dodatkowy szybki bufor trójstanowy przechowujący bieżące informacje z danej sesji zapisu/odczytu. Dodatkowe informacje o tym interfejsie i jego obsłudze w mikrokontrolerach z rodziny AT91sam7S można znaleźć w instrukcjach [1] i [2].

4. Realizacja systemu wizyjnego

4.1. Kamera

Każdy początkujący projektant systemów wizyjnych stawia sobie pytanie, jaką kamerę wybrać do realizacji założonego systemu wizyjnego. Pierwszym dylematem jest wybór typu sensora. Obecnie najpopularniejsze są sensory CCD (Charge Coupled Device) i CMOS (Complementary Metal Oxide Semiconductor). Sensory CCD są powszechnie stosowane w urządzeniach o dużej rozdzielczości tj. kamery i aparaty cyfrowe (choć czasem można spotkać wersję z przetwornikiem CMOS). W telefonach komórkowych stosowane są oba typy sensorów a w kamerach internetowych praktycznie wyłącznie przetworniki CMOS). W autonomicznych systemach wizyjnych najczęściej wykorzystywane są kamery z przetwornikiem CMOS ze względu na poniższe cechy w porównaniu do przetworników CCD:

- Mniejszy pobór energii (nawet kilkunastokrotnie) co powoduje dłuższy czas pracy przy zasilaniu bateryjnym.
- Wbudowane układy wewnętrzne (m.in. przetwornik ADC), które ograniczają do minimum liczbę zewnętrznych elementów potrzebnych do połączenia takiego przetwornika z mikrokontrolerem. (w praktyce jest to generator kwarcowy i kilka elementów R i C)
- Często szybsze działanie (nawet 500 fps) i dzięki temu ograniczenie efektu rozmazywania obrazu podczas ruchu elementów obrazu jak i samego przetwornika.
- Ograniczenie do minimum wpływu prześwietlenia(przepełnienia) danego piksela na ładunki zgromadzone w sąsiednich pikselach („reduced blooming”)
- Niższe ceny ze względu technologię produkcji CMOS.

Powodem, dla którego matryce CCD są częściej wykorzystywane w aparatach i kamerach jest to, że dają lepiej nasycony i kontrastowy obraz. Jest to szczególnie widoczne przy zmiennych warunkach oświetleniowych. Matryca CMOS jest tak zbudowana, że część powierzchni pojedynczego fotoelementu (piksela) jest zajęta przez układ odpowiedzialny za przetworzenie obrazu (grupa tranzystorów). Powoduje to, że nie wszystkie fotony niosące informacje o obrazie trafiają na powierzchnię aktywną fotoelementu. Tym samym przy słabym oświetleniu jakość obrazu ulega pogorszeniu. Wydłużenie czasu ekspozycji w takim

przypadku poprawia nieco obraz. Początkowo problemem była również gęstość upakowania elementów światłoczułych na standardowej matrycy 1/2,5 cala. Zmniejszanie częściowo już zajętej powierzchni aktywnej fotoelementu skutkowało słabszym sygnałem na jego wyjściu. Wykonanie np. 5 megapikselowej matrycy CMOS wymaga zastosowania trzy razy więcej tranzystorów. Obecnie technologia produkcji matryc CMOS pozwala już wytworzyć matryce składające się 8 milionów pikseli i wymiarach 1/2,5 cala (przy częstotliwości pracy 10 fps lub 30 fps dla zmniejszonej rozdzielczości – 2 megapiksele) [17]. W naszych założeniach idealne oddanie barw, a przez to jakość obrazu nie były najważniejsze. Ważniejszą rzeczą było wybranie przetwornika obrazu, który pozwoli na pracę ze stosunkowo dużą częstotliwością a jednocześnie będzie się charakteryzował niskim poborem energii i nie wymagał specjalnych układów pośredniczących do połączenia z mikrokontrolerem. Kierując się powyższymi kryteriami wybraliśmy moduł kamery z przetwornikiem CMOS typu OV6620. O wyborze tego modułu zadecydowały również duże możliwości regulacyjne jego parametrów poprzez dołączony interfejs (SCCB).

4.1.1. Budowa i parametry kamery

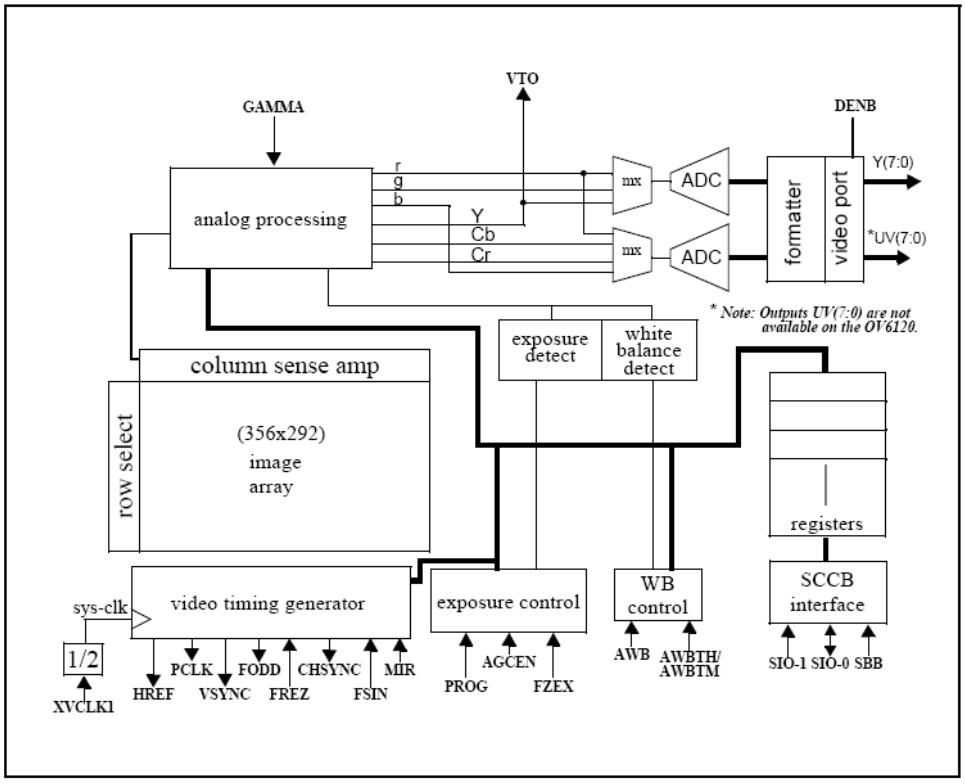
Spośród dostępnych kamer CMOS z przetwornikami obrazu o rozdzielczości CIF (Common Intermediate Format 352x288) wybrany został moduł kamery C3088 z przetwornikiem OV6620 firmy OmniVision [20]. Spośród innych przetworników o tej rozdzielczości (m.in. MI-0111 - Micron, PB-0100 – Photobit, PO3010 – Pixelplus) odznaczał się on największymi możliwościami regulacji parametrów kamery oraz największą szybkością (50 fps). Wybrane parametry kamery przedstawia tab. 4.1.

Lp.	Parametry przetwornika	
1	Rozdzielczość matrycy Rozdzielczość rzeczywista (format podstawowy obrazu)	356x292 CIF (352x288)
2	Wymiary pojedynczego piksela	9.0x8.2 µm
3	Tablica filtrów koloru	wzór Bayera
4	Zakres dynamiczny (Dynamic Range)	>72 dB
5	Stosunek sygnału do szumu	>48 dB
6	Poziomu zakłóceń o stałym wzorze (FPN)	<0,03% Vpp
7	Maksymalna prędkość robienia zdjęć	50 fps
7	Czas ekspozycji	500:1
9	Napięcie zasilania Zakres napięć akceptowanych na wejściach (stan H)	5 V DC 3,3÷5 V
10	Pobór energii - stan aktywny - stan oczekiwania	 <80 mW <30 µW

Tabela 4.1 Wybrane parametry charakterystyczne przetwornika OV6620

Wybrany przetwornik ma stosunkowo duże wymiary pojedynczych fotokomórek (pikseli). Jest to jego zaletą gdyż pojedynczy element światłoczuły o większej powierzchni jest w stanie przechwycić większą liczbę fotonów nawet przy słabszym oświetleniu. Tym samym stosunek sygnału do szumu ma większą wartość niż w bardzo gęsto upakowanej matrycy (dla tych samych warunków pracy). Dodatkowo efekt bloomingu jest tu bardzo ograniczony ze względu na dużą pojemność ładunkową takiej fotokomórki. Układ ten charakteryzuję się bardzo małym poziomem szumów stałych porównując do przetworników tej samej firmy, ale o wyższych rozdzielczościach. Z punktu widzenia zastosowań tego przetwornika w urządzeniach mobilnych kryterium oszczędności energii jest spełnione. W warunkach pracy pobiera on prąd o wartości zaledwie od kilku do kilkunastu mA zależnie od częstotliwości taktowania. Możliwe jest również ustawienie tego układu w stan uśpienia („Power down mode") w którym pobór prądu jest rzędu µA.

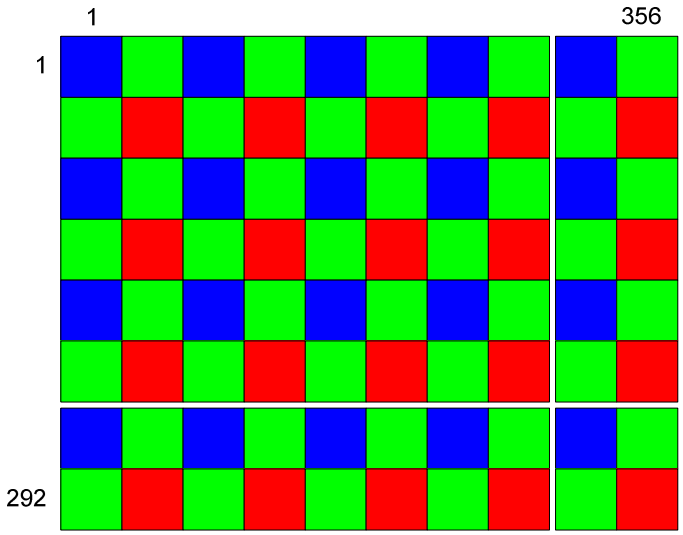
Schemat blokowy przetwornika obrazu OV6620 przedstawia rysunek 4.1. Sygnał analogowy z wyjść matrycy w postaci poziomów napięć zostaje poddany kondycjonowaniu wg ustawionych wartości parametrów regulacyjnych (m.in. separacja kolorów, wzmocnienie dla poszczególnych składowych, korekcja gamma, balans bieli, czas ekspozycji, parametry luminancji i chrominancji) Następnie na podstawie kolorów bazowych RGB obliczane są składowe YUV (YCrCb) i wartości te podawane są na dwa multipleksery. Do pierwszego multipleksera doprowadzony jest sygnał Y, R i G a do drugiego Cr, Cb, B i G. Połączenie takie jest celowe i determinuje ono możliwe sekwencje kolorów na wyjściach 8-bitowych Y i UV. Multipleksery są sprzężone z przetwornikami analogowo cyfrowymi i sterowane tak, aby proces próbkowania sygnału przebiegał prawidłowo. W rezultacie na wejściach przetworników A/D pojawiają się sekwencyjnie sygnały dołączone do multiplekserów i po próbkowaniu trafiają do bloku formatowania. Blok formatowanie spełnia również rolę multipleksera, który w zależności od ustawionego żądanego formatu danych przesyła wartości dyskretne na odpowiednie piny portów wyjściowych Y i UV. Wszystkie operacje są zsynchronizowane z zegarem systemowym, którego rolę pełni rezonator kwarcowy 17,73 MHz.



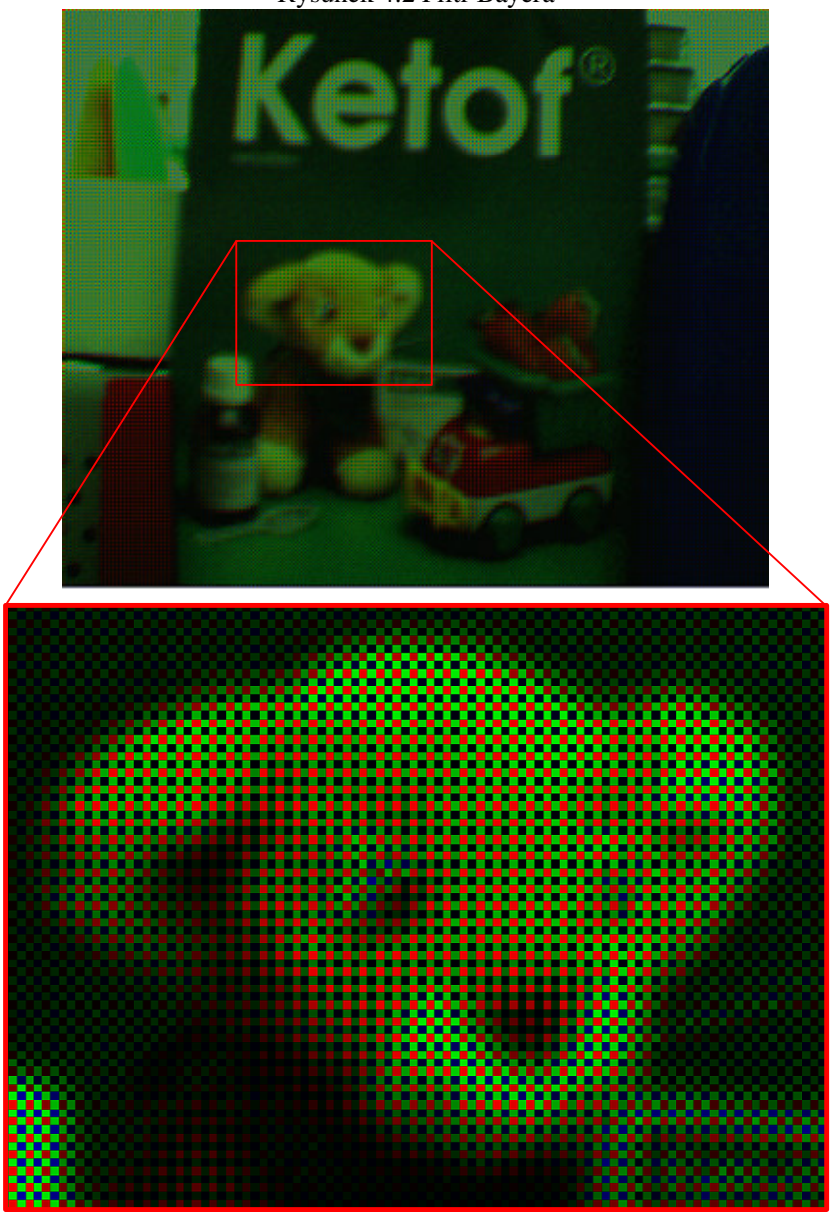
Rysunek 4.1 Schemat blokowy przetwornika OV6620 z instrukcji producenta [20]

Elementy światłoczułe matrycy CMOS pokryte są filtrami kolorowymi RGB wg wzoru Bayera (rysunek 4.2). Każdy element matrycy przepuszcza tylko jedną z trzech składowych światła R,G lub B. Pozostałe dwie składowe dla danego piksela są obliczane na podstawie wartości pikseli sąsiednich w procesie interpolacji opisanym w podpunkcie 5.2.2. Istnieją także inne wzory filtrów, które wg producentów matryc CMOS i CCD pozwalają na jeszcze lepsze odwzorowanie barw. Na uwagę zasługuje tu matryca Foveon X3, która jako jedyna pozwala rejestrować wszystkie trzy składowe RGB dla każdego piksela. Umożliwia to rzeczywiste odwzorowanie kolorów (bez potrzeby interpolacji brakujących składowych).

Większość stosowanych obecnie wzorów filtrów podobnie jak filtr Bayera posiada liczbę elementów zielonych równą sumie elementów dla pozostałych dwóch składowych (czerwonej i niebieskiej). Rozwiązanie takie ma na celu przybliżyć czułość takiej matrycy do czułości oka ludzkiego, które jest najbardziej wrażliwe na składową zieloną światła widzialnego.



Rysunek 4.2 Filtr Bayera



Rysunek 4.3 Format zdjęcia z kamery (RAW)

4.1.2. Skanowanie zdjęcia

Sposób przechwytywania zdjęcia z kamery ma duży wpływ na szybkość działania systemu. Metoda przechwytywania zastosowana przy przetwarzania obrazu na płycie głównej została opisana w rozdziale 5.2. Dla celów testowych i kalibracji parametrów kamery napisano graficzny interfejs użytkownika komunikujący się z płytą główną komputera przez interfejs RS-232. Ze względu na niską prędkość transmisji tego łącza i ograniczoną pamięć RAM mikrokontrolera do 16 KB należało zastosować metodę skanowania, która będzie najbardziej efektywna przy nałożonych ograniczeniach. W tym celu przeprowadzono obliczenia i pomiary na podstawie, których wybrano metodę i napisano moduły programu przechwytujące zdjęcia do komputera. Przeanalizowano następujące sposoby skanowania zdjęcia z kamery:

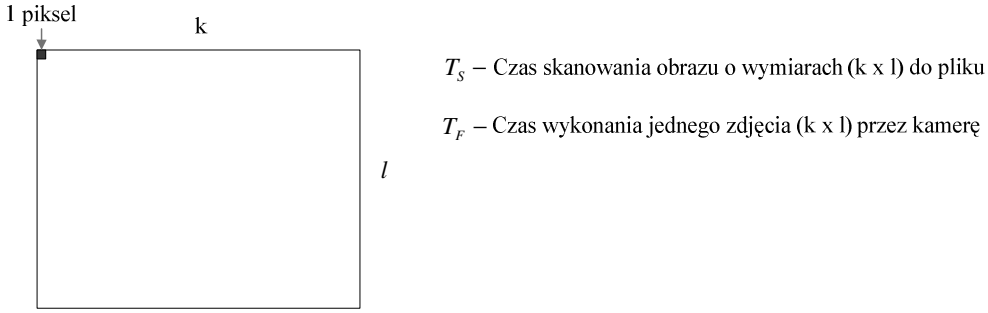
- Skanowanie pionowe
- Skanowanie poziome

Skanowanie pionowe polegało na odczycie jednego piksela z 1 linii poziomej matrycy i wysłania go przed nadejściem kolejnej. W efekcie w czasie potrzebnym na zrobienie jednego zdjęcia (Tf) wysyłana była do komputera tylko jedna pionowa linia obrazu (rysunek 4.4a). Do zeskanowania całego zdjęcia w formacie CIF potrzeba było aż 352 zdjęć wykonanych przez kamerę, co przy najszybszym możliwym w tym przypadku taktowaniu dawało czas około 2 minut. Metoda ta zaczerpnięta z pracy [18] została z tego powodu ostatecznie odrzucona.

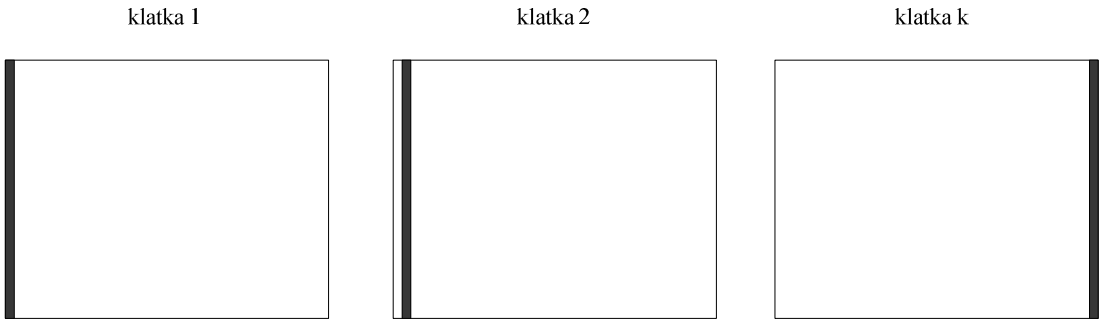
Skanowanie poziome jest trybem skanowania kamery, dlatego wykorzystanie tej metody w tym projekcie jest celowe. Podstawowy formatem, w jakim kamera robi zdjęcia jest format CIF (352x288). Zdjęcie (RAW) w takiej rozdzielczości zajmuje 99 KB pamięci. Wysyłany do komputera plik jest plikiem BMP (podpunkt 4.2.3) dlatego dodatkowo pamięć zajmowana przez zdjęcie zwiększa się o 1078 bajtów. Najszybszą metodą przechwycenia zdjęcia z kamery do komputera jest zapis całego zdjęcia z maksymalną szybkością do pamięci mikrokontrolera, a następnie przesłanie całej zawartości pamięci do terminala. Metoda ta może być zastosowana tylko w przypadku mikrokontrolera posiadającego pamięć RAM większą od pamięci zajmowanej przez przechwytywany obraz.

Skanowanie poziome 1-liniowe jest wolne podobnie jak skanowanie pionowe 1-liniowe. Różnica polega na tym, że tutaj odczytywana jest pojedyncza linia pozioma obrazu podczas trwania tylko jednego sygnału HREF, a następnie wysyłana jest w pętli znak po

znaku. Jedyną zaletą 1-liniowych metod skanowania jest niewielka wymagana wolna pamięć. Dla przedstawionej tu metody skanowania pionowego wymagany jest tylko 1 bajt wolnej pamięci na dane (1 piksel pobierany i wysyłany) a dla skanowania poziomego 1-liniowego *k* bajtów (*k* – szerokość obrazu w pikselach).

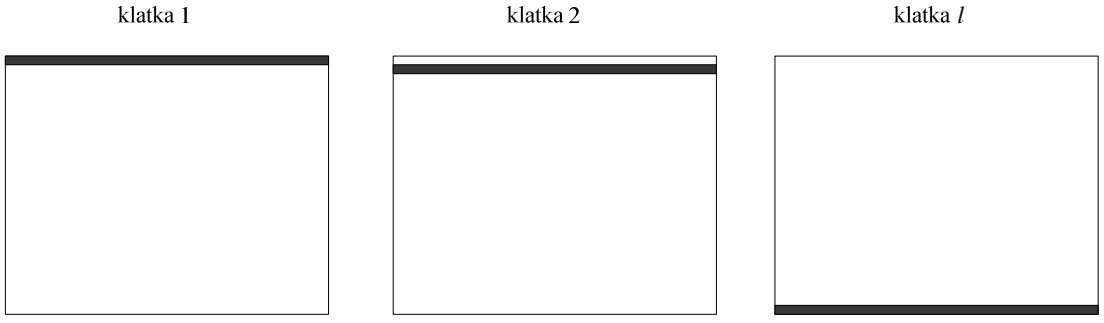


a) Skanowanie pionowe 1-liniowe



$$T_S = k * T_F$$

b) Skanowanie poziome 1-liniowe



$$T_S = l * T_F$$

Rysunek 4.4 Skanowanie 1-liniowe

W programie współpracującym z GUI zastosowano skanowanie poziome obszarowe (rysunek 4.5c). Polega ono na skanowaniu wycinka obrazu będącego wielokrotnością linii poziomej. Na podstawie informacji, jaką wolną pamięciom RAM dysponujemy należy tak dobrać wielkość obszaru, aby w wyniku dzielenia liczby linii obrazu przez liczbę linii obszaru uzyskać liczę naturalną (p).

T_S – Czas skanowania obrazu o wymiarach (k x l) do pliku

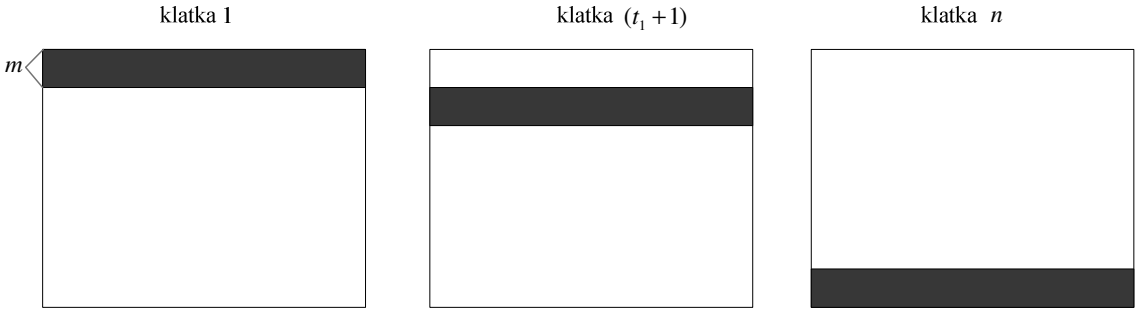
T_F – Czas wykonania jednego zdjęcia (k x l) przez kamerę

T_{WRi} – Czas skanowania obszaru m przez kamerę i jego wysłania do pliku (licząc od początku klatki)

m – Liczba linii objętych skanowaniem sekwencyjnym

n – Wymagana liczba klatek przy danym skanowaniu

c) Skanowanie poziome obszarowe $(T_{WRi} > T_F)$

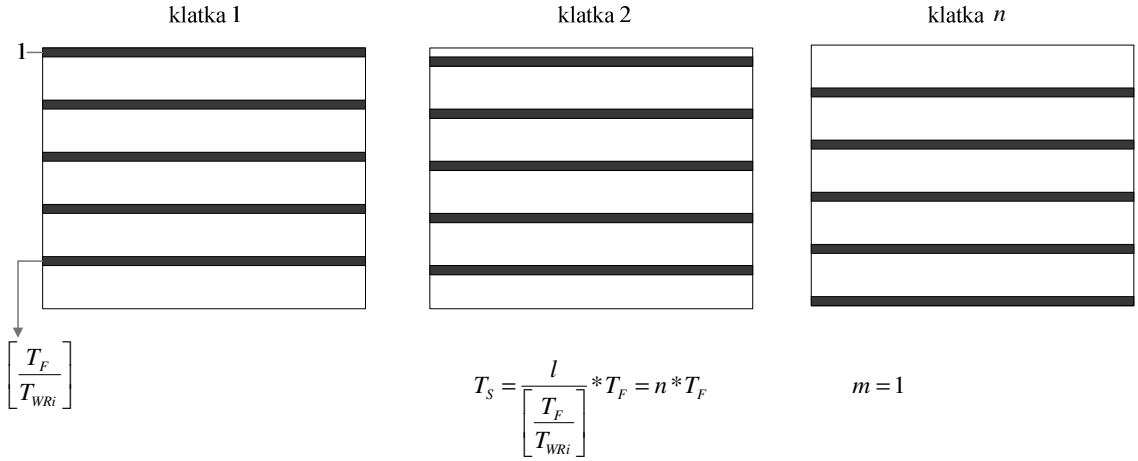


$$T_S = \sum_{i=1}^{p} t_i * T_F = n * T_F$$

$$p = \frac{l}{m} \quad gdzie \quad p \in N$$

$$t_i = \begin{cases} \dfrac{T_{WRi}}{T_F} & dla \quad \dfrac{T_{WRi}}{T_F} \in N \\ \left[\dfrac{T_{WRi}}{T_F}\right] + 1 & dla \quad \dfrac{T_{WRi}}{T_F} \notin N \end{cases}$$

d) Skanowanie poziome n-liniowe $(T_{WRi} < T_F)$



$$T_S = \frac{l}{\left[\dfrac{T_F}{T_{WRi}}\right]} * T_F = n * T_F \qquad m = 1$$

Rysunek 4.5 Skanowanie obszarowe i n-liniowe

Przed wyborem wielkości obszaru należy dokonać obliczeń wg wzorów na rysunku 4.5c. Jak wykazały obliczenia i pomiary nie zawsze wybór większego obszaru skanowania przy tym algorytmie przechwytywania zdjęcia jest optymalny. Dla dobranej częstotliwości skanowania zdjęcia wynoszącej dla formatu CIF 403 KHz i pamięci RAM 16 KB obliczono, że maksymalny obszar skanowania sekwencyjnego może zajmować 36 linii poziomych obrazu. Pomiary czasu skanowania całego obrazu wykazały, że czas ten jest taki sam zarówno dla 36 jak i 32 linii. Obliczenia wg wzorów na rysunku 4.5c potwierdziły wyniki otrzymane z pomiarów. Naturalnym był więc wybór obszaru składającego się z 32 linii, który przy tym samym czasie skanowania obrazu pozwolił zaoszczędzić około 1,4 KB pamięci RAM mikrokontrolera. Tabela 4.2 przedstawia wyniki obliczeń i pomiarów dla dwóch rozdzielczości obrazu przy skanowaniu obszarowym. Drobne różnice między wielkościami obliczonymi a pomierzonymi wynikają z uproszczenia wzorów 4.4, które nie uwzględniają czasu od momentu wysłania komendy do kamery do początku transmisji obrazu. Pomiary dodatkowo obarczone są błędami na poziomie 1%. Przedstawione wyniki pokazują, że uzyskane czasy przy skanowaniu obszarowym sekwencyjnym są zbliżone do maksymalnych teoretycznych wartości transferu plików wzorcowych o tych samych rozmiarach. Jedynym minusem tej metody jest wymaganie stosunkowo dużej wolnej pamięci RAM (zależnie od wybranego obszaru skanowania).

	CIF (352x288)	QCIF (176x144)
Częstotliwość pracy kamery	403KHz	554,2KHz
Obszar skanowania sekwencyjnego	352x32	176x72
Czas transferu obrazu wzorcowego – z obliczeń	8,9 s	2,3 s
Czas transferu obrazu wzorcowego – z pomiarów	9,1 s	2,5 s
Czas transferu obrazu z kamery – z obliczeń	12,8 s	2,6 s
Czas transferu obrazu z kamery – z pomiarów	13,0 s	2,8 s

Tabela 4.2 Czasy transferu obrazu w formacie CIF i QCIF

Ostatnią rozpatrywaną metodą skanowania było skanowanie poziome n-liniowe. Metoda polega na skanowaniu pojedynczych linii obrazu i ich natychmiastowym wysyłaniu. Odstępy między skanowanymi liniami z tej samej klatki są tak dobrane żeby można było w tym czasie wysłać łączem szeregowym ze skanowaną wcześniej linię. W każdej klatce jest skanowana taka sama liczba linii, która zależy od częstotliwości skanowania i formatu obrazu. Metoda ta pozwala uzyskać czasy transferu porównywalne z metodą skanowania obszarowego. Wymagana wolna pamięć RAM jest wielkości potrzebnej do przechowania tylko jednej linii obrazu, co jest dużą zaletą tej metody. Wadą tego typu skanowania jest to, że linie obrazu są wysyłane w kolejności innej niż skanuje kamera. Aby odtworzyć obraz z pliku

ze skanowany w ten sposób należy przywrócić linią obrazu ich pierwotną kolejność (indeksy). Biorąc pod uwagę wszystkie zalety i wady przedstawionych tu metod została wybrana metoda skanowania obszarowego która mimo, że wymaga większych zasobów wolnej pamięci RAM to pozwala na bezpośredni zapis do pliku obrazu w formacie skanowania kamery. Pozwala to na zastosowanie jako terminala dowolnego programu obsługującego interfejs RS-232, a plik zapisany na dysku w ten sposób jest obrazem w formacie BMP widzianym z kamery.

4.1.3. Opis tworzenia pliku BMP

Plik BMP jest to plik z grafiką rastrową, który może być zapisany w postaci nie skompresowanej lub z kompresją bezstratną RLE. W napisanym programie dane z kamery zapisywane są w pliku BMP bez kompresji w celu ich dalszej obróbki przez graficzny interfejs użytkownika. Plik w formacie BMP składa się zasadniczo z czterech części:

- Nagłówka pliku (14 bajtów) – „1”
- Nagłówka bitmapy (40 bajtów) –„2”
- Palety kolorów (opcjonalnie)
- Danych obrazu

Nagłówek pliku (File Header) zawiera informację o rodzaju pliku, jego długości w bajtach oraz pozycji danych obrazu licząc od początku pliku tab. 4.3. Nagłówek bitmapy (Bitmap Header) określa m.in.:

- szerokość i wysokość obrazu w pikselach
- liczbę bitów, która reprezentuje wartość pojedynczego piksela
- liczbę możliwych kolorów obrazu (paletę kolorów)
- algorytm kompresji – jeśli jest stosowany

Długość pliku BMP w bajtach obliczana jest na podstawie wzoru:

$S_{BMP} = S_H + S_P + S_D$

Gdzie:

S_H - długość nagłówka, $S_H = 54$

S_P - długość palety kolorów, $S_P = 4 * 2^n$, gdzie n to głębia bitowa

S_D - długość danych obrazu, $S_D = \frac{w * h * n}{8}$, gdzie w i h to szerokość i wysokość obrazu w pikselach.

W wykorzystywanym formacie danych wyjściowych kamery wartość każdego piksela jest reprezentowana przez 8 bitów stąd powyższy wzór upraszcza się do postaci:

$$S_{BMP} = 1078 + w * h$$

Paleta kolorów występuje tylko w trybach 8-bitowym i niższych. Dla trybu 8-bitowego każdy kolor jest reprezentowany przez 4 bajty składowe - kolejno B,G,R i alpha (nieużywany).

	offset	Rozmiar [b]	Opis	Ustawienia
1	0000h	2	Znacznik nagłówka (BM, BA, CI, CP, IC, PT)	„BM”
	0002h	4	Długość pliku w bajtach	102454 (26422)
	0006h	4	Zarezerwowane	0
	000Ah	4	Pozycja danych licząc od początku pliku	54
2	000Eh	4	Długość nagłówka (40 dla Windows)	40
	0012h	4	Szerokość obrazu w pikselach	352(176)
	0016h	4	Wysokość obrazu w pikselach (jeśli jest ujemna to obrazek jest zapisany od góry do dołu)	288(144)
	001Ah	2	Ilość planów (płatów) w obrazie	1
	001Ch	2	Liczba bitów na piksel (1, 4, 8, 16, 24, 32)	8
	001Eh	4	Algorytm kompresji „0” - brak „1” – 8-bitowa RLE „2” – 4-bitowa RLE	0
	0022h	4	Rozmiar rysunku (bitmapy) skompresowanego	0
	0026h	4	Rozdzielczość pozioma (ilość pikseli na metr)	0
	002Ah	4	Rozdzielczość pionowa (ilość pikseli na metr)	0
	002Eh	4	Liczba kolorów w palecie	256
	0032h	4	Liczba ważnych kolorów w palecie (stosowane przy animacji bitmapy) „0” – wszystkie kolory ważne	0

Tabela 4.3 Nagłówek pliku BMP

4.2. Optyka

Możliwości systemu wizyjnego w dużej mierze uzależnione są od rodzaju optyki. Położenie obiektywu, kąt nachylenia, ogniskowa i filtry decydują o rodzaju i jakości odbieranego obrazu. Właściwy dobór tych parametrów może radykalnie zmienić nakład pracy konieczny do rozpoznania obrazu. Na przykład celowe złe ustawienie ostrości może zastąpić programowy filtr uśredniający, a zmiana wysokości położenia kamery o kilka centymetrów może znacznie zwiększyć jej zasięg widzenia.

4.2.1. Obiektyw

Obiektyw jest to *„układ optyczny służący do projekcji obrazu pożądanej sceny na powierzchnię światłoczułą przetwornika obrazu”* [13]. Każdy obiektyw charakteryzuje się ogniskową (kątem widzenia), przesłoną i jasnością. Obiektywy dzielimy ze względu na [13]:

kąt widzenia:

- wąskokątne (teleobiektywy), ogniskowa większa od przekątnej przetwornika,
- standardowe, ogniskowa zbliżona do przekątnej przetwornika,
- szerokokątne, ogniskowa mniejsza od przekątnej przetwornika,

rodzaj przysłony:

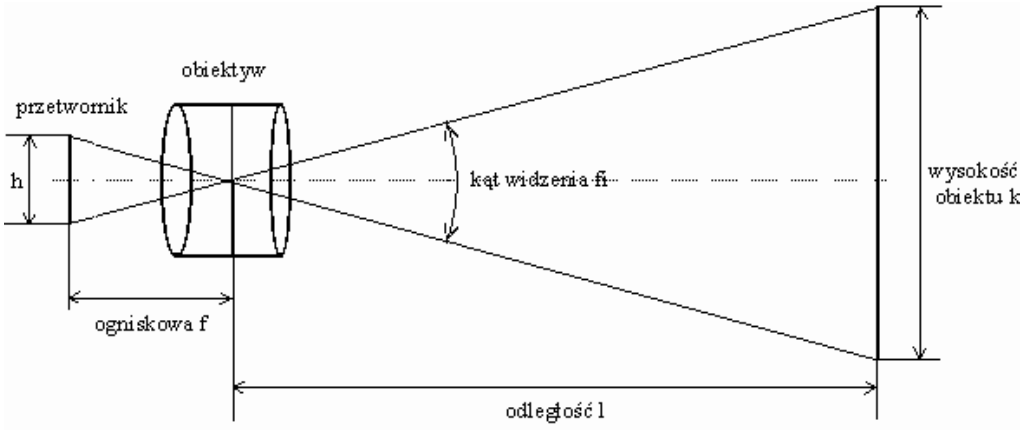
- bez przysłony, tanie, przeznaczone do stosowania w prostych kamerach,
- z przysłoną ręczną,
- z przysłoną automatyczną,

rodzaj ogniskowej:

- stała, produkowane obiektywy mają wartości typowe np. 2,5; 3,6; 4,0; 6,0; 12,0; 16,0; 25,0,
- zmienna (varifocal, zoom), zalecane gdy wielkość obiektu się zmienia lub wymagana jest swoboda doboru ogniskowej, także tu są pewne typowe wartości:3,5-8 (kamery 1/3”); 6-12 (kamery 1/2”) ,
- regulowana zdalnie (tak zwane obiektywy moto-zoom).

Kamera C3088, użyta w tej pracy, jest sprzedawana w zestawie z obiektywem f4.9 mm, F2.8 IR z mocowaniem M12xP0,5. Jest to obiektyw standardowy o niewygórowanych parametrach.

Aby ułatwić pracę systemu wizyjnego zostało założone, iż w polu widzenia kamery zawsze będzie widoczna linia krańcowa, bez względu na położenie robota na ringu. Przeciwnik, w miarę możliwości, nie powinien móc zasłonić pola widzenia. W celu sprawdzenia wykonalności tego założenia konieczne jest ustalenie zasięgu i kąta widzenia kamery.



Rysunek 4.6 Schemat układu optycznego [13]

Na podstawie znanych zależności trygonometrycznych można wyliczyć wzory na kąt widzenia kamery:

$$\varphi_h = 2arctg\left(\frac{h}{2f}\right)$$

$$\varphi_v = 2arctg\left(\frac{v}{2f}\right)$$

Zmienne *h* i *v* to odpowiednio wysokość i szerokość przetwornika kamery, a *f* to ogniskowa. Czyli φ_h to kąt widzenia w pionie, a φ_v w poziomie. Zastosowany w kamerze przetwornik ¼" ma zgodnie z instrukcją [20] wymiary 3,1 na 2,5 mm. Daje to więc:

$$\varphi_h = 2arctg\left(\frac{2{,}5}{9{,}8}\right) = 28{,}6°$$

$$\varphi_v = 2arctg\left(\frac{3{,}1}{9{,}8}\right) = 35{,}1°$$

Jest to mały kąt widzenia. Ring sumo ma średnicę 154 cm, każdy robot nie może przekroczyć wymiarów podstawy 20x20 cm. Poniższa tabela przedstawia zależność odległości, z jakiej przeciwnik zasłoni całe pole widzenia od ogniskowej obiektywu.

Ogniskowa [mm]	Kąt widzenia	Odległość [cm]
1,78	87,4	10,5
2	80,7	11,8
2,1	78,0	12,4
2,5	68,4	14,7
2,8	62,5	16,5
2,9	60,8	17,1
3,5	51,8	20,6
3,6	50,6	21,2
3,7	49,4	21,8
4	46,1	23,5
4,9	38,3	28,8
5	37,6	29,4
6	31,6	35,3
8	24,0	47,1

Tabela 4.4 Zależność minimalnej odległości widzenia przeciwnika od ogniskowej obiektywu

Wartości w powyższej tabelce zostały obliczone dla przetwornika o nieco większych rozmiarach, gdyż w testach okazało się, że podana szerokość 3,1 mm dotyczy obrazu telewizyjnego, a obraz cyfrowy jest jak dla przetwornika o szerokości 3,4 mm.

Z powyższej tabeli wynika, iż standardowy obiektyw umożliwia sprawne celowanie z odległości (od kamery) większej niż 3/2 długości robota. Przy założeniu, że kamera jest zamontowana na środku robota, daje to minimalna odległość celowania równą jednej długości robota. Jest to stanowczo za mały wynik. Dlatego obiektyw został wymieniony na model o mniejszej ogniskowej. Z uwagi na dostępność został wybrany obiektyw 2,1 mm. Ma on kąt widzenia poziomy 78º, a pionowy 58º. Ponieważ ma on ogniskową o połowę mniejszą od przekątnej przetwornika kamery, jest to obiektyw szerokokątny.

Jak widać na powyższej tabeli ogniskowa 2,1 mm dla przetwornika ¼” daje minimalną odległość celowania 12,4 cm, co przy założeniu, iż kamera jest zamontowana centralnie oznacza odległość 2,4 cm od przodu robota. Jest to dopuszczalnie mała odległość, która powinna zapewnić sprawne celowanie.

4.2.2. Położenie kamery

O jakości widzenia na odległość, przy stałej ogniskowej, decyduje sposób zamontowania kamery. Kamera zamontowana wysoko i skierowana w dół lepiej widzi poziome obiekty, jednak ma gorszą perspektywę widzenia obiektów pionowych niż kamera zamontowana nisko i poziomo. Autonomiczny system wizyjny robota mobilnego ma szukać

przecięcia linii (poziomej) z robotem (pionowym), dlatego kąt i wysokość umiejscowienia kamery musi być kompromisem miedzy widzeniem pionowym, a poziomym.

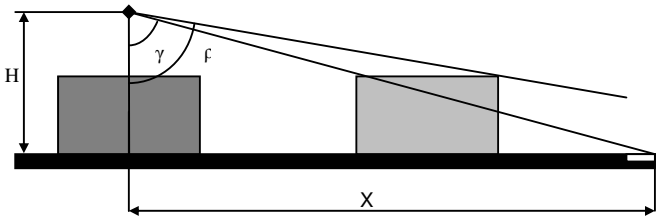
Najważniejszym parametrem jest odległość, z jakiej przeciwnik, nie niższy niż użyty w tej pracy robot (11 cm), będzie optycznie przecinał linię końcową przy zadanej wysokości zamontowania kamery.

$$\rho = \arctan\left(\frac{x'}{H-11}\right)$$

Powyższy wzór służy do obliczenia minimalnego kąta, pod jakim może być widziana tylna krawędź przeciwnika. Parametr x' to minimalna odległość od kamery do tylnej krawędzi przeciwnika. W tym przypadku jest to 30 cm (10+20). Natomiast H to wysokość zamontowania kamery liczona od powierzchni ringu. Wartość 11 to wysokość robota użytego w tej pracy.

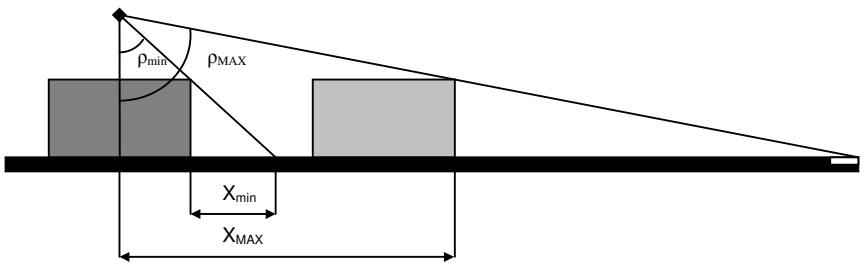
$$\gamma = \arctan\left(\frac{x}{H}\right)$$

Aby ustalić maksymalny zasięg X_{MAX} poprawnego działania, kąt ρ musi być zawsze większy (w obrębie widzenia) niż kąt γ, pod jakim widoczna jest linia końcowa. Ponieważ podczas startu roboty są na ogół ustawiane po środku obu połów ringu, na wprost siebie, zasięg ten powinien być większy niż ¾ średnicy ringu (¾ ·154 = 115 cm). Daje to gwarancję zauważenia przeciwnika już momencie startu. X_{MAX} jest tym większe im niżej umiejscowiona jest kamera.



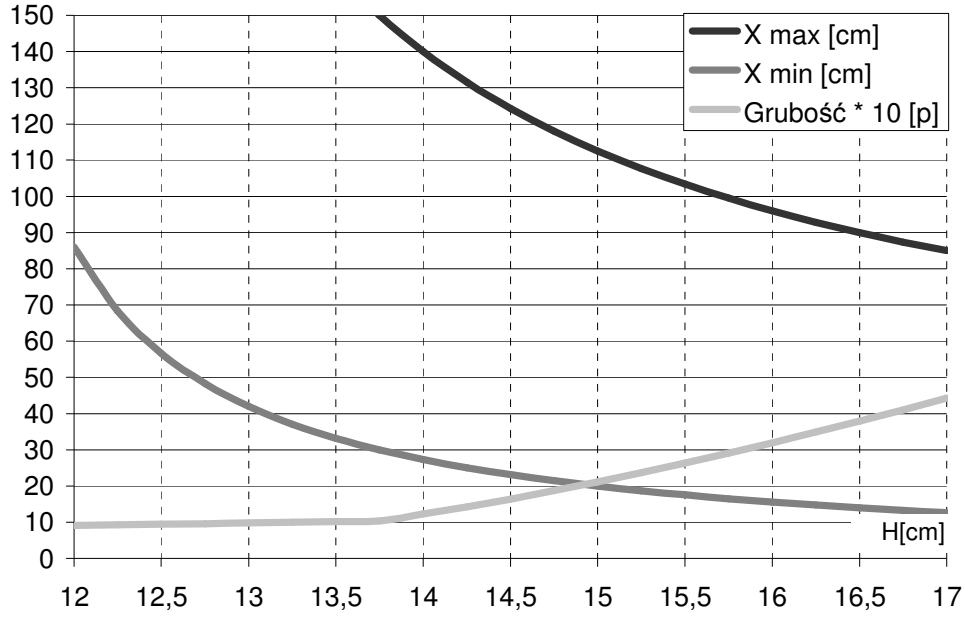
Rysunek 4.7 Ilustracja zasady optycznego przecięcia linii końcowej

Drugim ważnym parametrem jest zakres ślepego obszaru przed robotem X_{min}. Jest to przestrzeń, która jest dla kamery zasłonięta przez czoło robota. Jest ona tym mniejsza im wyżej lub im bliżej przedniej krawędzi jest umiejscowiona kamera.



Rysunek 4.8 Zasięg minimalny i maksymalny widzenia

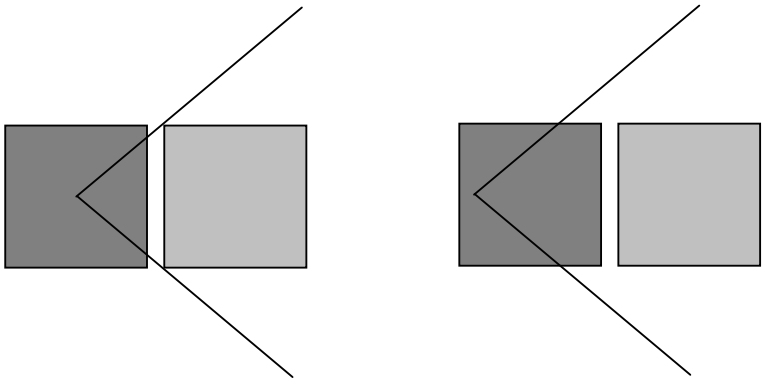
Ostatnim z parametrów zależnych od wysokości jest optyczna grubość linii końcowej, czyli liczbą pikseli przypadającej na nią, gdy znajduje się w najdalszej możliwej pozycji. Ponieważ linia jest namalowana płasko na krawędzi ringu, jest ona najlepiej widziana z góry. Z małej wysokości ulega ona optycznie spłaszczeniu. Aby została ona poprawnie wykryta powinna mieć optyczną grubość, co najmniej 1,5 piksela.



Rysunek 4.9 Zależność zakresu i jakości widzenia od wysokości zamontowania kamery

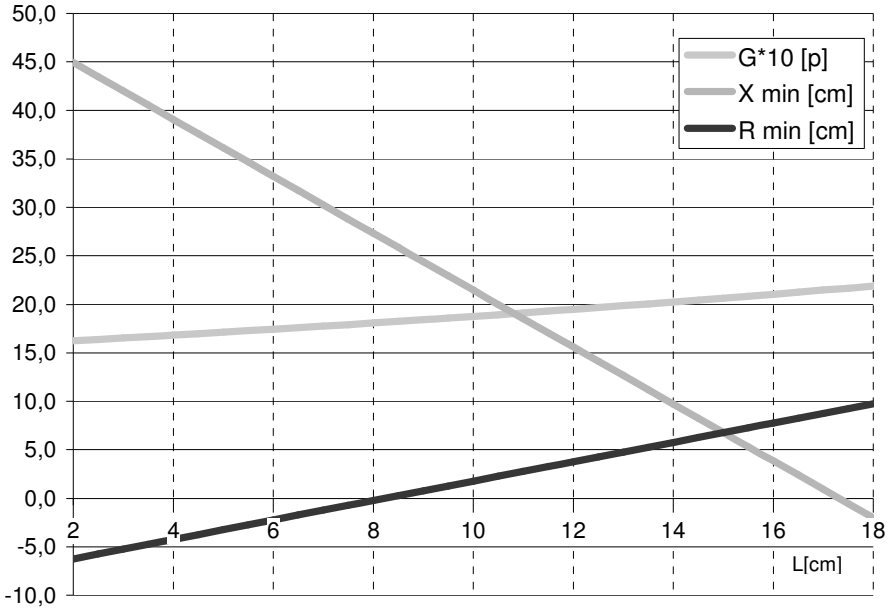
Z powyższego wykresu wynika, iż maksymalny zakres poprawnego działania kamery jest większy od założonego minimum 115 cm dla H mniejszego niż 15 cm. Podczas gdy optyczna grubość linii większa niż 1,5 piksela zaczyna się od H = 14,5 cm. Wynika stąd, że optymalna wysokość zamontowania kamery to 14,75 cm.

Wspomniane było już wcześniej o wpływie umiejscowienia kamery w poziomie na minimalny zasięg widzenia kamery. Ponieważ kąt widzenia i maksymalna szerokość przeciwnika są stałe, minimalna odległość widzenia oponenta jest również stała i wynosi 12,4 cm. Skoro jest to odległość od ogniskowej kamery do przedniej części przeciwnika, przesuwając kamerę w stronę tylnej ściany robota można zwiększyć minimalną odległość między przednią ścianą, a przeciwnikiem R z 12,4 do 2,4 cm. Przy uwzględnieniu wysokości zamontowania kamery ta odległość może być nawet ujemna, czyli przeciwnik nie byłby w stanie zasłonić całego pola widzenia.



Rysunek 4.10 Wpływ przesunięcia kamery na minimalną odległość widzenia

Zatem odległość kamery od tylnej ściany robota L ma wpływ na minimalną odległość widzenia przeciwnika R_{min}, zasięg ślepego pola X_{min} i na optyczną grubość linii, gdyż przesuwając kamerę do przodu zmniejsza się maksymalną odległość do linii końcowej.



Rysunek 4.11 Zależność zakresu i jakości widzenia od położenia kamery w poziomie

Zasięg minimalny widzenia jest ujemny, tj. granica poprawnego widzenia jest nieprzekraczalna, gdy L jest mniejsze niż 8 cm. Pole ślepe robota powinno być możliwie małe. Aby nie pomylić zasłonięcia linii końcowej przez czoło robota, z przecięciem linii przez wypychanego przeciwnika, maksymalna długość tego pola powinna być większa od długości robota, czyli 20 cm. Warunek ten jest spełniony dla L większego niż 10,5 cm. Niestety oznacza to, iż dla założonej wysokości umiejscowienia kamery nie ma optymalnego położenia w poziomie. Kompromisem jest zapewnienie bezbłędnego odczytu przecięcia linii kosztem pogodzenie się z błędem celowania przy małych odległościach, czyli gdy L≈10 cm tj. pośrodku robota.

Przy ustalonej pozycji H = 14,75 cm i L = 10 cm grubość optyczna linii w odległości ¾ średnicy ringu wynosi 1,88 piksela. Ponadto linię tę widać wtedy pod kątem 83º. Przy maksymalnej odległości od kamery, linia jest pod kątem 84º i ma grubość 1,11 piksela. Wynika stąd, iż pochylenie kamery o kącie widzenia pionowego 58º nie powinno być mniejsze niż 55º. Zwiększenie tego kąta zwiększy zakres widzenia w pionie o tło i całą sylwetkę przeciwnika, zmniejszenie tego kąta zredukuje zasięg działania. Ponadto przy tej ustalonej pozycji kamery zarys czoła robota jest widoczny pod kątem 65º, czyli przestanie on być widoczny, gdy odchylenie kamery od pionu przekroczy 94º. Zatem kąt ustawienia kamery powinien być w przedziale od 55 do 94º.

Ostateczna decyzja zależy od konstrukcji przedniej części robota, na którym system wizyjny ma być zainstalowany. Jeśli z ustalonej pozycji dobrze widać po bokach linię warto zachować mały kąt i pewność, że tło nie będzie przeszkadzało w rozpoznawaniu obrazu. Jeśli nic nie widać aż do 65º lepiej zamontować kamerę poziomo (90º). Daje to możliwość obserwacji całego robota przeciwnika z dowolnej odległości. Niestety widoczne jest wtedy również całe tło wraz możliwymi zakłóceniami z niego pochodzącymi.

4.3. Tryb graficzny

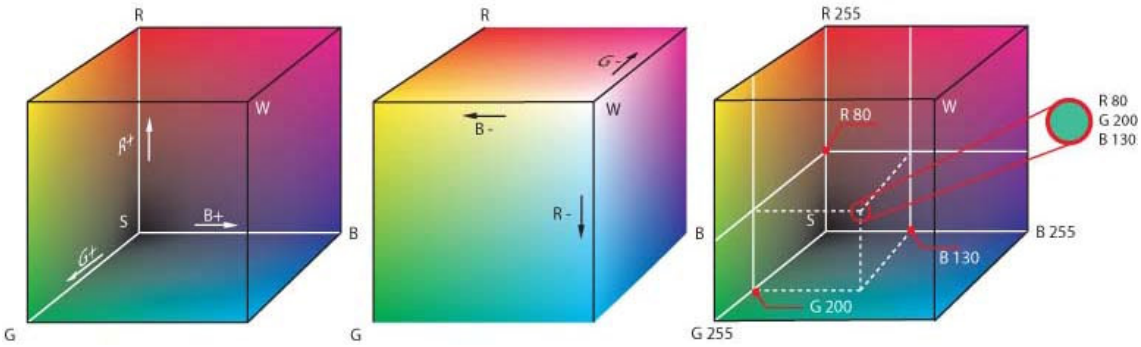
Wybór odpowiedniego trybu graficznego determinuje nakład pracy koniecznej do wykonania danej operacji graficznej. Występuje bardzo wiele trybów (palet), każdy z nich powstał specjalnie na potrzeby konkretnego zadania. Niektóre są na tyle uniwersalne, że wystarczy je nieznacznie zmodyfikować by spełniały wymagania danego projektu.

Podstawowa różnica między nimi dotyczy sposobu reprezentacji kolorów. W grafice komputerowej rozróżnia się trzy podstawowe palety: RGB, HSV i YUV. Wszystkie trzy używają 3 zmiennych do określenia wszystkich dostępnych kolorów i odcieni. Różnią się sposobem ich reprezentacji, czyli stosunkiem ilości składowych chromatycznych do luminescencyjnych i saturacyjnych.

4.3.1. RGB

RGB jest najpopularniejszym sposobem reprezentacji kolorów w grafice komputerowej. Swój sukces tryb ten zawdzięcza prostocie i dosłowności zapisu koloru. RGB to skrót od nazw trzech składowych chromatycznych Red, Green i Blue. Wartość każdej z tych zmiennych odpowiada jasności odpowiadającego jej koloru. Łącząc te trzy podstawowe

kolory można uzyskać dowolny kolor z zakresu widzenia człowieka. Można tą zasadę przedstawić w formie sześcianu RGB.



Rysunek 4.12 Paleta RGB [32]

Ponieważ monitory komputerowe wyświetlają obraz w ten sam sposób, odzwierciedlenie zapisu cyfrowego na ekranie jest dokładne. Niestety śledzenie jednego koloru dla wszystkich jego jasności i nasyceń wymaga analizy trzech składowych jednocześnie.

R	G	B	Zmiana	Kolor
209	52	31	Oryginał	
205	55	35	Nasycenie – 10	
214	48	26	Nasycenie + 10	
201	49	29	Jasność – 5	
218	54	32	Jasność + 5	
209	31	31	Odcień – 5	
209	31	23	Odcień + 5	

Tabela 4.5 Wpływ zmian barwy na składowe RGB

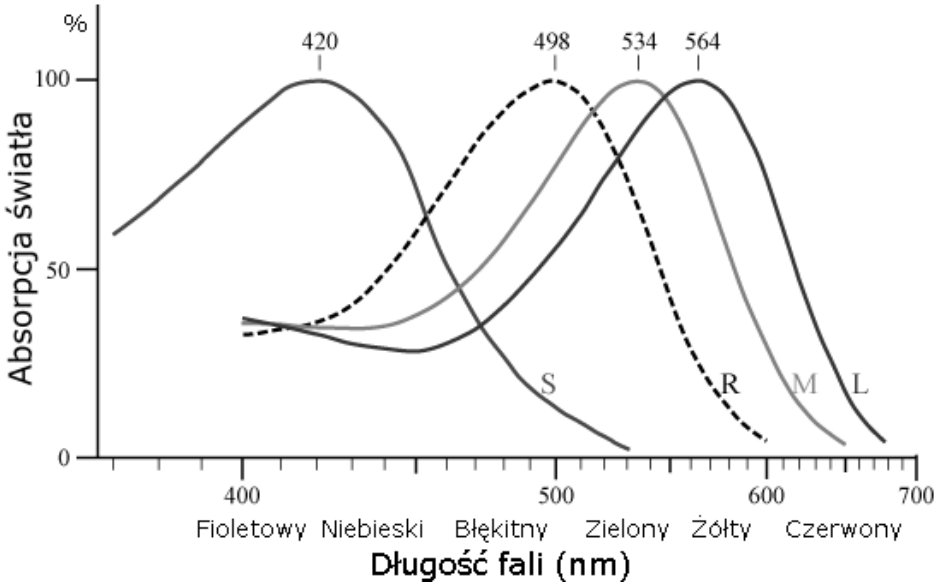
Jak widać na powyższej tabeli, pozornie ten sam kolor, przy niewielkiej zmianie nasycenia, jasności i odcienia, ma bardzo różne współrzędne RGB. Jest to spore utrudnienie przy szukaniu pojedynczej barwy.

4.3.2. YUV

YUV i wiele pochodnych tego systemu powstał dla obsługi telewizji kolorowej. Ponieważ niemożliwe było wymienienie wszystkich odbiorników czarnobiałych na kolorowe konieczne było opracowanie sposobu transmisji obrazu kolorowego wraz z czarnobiałym.

Składowa Y odpowiada ludzkiemu postrzeganiu jasności (luminescencji). Siatkówka oka ludzkiego składa się z trzech rodzajów czopków i jednego rodzaju pręcików. Czopki odpowiadają za widzenie kolorów (S, M, L). Są mało czułe na światło, ale szybko reagują na jego zmianę i dają wyraźny obraz kolorowy. Pręciki są bardzo czułe na światło, ale wolno

reagują na jego zmianę dając przy tym nie ostry obraz monochromatyczny (R). Połączenie bodźców od wszystkich receptorów siatkówki umożliwia barwne widzenia przy dowolnych warunkach oświetleniowych.



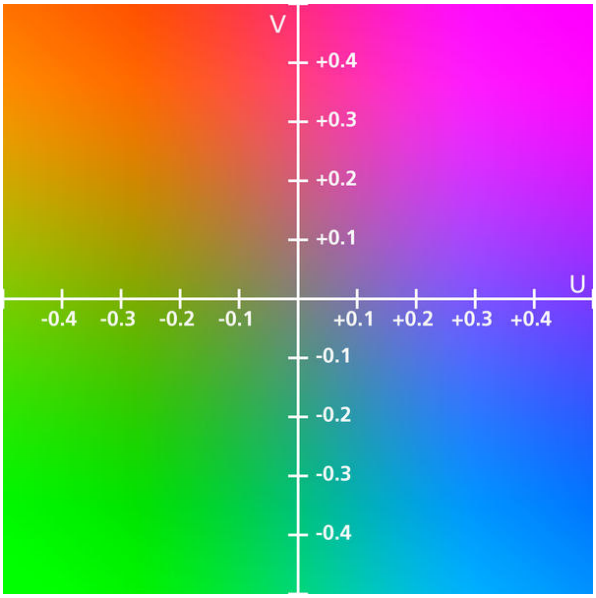
Rysunek 4.13 Względna absorpcja światła przez oko ludzkie [32]

Jak widać na powyższym rysunku widzenie jasności (R) jest bardzo zbliżone do widzenia zieleni. Dlatego w dużym przybliżeniu można uznać składową G obrazu RGB za obraz szarościowy. Aby poprawnie odwzorować jasność należy uwzględnić również pozostałe kolory w stopniu odzwierciedlającym ich wpływ na postrzeganą jasność. W ramach badań instytutu CIE LABS, na kontrolnej grupie ludzi, sprawdzono postrzeganie względnej jasności światła białego i kolorowego, aby ustalić dokładny wpływ koloru na postrzeganie jasności. Ostatecznie uzyskano następujący wzór na jasność:

$$Y = 0{,}299\ R + 0{,}587\ G + 0{,}114\ B$$

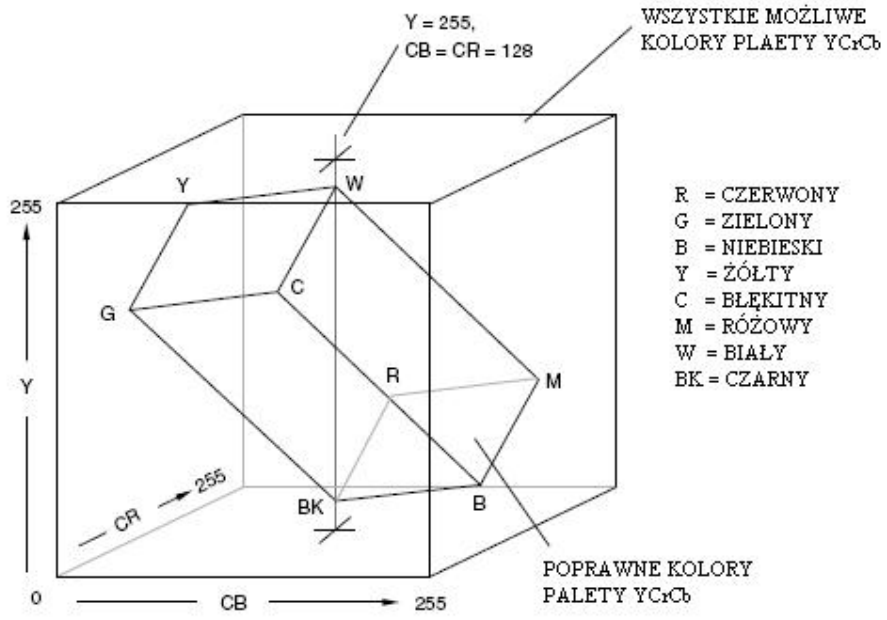
Składowa niebieska ma najmniejszy wpływ na jasność gdyż najmniej jest niebieskich czopków w ludzkiej siatkówce, a ich widmo absorpcji w najmniejszym stopniu pokrywa się z widmem absorpcji pręcików.

Pozostałe dwie składowe palety YUV to tak zwane składowe chromatyczne. Ich zestawienie tworzy paletę barw podstawowych.



Rysunek 4.14 Paleta kolorów UV trybu YUV

Powyższa paleta w zestawieniu ze składową Y tworzy pełną paletę barw RGB [32].



Rysunek 4.15 Nałożenie palety RGB na YCrCb [32]

Dokładny przelicznik składowych RGB na chromatyczne jest zależny od wybranego typu formatu. Typ bazowy to YUV:

$$U = -0{,}147\,R - 0{,}289\,G + 0{,}436\,B$$

$$V = \quad 0{,}615\,R - 0{,}515\,G - 0{,}1\,B$$

Pochodne systemu różnią się nieznacznie mnożnikami. YQI to typ formatu YUV używany powszechnie w systemie telewizyjnym NTSC:

$$Q = 0{,}212\,R - 0{,}523\,G + 0{,}311\,B$$
$$I = 0{,}596\,R - 0{,}275\,G - 0{,}321\,B$$

Inny często spotykany system telewizyjny to PAL używa on zapisu barw YPbPr:

$$Pb = -0{,}169\,R - 0{,}331\,G + 0{,}5\,B$$
$$Pr = 0{,}5\,R - 0{,}419\,G - 0{,}081\,B$$

Ostatni z popularnych trybów to YCbCr zwany też YCrCb. Jest on często spotykanym zamiennikiem YUV w kamerach cyfrowych:

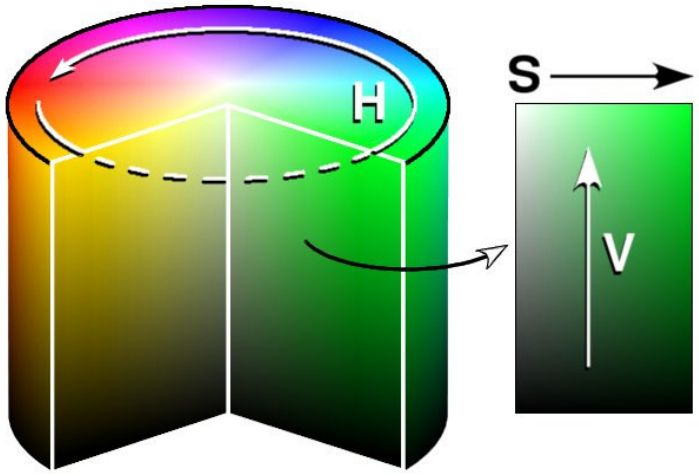
$$Cb = 128 - 0{,}169\,R - 0{,}331\,G + 0{,}5\,B$$
$$Cr = 128 + 0{,}5\,R - 0{,}419\,G + 0{,}081\,B$$

Jak widać jest to znormalizowany do liczb dodatnich tryb YPbPr. Wszystkie różnice w wagach wymienionych powyżej odmian YUV są spowodowane dopasowaniem do parametrów konkretnych odbiorników telewizyjnych bądź subiektywnego odczucia twórców.

Tryb YUV jest łatwy do zastosowania w analizie obrazu gdyż śledzenie jednego koloru przy różnym natężeniu oświetlenia wymaga obserwacji tylko składowych chromatycznych. Poszukiwanie jasnych obszarów wymaga zaś śledzenia tylko składowej luminescencyjnej. Ponadto YUV i jego odmiany są zawsze dostępne wprost zarówno z kamer cyfrowych jak i analogowych.

4.3.3. HSV

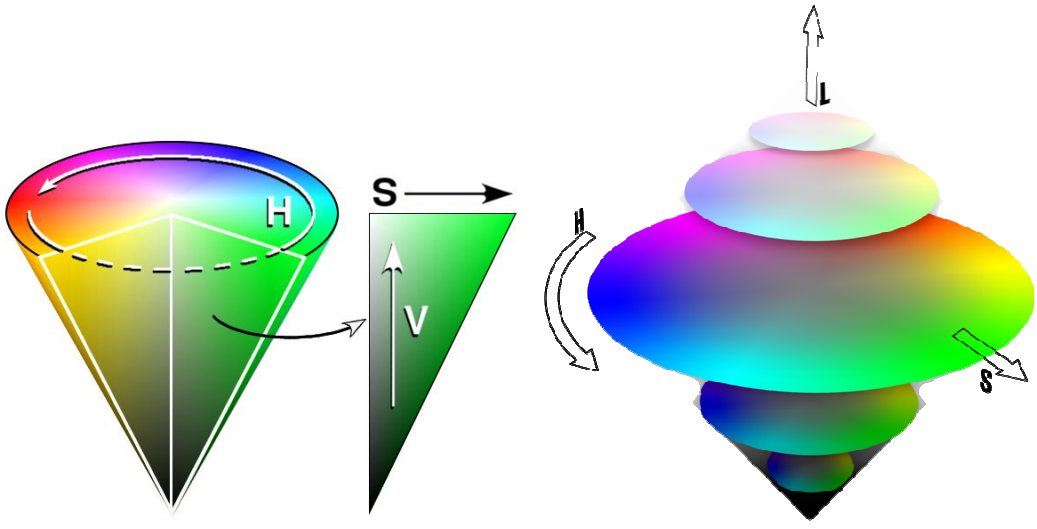
Najciekawszym i zarazem najlepszym, z punktu widzenia analizy obrazu, trybem jest HSV. Każda składowa odpowiada jednej cesze koloru. H od Hue (częstotliwość światła) odzwierciedla barwę w postaci dysku, gdzie H to kąt liczony od barwy czerwonej, po przez 120º czyli zieleń, aż po 240º tj. kolor niebieski. V od Value odpowiada za jasność, a S od Saturation, czyli nasycenie barwy.



Rysunek 4.16 Paleta HSV [32]

System ten pozwala na swobodne dobieranie barw i ich odcieni bez konieczności dokonywania żmudnych przeliczeń bądź poszukiwań. Wszystkie wariacje jednej barwy są dostępne wprost przez zmianę jednej tylko zmiennej. W przypadku poszukiwania jednej barwy należy śledzić tylko jedną zmienną H, tak samo jak przy poszukiwaniu jasnych obiektów V.

Jak widać na powyższym rysunku, dla małych wartości V kolory są optycznie prawie nierozróżnialne. Dlatego powstał okrojony system HSL. Bazuje on na założeniu, iż paleta H jest zdefiniowana dla L = ½ MAX, a nie V = MAX jak to ma miejsce w HSV.



Rysunek 4.17 Przejście z palety HSV na HSL [32]

Zapis HSV jest bardzo wygodny do analizy obrazu, niestety nie jest on dostępny wprost z kamery. Jego obliczenie wymaga przekształcenia składowych RGB przy pomocy tablic lub poniższych wzorów [32]:

$$H = \begin{cases} \text{undefined}, & \text{if } MAX = MIN \\ 60 \times \frac{G-B}{MAX-MIN} + 0, & \text{if } MAX = R \\ & \text{and } G \geq B \\ 60 \times \frac{G-B}{MAX-MIN} + 360, & \text{if } MAX = R \\ & \text{and } G < B \\ 60 \times \frac{B-R}{MAX-MIN} + 120, & \text{if } MAX = G \\ 60 \times \frac{R-G}{MAX-MIN} + 240, & \text{if } MAX = B \end{cases}$$

$$S = \begin{cases} 0, & \text{if } MAX = 0 \\ 1 - \frac{MIN}{MAX}, & \text{otherwise} \end{cases}$$

$$V = MAX$$

lub

$$S = \begin{cases} 0 & \text{if } L = 0 \\ \frac{MAX-MIN}{MAX+MIN} = \frac{MAX-MIN}{2L}, & \text{if } 0 < L \leq \frac{1}{2} \\ \frac{MAX-MIN}{2-(MAX+MIN)} = \frac{MAX-MIN}{2-2L}, & \text{if } L > \frac{1}{2} \end{cases}$$

$$L = \frac{1}{2}(MAX + MIN)$$

Jak wynika z powyższych wzorów, uzyskanie palety HSV lub HSL z RGB jest dosyć czasochłonne. Dlatego tę paletę stosuje się głównie w systemach typu Off-Line lub systemach On-Line o większej wydajności obliczeniowej.

5. Zastosowane algorytmy przetwarzania obrazu

5.1. Istniejące rozwiązania

Istniejące rozwiązania analizy obrazu cechują się bardzo dużą różnorodnością. Dzielą się one głównie na systemy uniwersalne i systemy dedykowane, które zostały napisane z myślą o jednym konkretnym zadaniu. Wszystkie te rozwiązania można scharakteryzować na podstawie kilku podstawowych cech:

- Sposób wyszukiwania obiektów (ruch, typ, lokalizacja znanego)
- Położenie jednostki obliczeniowej (sieć, jednostka stacjonarna, mobilna)
- Tempo analizy (czasu rzeczywistego, szybkie, Off-Line)
- Rodzaj analizy (detekcja, lokalizacja, rozpoznanie)
- Warunki pracy (stałe, znane, nieznane)

Pierwsza cecha determinuje konieczną moc obliczeniową do dalszej pracy. Najprostszym podejściem jest wyszukanie różnicy między dwoma obrazami [31]. Przy takim podejściu każdy obiekt, który zmienił położenie jest od razu wychwytywany przez system. Dalsza wstępna analiza musi już tylko odróżnić przypadkowe zakłócenia od szukanych obiektów. Kolejnym podejściem jest szukanie konkretnego typu obiektów przy pomocy specjalnego filtru. Można na przykład wydzielić w ten sposób kolorowe obiekty z szarego tła [22]. To podejście stosowane jest często w przemyśle z uwagi dużą uniwersalność i precyzję. Udoskonaleniem tej metody jest wyszukiwanie ściśle określonego obiektu. Zdjęcie wzorca porównywane jest z aktualnym kadrem w celu lokalizacji obiektu.

Druga cecha rozwiązuje problem mocy obliczeniowej potrzebnej do danego zadania. Ideałem jest sieć rozproszonych robotów komunikujących się ze sobą, w praktyce takie systemy nie działają jeszcze na masową skalę. Spotkać można natomiast systemy oparte na mobilnych jednostkach wyposażonych w mikroprocesor [5] lub komputer klasy PC. Taki system pozwala na zwiększenie pola działania i niezależności kosztem ograniczenia mocy obliczeniowej. Przy zawężeniu pola działania do zasięgu jednej kamery stosuje się często zewnętrzną stacjonarną jednostkę obliczeniową i zdalnie sterowany element wykonawczy [4]. W przypadku, gdy moc jest niewystarczająca lub potrzebny jest dostęp do większej ilości danych stosuje rozproszone systemy obliczeniowe lub rozproszone bazy danych [26].

Trzecia cecha informuje o tempie pracy sytemu. Zupełnie inaczej konstruuje się systemy, które muszą pracować szybko niż te, które muszą być precyzyjne lub mogą pozwolić sobie na znaczną zwłokę. Można wyróżnić trzy podejścia. Off-Line, czyli analiza zdjęcia dokonywana po fakcie. W wielu przypadkach proces akwizycji obrazu jest bardzo wolny [16] lub nie jest wymagane natychmiastowe podanie wyniku analizy. Wtedy system może przy bardzo rozbudowanych algorytmach analizować pojedyncze zdjęcia nawet przez kilka godzin. Czasem reakcja jest wymagana w czasie nie większym niż kilka-kilkanaście sekund. Ma to miejsce, gdy system współpracuje z człowiekiem lub, gdy na potrzeby danego projektu nie jest wymagana szybka reakcja [10]. Gdy reakcja na zmianę następuje w czasie krótszym niż trwa pobieranie jednego zdjęcia jest to system czasu rzeczywistego (Real-Time)[8]. Taka prędkość jest niezbędna w systemach koordynacji ruchu z przemieszczającym się obiektem. W podanym przykładzie [8] jest to śledzenie kolorowych szybko przemieszczających się obiektów.

Kolejna cecha określa stopień złożoności analizy. W najprostszym przypadku wystarczy określić trajektorię ruchu lub położenie obiektu [10]. Taki system działa z powodzeniem, gdy obiekt poszukiwany znacznie wyróżnia się z jednolitego tła. Jeśli obiekt jest ściśle zdefiniowany można porównać go ze wzorcem [16] w celu znalezienia różnic. Ułatwia to pracę człowiekowi gdyż wyróżnia istotne szczegóły eliminując znane obiekty. Ostatnim podejściem jest rozpoznanie na znalezionym obiekcie określonych cech bądź odczytanie napisów [12]. Odczyt dotyczy zarówno napisów cyfrowych (kod kreskowy), pisma drukowanego [12] jak i pisma odręcznego jak to ma miejsce w programach do odczytu tekstu ze skanera komputerowego.

Ostatnia cecha determinuje utrudnienia w pracy. Jeśli system pracuje w warunkach idealnych tj. przy niezmiennym otoczeniu i braku zakłóceń jego praca sprowadza się do lokalizacji określonych obiektów [31]. Jeśli otoczenie się porusza, ale jest znane [5] system musi najpierw zorientować się w otoczeniu. Robi to na podstawie porównania obiektów charakterystycznych (tzw. markerów) widocznych na zdjęciu ze wzorcem w pamięci. Po zorientowaniu samego siebie dokonuje analizy obrazu, który znajduje się już w znanym, stałym otoczeniu. Najtrudniejszym zdaniem jest znalezienie szukanego obiektu w nieznanym, zmiennym otoczeniu. Można tego dokonać, jeśli program jest na tyle zaawansowany by rozpoznać szukany obiekt na podstawie sposobu przemieszania się [15] lub jakiejś cechy wyróżniającej go z tła [9].

W ramach tej pracy magisterskiej został napisany program poszukujący przeciwnika na ringu Sumo, czyli obiektu o zdefiniowanym typie (obiekt przecina białą linię). Mikroprocesor analizujący obraz znajduje się na pokładzie robota i analizuje obraz na żywo.

Od programu analizującego wymagane jest tylko poprawne znalezienie przeciwnika i określenie jego położenia na osi X. Przy okazji system śledzi białą linię używaną jako marker i zapamiętuje jej położenie na osi Y. Ponieważ robot będzie poruszał się tylko wewnątrz obszaru okolonego białą linią, jest to ruchome, znane otoczenie.

5.2. Akwizycja

W rozdziale 4.1 była mowa o skanowaniu zdjęcia z matrycy kamery do formatu BMP. Ten proces polegał na pobraniu paczki danych i przekazaniu jej dalej poprzez interfejs RS-232 do kamery. Przy takim podejściu tempo skanowania ograniczonej jest od góry przez pojemność bufora pamięci SRAM i wydajność łącza. Podczas pobierania zdjęcia na potrzeby analizy proces skanowania przebiega zupełnie inaczej.

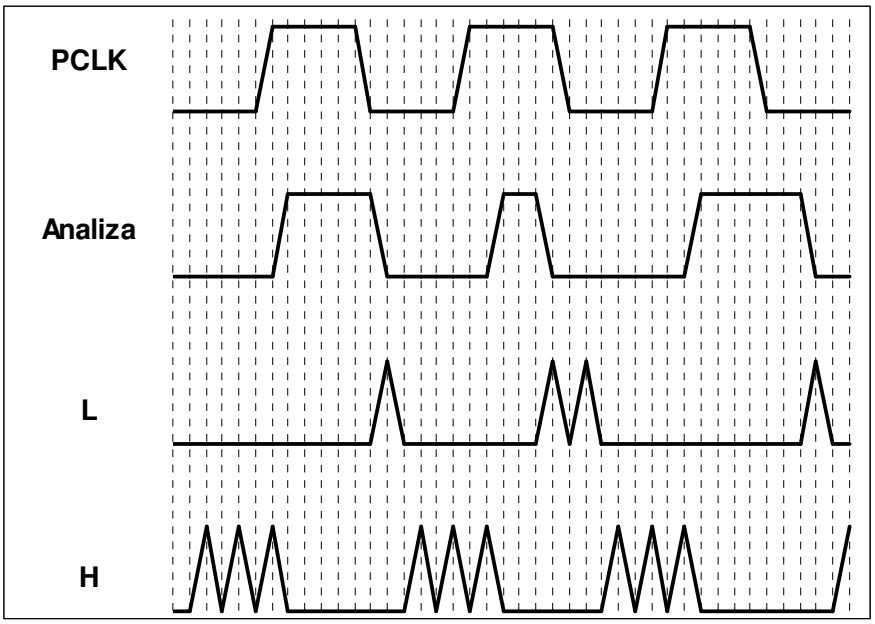
Z uwagi na ograniczoną pamięć podręczną (16 KB) nie jest możliwe pobranie całego zdjęcia w celu jego dalszej obróbki. Dlatego każdy piksel podczas pobierania jest najpierw interpolowany, następnie kwantowany, aby ostatecznie został posegmentowany. Cały proces ma na celu pozyskanie skompresowanego zdjęcia o maksymalnej możliwej rozdzielczości.

5.2.1. Synchronizacja

Kamera pracuje w systemie ciągłym przy własnym, stałym taktowaniu. Pierwszym zadaniem systemu jest dostosowanie się do rytmu przesyłania danych. Częstotliwość taktowania została tak dobrana żeby program akwizycyjny mógł zdążyć pobrać i wstępnie przetworzyć jeden piksel zanim zostanie przesłany następny. Każdy piksel kamera poprzedza wysłaniem sygnału PCLK. Pętla odczytująca czeka na ten sygnał i dopiero po jego nadejściu zezwala na odczytanie kolejnego piksela. Istnieją dwa sposoby synchronizacji z tym sygnałem:

- oczekiwanie na zbocze rosnące
- dopasowanie do stanu wysokiego

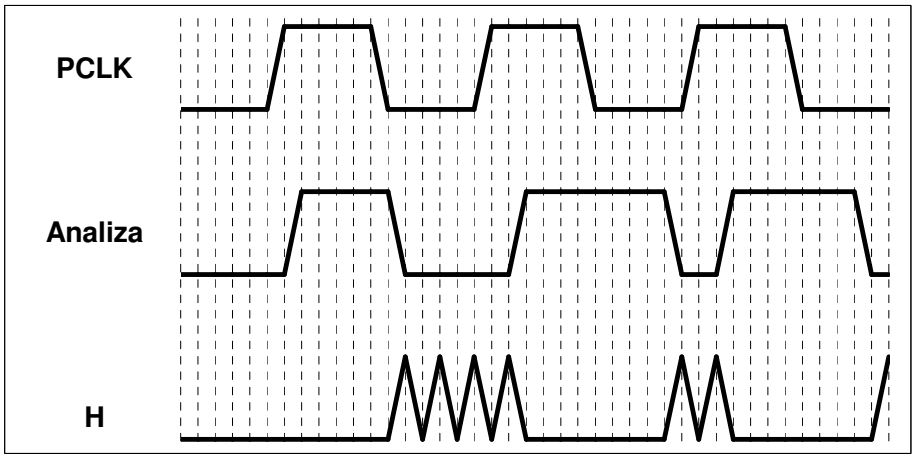
W pierwszym przypadku, aby wykryć zbocze rosnące należy zakończyć odczyt i analizę piksela odpowiednio wcześnie, aby nie pominąć stanu niskiego przed zboczem rosnącym. Przy takim podejściu niemożliwa jest utrata nawet jednego piksela. Odbywa się to jednak kosztem spowolnienia systemu przez zachowanie sporego marginesu czasowego.



Rysunek 5.1 Synchronizacja do zbocza rosnącego PCLK

Stan wysoki przebiegu L odzwierciedla pracę pętli oczekującej na stan niski sygnału PCLK, analogicznie przebieg H pokazuje, kiedy system sprawdza stan wysoki PCLK. Jak widać na powyższym rysunku zaraz po wykryciu L program przechodzi do pętli H. Gdy zostanie wykryty stan wysoki uruchamiana jest Analiza kolejnego piksela. Po jej zakończeniu od nowa program przystępuje do oczekiwania na stan niski.

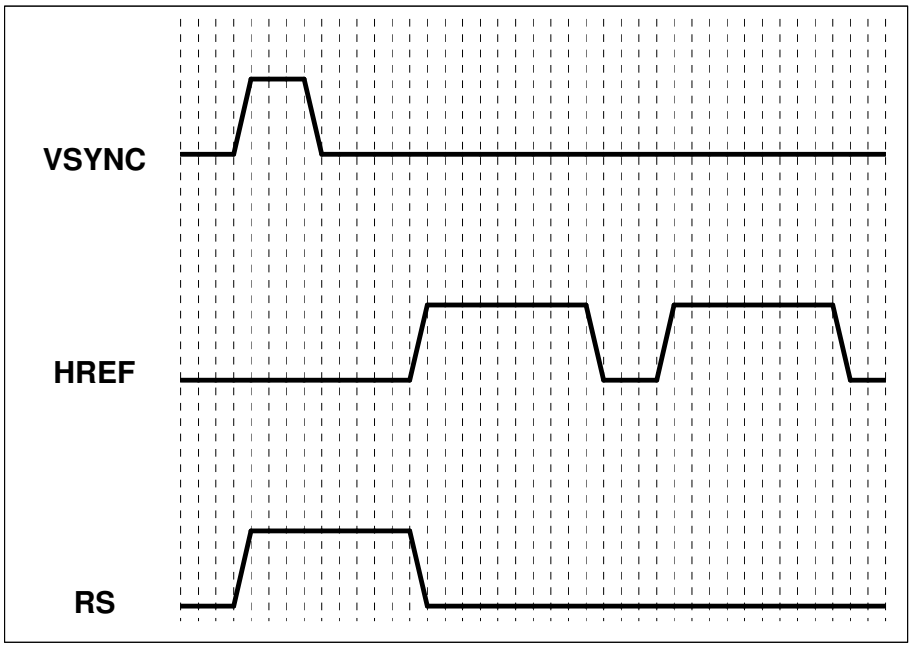
Drugi sposób synchronizujący opiera się na założeniu, że czas odczytu i analizy jednego piksela jest na pewno dłuższy niż pół okresu sygnału PCLK. Jeśli ten warunek zostanie zachowany nie jest konieczne sprawdzanie stanu niskiego, gdyż na pewno jest w momencie zakończenia analizy. Przy takim podejściu istnieje ryzyko, że zostanie utracony piksel, bądź odczytany dwukrotnie, jeśli czas odczytu i analizy jednego piksela będzie krótszy niż pół okresu PCLK lub dłuższy niż jeden okres. Daje to jednak sporą oszczędność czasową.



Rysunek 5.2 Synchronizacja do stanu wysokiego PCLK

Kolejnym sygnałem synchronizacyjnym jest HREF. Sygnał ten jest podawany przez cały czas, gdy przesyłana jest jedna linijka obrazu. Nosi on nazwę synchronizacji poziomej. Ze względów technicznych kamera rezerwuje sobie pewien czas po każdej linijce na swoje procedury wewnętrzne. W tym czasie HREF ma stan niski. System poświęca ten czas na uporządkowanie zapisu końca linijki i przygotowanie się na kolejną linijkę. Na wypadek gdyby synchronizacja PCLK zawiodła, system kończy odczyt kilka pikseli przed końcem linijki by ustrzec się odczytu tzw. śmieci, bądź rozsynchronizowania. Odczyt nowej linijki jest inicjowany dopiero po wykryciu zbocza rosnącego sygnału HREF. W tym wypadku z uwagi na duży zapas czasowy ten sposób synchronizacji jest lepszy.

Ostatnim sygnałem jest VSYNC. Jest o synchronizacja pionowa. Sygnał ten jest podawany w formie stosunkowo krótkiego stanu wysokiego($3*T_{HREF}$), przez pewien czas (ok. $13*T_{HREF}$) przed pierwszym sygnałem HREF. Aby wydłużyć czas na analizę ostatniej klatki i uniknąć zbędnego wyczekiwania na ten sygnał został opracowany na potrzeby tej pracy prosty układ pomocniczy. Przerzutnik RS jest włączany przez sygnał VSYNC, a wyłączany przez HREF. Czyli wydłuża się stan wysoki synchronizacji pionowej do nadejścia pierwszej synchronizacji poziomej. Dzięki temu usprawnieniu wystarczy po skończonej analizie ostatniego zdjęcia sprawdzić czy stan przerzutnika jest wysoki. Jeśli jest to oznacza, że można przygotować się do odczytu kolejnego zdjęcia. Jeśli system się nie wyrobił z analizą zdjęcia podczas przerwy między klatkami, bądź całą analizę zrobił przed sygnałem VSYNC to system czeka na nadejście sygnału VSYNC.



Rysunek 5.3 Sygnał układu wykrywającego synchronizację pionową

5.2.2. Interpolacja

Jak zostało wspomniane w rozdziale 4.1, kamera C3088 jest wyposażona w przetwornik OV6620. Przetwornik ten zawiera matrycę światłoczułą CMOS o rozdzielczości 356x292 pikseli. Jest to rozdzielczość siatki Bayera, czyli fizycznie matryca zawiera tylko 178x146 kompletnych makro-pikseli w formacie GRB 2:1:1. W celu zwiększenia rozdzielczości należy dokonać interpolacji pikseli znajdujących się na granicy makro-pikseli siatki 178x146. Format takiego zapisu nosi nazwę GRB 4:2:2, czyli każda składowa G występuje w 4 sąsiednich polach i analogicznie składowe R i B w dwóch. Dokładnie tak samo działa to w zapisie YCrCb 4:2:2, który wybraliśmy do analizy (patrz 7.3 CAM).

Stosuje się różne metody interpolacji [14]:

- Kwadratowa, gdzie interpoluje się wartości 8 pikseli sąsiednich i centralnego, czyli maskę 3x3. W wersji Bicubic maska ma rozmiary 4x4.
- Liniowa, czyli uzupełnianie brakujących składowych na podstawie sąsiadów położnych na jednej z wybranych linii przechodzących przez piksel środkowy. Często są to dwie lub więcej linii w wersji Bilinear.
- Gradientowa, w której kierunek uśredniania jest zależny od gradientu, tzn. do obliczenia wartości składowych piksela używana jest ta para pikseli, która ma najbardziej zbliżone wartości.
- Kimmela, który to proponuje wagowe uśrednianie kwadratowe.

Ponadto stosuje się różne algorytmy zawierające transformaty Laplace'a, Fouriera i wariacje wszystkich powyższych metod.

Z uwagi na chęć osiągnięcia jak największej wydajności systemu przy małej mocy obliczeniowej użytego procesora został stworzony bardzo uproszczony algorytm interpolacji. Ponieważ system ma wykrywać głównie jasne białe i jasne czerwone obiekty, interpolacja nie musi wiernie odzwierciedlać kolorów. W ramach tej pracy magisterskiej zostały przeprowadzone testy różnych sposobów interpolacji przy pomocy programu RAW (7.3). Okazało się, że wystarczającą dokładność przy minimalnym obciążeniu systemu można osiągnąć przez interpolację kwadratową z maską 2x2 z redukcją składowych chromatycznych.

Dane wejściowe mają postać:

R\K	1	2	3	4	...	353	354	355	356
1	B_{11}	G_{12}	B_{13}	G_{14}		B	G	B	G
2	G_{21}	R_{22}	G_{23}	R_{24}		G	R	G	R
3	B_{31}	G_{32}	B_{33}	G_{34}		B	G	B	G
4	G_{41}	R_{42}	G_{43}	R_{44}		G	R	G	R
5	B_{51}	G_{52}	B_{53}	G_{54}		B	G	B	G
...									
289	B	G	B	G		B	G	B	G
290	G	R	G	R		G	R	G	R
291	B	G	B	G		B	G	B	G
291	G	R	G	R		G	R	G	R

Tabela 5.1 Siatka Bayera kamery OV6620

Linijka	Kanał	n = 1	n = 2	n = 3	n = 4	n = 5
2	Y	Y_{21}	Y_{12}	Y_{23}	Y_{14}	Y_{25}
2	UV	Cb_{11}	Cr_{22}	Cb_{13}	Cr_{24}	Cb_{15}
3	Y	Y_{21}	Y_{32}	Y_{23}	Y_{34}	Y_{25}
3	UV	Cb_{31}	Cr_{22}	Cb_{33}	Cr_{24}	Cb_{35}

Tabela 5. 5.2 Postać danych wyjściowych trybu YG 16bit

Dane wyjściowe:

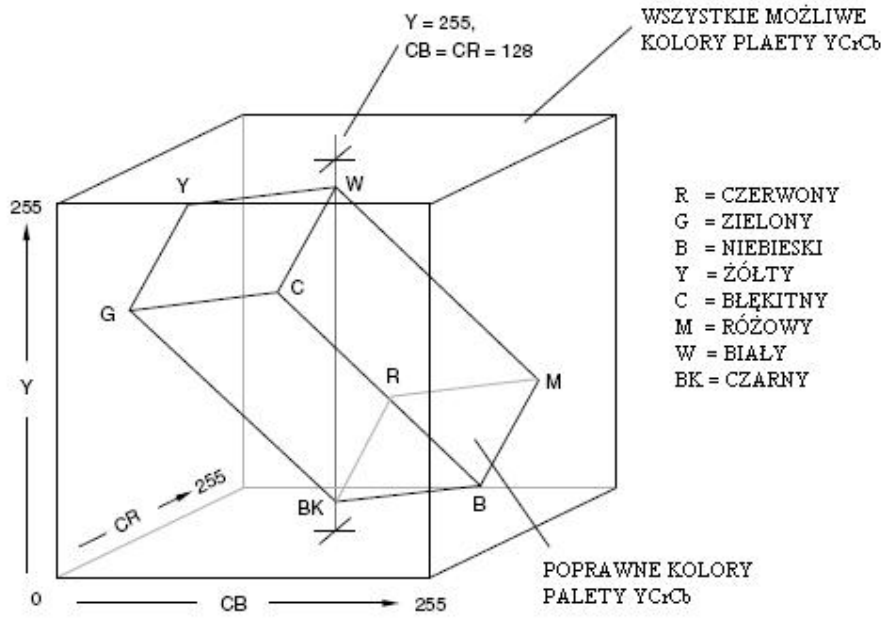
$$Y_n' = \frac{\left(Y_{xn} + Y_{x(n-1)}\right)}{2}$$

$$UV_n' = \frac{\left(UV_{xn} + UV_{x(n-1)}\right)}{2}$$

Czyli dla każdego n Y ma wartość średniej obu składowych Y wewnątrz kwadratu 2x2, a składowa chromatyczna ma wartość średniej z Cb i Cr.

Składowa luminescencyjna przenosi większość informacji o obrazie, więc ma kluczowy wpływ na jakość zdjęcia. Jednak pomimo zredukowania maski z 3x3 do 2x2 spadek rozdzielczości jest niewielki. Zwiększa się tylko rozmycie krawędzi, co w tym przypadku jest tylko zaletą.

Składowa chromatyczna po zsumowaniu osiąga maksimum dla koloru różowego. Kolor ten w zakładanych warunkach pracy robota nie powinien występować w tle, a jego obecność na przeciwniku tylko by ułatwiła jego wykrycie. Nieco niższe wartości suma osiąga dla koloru czerwonego i fioletowego. Kolor jasnoczerwony jest poszukiwanym przez system światłem lasera pokładowego. Kolor fioletowy, co prawda może wystąpić w tle, ale z uwagi na małą jasność tego koloru w formacie YCrCb zostanie on pominięty przez progowanie składowej Y.



Rysunek 5.4 Nałożenie palety RGB na YCrCb [32]

Można zauważyć, że najlepszą identyfikację koloru czerwonego da się uzyskać przy odczycie samej składowej Cr. Jednak dla uproszczenia programu został przyjęty zapis obarczony dopuszczalnym błędem.

5.2.3. Kwantyzacja i segmentacja

Obraz pierwotny po interpolacji można przedstawić jako macierz 356x292 pikseli o dwóch 8-bitowych parametrach Y i UV. Jest to zbiór 200 KB danych, które w tej postaci stanowią szum informacyjny. W celu dalszej sprawnej analizy należy z tego szumu wydobyć tylko istotne informacje.

Założenia projektu przewidują poszukiwanie białej linii okalającej czarny ring postawiony w znacznej odległości od ścian, przy górnym oświetleniu. W takich warunkach biała linia oraz odbite światło lasera są znacznie jaśniejsze od tła. Z tego powodu najwydajniej można odfiltrować istotne piksele po przez progowanie jasności.

Kwantowanie, czyli redukcja liczby kolorów pozwala oddzielić obiekty o zadanych progami kolorach/jasnościach od tła. Przy pomocy programu symulacyjnego CAM (7.3) udało się ustalić optymalną liczbę progów. Przy jednym progu jasności dostatecznie niskim by cała biała linia stanowiła ciągłość, jasne obiekty z tła były często nieodróżnialne od szukanej linii. Drugi próg jasności, z uwagi na to, że był wyższy, pozwolił wyodrębnić spośród jasnych obiektów te najjaśniejsze. W praktyce przy odpowiednio dobranych progach

(6.3) na większości zdjęć testowych tylko biała linii zawierała punkty przekraczające drugi próg (zwany dalej MaxY)

Reasumując, po kwantowaniu rozmiar macierzy obrazu się nie zmienił. Znikły natomiast wszystkie piksele o jasności mniejszej niż próg dolny (zwany dalej MinY). Macierz zawiera, więc sporo pustych obszarów i (czytając wierszami) odcinków o dwóch jasnościach. Obraz w postaci dużego zbioru pikseli jest trudny do odczytu. Redukcja ilości danych ułatwia analizę. Kolejnym krokiem jest, więc segmentacja.

Ponieważ kamera skanuje obraz poziomo konieczne było zastosowanie segmentacji do postaci odcinków poziomych. Program Kompresja (7.3) pozwolił dopracować sklejanie pikseli. Jak było wspomniane wcześniej system szuka białej linii. Wiadomo też, że zawiera ona piksele o jasności większej niż MinY i co najmniej jeden jaśniejszy niż MaxY. Dlatego podczas zapisu odcinek otrzymuje wartość jasności najjaśniejszego piksela do niego należącego. Kolejnymi informacjami związanymi z segmentem jest jego początek i koniec na osi X oraz indeks (używany dalej podczas grupowania 5.3). Położenie na osi Y jest zapisane w formie ukrytej jako współrzędna tablicy odcinków (Linie).

Po segmentacji obraz ma postać tablicy o wymiarach 286x13x4. Dzięki markerom końca danych przeszukiwanie tej tablicy jest szybkie, a rozmiar całkowity pozwala na umieszczenie jej w pamięci SRAM.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
#200540									Y=0 L1				Y=0 L2			
#200550	Y=0 L3				Y=0 Koniec											
#200560																
#200570													Y=1 Koniec			
#200580																
#200590																
#200600																
#200610	Y=2 L1				Y=2 L2				Y=2 L3				Y=2 L4			
#200620	Y=2 L5				Y=2 L6				Y=2 L7				Y=2 L8			
#200630	Y=2 L9				Y=2 L10				Y=2 L11				Y=2 L12			
#200640	Y=2 Koniec				Y=3 L1				Y=3 L2				Y=3 Koniec			
#200650																
...																
#203F20													Y=285 L1			
#203F30	Y=285 Koniec															
#203F40																
#203F50																

Tabela 5.3 Markery końca linii w tablicy Linie

5.2.4. Alokacja zmiennych

Jak zostało już wielokrotnie wspomniane, głównym problemem przy budowie systemu było szybkie zapisywanie i dostęp do dużych ilości danych w pamięci.

Użyty w pracy moduł MMsam7s wyposażony jest w pamięć szeregową DataFlash 4 MB. Niestety łącze szeregowe jest stanowczo za wolne by pamięć ta mogła być używana do przechowywania aktualnie analizowanego zdjęcia.

Moduł MMsam7s posiada procesor ARM7s z wbudowaną pamięcią równoległą SRAM 16 Kb. Jest ona stosunkowo mała, ale za to bardzo szybka. W czasie, gdy procesor odczytuje 1 bajt z losowo wybranej komórki pamięci szeregowej, można odczytać 56 bajtów z pamięci równoległej. Ponieważ system wizyjny był budowany z nastawieniem głównie na wydajność do przechowywania danych została wybrana pamięć SRAM.

Wynikiem kwantyzacji i segmentacji jest tablica o rozmiarze 286x13x4 bajtów. Wymiary te nie są przypadkowe. Minimalna ilość danych w jednym segmencie to: współrzędne X początku i końca odcinka (2 bajty), informacja o kolorze lub rodzaju segmentu i numer indeksu przydatny przy grupowaniu. Ponadto potrzebna jest informacja o współrzędnej Y. Istnieją dwa sposoby zapisu takich informacji:

- Stały rozmiar tablicy, gdzie numer wiersza tablicy jest współrzędną Y.
- Tablica w formie listy dynamicznej zapisanymi wszystkimi 5 danymi jawnie.

Ponieważ druga opcja nie ma z góry zdefiniowanego rozmiaru i istnieje ryzyko utraty lub nadpisania danych, została wybrana opcja pierwsza.

Druga tablica potrzebna tworzona podczas odczytywania obrazu musi przechowywać informacje o składowej UV. Ponieważ ta część programu została napisana tylko dla jednego typu przeciwnika (5.5 Przypadki) jest ona bardzo uproszczona. Z tego powodu tablica przechowuje tylko informacje o pierwszym i ostatnim wystąpieniu pikseli czerwonych w danej linijce.

Ponadto potrzebna jest jeszcze pamięć na tablicę Bloków, czyli wyniku grupowania (5.3) oraz na stos i pojedyncze zmienne. Zakładane wymiary tabel to:

Linie [Max_L][Max_J][4]

Laser[Max_L][2]

Bloki[Max_B][6]

Do obliczenia stałej Max_J została użyta następująca nierówność:

$$Max\_L \cdot Max\_J \cdot 4 + Max\_L \cdot 2 < 16Kb$$

Po uproszczeniu:

$$(2 \cdot Max\_J + 1) < \frac{8Kb}{Max\_L}$$

Wiadomo, że rozmiar pionowy matrycy kamery to 292, więc Max_L ≤ 292 czyli:

$$2 \cdot Max\_J + 1 < 28.05$$

$$Max\_J < 13.53$$

Dlatego Max_J zostało przyjęte jako 13.

Przestrzeń na zmienne kompilatora powinna znajdować się na początku pamięci, aby ich adresowanie było prostsze. Natomiast stos musi znajdować się na końcu pamięci. Z tego powodu konieczne było zrobienie odstępu od początku pamięci do pierwszej tabeli i umieszczenie drugiej tak by było łatwe jej adresowanie. Zostało przyjęte, że tablica Laser jako mniejsza będzie pierwsza. Współrzędna Y została przenumerowana o zmienną Y_0, aby zrobić odstęp od początku pamięci do początku tablicy przy zachowaniu prostego obliczania adresów. Jako że tablica Linie składa się z komórek 52 bajtowych (13x4), a tablica Laser z 2-bajtowych, odstęp początkowy został ustalony na podstawie wielokrotność tych dwóch liczb, czyli $Y_0 = 26$. Przy takim zapisie tablica Laser zaczyna się od adresu 34h (26x2=52), a tablica Linie od 548h (26x52=1352). Aby zapewnić odstęp od końca pamięci Max_L został zredukowane do 286 linii. Liczba ta została tak dobrana by zapewnić dostatecznie dużo miejsca na stos i uzyskać adresy równe wielokrotności 4, gdyż takie jest domyśle adresowanie (integer = 4b) Ostatecznie mapa pamięci wygląda tak:

SRAM 16Kb	Adres	Zawartość	Rozmiar [b]	Odstęp od początku
	#200000h #200033h	Przestrzeń dla kompilatora	52	0
	#200034h #20026Fh	Laser[286][2]	572	Y_0 * 2
	#200270h #2002D7h	Zmienne systemowe	104	12 * 52
	#2002D8h #200547h	Bloki[104][6]	624	14 * 52
	#200548h #203F5Fh	Linie[286][13][4]	14872	Y_0 * 52
	#203F60h #203FFFh	Stos[40][4]	160	(Max_L+Y_0) * 52

Tabela 5.4 Mapa pamięci SRAM

5.2.5. Opis algorytmu

Zaraz po wywołaniu funkcja Odczyt ładuje do zmiennej MinY wynik kalibracji (czyli dolny próg kwantowania) oraz ustawia Adres_UV na adres na początku tablicy Laser. Jest to konieczne na wstępie, gdyż dalej adres tej tablicy jest już tylko inkrementowany.

W kolejnym kroku program wchodzi w pętlę oczekującą na sygnał synchronizacji pionowej.

Po uzyskaniu sygnału synchronizacyjnego, włączana jest główna pętla programu. Warunkiem jej zakończenia i tym samym zamknięcia programu Odczyt, jest odczytanie całego zdjęcia. Za koniec zdjęcia została uznana linia numer 286. Jak zostało wspomniane wcześniej, współrzędna Y jest przesunięta o 26, więc linia końcowa wypada dla Y = 312. Dla każdej rozpoczętej linijki zdjęcia zerowane są zmienne śledzące zmiany: Ostatni, Koniec_UV oraz współrzędna X. Ponieważ tablica Linie nie jest w całości uzupełniana, a numer wiersza tej tabeli odpowiada numerowi wiersza zdjęcia, kolejnym krokiem jest ustawienie zmiennej Adres_Y na początek odpowiedniego wiersza w tabeli. Adres względem początku pamięci SRAM jest iloczynem Y i długości wiersza, czyli 52.

Następnym etapem jest wejście w pętlę odczytującą jeden wiersz ($X \in <0,312)$). Pierwszym krokiem jest odczytanie danych z kamery. Dane są dostępne po sygnale synchronizacji PCLK, dlatego funkcja wpierw synchronizuje się z tym sygnałem, aby mieć pewność, że odczytuje świeże informacje. Dalej piksel odczytany z portu 32-bitowego rozbijany jest na składową Y i UV. Później piksel jest interpolowany. Proces ten dokładnie zostanie omówiony w podpunkcie 5.2.6.

Kolejnym krokiem jest kwantyzacja połączona ze wstępną segmentacją. Ma ona na celu przyporządkowanie odebranego piksela. Możliwe są 4 rodzaje odebranych pikseli:

- Kolejny element tła : Nowy < MinY, Ostatni < MinY
- Pierwszy element jasnego odcinka : Nowy > MinY, Ostatni < MinY
- Kolejny piksel jasnego odcinka : Nowy > MinY, Ostatni > MinY
- Koniec odcinka : Nowy < MinY, Ostatni < MinY

Jeśli został wykryty początek to program zapisuje do tablicy Linie współrzędną X początku odcinka, a do zmiennej Kolor jasność aktualnego piksela. Następnie przygotowuje się do odczytanie kolejnego piksela, czyli przepisuje zmienną Nowy do Ostatni, inkrementuje X i wraca do pętli odczytującej wiersz.

Jeśli został wykryty koniec odcinka to program zapisuje współrzędną X końca, Kolor i zerowy Indeks do tablicy Linie. Następnie inkrementuje Adres_Y i zeruje zmienną Kolor. Po czym tak jak w poprzednim przypadku i dwóch następnych odczytuje nowy piksel.

Kolejną możliwością jest środek odcinka. W tym przypadku pierwszym krokiem jest sprawdzenie składowej UV, jako że program jest nią zainteresowany tylko w obrębie jasnych obiektów. Funkcja Laser sprawdza czy składowa UV przekracza żądany próg MinUV. Jeśli tak to sprawdza czy w danej linijce było już wykryte światło lasera. Jeśli nie było to zapisuje współrzędną X do zmiennej Początek_UV, po czym w obu przypadkach zapisuje X do zmiennej Koniec_UV. Taki zapis daje pojęcie tylko o pierwszym i ostatnim wystąpieniu lasera w danej linijce, ale w połączeniu z informacją o linii białej jest wystarczający. Następnym krokiem jest aktualizacja zmiennej Kolor. Ponieważ system szuka przede wszystkim najjaśniejszych obiektów, cały segment otrzymuje jasność (Kolor) najjaśniejszego piksela.

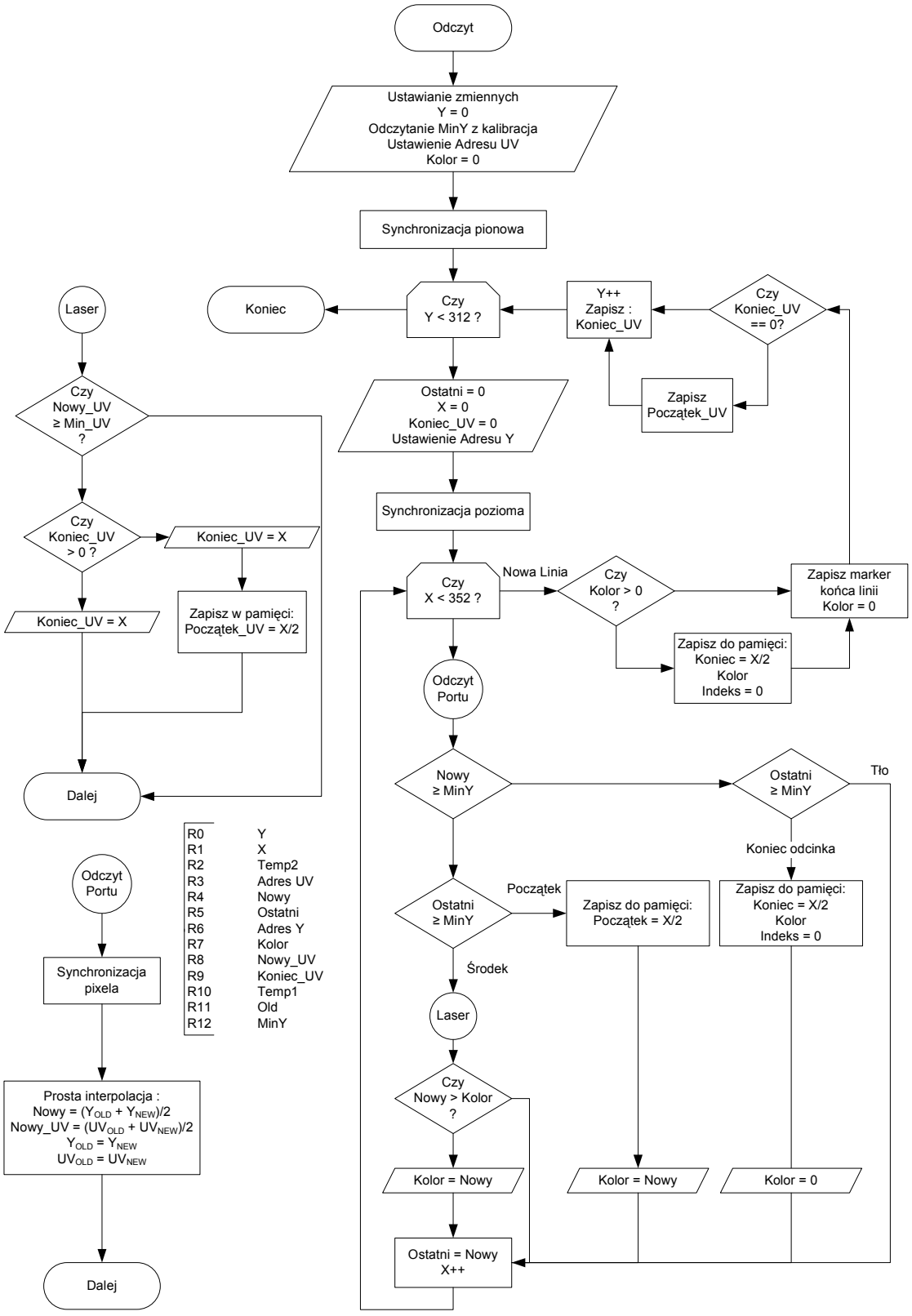
Ostatnim przypadkiem jest Tło. Gdy, ani obecny piksela, ani poprzedni nie są wystarczająco jasne, program przechodzi do kolejnego odczytu.

Pętla odczytująca wiersz kończy się, gdy X = 352, czyli dla bezpieczeństwa tuż przed końcem właściwej linijki. Korzystając z przerwy między końcem jednej, a początkiem drugiej linijki, przesyłanej przez kamerę, system podsumowuje ostatni wiersz.

Pierwszym etapem podsumowania jest sprawdzenie czy ostatni segment został zakończony. Jeżeli nie, to zostaje zapisany koniec odcinka. Następnie system zapisuje w kolejnej komórce tablicy Linie marker końca linii. Ma to na celu unikanie przeszukiwania pustych wierszy tabeli.

Drugim etapem jest podsumowanie segmentu UV. Wpierw, jeśli nie został wykryty laser, program zapisuje Początek_UV, a później Koniec_UV.

Po tych etapach pozostała tylko inkrementacja współrzędnej Y i program wraca na początek głównej pętli.



Rysunek 5.5 Schemat blokowy programu Odczyt.

5.2.6. Optymalizacja kodu w ASM

W większości aplikacji, jak to zostało przedstawione w 5.1, analiza jest albo uproszczona do minimum albo powierzona mocnym komputerom. W ramach tej pracy została przeprowadzona udana próba zwiększenia możliwości analizy obrazu przez prosty, nie graficzny, procesor. Aby to było możliwe konieczne były liczne optymalizacje.

Pierwsze usprawnienie dotyczyło interpolacji. Omówiony wcześniej (5.2.2.) algorytm przewiduje dwie sumy i dwa dzielenia. Ponieważ dzielenia są tylko przez 2 można je było zrealizować po przez przesunięcia binarne, które są nieporównywalnie szybsze. Poniżej jest zapis w formie programu i graficzne przedstawienie rejestrów wpisywanych w danej linijce.

Port = PDSR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	...	2	1	0
Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	UV_7	UV_6	UV_5	UV_4	UV_3	UV_2	UV_1	UV_0	X	X	...	X	X	X

UV_{NEW} = Port >> 16

31	30	29	...	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	...	0	0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	UV_7	UV_6	UV_5	UV_4	UV_3	UV_2	UV_1	UV_0

UV_{NEW} = UV_{NEW} & 0xFF

31	30	29	...	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	...	0	0	0	0	0	0	0	0	0	0	UV_7	UV_6	UV_5	UV_4	UV_3	UV_2	UV_1	UV_0

Y_{NEW} = Port >> 24

31	30	29	...	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	...	0	0	0	0	0	0	0	0	0	0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0

Nowy = (Y_{NEW} + Y_{OLD})>>1

31	30	29	...	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	...	0	0	0	0	0	0	0	0	0	0	Y'_8	Y'_7	Y'_6	Y'_5	Y'_4	Y'_3	Y'_2	Y'_1

Nowy_UV = (UV_{NEW} + UV_{OLD})>>1
Y_{OLD} = Y_{NEW}
UV_{OLD} = UV_{NEW}

W takim zapisie aż 7 rejestrów jest zablokowanych przez stałe w celu obliczenia dwóch wartości. Ponadto cały proces trwa 7 taktów procesora. Jest to pozornie krótki czas, jednak w ostatecznej postaci średni czas potrzebny na odczyt i analizę to 36 taktów, czyli odczyt zajmuje ¼ całego czasu. Dlatego konieczne było usprawnienie tej funkcji. Zmienne UV i Y są 8 bitowe, a procesor jest 32-bitowy. Ponieważ na obu składowych wykonywane są te same operacje, zostały one zespolone.

Port = PDSR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	...	2	1	0
Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	0	UV_7	UV_6	UV_5	UV_4	UV_3	UV_2	UV_1	UV_0	X	X	...	X	X	X

Temp = Old + Port >> 17

31	30	29	...	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	...	0	0	0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	0	UV_7	UV_6	UV_5	UV_4	UV_3	UV_2	UV_1

31	30	29	...	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	...	0	0	Y'_8	Y'_7	Y'_6	Y'_5	Y'_4	Y'_3	Y'_2	Y'_1	UV'_8	UV'_7	UV'_6	UV'_5	UV'_4	UV'_3	UV'_2	UV'_1

Nowy_UV = Temp & 0xFF
Nowy = Temp >> 8
Old = Port >> 17

W pierwotnej wersji port został podzielony przed interpolacją w drugiej dopiero po, w efekcie czego czas potrzebny na wykonanie tej funkcji wynosi 4 takty. Daje to prawie

dwukrotne przyśpieszenie najczęściej używanej funkcji. Przy okazji liczba używanych rejestrów spadła do 5, z czego Temp i Port (Temp2) są używane tylko jako zmienne pomocnicze.

Drugim usprawnieniem była reorganizacja instrukcji warunkowych. Procesor ARM7 fizycznie nie posiada osobnej funkcji warunkowej. Warunki zostały wbudowane w wszystkie funkcje procesora. Dzięki temu zabiegowi od momentu porównanie dwóch zmiennych do kolejnej funkcji testującej wszystkie operacje mogą uzależniać swoje wykonanie od odpowiedniej relacji między zmiennymi. Procesor rozróżnia 11 podstawowych warunków:

EQ	=	równe
NE	≠	różne
CS, GE	≥	większe lub równe
CC, LT	<	mniejsze
HI, GT	>	większe
LS, LE	≤	mniejsze lub równe
MI	< 0	ujemne
PL	≥ 0	dodatnie lub zerowe
VS	> max	przepełnienie
VC	≤ max	brak przepełnienia
ALL	∀	zawsze

Tabela 5.5 Warunki wykonania

Na przykład:

```
CMP     R1,R2
MOVGT   R3,R1
MOVLT   R3,R2
BEQ     Równe
```

Powyższa funkcja porównuje rejestr R1 i R2. Jeśli R1 jest większe od R2 to R3 przyjmuje wartość R1. Jeśli jest mniejsze to R3 przyjmuje wartość R2. Jest to więc funkcja znajdująca maksimum. Na koniec, jeśli R1 było równe R2 to zostanie wykonany skok (Branch) do etykiety „Równe”.

Wykonanie większości funkcji zajmuje od 1 do 2 taktów procesora. Ominięcie pojedynczej funkcji, przy niespełniony warunku, trwa zawsze jeden takt. Instrukcja skoku, która jest nieodzowna w operacja warunkowych, trwa aż 3 takty. Dlatego istotnym jest takie ułożenie warunków, aby newralgiczne dla prędkości funkcje były wykonywane jak najszybciej. Daje to różnicę 2 taktów między operacją, do której się skacze, a tą która występuje wprost po warunku.

Blok wyboru typu piksela składa się z trzech testów, które odróżniają w dwóch krokach jasną linię od tła, stwierdzając przy okazji czy jest to Początek, Środek, czy Koniec_Linii. Wykonanie funkcji Początek trwa 4 takty, Koniec_Linii 8, a Środek od 6 do 10

w zależności od tego czy był wykryty laser czy też nie. Analiza Tła nie zajmuje czasu procesorowi. Ponieważ procesor musi dopasować się do rytmu podawania danych przez kamerę, konieczne jest wyrównanie czasów potrzebnych na analizę różnych pikseli. Zmiana warunku mniejszy niż na większy lub równy MinY pozwala regulować czas wykonania 2 warunków od 4 przez 6 do 8 taktów.

```
CMP R5,R12                    CMP R5,R12
BGE  Koniec_odcinka           BLT  Tło:
Tło:                          Koniec_odcinka:
….                            ....
Koniec_odcinka:               Tło:
....                          ....
```

W pierwszym przypadku wywołanie funkcji Koniec_odcinka trwa 4 takty, a w drugim tylko 2. Jednak w obu przypadkach wykonywany jest ten sam warunek:

jeśli Ostatni<MinY to Tło
jeśli nie to Koniec_odcinka

W połączeniu z testem Nowy<MinY daje to zapowiadane 4 do 8 taktów.

Po wyrównaniu czasów, wraz z warunkami, Początek trwa 10 taktów, Koniec_odcinka 14, a Środek od 10 do 14. Tło jako najszybsze zostało najbardziej zwolnione, z 0 do 8 taktów. Czyli różnica między czasem minimalnym a maksymalnym spadła z 10 do 6 taktów.

Trzecim usprawnieniem było rozdzielenie bloku przygotowania do odczytu kolejnego piksela. Na schemacie blokowym (rysunek 5.5) jest on przedstawiony jako wspólny element: Ostatni = Nowy; X++. Strzałki wskazują na to, iż wyniki wszystkich warunków kończą się skokiem do tego bloku. Program tak skonstruowany był krótszy i czytelniejszy, jednak wolniejszy gdyż każda funkcja poświęcała 3 takty na skok. W programie zoptymalizowanym ten blok został powielony. Funkcje Początek, Środek, Koniec_odcinka i Tło kończą się kopią tego bloku zamiast skokiem do niego. Dało to oszczędność kolejnych 3 taktów.

Czwarte i zarazem najważniejsze usprawnienie dotyczyło wykorzystania rejestrów. Procesor ARM7s posiada ponad 30 rejestrów. Niestety większość z nich pełni specjalne funkcje lub jest dostępna tylko w pewnych wyjątkowych momentach. Na przykład podczas wykonywania przerwania. Do dyspozycji użytkownika pozostaje tylko 17 rejestrów. Rejestry od R0 do R12 służą do wykonywania operacji arytmetycznych i przechowywania zmiennych. Rejestr PC (R17) wskazuje na aktualnie wykonywaną linijkę programu, jego zmiana powoduje skok wewnątrz programu pod wskazany adres. LR(R14) jest używany przez niektóre operacje skoku jako adres powrotny. Teoretycznie może być używany jak rejestr

arytmetyczny, jednak przy zachowaniu uwagi na wspomniane operacje skoku z powrotem. Często używany, ale nie jawnie CPSR(R15) przechowuje wyniki testów logicznych i kody błędów. Od jego zawartości funkcje warunkowe uzależniają swoje wykonanie. Ostatnim z ogólnie dostępnych rejestrów jest SP(R13). Stos jest miejscem gdzie przechowywane są chwilowo nie potrzebne dane, aby zwolnić pozostałe rejestry. SP wskazuje na ostatnio dołożoną wartość do stosu i jest inkrementowany przy każdym dodaniu zmiennej, a dekrementowany przy jej odczycie.

Procesor o architekturze ARM7TDI może wykonywać wszystkie operacje tylko na rejestrach, które są 32-bitowe. Wyjątkiem są operacje na danych liczbowych 8-bitowych i odczyt/zapis z pamięci, ale nawet wtedy adres źródłowy lub docelowy musi być podany za pośrednictwem rejestru. Dlatego należało rozdzielić zmienne między wszystkie dostępne rejestry, tak by system osiągną największą wydajność i nie marnował czasu na każdorazowe odczytywanie danych z pamięci SRAM. Drogą eliminacji rejestry zostały rozporządzone w sposób następujący:

R0	Y
R1	X
R2	Temp2
R3	Adres_UV
R4	Nowy_Y
R5	Ostatni
R6	Adres_Y
R7	Kolor
R8	Nowy_UV
R9	Koniec_UV
R10	Temp1
R11	Old
R12	MinY

Pierwsze dwa rejestry odpowiadają za indeksy pętli. Ponieważ w każdym przebiegu pętli są one konieczne, zostały jako pierwsze rozlokowane w rejestrach. Zmienne Temp1(R10) i Temp2(R2) przechowują tymczasowo wyniki częściowe różnych obliczeń. Nowy_Y(R4) i Nowy_UV(R8) zawierają odpowiednio składową Y i UV aktualnie analizowanego piksela. Ostatni(R5) przechowuje ostatnią wartość Y piksela, a Old poprzednią wartość portu, konieczną do interpolacji. Zmienna Kolor(R7) zawiera maksymalny kolor (jasność) aktualnego segmentu, bądź informuje o braku otwartego

segmentu. Analogicznie Koniec_UV przechowuje pozycje końca segmentu UV lub informuje o jego braku w linijce. Zmienne adresowe Adres_Y(R6) i Adres_UV(R3) pozwalają uniknąć każdorazowego obliczania adresów edytowanych komórek tablic, a kontrolowana inkrementacji (która jest szybsza) umożliwia zredukować do absolutnego minimum obliczanie adresów. Ostatnim rejestrem jest R12, czyli MinY. Zmienna ta jest, co prawda, dostępna w pamięci SRAM, ale każdorazowa operacja odczytu trwa w sumie aż 5 taktów, więc umieszczenie jej dodatkowo w rejestrze daje znaczne przyśpieszenie.

Aby udowodnić skuteczność, wymyślonych na potrzeby tej pracy, technik optymalizacji został przeprowadzony test porównawczy. Program w C++, napisany dosłownie na podstawie algorytmu przedstawionego na rysunku 5.5, został skompilowany do Asemblera przy pomocy programu IAR [13]. Następnie włączano kolejne stopnie automatycznej optymalizacji i sprawdzono czas (w taktach procesora) trwania odczytu i analizy różnych typów pikseli.

Rodzaj piksela	Bez Optymalizacji	Średnia Optymalizacja	Maksymalna Optymalizacja	Omawiana technika
Tło	36	33	27	17
Początek segmentu	52	35	30	19
Środek segmentu:				
Bez Lasera	46	38	32	19
Początek Lasera	55	41	35	21
Środek Lasera	48	40	32	20
Koniec segmentu	83	38	32	21
Granica stabilności	67	41	33	21
Delta	47	8	8	4
Maksimum	83	41	35	21

Tabela 5.6 Porównanie ilości taktów potrzebnych na analizę przy różnych technikach optymalizacji

Wyniki testu omawianej techniki optymalizacji różnią się od obliczonych, gdyż program symulacyjny nie jest dokładny. Ponieważ błędy symulatora są systematyczne, a testy wszystkich 4 programów zostały przeprowadzone w ten sam sposób, relacje między wynikami można uznać za wiarygodne.

Z testów wyraźnie wynika, że optymalizacja dedykowana, z wykorzystaniem wszystkich rejestrów, daje rezultaty o 50% lepsze od najlepszej optymalizacji automatycznej. Ponadto optymalizacja dedykowana, w odróżnieniu od automatycznej, zachowała przejrzystość kodu i ciągi logiczne wewnątrz bloków. Wynika stad, że zastosowanie procedur automatycznej optymalizacji i dalsza reorganizacja rejestrów pozwoliłaby uzyskać jeszcze większy stopień optymalizacji. Dalsze prace zostały jednak zaniechane z uwagi na otrzymanie zadawalającej prędkości (5.2.7).

Granica stabilność wymieniona w powyższej tabelce została oszacowana na podstawie maksymalnego czasu potrzebnego na odczyt i analizę pojedynczego piksela i najwolniejszej funkcji, która może po nim wystąpić. Aby nie wypaść z synchronizmu procesor nie może zgubić sygnału PCLK, czyli nie może zgubić piksela. Sygnał PCLK jest przebiegiem prostokątnym o wypełnieniu 50%. Żeby uniknąć przegapienia stanu wysokiego, czas najdłuższego procesu nie może przekroczyć 3/2 okresu, a suma obu procesów musi być mniejsza od dwóch okresów.

Dla przykładu, przy optymalizacji średniej najwolniejszy jest odczyt piksela ze środka segmentu przy pierwszym wystąpieniu lasera w linijce (41). Po tym zdarzeniu może nastąpić:

- Ciąg dalszy segmentu z laserem (40)
- Ciąg dalszy segmentu, ale bez lasera (38)
- Koniec segmentu (38)

Suma najwolniejszych to 81 taktów, czyli granica stabilności to 41 (40,5).

5.2.7. Analiza prędkości

Aby dostosować rytm pracy kamery do maksymalnego tempa odczytywania przez mikroprocesor, należało najpierw oszacować możliwości systemu. Wymieniane w poprzednim podpunkcie czasy były wynikiem symulacji w debuggerze lub były sumą tylko istotnych dla danego zagadnienia funkcji. Cały proces odczytu zgodnie ze schematem z rysunku 5.5 składa się z:

- Sprawdzenia warunku zakończenia pętli odczytującej wiersz
- Synchronizacji z PCLK
- Odczytu i interpolacji piksela
- Określenia przynależności piksela
- Podjęcia adekwatnych działań w zależności od wyniku poprzedniego testu
- Powrotu do początku pętli

Większość z tych funkcji ma stały czas wykonania wynoszący 18 taktów plus 7, jeśli poprzednia funkcja skończyła się przed stanem wysokim PCLK. Określanie przynależności wraz z odpowiednimi reakcjami wynoszą odpowiednio:

- Tło (8)
- Początek segmentu (10)
- Środek bez Lasera (10)
- Środek z Laserem (13)

- Środek z początkiem Lasera (14)
- Koniec segmentu (14)

Reasumując łączny czas odczytu to od 26 do 32 taktów. Granica stabilności, zgodnie z założeniem z podpunktu poprzedniego, wynosi 31,5 taktu. Kamera jest wyposażona w zegar 8,86 MHz z programowalnym dzielnikiem od 1 do 64. Dzieląc częstotliwość zegara procesora ARM7s (T) przez ustawioną częstotliwość taktowania kamery (PCLK) można określić ile taktów procesora przypadnie na jeden takt kamery, czyli PCLK.

Dzielnik	PCLK[MHz]	T/PCLK	Zdjęcie[ns]	Zdjęć/s	Analiza [µs]
1	8,87	5,4	20	50,0	1816
2	4,43	10,8	41	24,4	3633
3	2,96	16,3	61	16,4	5449
4	2,22	21,7	81	12,3	7266
5	1,77	27,1	102	9,8	9082
6	1,48	32,5	122	8,2	10898
7	1,27	37,9	142	7,0	12715
8	1,11	43,4	163	6,1	14531
9	0,99	48,8	183	5,5	16348
10	0,89	54,2	203	4,9	18164
11	0,81	59,6	223	4,5	19981
12	0,74	65,0	244	4,1	21797
13	0,68	70,5	264	3,8	23613
14	0,63	75,9	284	3,5	25430
15	0,59	81,3	305	3,3	27246
16	0,55	86,7	325	3,1	29063

Tabela 5.5. 7 Prędkości odczytu jednego zdjęcia

Czas zrobienia całego zdjęcia został oszacowany na podstawie pomiaru rytmu pracy kamery. Każda linijka składała się z 356 pikseli, które były podawane w czasie, gdy HREF miał stan wysoki. Po nim następowała przerwa trwająca średnio tyle, co 214 pikseli (PCLK). Każda klatka była poprzedzona sygnałem VSYNC trwającym mniej więcej tyle, co 3 takty HREF i przerwą o długości 13 taktów HREF. Po zsumowaniu tych wszystkich czasów okazało się, że ze skanowanie całego zdjęcia o wymiarach 356x292 trwa tyle, co 570x316 taktów PCLK. Wynik ten podzielony przez częstotliwość taktowania kamery daje czas potrzebny na zrobienie zdjęcia. Wartość kolumny Analiza jest czasem wolnym między końcem danych z jednego zdjęcia, początkiem danych kolejnego. Czas ten będzie potrzebny do oszacowania możliwości wyrobienia się systemu z analizą między jednym, a drugim zdjęciem. Szczegóły tego procesu zostaną omówione w podpunkcie 0.

Z wyliczeń wcześniejszych wynikało, że system potrzebuje, co najmniej 32 takty na jeden piksel i nie więcej niż 52(26x2). Daje to dzielnik od 6 do 9, czyli od 8 do 5 zdjęć na sekundę. Założenia projektu przewidywały osiągnięcie wydajności analizy, co najmniej 1 zdjęcia na sekundę.

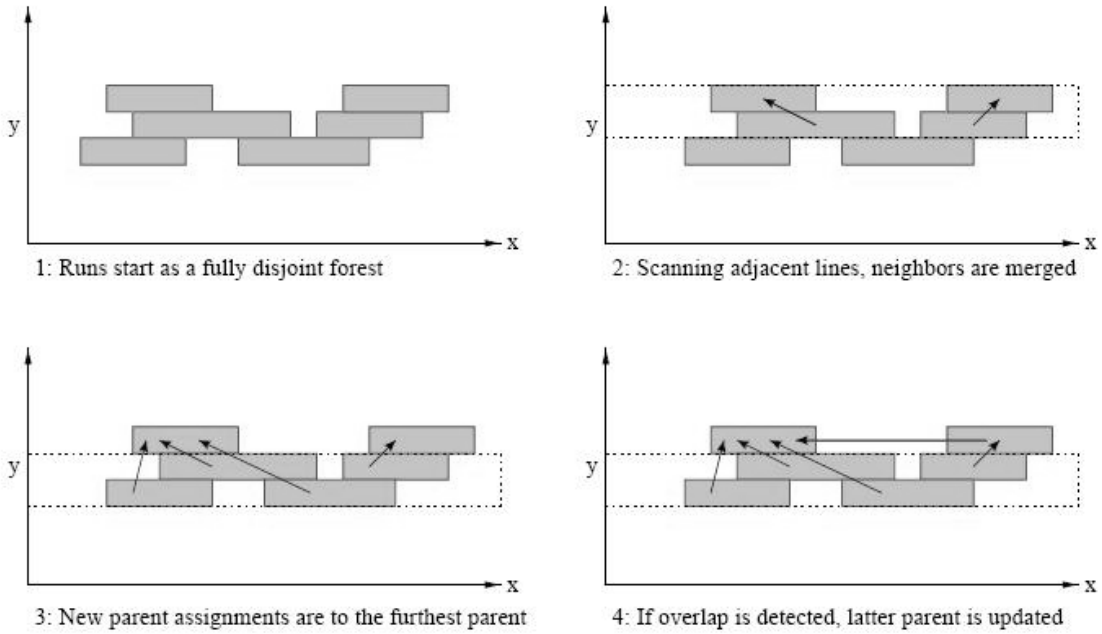
5.3. Grupowanie

5.3.1. Idea

Oryginalne zdjęcie z kamery C3088 składa się 103952 pikseli. Procedura odczytu, opisana w powyższym podpunkcie, zredukowała tę ilość danych do dwóch tablic: Linie i Laser. Tablica Linie składa się z 286 wierszy po 13 komórek. Każda komórka zawiera informacje o początku i końcu jasnego segmentu oraz o jego jasności i numerze indeksu. Wszystkie zapisane w ten sposób segmenty są podejrzane o bycie szukaną białą linią. Są one zapisane w formie zbioru poziomych, jednowymiarowych odcinków. Aby możliwa była ich sprawna interpretacja muszą zostać wpierw zespolone w pionie by utworzyły dwuwymiarowe bloki. Proces ten został nazwany Grupowaniem.

5.3.2. Sąsiedztwo

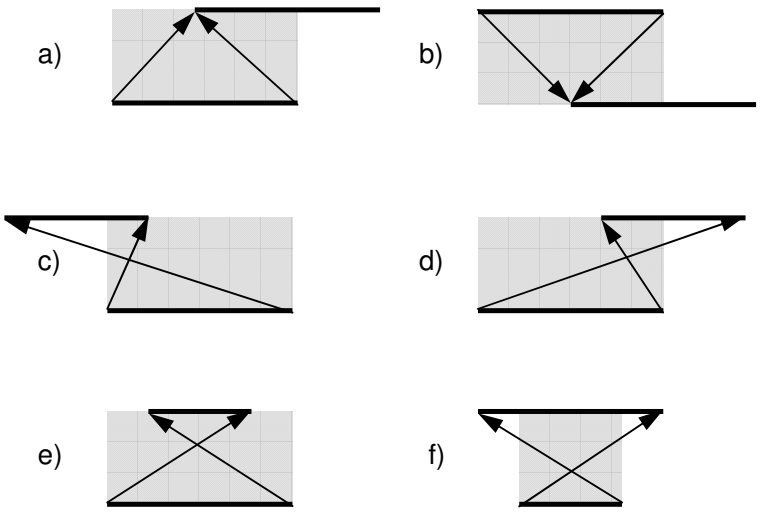
Wzorem do skonstruowania funkcji grupującej był popularny, uniwersalny moduł analizy obrazu CMVison [6]. Program ten w pierwszym etapie swojej pracy skleja poziomo piksele tworząc segmenty tzw. Runy. Każdy Run zawiera informacje o położeniu na osi Y, swoim początku na osi X i długości. Ponadto przechowuje uśredniony kolor segmentu i numer indeksu zwany rodzicem. Kolejnym etapem jest sklejanie Runów w bloki zwane Regionami. Podczas tego sklejania wybierany jest skrajny górny, prawy Run, zwany Rodzicem i w kolejnych przebiegach funkcji sąsiadujące z nim Runy otrzymują jego indeks. Runy oddzielone w poziomie przez przerwę, jeśli mają pośrednią styczność, zostają również zespolone. W ten sposób grupa sąsiadujących segmentów dziedziczy po Rodzicu numer indeksu, który wskazuje na Region. Każdy Region zawiera informacje o kolorze, skrajnych współrzędnych x i y, polu powierzchni, położeniu środka regionu oraz indeks własny i runu początkowego.



Rysunek 5.6 Przykład sposobu sklejania Runów przez program CMVison [7]

Technika zastosowana w CMVision daje bardzo dobre rezultaty przy pracy na komputerach klasy PC. W ramach pracy magisterskiej na Uniwersytecie Carnegi Mellon w Pittsburgu [7], zespołowi studentów na komputerze z procesorem Pentium®3 700MHz udało się uzyskać wydajność 30 zdjęć na sekundę przy rozdzielczości 640x480 i rozpoznawaniu 32 kolorów. Niestety algorytmy CMVision wymagają dostępu do bardzo dużych ilości pamięci operacyjnej, a podana wydajność nie zawierała czasu potrzebnego na pobranie zdjęcia, gdyż odbywało się to równolegle za pośrednictwem Frame Grabbera.

W ramach tej pracy został napisany podobny algorytm zawierający usprawnienia i znaczne uproszczenia.



Rysunek 5.7 Schematy analizy sąsiedztwa.

Program CMVision przewiduje dwa przypadki sąsiedztwa. Pierwszy, gdy początek górnego odcinka a) znajduje się wewnątrz obszaru nad odcinkiem dolnym i drugi, gdy początek dolnego odcinka b) znajduje się wewnątrz obszaru pod górnym odcinkiem. Sprawdzenie tej zależności wymaga każdorazowo wykonania dwóch sum arytmetycznych, dwóch iloczynów logicznych, czterech porównań i jednej sumy logicznej.

$$\begin{array}{l}(R2_{MIN} \le R1_{MIN}) \& (R1_{MIN} < (R2_{MIN} + R2_{WIDTH})) \quad LUB \\ (R1_{MIN} \le R2_{MIN}) \& (R2_{MIN} < (R1_{MIN} + R1_{WIDTH}))\end{array}$$

Ten skomplikowany system testu można zastąpić jego wersją uproszczoną, a tak samo skuteczną.

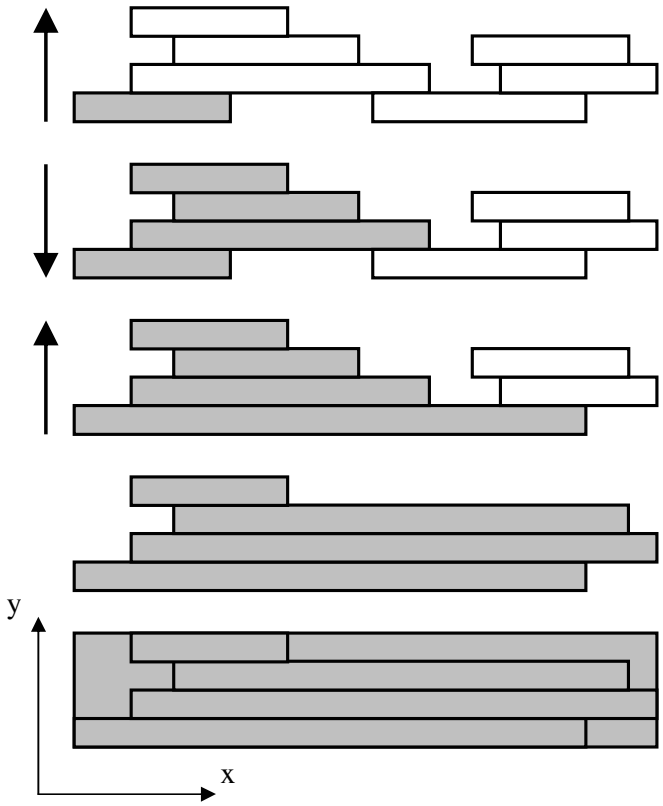
Odcinki sąsiadują ze sobą, jeśli między ich współrzędnymi X występuje część wspólna. Program CMVision wyszukiwał osobno czy początek jednego odcinka zawiera się w drugim lub odwrotnie, czy początek drugiego zawiera się w odcinku pierwszym. Można wykazać, że jeżeli początek dolnego odcinka jest położony wcześniej niż koniec górnego i równocześnie koniec dolnego odcinka jest dalej niż początek górnego, odcinki te mają część wspólną.

Jeśli odcinek górny jest przesunięty w lewo c) względem dolnego to dopóki jego koniec jest za początkiem dolnego, mają one część wspólną. Drugi warunek jest wtedy zawsze spełniony, gdyż początek górnego jest znacznie wcześniej niż początek dolnego, a tym bardziej niż jego koniec, który jest znacznie dalej. W przypadku przeciwnym, gdy odcinek górny jest przesunięty w prawo d) dopóki jego początek jest przed końcem dolnego mają one część wspólną. Analogicznie, drugi warunek jest zawsze spełniony. Jeśli odcinek górny w całości zawiera się dolnym e), oba warunki są spełnione, podobnie jak w przypadku przeciwnym f).

$$(R2_{MIN} \le R1_{MAX}) \& (R2_{MAX} \ge R1_{MIN})$$

Ponadto program CMVision przewiduje jednorazowe przeszukanie całej tablicy Runów i stopniowe aktualizowanie zespolonych grup Runów (rysunek 5.6). Zapis w formie listy i operacje na adresach w przypadku procesora ARM7s się nie sprawdzają. Program napisany na potrzeby tej pracy został więc usprawniony. Zamiast stopniowego rozrastania i spajania małych bloku program Grupuj od razu stara się stworzyć jeden duży blok. W tym celu począwszy od pierwszego znalezionego bloku, przeszukuje tablicę w górę, aż do momentu, gdy nie znajdzie już sąsiedztwa. Następnie od najwyższego sąsiada ponawia przeszukiwanie w dół, aby dokleić sąsiadów, którzy byli odseparowania w poziomie, a

sąsiadowali w pionie. Przeszukiwanie podobnie jak poprzednio kończy na najniższym sąsiedzie. Dla pewności przeszukiwanie w górę jest ponawiane. W trakcie doklejania kolejnych segmentów, jeśli w linijce zostało znalezionych kilka obiektów, pierwszy z nich otrzymuje skrajne wymiary, a pozostałe zostają skasowane. W ten sposób, po każdym doklejeniu redukowana jest liczba segmentów, więc przyspiesza się przeszukiwanie powrotne.



Rysunek 5.8 Schemat sposobu szukania sąsiadów w pionie

Przeszukiwanie według podanego, szybszego algorytmu jest możliwe, ponieważ poszukiwane białe linie są w przybliżeniu poziomymi łukami lub prostokątami. Jest to przykład przewagi programu dedykowanego nad uniwersalnym.

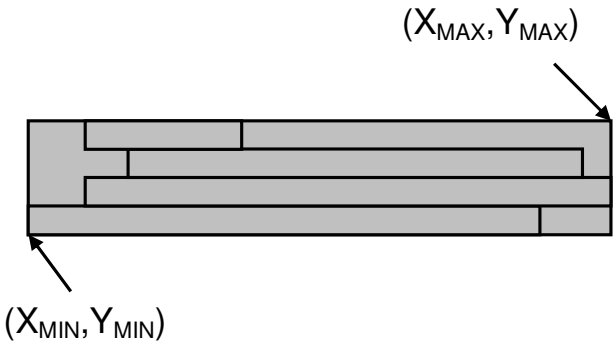
5.3.3. Opis bloku

Jak zostało już wielokrotnie w tej pracy wspomniane, celem pracy autonomicznego systemu wizyjnego robota mobilnego jest zlokalizowanie przeciwnika na ringu sumo. Ring jest okolony przez białą linię. Z uwagi na odpowiednie dobranie obiektywu (4.2) i ustawienie kamery, przeciwnik, jeśli będzie w polu widzenia, na pewno przetnie linię krańcową. Istnieją dwie możliwe sytuacje:

- Przeciwnik jest równie jasny, co biała linia i zlewa się z nią
- Przeciwnik jest ciemniejszy od białej linii i widać jej przecięcie

W pierwszym przypadku, gdy robot jest bardzo jasny, na pewno będzie widać odbicie światła lasera pokładowego na jego powierzchni. Wykrycie przeciwnika sprowadza się wtedy do znalezienia czerwonego odcinka na dużej białej linii. W przypadku drugim robot jest na tyle ciemny, że wyraźnie widać granicę między nim a białą linią. Wtedy przerwa w wykrytej linii jest uznawana za przeciwnika.

Na podstawie powyższych założeń można przyjąć, że system potrzebuje informacje tylko o wymiarach szukanych obiektów i ich lokalizacji. System wizyjny w trakcie zespalania segmentów zapamiętuje ich skrajne współrzędne. Otrzymane w ten sposób wartości X_{MIN}, X_{MAX}, Y_{MIN}, Y_{MAX} wyznaczają współrzędne wierzchołków prostokąta opisanego na połączonych segmentach. Wszystkie te prostokąty, zwane dalej Blokami, są zebrane w tablicy Bloki. Maksymalna ilość bloków wykrywana przez system to 104. Ograniczenie odgórne jest konieczne ze względu na mała ilość pozostałej pamięci SRAM. Ponieważ przeszukanie zdjęcia zaczyna się od dołu, pierwsze wykryte obiekty będą albo szukaną linią, albo fragmentem malowania przeciwnika. Zatem wszystkie nadmiarowe bloki, które mogłyby zostać pominięte, na pewno są elementami tła, więc nie stanowią obiektu zainteresowania systemu.



Rysunek 5.9 Opis bloku

5.3.4. Opis algorytmu

Program Grupuj składa się z dwóch części. Pierwsza część to podwójna pętla przeszukująca cały posegmentowany obrazek w poszukiwaniu nowego jądra (nieużywanego segmentu o jasności powyżej MaxY). Druga część łączy w blok wszystkie segmenty sąsiadujące z nowym jądrem.

Każdy segment podczas segmentowania otrzymuje zerowy indeks, oznacza on obiekt jeszcze nie pogrupowany. Ponadto zapisywana jest jasność najjaśniejszego piksela z

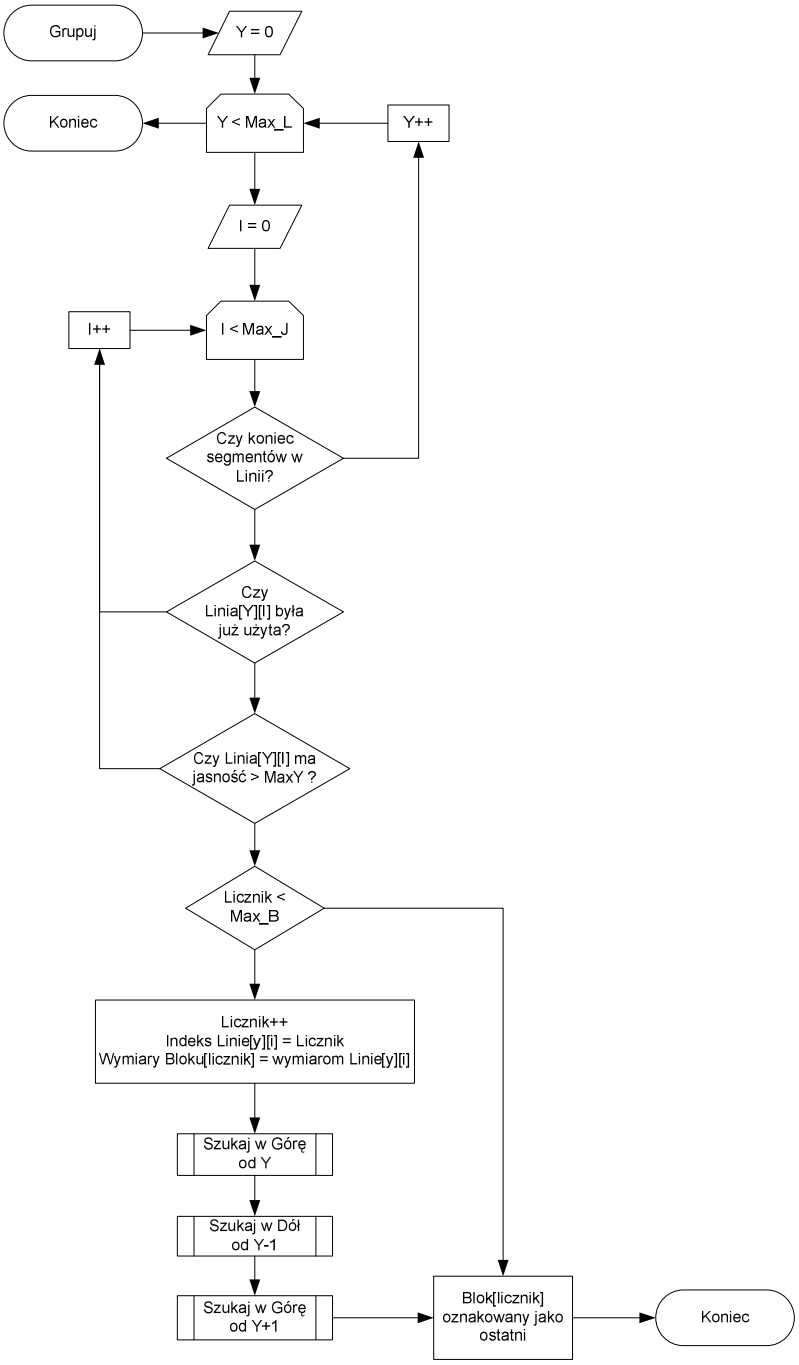
segmentu. Każda odczytana linia kończona jest markerem końca danych, ma to na celu uniknięcie przeszukiwania pustych obszarów tablicy Linie. Pierwsza część programu Grupuj w pierwszej kolejności sprawdza czy nie został już osiągnięty ten marker. Jeśli tak to przechodzi do kolejnej linii, jeśli nie to przechodzi do kolejnych testów. Drugim testem jest sprawdzenie czy segment został już przyłączony do innego bloku. Jeśli jego indeks pozostał zerowy to nie należy on do nikogo, w przeciwnym wypadku ma on numer bloku.

Na koniec sprawdzana jest jasność. Aby lepiej odczytywać obraz, segmenty zostały podzielone na trzy kategorie: tło, jasne obiekty i bardzo jasne obiekty. Te ostatnie najprawdopodobniej są białymi liniami. Niestety ze względu na zmienne warunki oświetlenia i różne zakłócenia linia nie w całości składa się z bardzo jasnych segmentów. Aby uniknąć poszarpania linii, segmenty jasne sklejają segmenty bardzo jasne w bloki. Łączenie w bloki zaczyna się od pierwszego od dołu bardzo jasnego segmentu, czyli jądra.

Po znalezieniu jądra bloku program przechodzi do drugiej części, czyli łączenia segmentów. Każdy nowy blok otrzymuje unikalny, inkrementowany numer. W trzech przebiegach programy Szukaj w Górę i Szukaj w Dół wynajdują kolejne segmenty sąsiadujące z jądrem. Oba te programy po dojściu do ostatniego w danym kierunku obiektu zwracają współrzędną Y i wymiary skrajnego segmentu. Przeszukiwanie w stronę przeciwną zaczyna się od końca ostatniego przeszukania. Po trzech przeszukaninach blok uznaje się z kompletny, a jego skrajne współrzędne zapisane zostają do tablicy Bloki.

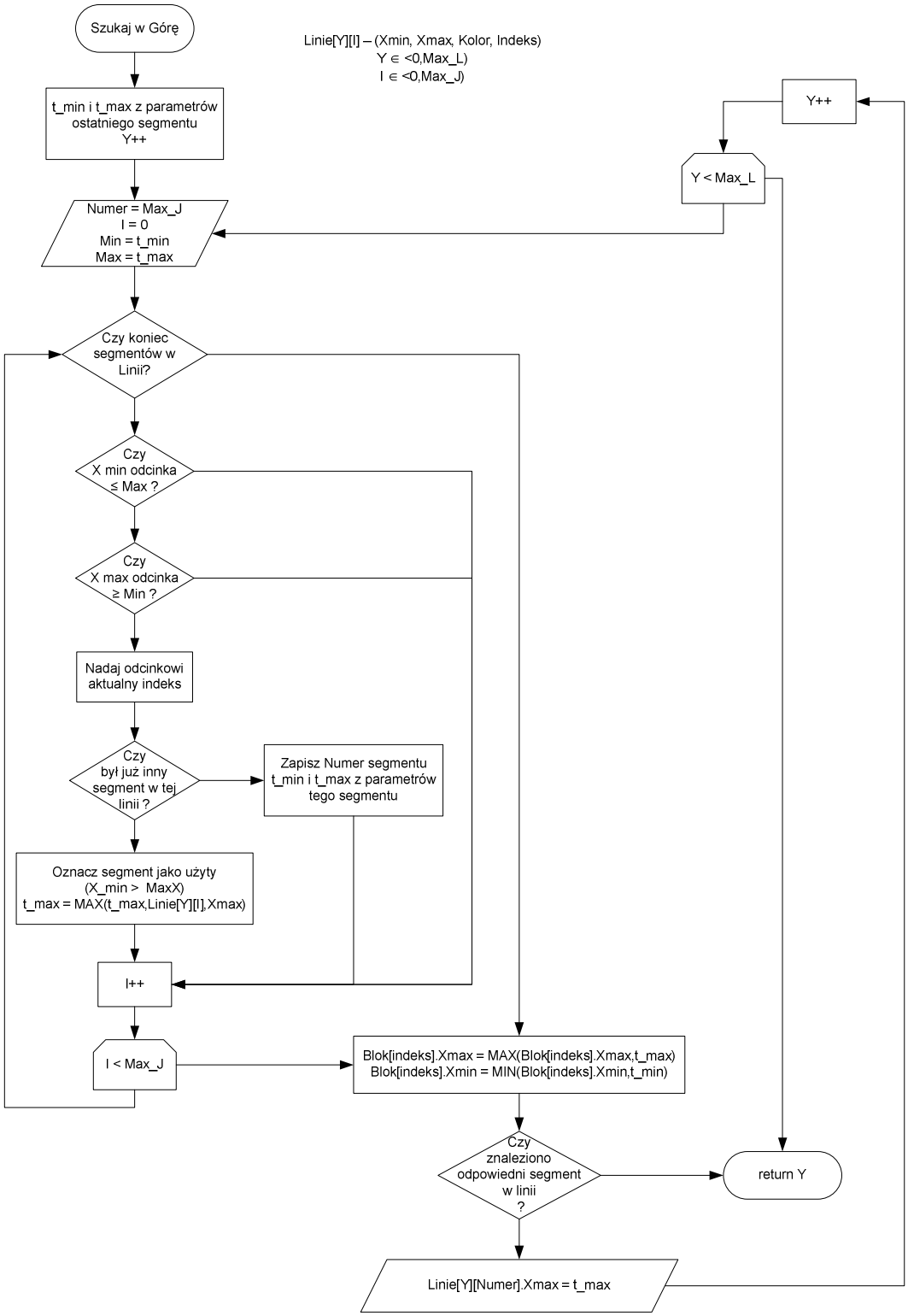
Po osiągnięciu końca zdjęcia lub końca tablicy Bloki program Grupuj kończy pracę.

Programy przeszukujące w górę i w dół postępują zgodnie z opisanym we wcześniejszym podpunkcie algorytmem. Na początek określają wymiary pierwszego segmentu. W przypadku przeszukiwania w górę, w kolejny kroku program sprawdza, które spośród segmentów leżących powyżej znajdują się w granicach pierwszego segmentu. Początek pierwszego i koniec ostatniego znalezionego stają się granicami kolejnego kroku przeszukiwania i zostają zapisane do pierwszego segmentu. Pozostałe segmenty z bloku zostają oznaczone jako użyte (skasowane). Wszystkim obiektom należącym do danego bloku nadany zostaje indeks równy numerowi bloku. W przypadku znalezienia podczas przeszukiwania obiektu wychodzącego poza wymiary bloku, blok zostaje od razu powiększony.



Rysunek 5.10 Schemat blokowy programu Grupuj

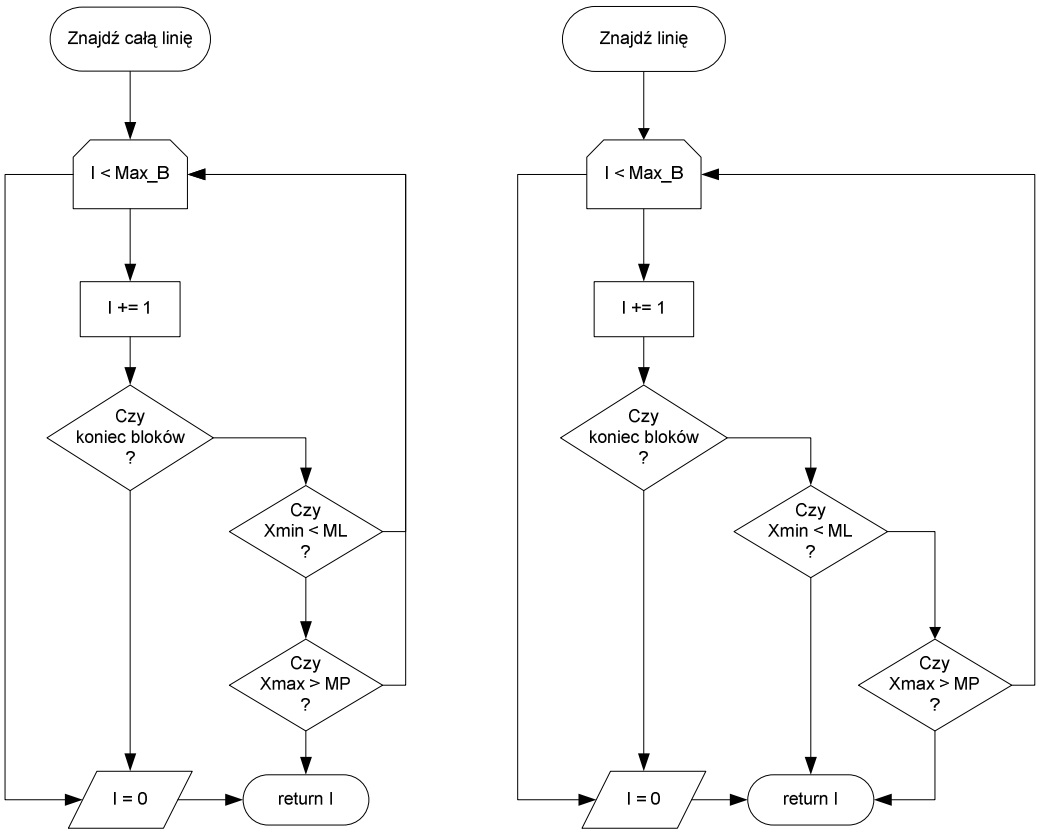
Program Szukaj w Dół różni się od Szukaj w Górę (poniższy rysunek) tylko kierunkiem przeszukiwania. Czyli zamiast inkrementacji numeru wiersza (Y++) jest dekrementacja (Y--), a za koniec obrazka uznany jest Y = 0 zamiast Y = Max_L.



Rysunek 5.11 Schemat blokowy programu Szukaj w Górę

5.4. Znajdowanie linii

Wynikiem pracy programu Grupuj jest zbiór maksymalnie 104 bloków. W większości testów ilość wykrytych obiektów nie przekraczała 20. Każdy z nich jest potencjalnie szukaną białą linią. Aby wśród nich znaleźć właściwą linię powstały dwa programy: Znajdz_linie i Znajdz_cala_linie. Pierwszy z programów poszukuje linii, która zaczyna się przy lewej lub prawej krawędzi obrazu i urywa gdzieś w środku kadru, a drugi szuka linii ciągłej przez całą szerokość obrazu. W praktyce dla bezpieczeństwa za krawędzie obrazu uznane zostały marginesy oddalone o kilkadziesiąt pikseli od właściwych krawędzi. Zostało tak zrobione z powodu kłopotów z interpolacją pikseli przy krawędziach i zakłóceniach wynikających z aberracji sferycznej.



Rysunek 5.12 Schematy blokowe programów Znajdz_linie i Znajdz_cala_linie

Oba programy są uruchamiane z podaniem numeru bloku, od którego mają zacząć pracę. Umożliwia to przeszukanie całej tablicy Bloki jak i kontynuację poszukiwań w przypadku, gdy ostatni znaleziony obiekt nie spełniał wymagań programu, który wywołał funkcję szukającą.

Programy szukające linii składają się tylko z jednej pętli odczytującej kolejne elementy tablicy Bloki. Pierwsze testy sprawdzają czy są jeszcze jakieś bloki do odczytania. Możliwe jest, iż numer bloku osiągnął maksimum, lub kolejny blok zawiera marker końca tablicy. W obu przypadkach programy kończą pracę zwracają zerowy numer bloku. Jest to sygnał dla programu wywołującego, że nic nie zostało znalezione.

Kolejne dwa testy dotyczą kontaktu linii z marginesami. Wpierw sprawdzane jest czy początek bloku jest przed marginesem lewym (ML). Następnie czy koniec bloku wypada za marginesem prawym (MP). W tym miejscu program Znajdz_linie różni się od Znajdz_cala_linię. Przy poszukiwaniu fragmentu białej linii istotne jest by dotykała ona co najmniej jednego marginesu. Dlatego program kończy pracę i zwraca numer linii, gdy spełniony jest którykolwiek z warunków. Linia ciągła rozpoznawana jest po tym, że dotyka obu marginesów. Przy niespełnieniu obu warunków dla fragmentu linii i niespełnieniu chociażby jednego dla całej linii, programy inkrementują numer bloku i ponawiają proces poszukiwania.

5.5. Analiza

5.5.1. Wstęp

Program Analiza jest właściwym programem dokonującym interpretacji zdjęcia. Dzięki złożonemu, kilkuetapowemu przygotowaniu danych, ten program może mieć stosunkowo prostą konstrukcję i dokonywać analizy w krótkim czasie.

Podstawą analizy są dwa programy szukające linii, omówione w poprzednim podpunkcie. Program posiłkuje się nimi w poszukiwaniu białej linii i przerwy w niej. Wynikiem tej pracy jest podanie wysokości, na jakiej została znaleziona linia i położenia brzegów przerwy w białej linii, jeśli została wykryta. W przypadku przeciwnika zlewającego się z linią, program podaje położenie czerwonej linii.

Program ten podaje tylko współrzędne, a nie podejmuje decyzji o zmianie kierunku, gdyż opisywany w tej pracy Autonomiczny System Wizyjny Robota Mobilnego jest tylko dodatkowym czujnikiem dla robota. Ostateczną decyzję podejmuje system operacyjny robota mobilnego na podstawie wyników pomiarów ze wszystkich zainstalowanych czujników.

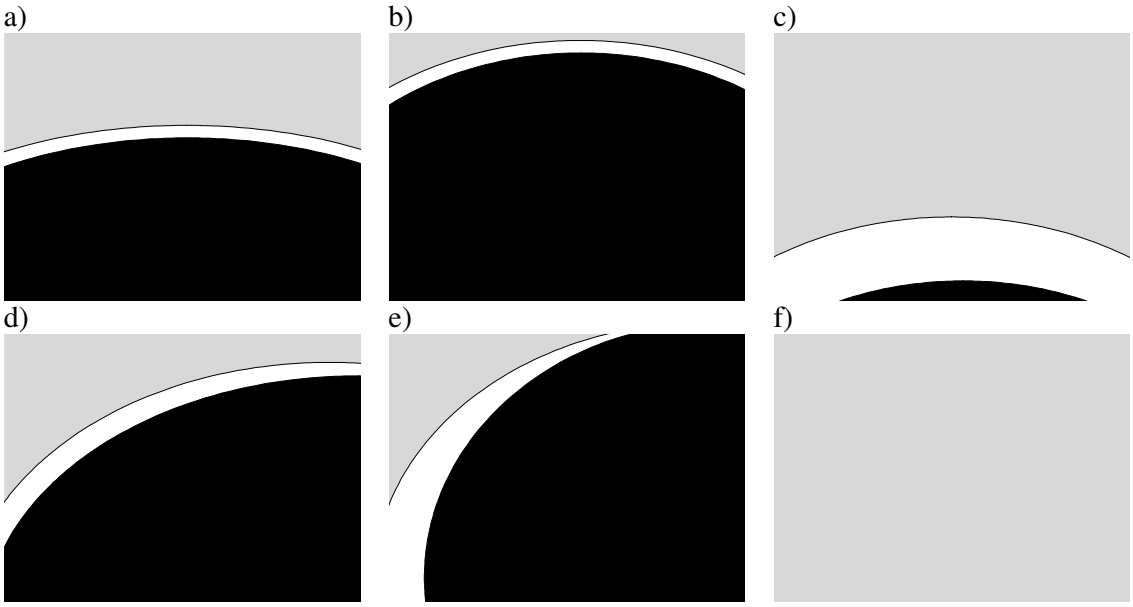
Na potrzeby demonstracji program został rozszerzony o procedury Celuj, Szukaj i Obrót, które zastępują złożony system decyzyjny robota.

5.5.2. Przypadki

Robot Sumo spotyka się na ringu z wieloma skrajnie różnymi przeciwnikami oraz może znaleźć się w niezliczonej ilości możliwych pozycji. System wizyjny powinien w możliwie największej ilości przypadków podać poprawne położenie białej linii i przeciwnika. Wszystkie możliwe przypadki można podzielić na cztery kategorie ze względu rodzaj problemu identyfikacji obrazu:

- Położenie białej linii
- Pozycja przeciwnika
- Przeciwnik zamaskowany
- Specyficzne tło

Pierwszym i najważniejszym zadaniem systemu jest zlokalizowanie linii krańcowej ringu sumo. W momencie startu linia ta najprawdopodobniej będzie po środku kadru a).



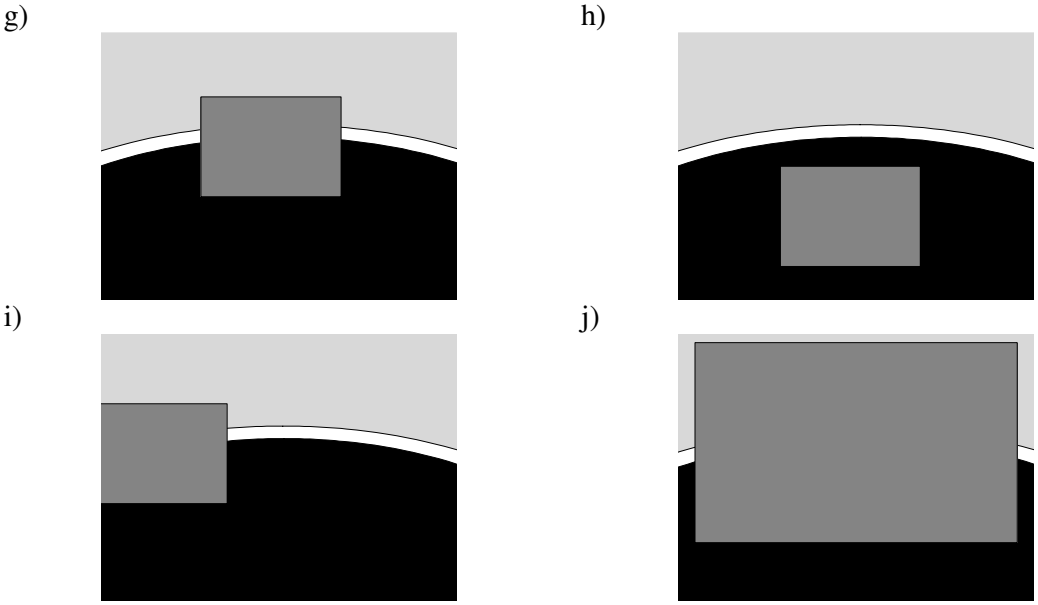
Rysunek 5.13 Położenie białej linii

Poza położeniem środkowym możliwe są jeszcze dwa skrajne na osi Y. Gdy robot cofnie się aż do krawędzi tyłem, linia krańcowa powinna nadal być widoczna na samej górze kadru b). Przy dojechaniu przodem do krawędzi ringu c), przy odpowiednim ustawieniu kamery (4.2) powinno wciąż być widać linię. Przed najechaniem na linię krańcową chronią robota dodatkowo czujniki krawędziowe. Umieszczone na wszystkich rogach robota małe

czujniki odbiciowe informują go o najechaniu na coś jaśniejszego od maty ringu. Mają one pierwszeństwo przed wszystkimi pozostałymi czujnikami.

Podczas walki istnieje ryzyko zostania zepchniętym bokiem do krawędzi ringu. W tej sytuacji d) linia, chociaż wciąż widoczna w całości, po opisaniu na niej prostokąta jest przedstawiona jako bardzo gruba. System operacyjny robota powinien ten fakty zinterpretować jak zagrożenie. Jeśli zostanie ono zlekceważone kolejnym możliwym obrazem z kamery będzie e) linia, która już nie w całości mieści się w poziome w kadrze. Jednak wtedy zamiast Autonomicznego Systemu Wizyjnego Robota Mobilnego powinny zareagować czujniki krawędziowe. Ostatnim z tej kategorii obrazem jest kadr niezawierający żadnych rozpoznawalnych obiektów f). Jest to sytuacja awaryjna, którą program ogłasza przez wyzerowanie pozycji białej linii.

Po zlokalizowaniu białej linii system wizyjny musi znaleźć w jej obrębie przeciwnika.

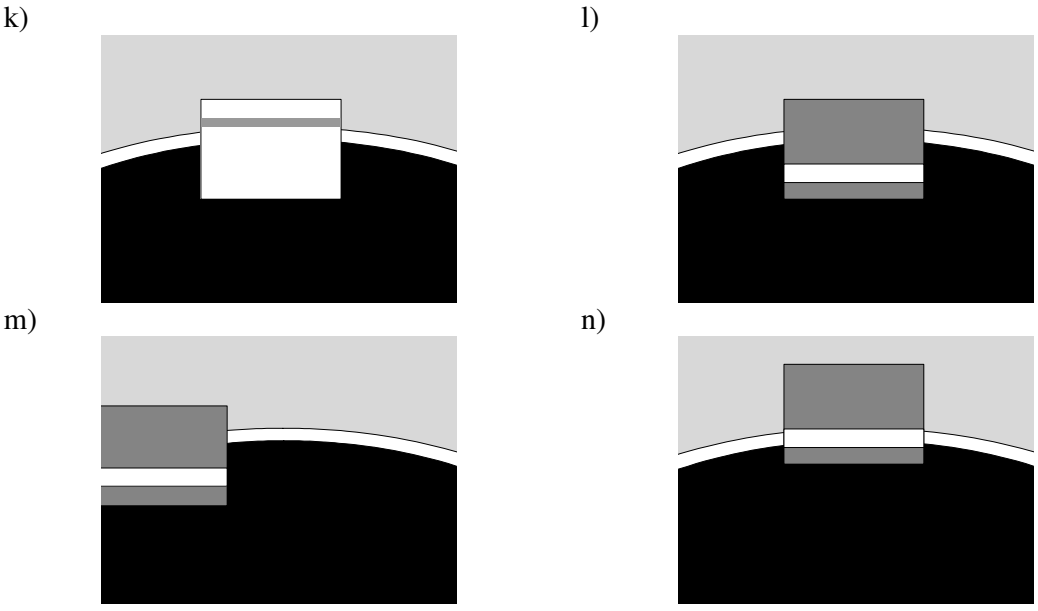


Rysunek 5.14 Pozycja przeciwnika

Podczas startu, lub już po znalezieniu i wycelowaniu przeciwnik powinien znaleźć się po środku kadru g). Kamera została tak zamontowana, aby nawet bardzo niski przeciwnik przecinał białą linię. W kilkuletniej historii zawodów Sumo w Polsce tylko dwa roboty były dostatecznie niskie by powyższego założenia nie spełnić h) (oba z uwagi na zbyt małą wysokość nie zawierały dostatecznie silnego napędu by wypchnąć przeciwnika). Na szczęście tylko przy niektórych względnych położeniach przeciwnika i robota z kamerą, możliwe jest nie przecięcie linii przez przeciwnika naprzeciwko. W pozostałych przypadkach warunek jest spełniony, więc ten problem można pominąć.

Podczas poszukiwania przeciwnika istotnym momentem jest jego wykrycie przy krawędzi kadru. Jest to możliwe, gdy szukamy przeciwnika lub, gdy ucieka nam on z pola widzenia. W takiej sytuacji i) program Analiza podaje położenie przerwy jako odcinek od krawędzi kadru do początku lub końca wykrytej białej linii. Decyzją robota powinien być skręt w stronę przeciwnika.

Po wycelowaniu i dojechaniu do oponenta optycznie zwiększą się jego rozmiar j). Ponieważ istnieje ryzyko, że zasłoni on sobą cały kadr, kamera musi być odpowiednio cofnięta i posiadać dość duży kąt widzenia (4.2). Przy wykryciu bardzo szerokiej przerwy robot nie powinien już korygować kierunku, tylko atakować.



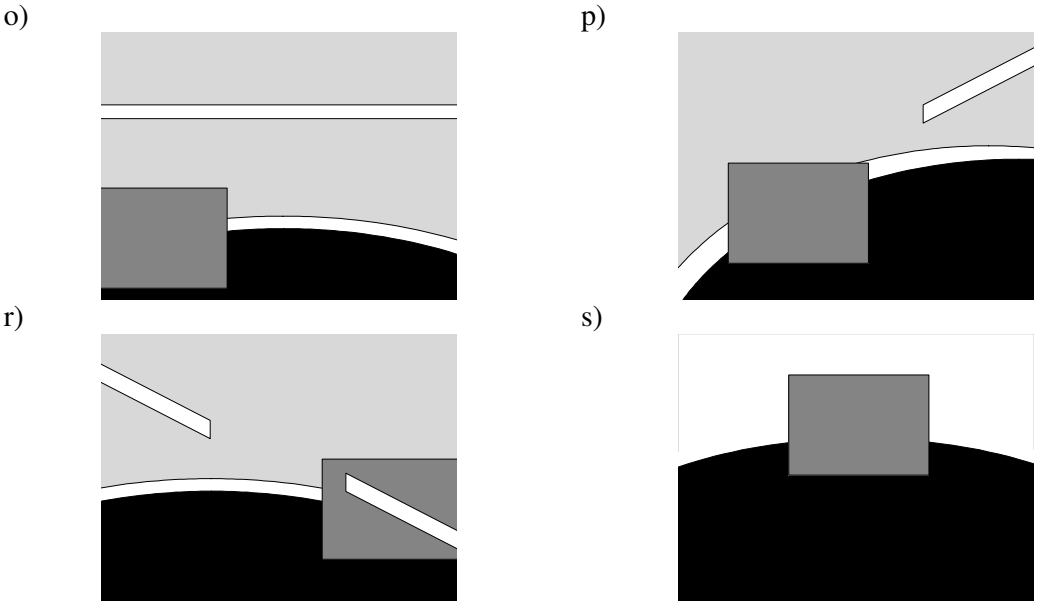
Rysunek 5.15 Przeciwnik zamaskowany

Przeciwnicy mogą specjalnie lub przypadkiem stosować maskowanie. Większość robotów Sumo, aby ukryć się przed sensorami podczerwieni, stosuje czarne malowanie. Część czarnych farb poza pochłanianiem całego widma światła widzialnego, pochłania również światło podczerwone, na przykład farba H.B.Fuller typu Omega - RAL 9005. Jednak niektóre z robotów są pomalowane na biało k) lub srebrno. Z uwagi na dużą ilość odbitego światła od takiego koloru, przeciwnik może być niemożliwy do odróżnienia od białej linii. W tym wypadku będzie widoczne odbicie linijki świetlnej generowanej przez zamontowany na pokładzie laser.

Część z zawodników stara się zmylić innych malując białą linię na płaskiej płytce umieszczonej z przodu lub z tyłu robota. Czujniki krawędziowe teoretycznie mogą ją zinterpretować jako koniec ringu i wydać polecenie zmiany kierunku jazdy. Ta sama fałszywa linia może zmylić system wizyjny l). Dlatego oprócz podawania współrzędnych znalezionej

przerwy w białej linii, program zapamiętuje ostatnią dobrą współrzędną Y. Na kolejnym zdjęciu w pierwszej kolejność szuka linii w okolicy ostatniej dobrej. Dopiero przy jej braku przeszukuje całe zdjęcie. Jeśli na zdjęciu jest widoczna właściwa linia i fałszywa to na podstawie wyniku poprzedniego pomiaru, linia dobra powinna zostać poprawnie zidentyfikowana. Wyjątkiem od tej reguły może być sytuacja, gdy taki robot znajdzie się przy krawędzi zdjęcia m). W takim przypadku teoretycznie widoczne są dwa fragmenty białej linii, leżące na tej samej wysokości. Wynik analizy powinien wskazać na małą przerwę położoną przy jednej z krawędzi przeciwnika. Ponieważ przerwa będzie mała i położona raczej z boku, system robota powinien na to zareagować skrętem w stronę przerwy, a tym samym przejście do sytuacji, gdy oba fragmenty linii są dobrze widoczne, a przeciwnik jest na wprost.

Osobnym przypadkiem jest pomylenie fałszywej linii z linią właściwą n). Bardziej prawdopodobne jest, że linia będzie położona nisko lub, że robot będzie cały biały k). W praktyce taka sytuacja może zaistnieć tylko, gdy przeciwnik wypadnie poza ring. Dlatego nie ma osobnej reakcji na takie zdjęcie.



Rysunek 5.16 Specyficzne tło

Ostatnią kategorią obrazu jest specyficzne tło. Tło praktycznie może zawierać dowolne elementy. Mogą tam również występować ciągłe, białe linie, na przykład w postaci listwy przypodłogowej. Dzięki śledzeniu ostatniej dobrej linii system powinien zignorować ciągłe linie o), gdy w okolicy poprzedniej jest wykryty fragment białej linii.

Oprócz ciągłych linii możliwe jest wykrycie fragmentu linii w tle, która wypada na przedłużeniu wykrytego fragmentu białej linii p) lub elementu malowania przeciwnika r). Ponieważ program Analiza nie szuka drugiego fragmentu na przedłużeniu wykrytego tylko

mniej więcej na tej samej wysokości, oba te przypadki są niegroźne dla poprawnego działania.

Praktycznie możliwe jest, iż tło (ściany) optycznie zleje się z białą linią krańcową s) powodując, że nie będzie możliwe ich rozróżnienie. W warunkach testowych taka sytuacja miała miejsce. Poprawnie wykonana kalibracja parametrów kamery powinna przy górnym oświetleniu uniemożliwić powstanie takiej sytuacji. Ponadto odpowiednio dobrany kąt (4.2) ustawienia kamery spowoduje, że w większości przypadków bezpośrednio za linią będzie widoczny pasek podłogi. Zgodnie z regulaminem zawodów, wokół ringu powinien zostać zachowany metr wolnej przestrzeni, a podłoga nie może być biała. Dlatego teoretycznie problemu z tym przypadkiem s) nie ma.

5.5.3. Opis algorytmu

Program analiza rozpoczyna pracę od wyczyszczenia zmiennych ustawionych w poprzednim cyklu pracy. Najpierw zeruje współrzędne przerwy następnie przywraca pozycje marginesów i wylicza okolice ostatniej dobrej linii. Okolice współrzędnej Y ostatniej wykrytej białej linii są na tyle szerokie by mieć pewność, iż po zmianie pozycji robota od ostatniego zdjęcia, linia nie wysunie się poza nie. Domyślnie ustawione są one na + 50 i – 100. Dokładne ich ustawienie zależy od prędkości robota i szybkości pracy kamery.

Kolejnym krokiem jest poszukiwanie całej linii w okolicy ostatniej dobrej. Pętla ta kończy się tylko w dwóch przypadkach: gdy program Znajdz_cala_linie nie może już znaleźć więcej linii lub, gdy znaleziona linia jest wyznaczonej okolicy.

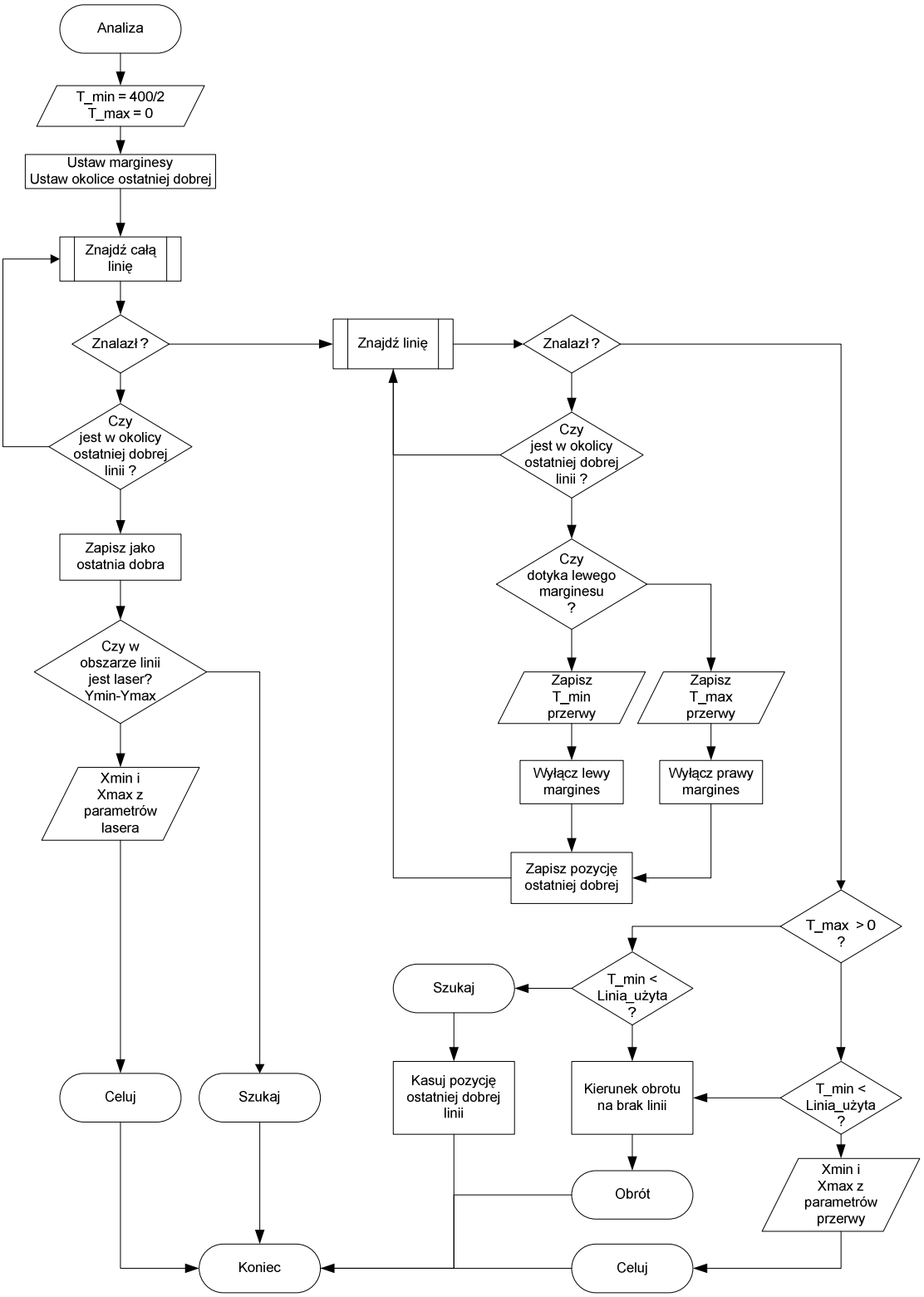
Kiedy zostanie wykryta dobra linia program sprawdza czy w jej obrębie występuje światło lasera. W tym celu przeszukuje tablicę Laser od Y_{MIN} do Y_{MAX}. Jeśli zostaną znalezione wzmianki o wystąpieniu lasera to program zwraca jego skrajne współrzędne i uruchamia procedurę Celuj. Jeśli nie to uznaje, że przeciwnika nie ma w zasięgu kamery i wybiera procedurę Szukaj.

Poza przypadkiem, kiedy kamera nie widzi przeciwnika lub przeciwnik jest koloru białego program powinien nie znaleźć całej linii. Wtedy przystępuje poszukiwania jej fragmentów. Podobnie jak poprzednio głównym obiektem zainteresowania są linii położone w okolicy ostatniej dobrej. Jeśli znaleziony obiekt spełnia powyższe kryteria to program poszukuje drugiego fragmentu. W tym celu wyłącza margines przy wykrytej linii. Po znalezieniu pary lub jej braku następuje jej klasyfikacja.

Możliwe są trzy przypadki:

- Znaleziono dwa fragmenty linii
- Jest tylko jeden fragment
- Nic nie zostało znalezione

W pierwszym przypadku program zwraca współrzędne przerwy i uruchamiana jest procedura Celuj. W drugim przypadku brakuje drugiego fragmentu linii. W celu sprawdzenia czy w tym miejscu jest przeciwnik program wywołuje procedurę Obrót z podaniem strony, z której nie ma linii. Ostatnią możliwością jest brak linii. Może to być spowodowane błędem odczytu lub wyskoczeniem właściwej linii z wyznaczonych okolic jej poszukiwania. Aby to sprawdzić włączana jest procedura Szukaj. Po jej zakończeniu dla pewności program kasuje pozycję ostatniej dobrej linii, żeby podczas kolejnego cyklu pracy nie kierować się jej położeniem, tylko szukać wszystkich linii na zdjęciu.



Rysunek 5.17 Schemat blokowy programu Analiza

5.5.4. Dodatkowe procedury

Na potrzeby demonstracji program został rozszerzony o procedury Celuj, Szukaj i Obrót, które zastępują złożony system decyzyjny robota.

Celuj – procedura decyduje jak należy zachować się na widok przeciwnika. Jeśli przeciwnik znajduje się blisko wszelkie manewry mogą tylko zmniejszyć szanse w bezpośrednim starciu. Dlatego w takiej sytuacji wykonywane jest polecenie jazdy naprzód. Jeśli przeciwnik jest położony dalej należy zdecydować czy jest na tyle odsunięty od środka kadru, że warto skręcać czy też nie. Jest to typowy regulator histerezowy, powyżej ustalonego odchylenia od środka decyduje się na obrót w danym kierunku, poniżej nakazuje kontynuację jazdy na wprost.

Szukaj – procedura ta ma umożliwić odnalezienie przeciwnika. Gdy robot wciąż dobrze widzi linię krańcową Szukaj wydaje polecenie obrotu w miejscu w celu znalezienia przeciwnika. Jeśli współrzędna ostatniej dobrej linii została skasowana obrót jest kontynuowany, a ponadto procedura sygnalizuje błąd.

Obrót – w aplikacji symulacyjnej zastępuje sterownik napędu robota. Każdy robot mobilny posiada sterownik silników. Jest to odpowiedni układ tranzystorów lub gotowy mostek H załączający zasilanie napędów. Procedura Obrót wyświetla na kilku diodach symulację reakcji robota.

6. Kalibracja parametrów obrazu

6.1. Opis możliwości regulacji kamery

Przetwornik obrazu OV6620, w który wyposażono kamerę C3088 posiada 80 rejestrów, w których przechowywane są dane odpowiadające za parametry pracy kamery. Do dyspozycji użytkownika są 52 rejestry 8-bitowe, w których może on ustawiać pojedyncze bity i tym samym regulować poszczególne parametry kamery. Ustawienia rejestrów kamery dokonuję się przez interfejs SCCB opisany w punkcie 3.5.1. Napisane oprogramowanie dla tego interfejsu wraz GUI umożliwia dostęp do wszystkich parametrów regulowanych, jakie posiada ta kamera. Wśród całej grupy rejestrów można wyróżnić rejestry odpowiedzialne za:

- Parametry sygnałów synchronizacji (VSYNC, HREF i PCLK)
- Format wyjściowy obrazu
- Rozdzielczość wyjściową obrazu
- Ustawienia wstępne układów automatycznej regulacji (AEC, AGC, AWB)
- Ustawienie parametrów tj. jasność, kontrast, ostrość i nasycenie

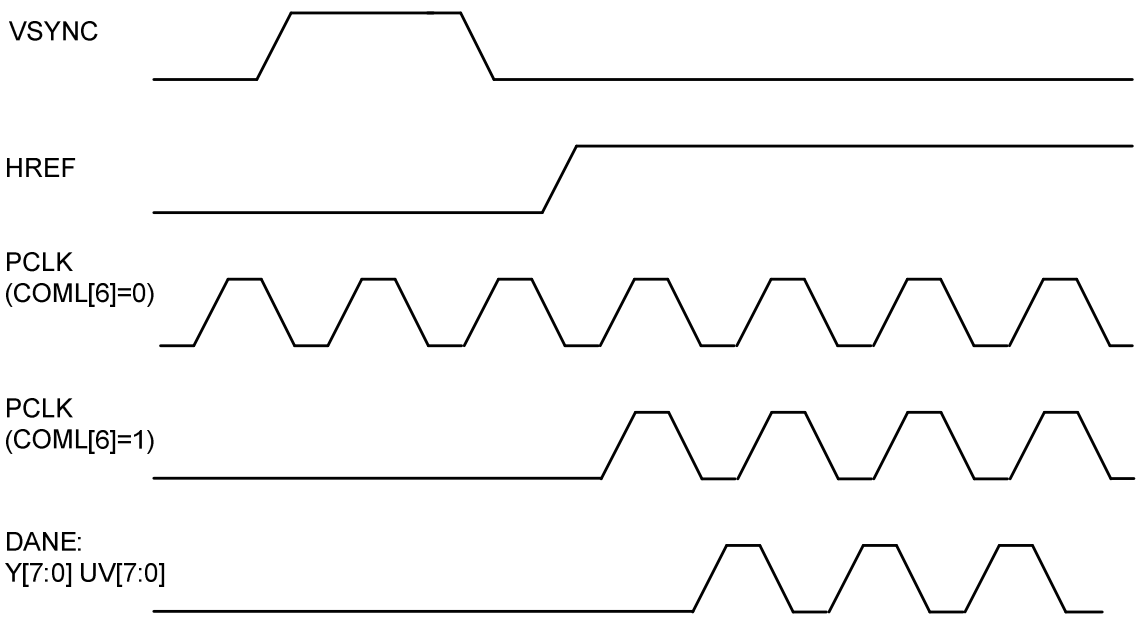
Kamera wysyła pojedyncze bajty (piksele) obrazu w takt sygnału PCLK (zbocze rosnące lub malejące). Częstotliwość tą można regulować za pomocą 6 młodszych bitów rejestru CLKRC oraz najstarszego bitu rejestru COMO. Domyślnie najstarszy bit rejestru 0x3Eh jest ustawiony a jego wyzerowanie powoduje dwukrotne zmniejszenie częstotliwości PCLK. Wzór na obliczenie częstotliwości PCLK ma postać:

$$PCLK = \frac{17{,}73 * 10^6}{(CLKRC[5:0]+1) * 2k} \quad \text{,gdzie} \quad k = \begin{cases} 1 \quad dla \quad COMO[7] = 1 \\ 2 \quad dla \quad COMO[7] = 0 \end{cases}$$

Stan aktywny (polaryzacja) poszczególnych sygnałów synchronizacji może być ustawiany w rejestrach: CLKRC[7:5] dla VSYNC, COMC[3] dla HREF i COMD[6], dla PCLK. W trybie domyślnym kamery stanem aktywnym sygnałów synchronizacji jest stan wysoki i ten tryb był wykorzystywany. Po resecie (COMA[7]=1) kamera pracuje z częstotliwością PCLK = 8,86 MHz w trybie 16-bitowym YUV. Częstotliwość ta była zmieniana i ustawiana w zależności od formatu i trybu skanowania (punkt 4.1.2).

Kamera posiada rejestr COML, który dla ustawienia COML[6] = 1 powoduje, że sygnał PCLK jest generowany tylko wtedy, gdy HREF jest w stanie wysokim. Ustawienie to znacznie uprościło program skanujący a dodatkowo przyspieszyło jego pracę. W domyślnych

ustawieniach kamery, PCLK jest generowane cały czas, (free running) a przez to, aby otrzymać właściwe dane z kamery należy dla każdej linii obrazu synchronizować się z HREF a następnie z PCLK. W przypadku, gdy PCLK jest generowane tylko wtedy, gdy HREF ma stan wysoki to po nadejściu sygnału nowej klatki (VSYNC) wystarczy zsynchronizować się z PCLK. Rysunek 6.1 przedstawia przebiegi sygnałów kamery dla omawianych ustawień rejestrów.



Rysunek 6.1 Przebiegi sygnałów kamery przy zmianach ustawień rejestru COML

Dane z kamery mogą być przesyłane w formacie 16, 8 lub 4-bitowy. Dodatkowo dostępny jest tryb RGB i YUV dla każdego formatu. Do określenia formatu danych wyjściowych służą rejestry COMA, COMB, COMC, COMD oraz COMH. Bit COMA[3] ustala tryb pracy na RGB (1) lub YCrCb (0). Bit COMB[5] pozwala ustalić czy dane będą pojawiały się na obu portach (Y i UV) czy tylko na porcie Y. Poprzez negację bitu COMD[0] możemy zmienić sekwencję bajtów wyjściowych (np. z UVUV na VUVU dla formatu 16-bitowego). Kolejne możliwości dają ustawienia bitów rejestru COMH. Ustawiając stan COMH[7]=1 ustalamy tryb 8-bitowy „On-Line" RGB, z którego przy sekwencyjnym skanowaniu uzyskujemy dokładne odwzorowanie siatki Bayera. W programie wykorzystywano również tryb YG (COMH[2]=1), w którym na port Y wysyłana jest tylko składowa G a na port UV naprzemiennie składowe B i R.

W tabeli 6.1 przedstawiono poszczególne tryby obsługiwane przez kamerę z podaniem sekwencji danych na portach wyjściowych.

Format 16-bitowy				
	Tryb domyślny (16-bit)		Tryb YG	
	Port Y	Port UV	Port Y	Port UV
1 linia	niestabilne	B11 G12 B13 G14	niestabilne	niestabilne
2 linia	G21 R22 G23 R24	B11 G12 B13 G14	G21 G12 G23 G14	B11 R22 B13 R24
3 linia	G21 R22 G23 R24	B31 G23 B33 G34	G21 G32 G23 G34	B31 R22 B33 R24
Format 8-bitowy				
	Tryb One line		Tryb 8--bit	
	Port Y		Port Y	
1 linia	B11 G12 B13 G14		niestabilne	
2 linia	G21 R22 G23 R24		B11 G21 R22 G12	
3 linia	B31 G32 B33 G34		B31 G21 R22 G32	

Tabela 6.1 Tryby pracy kamery

Podstawową rozdzielczością robienia zdjęć przez kamerę jest rozdzielczość 352x288 (CIF). Ustawiając stan bitu COMC[5] = 1 zmieniamy rozdzielczość na 176x144 (QCIF).

Zaletą tego przetwornika jest łatwa regulacja takich parametrów jak: jasność, kontrast, ostrość czy nasycenie. Tabela 6.2 przedstawia adresy rejestrów regulujących te parametry wraz z zakresami regulacji.

Parametr	Adres rejestru	Wartość domyślna	Zakres regulacji
Nasycenie (Saturation)	0x3	0x80	0x0÷0xFF
Kontrast (Contrast)	0x5	0x48	0x0÷0xFF
Jasność (Brightness)	0x6	0x80	0x0÷0xFF
Ostrość (Sharpness)	0x7	0xC6	Bity [7:4] - ustawienia progu (0÷80 mV) Bity [3:0] - regulacja

Tabela 6.2 Regulacja parametrów kamery

6.2. Ustawianie parametrów kamery

Jak wspomniano w podpunkcie 6.1 kamera posiada 52 rejestry regulacyjne. Po dokonaniu zmian wartości w tych rejestrach są one zapamiętywane do czasu wprowadzenia nowych zmian lub wyłączenia napięcia zasilania. Wyłączenie zasilania powoduje, że zapisane wartości są tracone. Po włączeniu kamery zostaje ona uruchomiona z parametrami domyślnymi, które zostały wyszczególnione w instrukcji [20].

Kamera ma domyślnie włączone wszystkie układy automatycznej regulacji (AEC, AGC, AWB). Jak pokazały testy, dokonując nastaw ręcznych przy wyłączonych układach automatycznych można uzyskać lepsze wyniki kalibracji. Niezadowalające wyniki działania automatów były spowodowane najprawdopodobniej tym, że wszystkie układy automatycznej regulacji zostały zaprojektowane tak, aby pracowały optymalnie przy maksymalnej

częstotliwości skanowania kamery. Autorzy systemu CMUcam2 w pracy [27] potwierdzali lepsze działanie układów automatycznej regulacji przy skanowaniu z pełną prędkością. Podczas testowania systemu problemem był też długi czas stabilizacji układów automatycznej regulacji. Występuje on tylko w przypadku zmiany formatu wyjściowego obrazu lub szybkiej zmiany oświetlenia w dużych granicach. Przy skanowaniu zdjęcia z częstotliwością 403 KHz i domyślnych ustawieniach szybkości reakcji automatów poprawny obraz otrzymywano dopiero po około 40 s. Po zmianach na szybką stabilizację automatów (COMK[1]=1, COMK[3]=1) ograniczono ten czas do połowy, co jednak nadal było dużą wartością. Badania wykazały, że czas stabilizacji kamery zależy odwrotnie proporcjonalnie do częstotliwości skanowania. W instrukcji kamery nie poruszono problemu stabilizacji jej parametrów. Sformułowany wniosek potwierdza jednak informacja podana w projekcie CMUcam2 [8] wg której czas stabilizacji kamery przy maksymalnej prędkości skanowania wynosi 5 s.

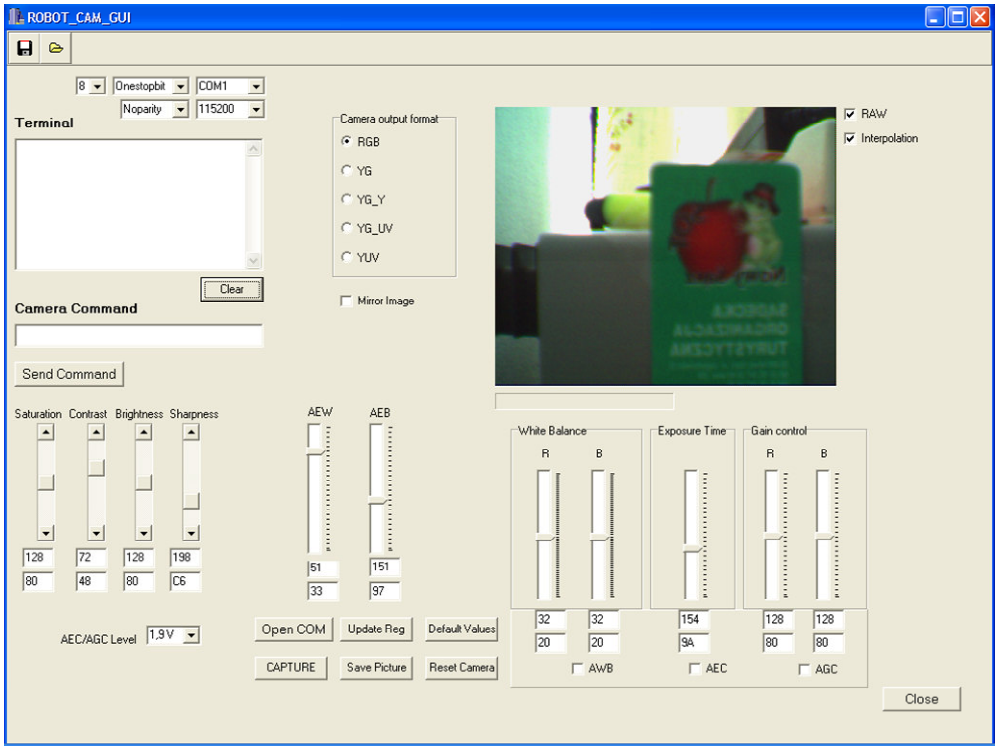
W pewnych przypadkach mimo dobrych nastaw kamery mogą pojawiać się zakłócenia obrazu. Przykładem może tu być świetlówka, której tętnienia strumienia świetlnego prowadzą do powstawania zjawiska stroboskopowego. Na obrazie tętnienia strumienia objawiają się powstawaniem ciemniejszych i jaśniejszych pasów. Są one szczególnie widoczne, kiedy częstotliwość skanowania kamery jest wielokrotnością częstotliwości zmian strumienia.

6.2.1. Graficzny interfejs użytkownika

Do celów nastaw i testów parametrów kamery napisano program ROBOT_CAM_GUI w środowisku Borland® C++ Builder (rysunek 6.2). Program posiada pełną obsługę łącza RS-232 wykorzystując funkcje z interfejsu programistycznego systemu Windows (API). Pozwala na podgląd i ustawianie rejestrów kamery zarówno z wykorzystaniem elementów graficznych (suwaki, przełączniki) jak i przez wprowadzanie rozkazów z klawiatury. Takie połączenie klasycznego terminala ze środowiskiem graficznym pozwala na pełną regulację i podgląd parametrów kamery..

Interfejs umożliwia ustawienie wszystkich podstawowych parametrów transmisji szeregowej RS-232 tj. nr portu, prędkość transmisji, parzystość, liczbę bitów danych i liczbę bitów stopu. Może dzięki temu zastępować w niektórych aplikacjach terminal Windows. Aplikacja została napisana dla celów testowych i edukacyjnych, dlatego nie zawiera wszystkich metod poprawiających niezawodność aplikacji. Informuje ona jednak m.in. o:

- Braku odpowiedzi ze strony urządzenia, do którego wysłano rozkaz
- Przerwaniu transmisji obrazu



Rysunek 6.2 Graficzny interfejs użytkownika ROBOT_CAM_GUI

Część graficzna interfejsu pozwala na manualne ustawianie układów automatycznej kalibracji (AEC, AGC, AWB) jak i ich pracę w normalnym trybie. Dodatkowo regulowane jest napięcie odniesienia będące stosunkiem AEC/AGC. Program daje możliwość robienia zdjęć w pięciu trybach: RGB, YG, YG_Y, YG_UV i YUV, przy czym w 2 trybach obraz jest bezpośrednio interpolowany. Opcja „Mirror Image" pozwala na uzyskanie odbicia lustrzanego obrazu. Pod każdym suwakiem znajdują się dwa pola pokazujące jego aktualną wartość. Pierwsze pole wyświetla wartość w systemie dziesiętnym a drugie w systemie heksadecymalnym tak, aby łatwiej było porównywać otrzymywane wartości z wielkościami z instrukcji kamery. Obsługa terminala polega na wpisaniu odpowiedniej komendy w polu Camera Command a następnie potwierdzeniu przyciskiem Send Command. Jeżeli podana komenda nie zostanie rozpoznana to system poinformuje użytkownika o tym fakcie w polu Terminal. Zawartość tego pola jest kasowana przyciskiem Clear. Obsługa kamery z poziomu menu graficznego polega na następujących krokach:

- Otwarciu portu szeregowego (przycisk Open COM)
- Ustawieniu żądanych parametrów poprzez manipulowanie el. graficznymi
- Uaktualnieniu stanów rejestrów przyciskiem Update Reg
- Wywołaniu procedury przechwytywania zdjęcia wciskając CAPTURE
- Zapisaniu zdjęcia do pliku „.bmp" przyciskiem Save Picture

Podczas skanowania zdjęcia wskaźnik wykonania procesu pokazuje procentową liczbę danych faktycznie odebranych w stosunku całego rozmiaru pliku. Po naciśnięciu przycisku Save Picture następuje zachowanie obrazu w katalogu, z którego uruchomiono program. Nazwa zapisanego pliku zawiera dokładną datę i czas wykonania zdjęcia. Parametry ustawień programu można zapisać pod tą samą nazwą korzystając z górnego menu. Ustawienia zostaną zapisane w pliku o rozszerzeniu „.INI", które jest typowym rozszerzeniem stosowanym do zapisu ustawień programowych w systemie Windows. Wybrane ustawienia programu możemy załadować korzystając z opcji otwórz z górnego menu i wybierając żądany plik. W programie uwzględniono również możliwość szybkiego resetu kamery przyciskiem Reset Camera oraz funkcję przywracającą domyślne ustawienia kamery (Default Values).

6.2.2. Przykładowe ustawienia

Jak wspomniano wcześniej program został tak napisany żeby można było dokonywać operacji odczytu i zapisu zarówno na rejestrach jak ich pojedynczych bitach. W tym celu utworzono komendy:

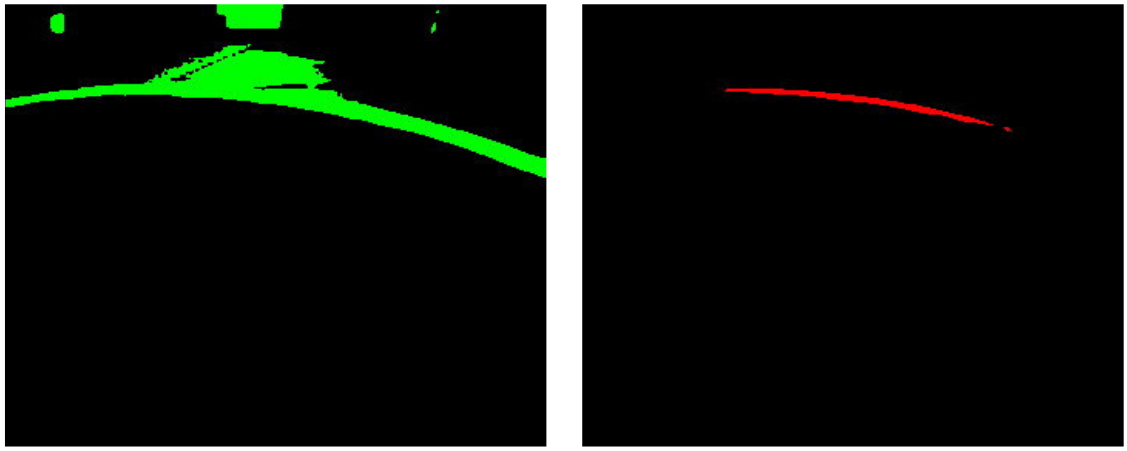
- Odczytu
  - „read" – odczytuje wartość rejestru ostatnio zapisanego
  - „read 0x11" – odczytuje wartość rejestru o podanym adresie (tutaj 0x11)
  - „scan 0x10 0x5" – odczytuje podaną liczbę rejestrów (argument 2) począwszy od rejestru, którego adres podano jako pierwszy argument. W przykładzie następuje kolejno odczyt 5 rejestrów zaczynając od rejestru 0x10.
- Zapisu
  - „write 0x10 0x5" – zapisuje do rejestru o adresie 0x10 (argument 1) wartość 5 (argument 2)
  - „setbits 0x10 0x4" – ustawia w rejestrze 0x10 (argument 1) bity zgodnie z maską podaną jako 2 argument. W przykładzie ustawia bit 2 rejestru 0x10
  - „clearbits 0x10 0x4" – zeruje w rejestrze 0x10 (argument 1) bity zgodnie z maską podaną jako 2 argument. W przykładzie zeruje bit 2 rejestru 0x10

Aby dokonywać ręcznej regulacji wzmocnienia dla kanałów B i R kamery należy wcześniej wyłączyć automatyczny balans bieli (AWB), ponieważ ma on bezpośredni wpływ na wartości tych parametrów. W przypadku słabej widoczności lasera należy zwiększyć

wzmocnienie dla kanału R. W sytuacji ciągłego ustawiania przez AEC zbyt długiego czasu ekspozycji należy zmniejszyć poziom AEC/AGC (dokonując np. wpisu write 0x4E 0x20).

6.3. Algorytm automatycznej kalibracji

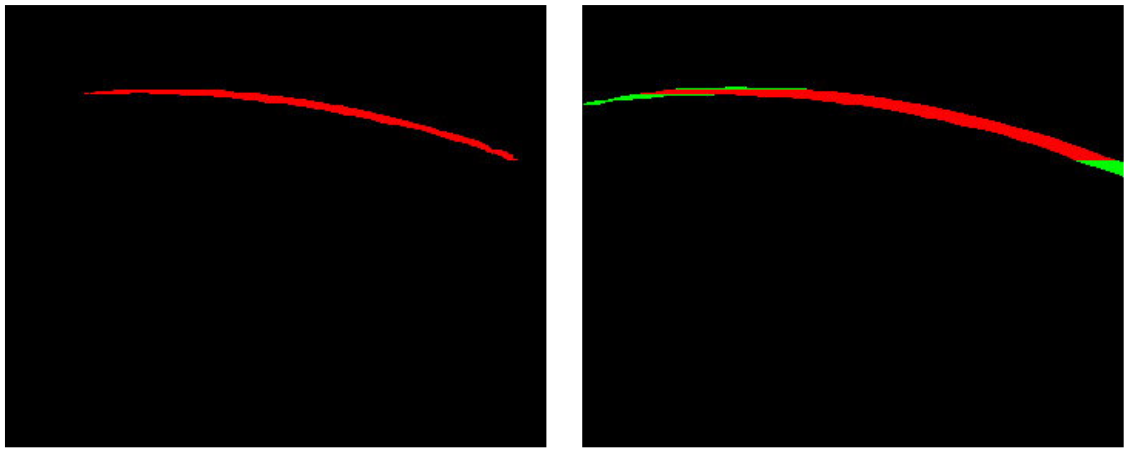
Algorytm kwantowania bazuje na dwóch progach jasności. Przy odpowiednim dobraniu tych progów obraz po kwantowaniu zawiera tylko szukaną linię widoczną od jednej do drugiej krawędzi kadru. Po serii testów w programie Kompresja, zostało ustalone, że najlepszy stosunek jakości obrazu do czasu edycji i zajętej pamięci, osiąga się dla dwóch progów jasności.



Jeden niski próg Jeden wysoki próg

Rysunek 6.3 Wpływ wysokości jednego progu na jakość obrazu

Przy jednym niskim progu (zdjęcie po lewej) cała linia jest gruba i dobrze widoczna od marginesu do marginesu. Niestety z uwagi na zbytnie obniżenie tego progu jasności linia zlewa się z tłem i pojawiają się obiekty, które nie przynależą do szukanej linii. Przy jednym wysokim progu (zdjęcie prawej) może i często występuje nieciągłość linii.



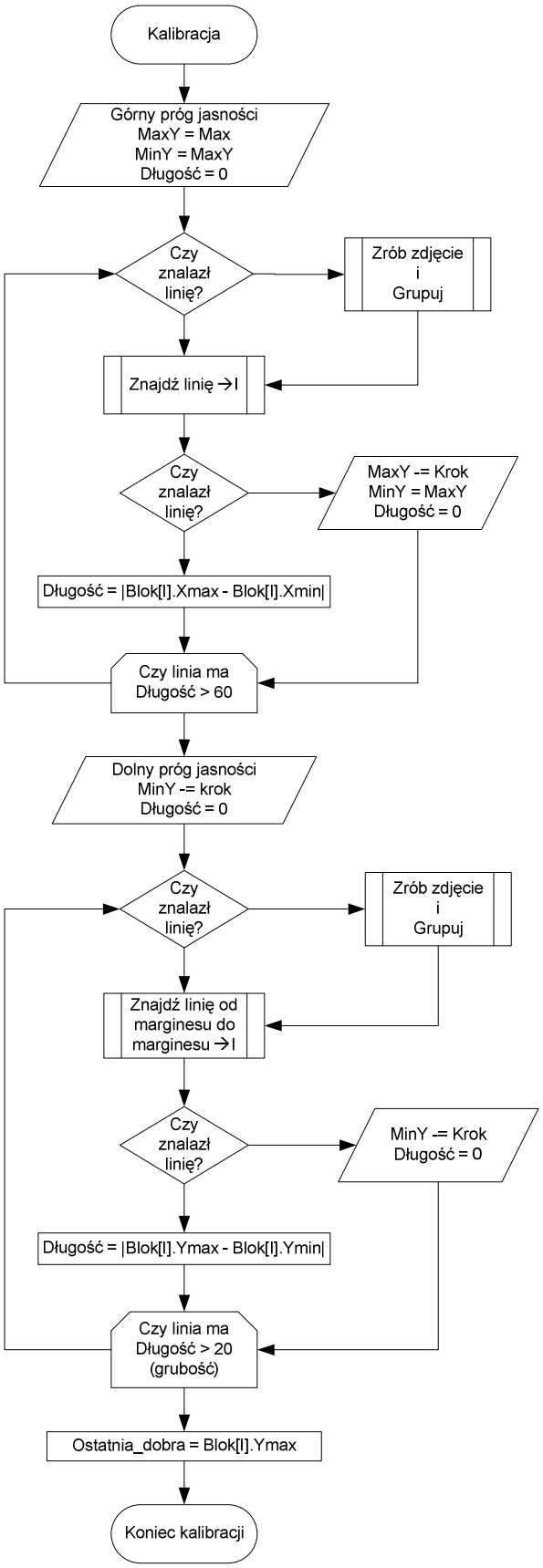
Dobrze ustawiony górny próg jasności Obraz po kalibracji

Rysunek 6.4 Dopasowanie progów

Program kalibracyjny w pierwszym kroku obniża próg MaxY tak długo aż znaleziona linia będzie dotykała marginesu i miała długość większą niż założone minimum. Margines odseparowuje obraz od krawędzi, na której mogą występować zakłamania optyczne. Ponadto regulując szerokość marginesów można wpłynąć na jakość obrazu po kalibracji. Węższy margines oznacza jaśniejszy (pełniejszy) obraz, szerszy oznacza ryzyko pojawienia się zbędnych obiektów w tle. Podobnie działa regulacja zdefiniowanej minimalnej długości linii. Pozwala ona zignorować obiekty z tła, jeśli ich długość jest mniejsza. Z definicji wynika, że linia końcowa powinna być najdłuższym obiektem z tła.

Drugim krokiem kalibracji jest stopniowe obniżanie progu MinY do poziomu, przy którym znaleziona linia będzie dobrze widoczna od jednego do drugiego marginesu i będzie miała odpowiednią grubość. Regulując grubość linii można zwiększyć szanse na poprawne ustawienie dolnego progu. Dla przyspieszenia pracy programu przyjmuje się, że poziom początkowy jest równy progowi górnemu.

Końcowy wynik kalibracji widoczny na powyższym zdjęciu został wynosi: MaxY = 172 i MinY = 118. Dokładne wartości progów są zależne od wybranego kroku kalibracji. Im krok jest większy tym szybciej i mniej dokładnie pracuje program. Domyślnie jest to 6. Ponieważ program zaczyna kalibrację od wartości 238, więc próg górny został ustalony po 11 krokach, a dolny po kolejnych 9. Łącznie program Kalibracja musiał zrobić i przeanalizować 20 zdjęć, aby ustalić odpowiednie poziomy jasności. Dokładny schemat blokowy działania tego programu przedstawia poniższy rysunek.

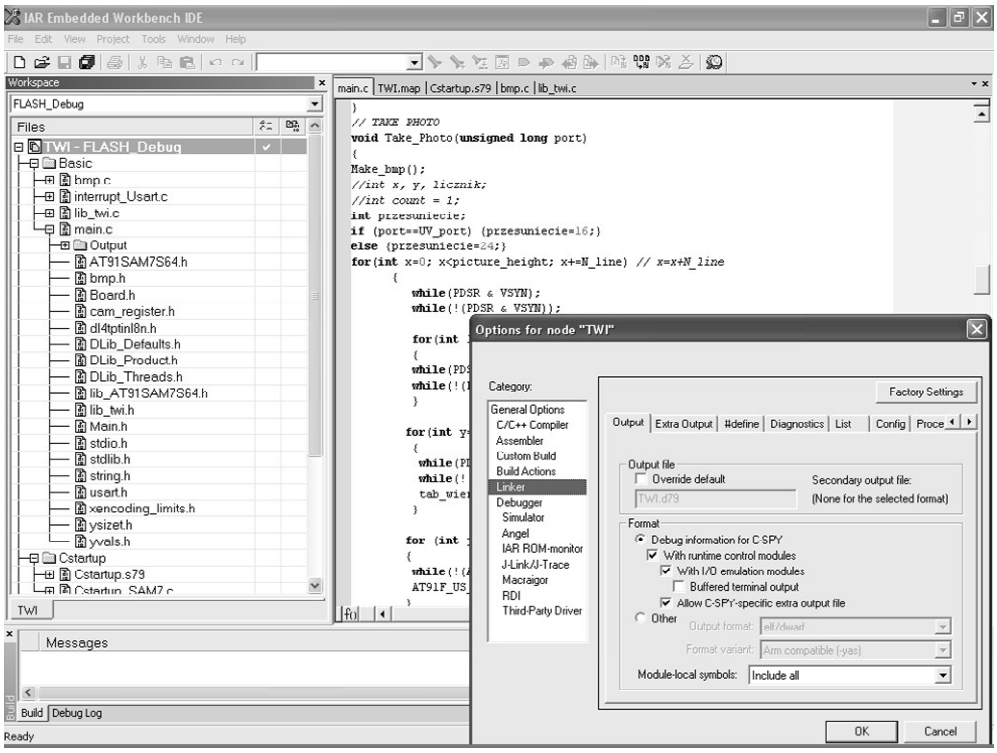


Rysunek 6.5 Schemat blokowy kalibracji

7. Środowiska programistyczne użyte w projekcie

7.1. Środowiska programowania mikroprocesorów

Do programowania i debuggowania mikrokontrolerów AT91SAM7S64 wykorzystywano środowisko CrossWorks firmy Rowley Associates Ltd. oraz IAR Embedded Workbench firmy IAR Systems (rysunek 7.1). Oba środowiska oferują pełne wsparcie dla mikrokontrolerów z rdzeniem ARM. CrossWorks jest programem wyposażonym w biblioteki, w których deklaracje nazw rejestrów i bitów rejestrów są zgodne z nazwami zawartymi w instrukcji mikrokontrolera. Ułatwia to znacznie pisanie programów szczególnie osobom zaczynającym programowanie mikrokontrolerów z rdzeniem ARM.



Rysunek 7.1 IAR – ustawienia parametrów konfiguracyjnych projektu

W programie IAR ustawianie pojedynczych rejestrów i bitów też jest możliwe, ale ich nazwy nie zawsze odpowiadają nazwom z instrukcji (np. dodatkowe litery w oznaczeniach). Program ten posiada jednak znacznie większe biblioteki funkcyjne, które są przydatne przy pisaniu dużych programów. Dodatkowym atutem tego środowiska jest większa liczba przykładowych aplikacji dostępnych na stronach producenta. Pierwsze wersje programów napisane w języku C w programie CrossWorks zostały przeniesione na platformę IAR i ostateczny program przetwarzania obrazu z kamery został napisany w tym środowisku

(Asembler i C). Obszerny opis środowiska znajduje się w helpie programu. W tym miejscu zostaną podane jedynie niezbędne ustawienia dla poprawnego zaprogramowania mikrokontrolera z wyszczególnieniem rozszerzeń plików składowych.

Przed załadowaniem programu do mikrokontrolera należy wybrać konfigurację projektu (Project - Edit Confiurations). Program będzie ładowany do pamięci FLASH, dlatego należy wybrać konfigurację FLASH_Debug. W programach dołączonych do płyty CD wszystkie parametry są już ustawione w tej konfiguracji. W przypadku problemów z programowaniem należy wybrać Project – Options (rysunek 7.1) i sprawdzić czy parametry są ustawione wg tabeli 7.1.

Category	Zakładka	Pole, które należy dodatkowo zaznaczyć (1) lub dokonać wyboru (2)
Linker	Output	(1) Allow c-SPY-specyfic extra output file
	Extra Output	(1) Generale extra output file (2) Output format - simple code
	Config	(2) Linker command file - at91SAM7S64_NoRemap.xcl
Debugger	Setup	(2) Driver - Macraigor (2)Use macro file - SAM7.mac
	Download	(1) Use flash loader(s)
Macraigor	Macraigor	(2) OCD Interface device - Wiggler

Tabela 7.1 Ustawienia dla konfiguracji FLASH_Debug

W każdym projekcie tworzonym w środowisku IAR dla mikrokontrolera AT91SAM7S64 powinny znaleźć się pliki z bibliotekami:

- AT91SAM7S64.h
- Lib_AT91SAM7S64.h

Biblioteka AT91SAM7S64.h zawiera definicję rejestrów mikrokontrolera, dzięki czemu możemy odwoływać się do nich za pomocą nazw a nie adresów. W bibliotece Lib_AT91SAM7S64.h znajdują się gotowe funkcje wykonujące różne operacje na poszczególnych grupach rejestrów. Podczas pisania programu w C często korzystano z funkcji umieszczonych w tej bibliotece. Należy jednak pamiętać, że nie wszystkie znajdujące się tam funkcje są optymalnie napisane do danego zastosowania (np. funkcja kontroli bufora RS-232). W pewnych przypadkach lepiej jest operować bezpośrednio na nazwach rejestrów, jeśli może to skrócić kod programu.

Pliki SAM7.mac i SAM7_RAM.mac zawierają niezbędne ustawienia procesora przed załadowaniem i debuggowaniem programu odpowiednio w pamięci FLASH i SRAM.

W plikach at91SAM7S64_NoRemap.xcl i at91SAM7S64_16KRAM.xcl znajdują się informacje dla linkera takie jak definicja mapowania pamięci, wielkość poszczególnych

stosów (stos przerwań, programu w danym trybie) oraz rozmiar bufora dla zmiennych alokowanych dynamicznie.

Po zlinkowaniu projektu (F7) utworzony zostaje plik o rozszerzeniu „.d79" z informacjami dla debuggera oraz plik o rozszerzeniu „.map". W pliku tym znajdują się dokładne informacje o alokacji i rozmiarach poszczególnych stałych i zmiennych segmentów programowych w pamięci FLASH i SRAM. Na końcu tego pliku znajduje się podsumowanie zawierające informację o łącznej pamięci zajmowanej przez:

- kod wykonywalny (CODE)
- dane umieszczone w pamięci SRAM (DATA)
- dane umieszczone w pamięci ROM (CONST)

Automatyczne załadowanie wszystkich plików projektu wraz z jego ustawieniami następuje po otworzeniu pliku o rozszerzeniu „.eww".

7.2. Środowiska testowe

Podczas uruchamiania i testowania systemu korzystano z kilku dodatkowych programów z zachowaniem warunków licencyjnych producenta oprogramowania. Poszczególne programy były wykorzystywane do:

- Modyfikowania programu odczytu z kamery na mikrokontrolerze Atmega32
  - WinAVR
- Analizy zawartości pliku BMP otrzymywanego bezpośrednio z kamery oraz zrzutów pamięci SRAM mikrokontrolera.
  - 010 Editor
  - Hex Workshop
- Obsługi interfejsu RS-232 (terminal)
  - RealTerm
- Programowania, kasowania i podglądu pamięci FLASH i SRAM
  - SAM Boot Assistant (SAM-BA)
  - Macraigor Flash Programmer
  - Macraigor OCD Commander

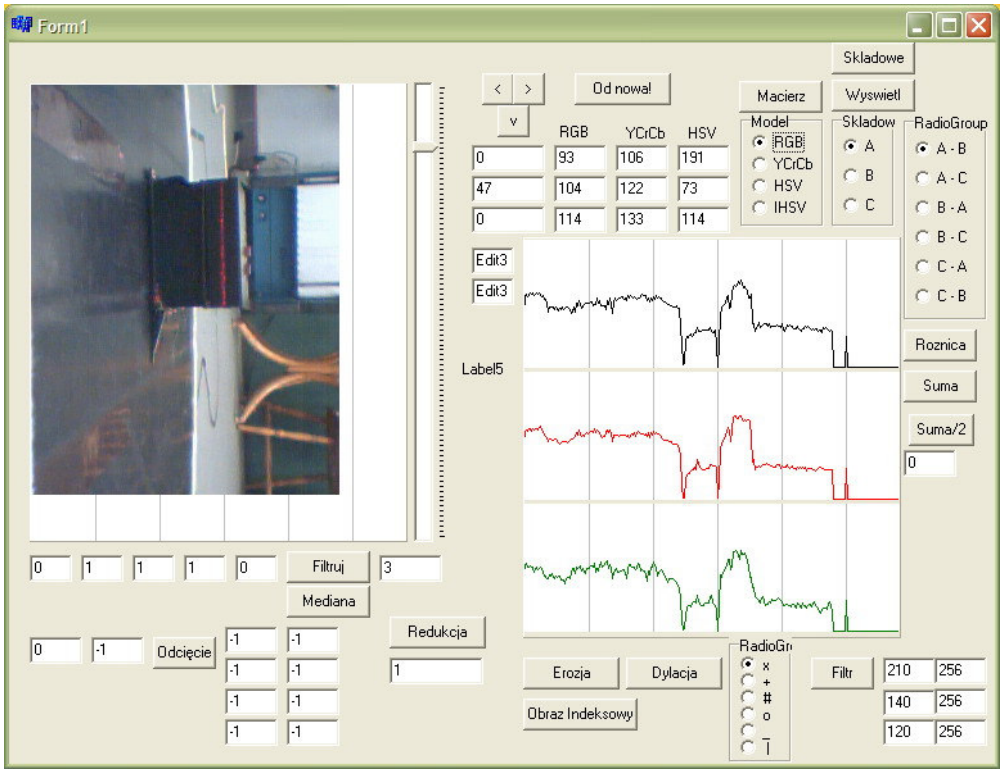
7.3. Programy symulacyjne

Podstawową przewagą programów pisanych dla komputerów klasy PC nad programami dla mikroprocesorów jest możliwość łatwego wglądu na wszystkie zmienne. Jest to spore ułatwienie zwłaszcza, jeśli możliwe jest śledzenie zmian ich wartości w trakcie normalnej pracy programu. Oczywiście mikroprocesory też mają możliwość podglądu ich pracy po przez specjalne łącza np.: JTAG i odpowiednie Debuggery. Są one jednak wolniejsze, gdyż przy każdym kroku programu wszystkie obserwowane i kontrolowane zmienne muszą być transmitowane raz w jedną, raz w drugą stronę między komputerem, a mikroprocesorem. Ta transmisja staje się uciążliwa, gdy dotyczy dużych ilości danych, jakimi są zdjęcia. Dlatego aby ułatwić pracę nad Autonomicznym systemem wizyjnym robota mobilnego, zostało zrobionych sześć programów testujących założenia dla kolejnych etapów tworzenia systemu.

Wszystkie te programy zostały napisane przy pomocy darmowej, nie komercyjnej wersji środowiska Borland® Builder C++.

7.3.1. Cam

Program o roboczej nazwie Cam służył do sprawdzenia podstawowych założeń analizy obrazu jeszcze przed pierwszym uruchomieniem kamery. Pozwala on na załadowanie pliku test01.bmp i edycję jego w trzech trybach graficznych: RGB, YCrCb i HSV.



Rysunek 7.2 Program Cam

Okno z obrazkiem służy do poglądu obecnego stanu edytowanego zdjęcia. Suwak po prawej stronie pozwala wybrać wysokość, na jakiej ma zostać zrobiony przekrój. Pionowe szare linii pozwalają zorientować się w przekroju.

Przekroje widoczne jako trzy wykresy po prawej stronie okienka przedstawiają zmiany trzech składowych zdjęcia w wybranym trybie graficznym. Kolejność wykresów jest zgodna z kolejność składowych w nazwie, czyli pierwszy wykres to R, Y lub H, w zależności od wybranego trybu. Oś pionowa jest odwrotnością wartości składowej, czyli im niższe położenie na wykresie tym większa wartość. Układ ten jest wynikiem uproszczenia budowy programu. Oś poziom odpowiada osi poziomej zdjęcia, a położenie szarych linii odpowiada liniom ze zdjęcia.

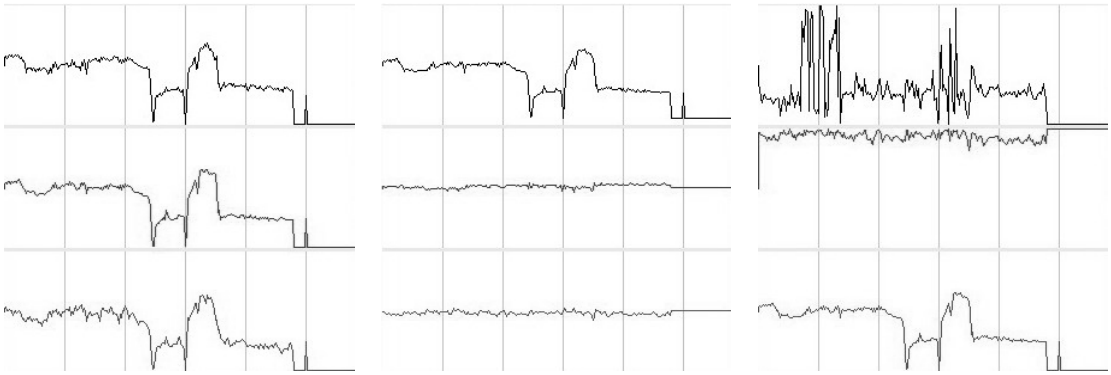
Górne pola podają położenie wybranego piksela i jego współrzędne we wszystkich trzech trybach graficznych. Pola wyboru obok pozwalają wybrać tryb graficzny, w jakim przeprowadzana jest analiza i składową, która ma być edytowana.

Pola poniżej zdjęcia służą do wyboru maski filtra, który zostanie użyty po wciśnięciu klawisza Filtruj. Jest to filtracja liniowa jednowymiarowa uśredniająca lub medianowa. Poniżej znajdują się pola do kwantowania. Dla wybranej składowej ustawia się zakres kwantowany (od – do). Następnie po prawej stronie przycisku Odcięcie dobiera się wartości, jakimi ma być zastąpiona wybrana składowa i wszystkie pozostałe wewnątrz i na zewnątrz ustawionego zakresu. Wartość -1 oznacza brak zmiany.

Ponadto program ten pozwala testować filtry morfologiczne takie jak erozja czy dylacja i różne filtry dwu wymiarowe, a także dokonywać operacji matematycznych na dwóch wybranych składowych dla całego obrazka. Ta część programu wykazała, iż pomimo dużej skuteczności tych operacji z uwagi na ich zbyt duże wymagania obliczeniowe i pamięciowe nie mogą być one zastosowane w tym systemie wizyjnym.

Program Cam umożliwił wybranie odpowiedniego trybu graficznego dla potrzeb Autonomicznego systemu wizyjnego robota mobilnego. Jak zostało wspomniane w punkcie 4.2 każdy z trzech podstawowych trybów graficznych różni się sposobem rozłożeniem informacji o pikselach na trzy składowe.

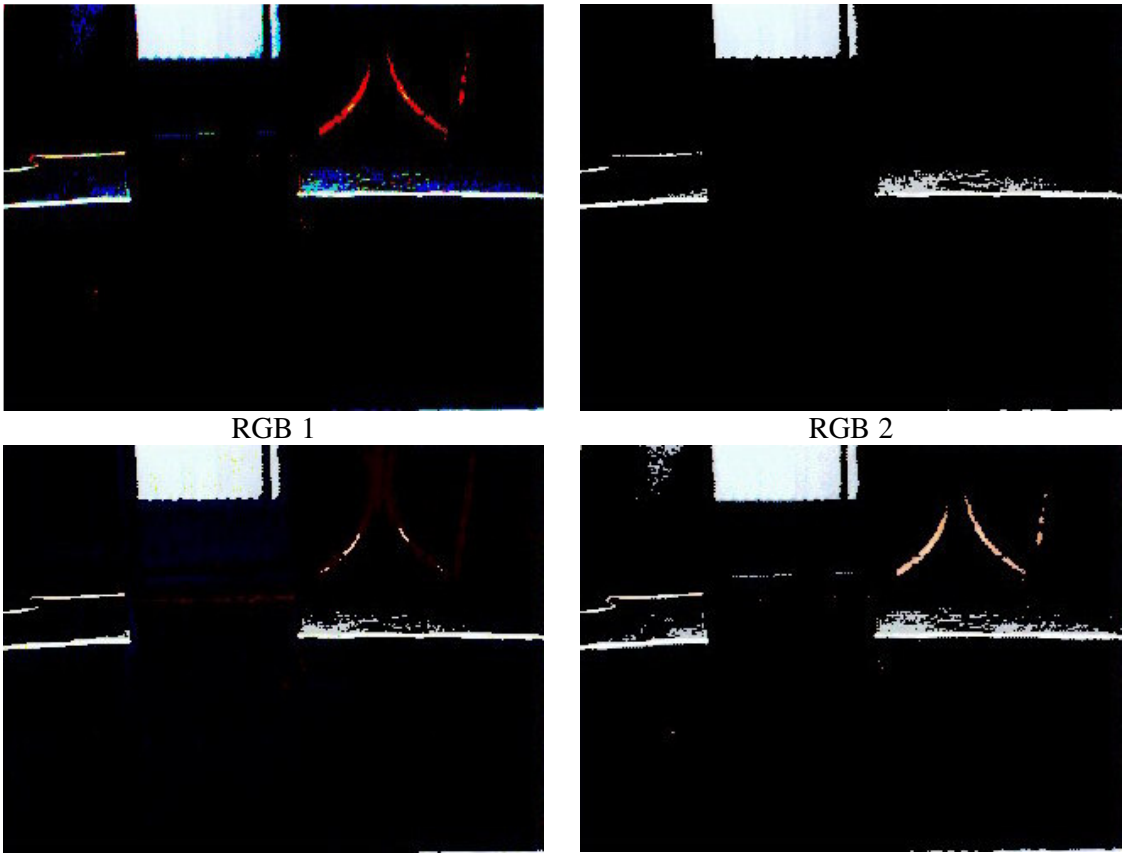
Pierwszy test dotyczył porównania zapisu białej linii na wysokości podanej na poprzednim rysunku.



Rysunek 7.3 Porównanie zapisu białej linii przez trzy tryby graficzne

Jak widać powyżej wykresy RGB mają dwa wspólne minima, czyli maksymalne wartości zmiennych R, G i B. Na wcześniejszym zdjęciu widać w tych miejscach linię i biały przewód. Te same miejsca można odczytać patrząc tylko na współrzędną Y z YCrCb lub na V z HSV. Czyli te dwa ostatnie zapisy są lepsze przy poszukiwaniu linii.

Drugi test dotyczył nakładu pracy potrzebnego do wydobycia samej linii ze zdjęcia.

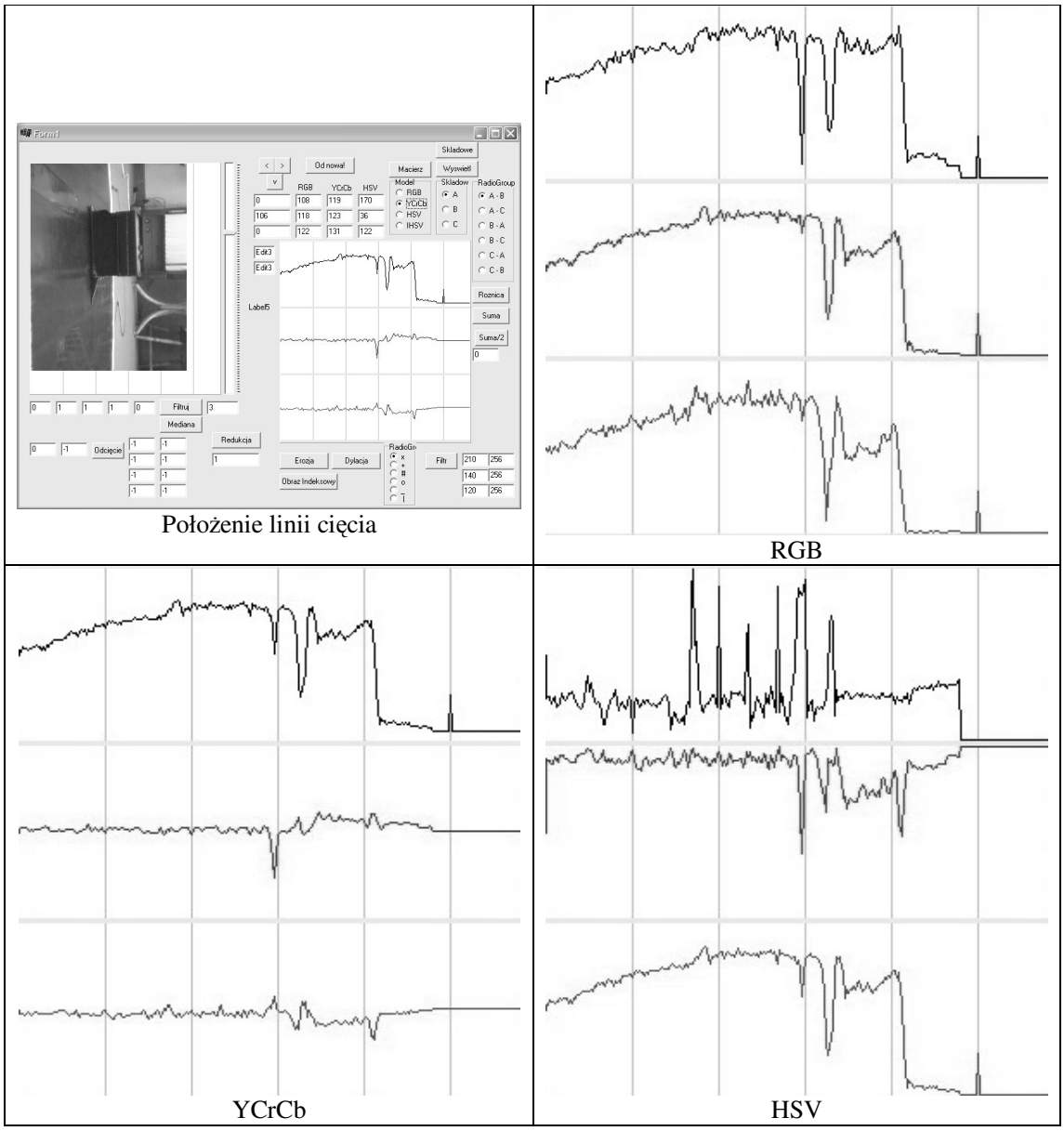


Rysunek 7.4 Porównanie sposobów wydobycia białej linii ze zdjęcia

Obrazek RGB1 przedstawia wynik odcięcia osobno każdej z trzech składowych. Ponieważ nie ich maksima są wspólne tylko dla koloru białego, a dla pozostały kolorów ich wartości znacznie różnią się od siebie, na tym zdjęciu widać sporo kolorów w niespodziewanych miejscach. W celu uniknięcia tego błędu w RGB2 zostały kolejno wyzerowane trzy składowe poza wybranym obszarem jednej z nich. Efektem tej poprawki jest pozostawienie na zdjęciu tylko odcieni szarości.

Ten sam efekt, co w RGB2 został uzyskany po przez odcięcie na składowej Y na obrazku YCrCb. Nieco inny obraz powstał przy analogicznym odcięciu V na obrazku HSV. Różnica między tymi dwoma zdjęciami wynika ze sposobu zapisu jasności. W HSV wszystkie kolory mają równy wpływ na jasność, dlatego wewnątrz przerwy w linii widać słaby pasek czerwonego lasera oraz wyraźnie widać jasnobrązowy wieszak w tle.

Trzeci test miał na celu, podobnie jak poprzednio, wybranie najlepszego trybu zapisu czerwonej linii, czyli odbitego światła lasera.



Rysunek 7.5 Porównanie zapisu czerwonej linii przez trzy tryby graficzne

Pierwsze z powyższych zdjęć przedstawia okno programu z zaznaczoną wysokością, na której został przeprowadzony przekrój. Obok na wykresie RGB widać dwa ciekawe miejsca. Wspólne minimum trzech składowych to odbicie światła od górnej części robota. Minimum pierwszej składowej, czyli R to miejsce gdzie występuje światło lasera. Czyli wyszukując ekstremum składowej R, gdy pozostałe dwie mają przeciwną wartość pozwala znaleźć kolor czerwony.

Zupełnie inaczej przedstawia się sprawa z wykresem YCrCb. Składowe chromatyczne są tak dobrane by niosły jak najmniej informacji, taka jest specyfika tego trybu zapisu. Wahania ich wartości są wyraźne dopiero przy czystych kolorach. Na tym zdjęciu widać, że składowa Cr ma wyraźne minimum w tym samym momencie, co składowa Y. W zależności

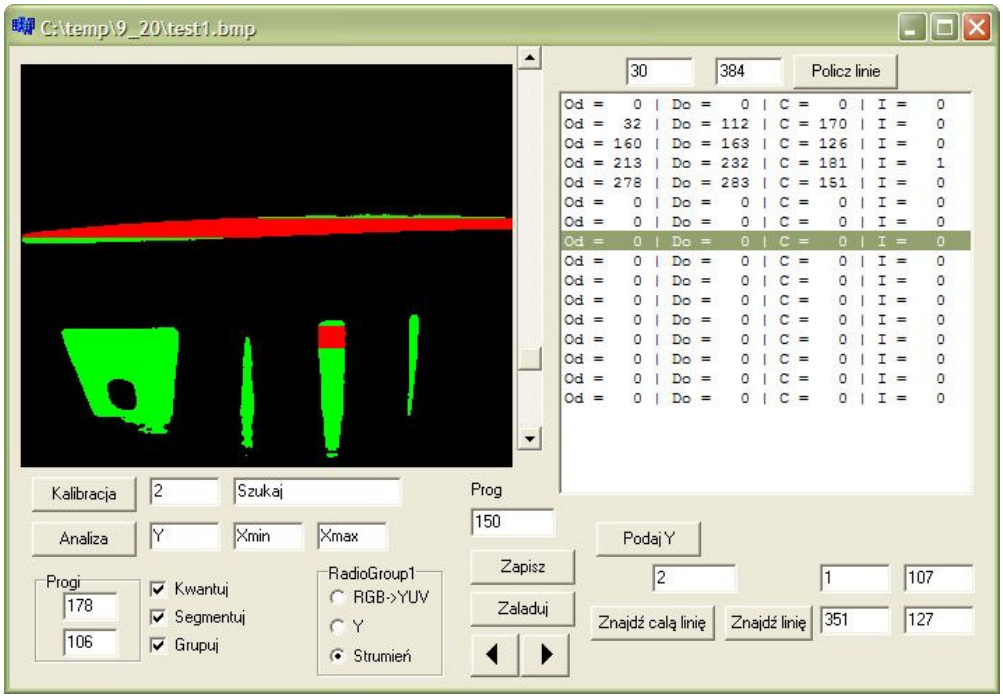
od odcienia czerwieni, składowa Cb będzie miała minimum albo maksimum, więc nie stanowi ona dobrego wyznacznika. Reasumując odczyt niezbyt czystego koloru na czarnym tle jest w tym trybie trudny, a raczej nie pewny. Dlatego w systemach używających YCrCb na ogół stosuje się markery o bardzo wyrazistych kolorach.

Ostatni wykres przedstawia tryb HSV. Składowa V wskazuje na niewielkie wahnięcie jasności, podczas gdy S, czyli nasycenie pokazuje wyraźną zmianę. Oznacza to, iż w danym miejscu jest jakiś w miarę czysty kolor. Odczytując wartość H dla przedziałów gdzie nasycenie jest znaczne można odczytać kolory. Minimalna lub maksymalna wartość H oznacza kolor czerwony. Jest tak, ponieważ kolor czerwony występuje dla zerowego kąta palety H, czyli jego odcienie zaczynają się od kąta -60º (300º), a kończą na 60º. Oznacza to, iż w tym trybie również należy obserwować więcej niż jedną współrzędną w celu znalezienia czerwonego koloru.

Podsumowując wyniki testów przeprowadzonych przy pomocy tego programu, stwierdzono, iż najlepszym trybem do poszukiwania białej linii i lasera jest YCrCb. Tryb HSV został odrzucony, ponieważ nie jest on dostępny wprost z żadnej z kamer na rynku, a jego wyliczenie jest albo niedokładne albo czasochłonne. Tryb RGB odpadł gdyż jego użycie wymaga stałej obserwacji wszystkich trzech zmiennych bez względu na rodzaj poszukiwanego obiektu

7.3.2. Kompresja

Program Kompresja był drugim programem powstałym w celu sprawdzenia założeń przed właściwym uruchomieniem kamery. Program odczytuje obraz w jednym z trzech możliwych formatów i następnie dokonuje kwantyzacji, segmentacji i grupowania. W trakcie pracy możliwy jest podgląd na wszystkie zmienne istotne w trakcie pracy. Ponadto możliwy jest podgląd na odczyt kilku obrazów przy tych samych nastawach kwantyzacji.



Rysunek 7.6 Program Kompresja

Obrazek widoczny na zdjęciu powyżej jest kwantowanym i pogrupowanym zdjęciem planszy testowej zawierającej parę białych obiektów i linię. Kolor czerwony odpowiada pikselom o jasności powyżej MaxY = 178 (Progi), a kolor zielony pikselom powyżej MinY = 106. Wymienione powyżej progi zostały policzone przez program Kalibracja, wywoływany przez przycisk Kalibracja.

Zdjęcie zostało odczytane w trybie Strumień, jest to odczyt danych z kamery zapisanych do pliku jako obraz RAW. W tym trybie program dokonuje interpolacji obrazu zgodnie z założeniami z programu opisanego w kolejnym podpunkcie. Ponadto program odczytuje zdjęcia w formacie YUV i RGB, zamieniając te ostatnie na YUV. Ten wybór pozwolił przejść kolejne etapy od zdjęcia z obcej kamery do strumienia danych z kamery właściwej.

Długa lista po prawej stronie to podgląda na tablicę Linie dla wskazanej przez suwak linii. Jak widać na zdjęciu i w poglądzie tablicy, na wybranej wysokości występują trzy obiekty, z czego tylko jeden zawiera piksele o jasności powyżej MaxY. Odcinek należący do tego obiektu został dodatkowo oznaczony w tablicy Linie indeksem = 1, co oznacza że został on przydzielony do bloku numer 1 w procesie grupowania.

Powyżej listy znajdują się dwie liczby. Większa z nich oznacza ilość segmentów wykrytych, a mniejsza ilość segmentów o jasności powyżej MaxY, czyli potencjalnych jąder bloków.

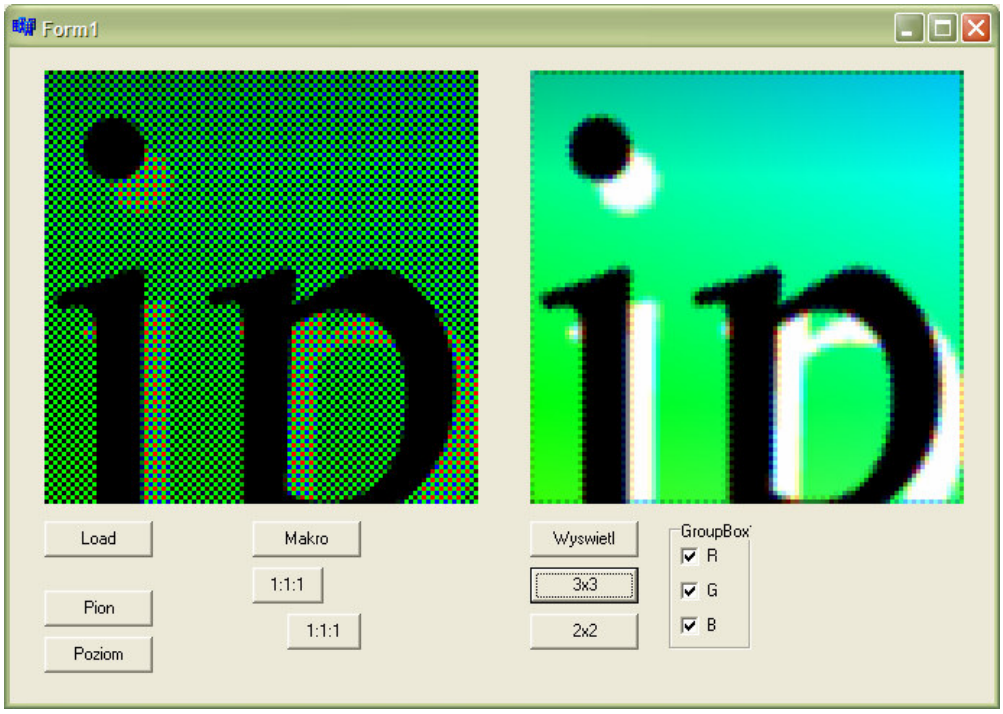
Poniżej listy jest zespół klawiszy i pól tekstowych służących do testowania algorytmu wyszukiwania linii, który został opisany w punkcie 5.4. Wynikiem poszukiwania całej linii jest blok numer 2, który zaczyna się od Y = 107, a kończy na Y = 127 i rozciąga się od X = 1 do X = 351. Jest to widoczna na zdjęciu pozioma czerwona linia.

Wynikiem pracy nad tym programem było dopracowanie podstawowych funkcji systemu wizyjnego. Udało się ustalić, iż najlepsze wyniki kwantyzacji da się uzyskać przy zastosowaniu dwóch progów, gdyż jeden wysoki próg dawał linie poszarpane lub, gdy był zbyt niski, linie które zlewały się z tłem. Segmentacja została uproszczona, ponieważ program wykazał przewagę segmentów zawierających dane tylko najjaśniejszego piksele nad osobnymi segmentami z każdego progu. Ponadto zostały sprawdzone założenia łączenia w bloki, które zostało opracowane w programie Test_zapisu.

Najważniejszym osiągnięciem programu Kompresja było oszacowanie wielkości tablic potrzebnych do przechowywania skompresowanego obrazu. Polegało to na sprawdzeniu ile segmentów i bloków przypada przy danym sposobie zapisu na przeciętne zdjęcie oraz obliczeniu ile one zajmą miejsca w pamięci.

7.3.3. Raw

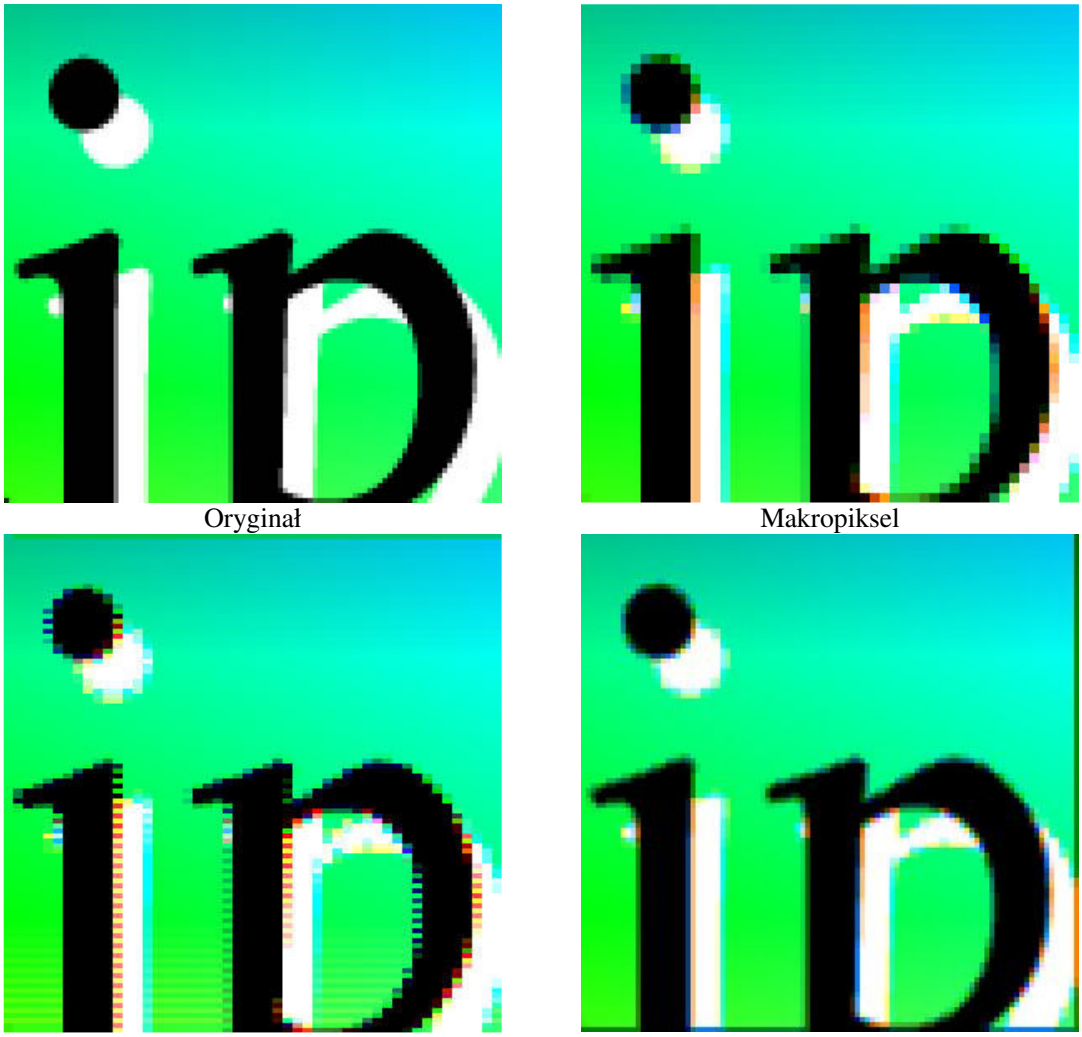
Obraz pochodzący wprost z kamery cyfrowej ma postać strumienia bajtów odwzorowujący siatkę Bayera przetwornika CMOS lub CCD. Oznacza to, że zamiast pikseli składających się z trzech składowych RGB lub YUV, są pojedyncze składowe dla każdego piksela. Aby uzyskać właściwe zdjęcie należy interpolować kolory. Dokładniej ten proces został omówiony w podpunkcie 5.2.2.



Rysunek 7.7 Program Raw

Istnieje wiele sposobów interpolacji. Najprostsza metoda bazuje na uśrednieniu brakujących składowych na podstawie elementów z sąsiednich komórek. Uśredniając wartości z pikseli ograniczonych przez siatkę 2x2 można uzyskać bardzo szybko obraz kolorowy, ale kosztem zakłamań w kolorach. Przy masce 3x3 lub większej zaczyna pojawiać się rozmycie. Dodatkowym problemem jest pojawianie się nowych kolorów po uśrednieniu. Jeśli występuje duży gradient jasności to dla pikseli, których maska uśrednienia obejmuje granicę jasności część składowych pikseli należy do obszaru jasnego, a część do ciemnego. Oznacza to, że na granicy koloru białego i czarnego może pojawić się na przykład kolor czerwony, gdy akurat tylko on z uśrednianej grupy należał do koloru białego. Ten problem jest rozwiązywany przez algorytmy interpolacji biorące pod uwagę gradient jasności.

Z uwagi na ograniczoną moc procesora i brak dostępu do całego zdjęcia podczas interpolacji, konieczne było znalezienie szybkiego algorytmu dającego najlepsze rezultaty w tych warunkach.



Oryginał Makropiksel

Poziomo 2x2

Rysunek 7.8 Porównanie różnych sposobów interpolacji.

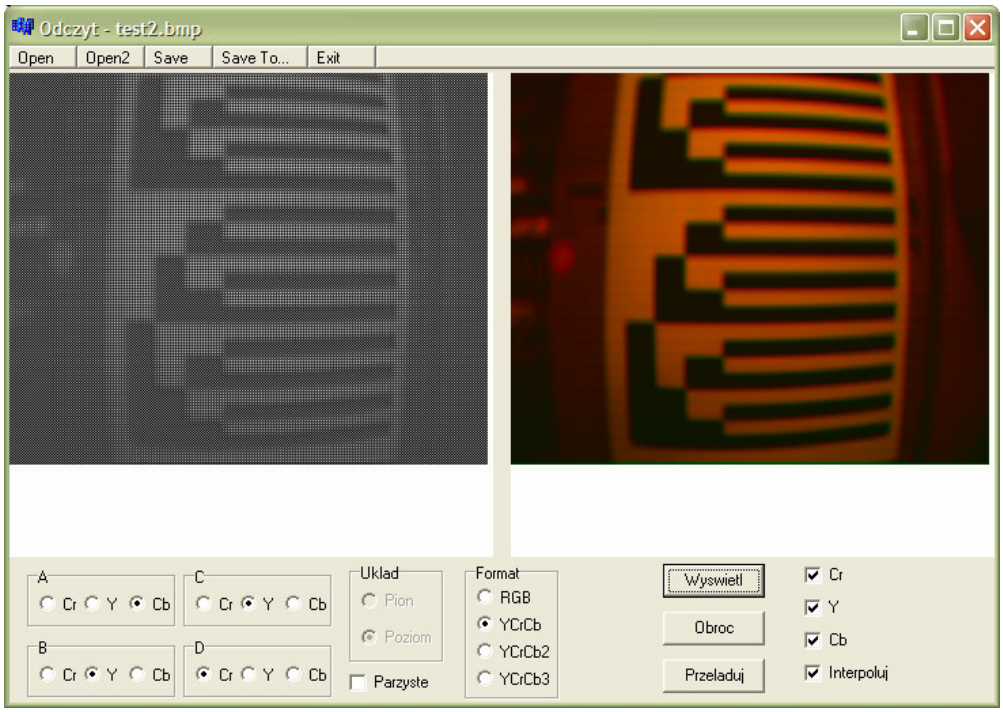
Zdjęcie oryginalne zostało zamienione na siatkę Bayera aby porównać różne metody interpolacji ze wzorcem. Ponieważ teoretyczna rozdzielczość (w tym wypadku 100x100) oznacza liczbę pojedynczych komórek składających się tylko z jednego koloru, a nie faktycznych trójkolorowych pikseli, pierwsza próba interpolacji opiera na komórkach 2x2 tworzących jeden tzw. makropiksel. Jak widać takie podejście zaowocowało zmniejszeniem rozdzielczości do 50x50 dodatkowo tworząc spore pola z fałszywymi kolorami. Drugim podejściem była interpolacja w poziomie, czyli z wykorzystaniem sąsiadów tylko z jednej strony, ale bez uśredniania. Efektem tego zabiegu jest obraz o rozdzielczości 100x50. Ostatnią próbą było uśrednienie z maską 2x2. Jak widać jest to kolorystycznie najbardziej zbliżony do oryginału, ale pozbawiony ostrych krawędzi.

Ostatecznie testy na tym programie zostały przerwane z powodu odkrycia możliwości zakupionej kamery. Kamera podaje wiersze obrazu parami, powtarzając ostatnią linijkę. Możliwe jest więc otrzymanie pełnej rozdzielczości bez zapamiętywania całego obrazu przy

uśrednieniu tylko składowej Y (G) i pobraniu pozostałych składowych makropikseli. Jest to zatem podobna metoda do przedstawionej powyżej jako makropiksel tylko, że z pobraniem tych samych składowych do czterech sąsiadujących ze sobą pikseli zamiast do jednego. Dokładnie działanie tego algorytmu zostało przedstawione w podpunkcie 5.2.2, a wynik działania interpolacji w kolejnym podpunkcie tego rozdziału.

7.3.4. Odczyt

Program Odczyt powstał w celu sprawnego odczytu plików w formacie RAW bez względu na tryb zapisu zdjęcia i układ składowych kolorów.



Rysunek 7.9 Program Odczyt

Ramka z po lewej stronie służy do podania układu składowych kolorów. Przy różnych sposobach odczytu kolejność kolorów znacznie się różni. Domyślny układ jest taki jak widać na zdjęciu. W trybie YG, w co drugiej linijce są składowe luminescencyjne, a w pozostałych chromatyczne. Te i dowolne inne kombinacje należy w tym miejscu podać by zdjęcie RAW zostało poprawnie odczytane.

Pole wyboru formatu pozwala odczytywać zdjęcie w formacie RGB lub kilku różnych odmianach YCrCB, różniących się wagami składowych chromatycznych.

Pola wyboru po prawej stronie pozwalają wybrać, które składowe obrazu mają zostać wyświetlone. Ta opcja była bardzo ważna przy ustalaniu przyczyn, dla których we wstępnych

etapach pracy kamera nie działała prawidłowo. Ostatnia opcja to interpolacja wspomniana w poprzednim podpunkcie.

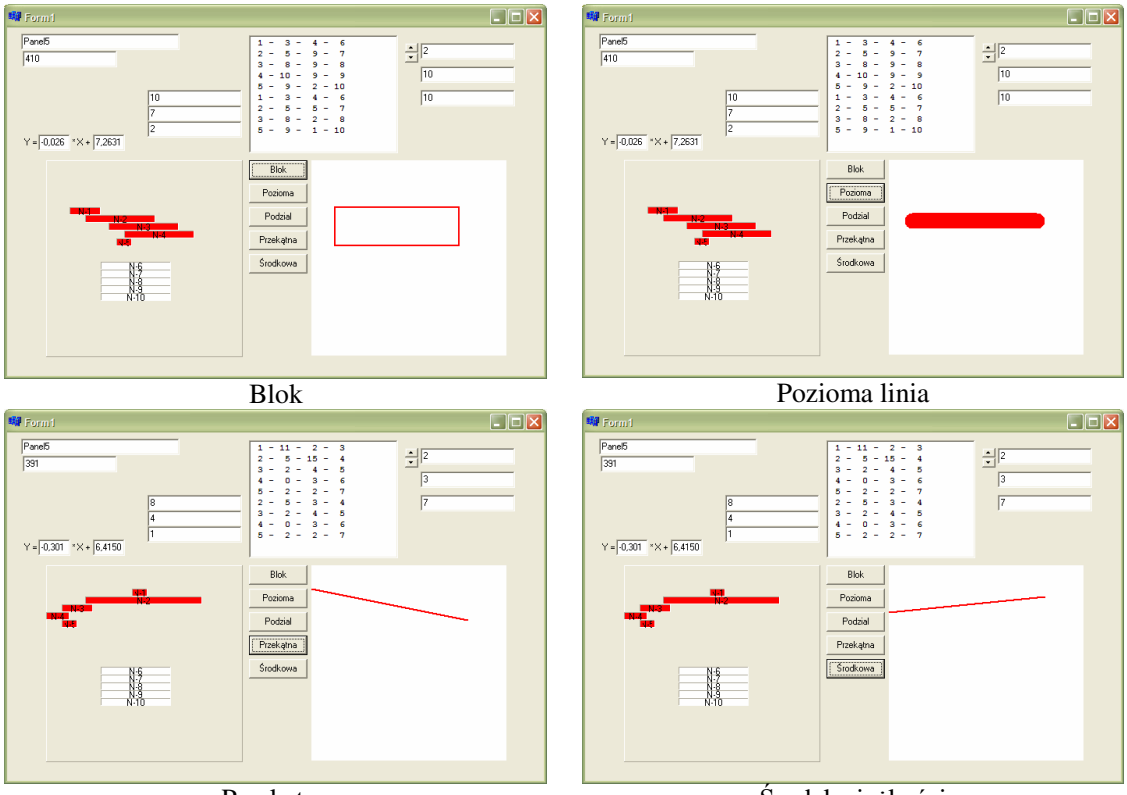
Klawisz Wyświetl wyświetla ponownie zdjęcie po zmianie wybranych składowych. Obróć przekręca zdjęcie o 90º. Jest to przydatne, gdyż w początkowym etapie prac łatwiej było uzyskać z kamery zdjęcie pionowe niż poziome, a ostatecznie kamera została zamontowana do góry nogami, aby rozpoczynać skanowanie od dołu. Ostatni klawisz służy do przeładowania zdjęcie w przypadku, gdy został źle wybrany układ składowych piksela lub, gdy chce się zmienić format odczytu zdjęcia.

Przyciski na górnym pasku pozwalają na odczyt zwykłego zdjęcia lub zdjęcia z osobno zapisanymi składowymi chromatycznymi i luminescencyjnymi oraz na zapis zdjęcia przerobionego pod domyślną nazwą, lub dowolnie wybraną. Domyślna nazwa zdjęcia to <nazwa oryginalna>_.bmp. Ten format pozwala na szybki zapis obok zdjęcia RAW jego odpowiednika w formie RGB po interpolacji.

Po wyjściu z programu przy pomocy przycisku Exit program zapisuje do pliku tekstowego wybrane ustawienia programu. Przy kolejnym włączeniu, jeśli zostanie wykryty ten plik, program wczyta z niego ustawienia, w przeciwnym wypadku załaduje ustawienia domyślne.

7.3.5. WSP

Po otrzymaniu obrazu w formie skompresowanej, czyli zbioru odcinków, należy je scalić w jeden blok lub wektor. Forma zapisu powinna zawierać jak najwięcej informacji o scalonych segmentach i względnym położeniu przy zachowaniu jak najmniejszej objętości danych. Program WSP służył do sprawdzania na modelu jak różne sposoby opisu zachowują się dla różnych układów segmentów.



Rysunek 7.10 Program WSP

Pole po lewej stronie z czerwonymi odcinkami służy do budowania z odcinków (o zmiennej długości) testowej grupy segmentów. Przyciski po środku pozwalają wybrać jeden z kilku sposobów opisu bloku, a okno po lewej pokazuje jak wygląda model narysowany na podstawie opisu.

Pola tekstowe w górnej części wyświetlają różne zmienne używane przez algorytmy opisujące blok. Ich znaczenie było istotne tylko dla sprawdzenia poprawności działania programu.

Najprostszy sposób opisu grupy odcinków to zbudowanie prostokąta opisanego na nich. Prostokąt, zwany dalej Blokiem, wymaga podania tylko współrzędnych dwóch przeciwległych wierzchołków, których obliczenie wymaga jedynie znalezienia skrajnych

współrzędnych segmentów. Wadą bloku jest zakrywanie swoją powierzchnią obszarów, które nie przynależą do wybranych segmentów. Jest to najbardziej widoczne, gdy blokiem zostanie opisany odcinek nachylony pod kątem 45º. Bez względu na jego grubość blok opisujący będzie miał te same wymiary.

Drugą techniką, wymyśloną na potrzeby tej pracy, jest aproksymacja linią poziomą, a dokładniej poziomym prostokątem. Linia ta znajduje się dokładnie pośrodku prostokąta opisanego na wybranych segmentach, zaczyna i kończy się na skrajnych współrzędnych poziomych segmentów, a jej grubość jest zależna od łącznej ich „masy". Masa jest liczona na podstawie długości odcinków. Czyli każdy piksel jest jedną jednostką, a łączna długość segmentów masą. Ten zapis pokazuje odkąd, dokąd jest obiekt i odzwierciedla jego grubość, ale w przypadku odcinków skośnych zupełnie nie odzwierciedla obszaru, na którym się znajdują.

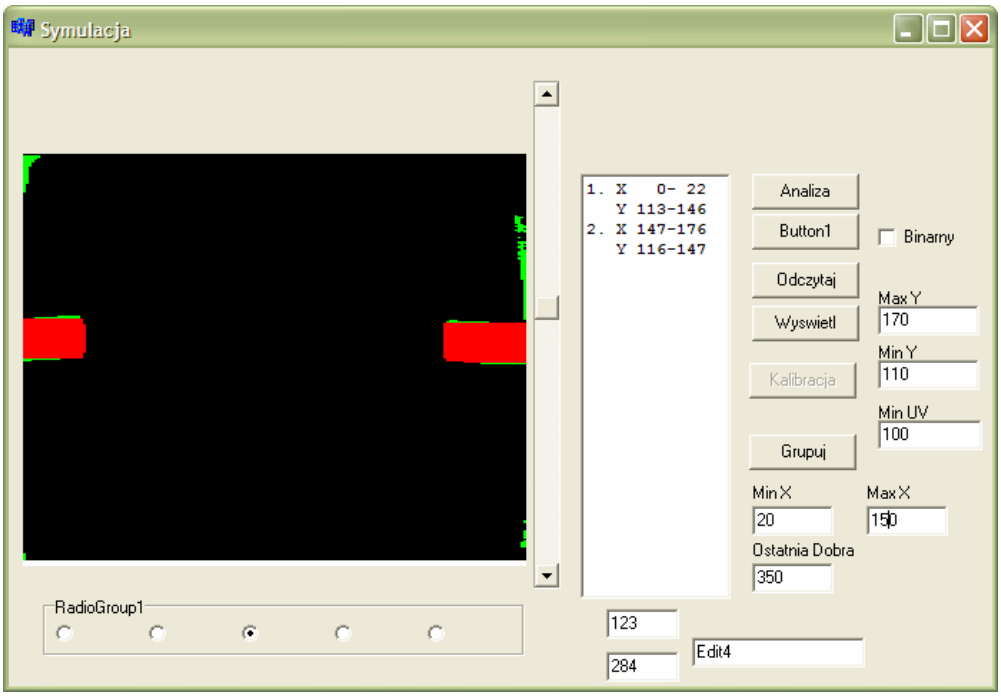
W ramach próby rozwiązania problemu ukośnych odcinków została sprawdzona koncepcja opisu grupy segmentów ich przekątną. Niestety wybór odpowiedniej przekątnej przy różnych, czasem skomplikowanych układach segmentów, był nieskuteczny. Powyższe zdjęcie pokazuje jeden z bardziej absurdalnych wyników. Ta technika po dopracowaniu mogłaby dobrze odzwierciedlać położenie segmentów, ale aby tego dokonać trzeba by tworząc blok przeliczać współczynniki wszystkich składowych, a to jest czasochłonne.

Najlepszą techniką opisu jest analiza środka ciężkości bloku. Pierwszym etapem jest ustalenie środka ciężkości całego bloku. Następnie po podzieleniu poziom bloku na wysokości jego środka ciężkości należy wyznaczyć analogicznie środek ciężkości jednej z otrzymanych połówek. Odcinek poprowadzony przez tak otrzymane punkty odzwierciedla wypadkowe nachylenie bloku. Niestety, gdy blok składa się z długich odcinków, położonych mniej więcej jeden nad drugim, wypadkowy odcinek jest pionowy, mimo iż wyraźnie widać, iż jest poziomy. Dalsze przeliczenia poprawnego opisu wymagają równań trygonometrycznych, które są prawie nie osiągalne dla procesora ARM7S.

Ostatecznie z uwagi na szybkość działania i największą nieomylność został wybrany opis przy pomocy bloku. Jest to też najczęściej używany system w wizji robotów. W bardziej zaawansowanych systemach oprócz współrzędnych dwóch wierzchołków i koloru używa się też masy składowych, wypadkowego nachylenia i innych przydatnych w danym projekcie cech.

7.3.6. Test_zapisu

Ostatnim symulatorem jest program o roboczej nazwie Test_zapisu. Po skończeniu całego algorytmu analizy obrazu, musiał on zostać sprawdzony w praktyce. Program Test_zapisu umożliwia sprawdzenie wyników działania każdej ze składowych Autonomicznego systemu wizyjnego robota mobilnego na zdjęciu pobranym z kamery lub na podstawie danych odczytanych z pamięci SRAM.



Rysunek 7.11 Program Test_zapisu

Nazwa Test_zapisu pochodzi od pierwszej funkcji dodanej do tego programu, czyli do próby odtworzenia zdjęcia na podstawie jego obrazu w formie tablic zapisanych w pamięci SRAM mikroprocesora. Po wciśnięciu przycisku Odczytaj, program odczytuje plik binarny o wielkości 16 KB, który jest dokładną kopią pamięci mikroprocesora. Program Test_zapisu używa identycznych funkcji i deklaracji jak właściwy system, dzięki temu odczytuje zapisane tablice tak jakby były jego własnymi. W ten sposób zostały usunięte wszelkie błędy w zapisie zdjęcia. Następnie program wyświetla odczytanie dane i podaje wnioski wysnute przez mikroprocesor, czyli: położenie przerwy wysokość, na jakiej jest linia, poziomy kalibracji i sugerowany kierunek zwrotu (kropka na pasku pod zdjęciem).

Drugą cechą programu jest symulowanie pracy mikroprocesora. Jeśli zostanie odczytane zdjęcie typu BMP zamiast kopi pamięci BIN, program wyświetli zdjęcie w

oryginale. Następnie po wciśnięciu klawisza Kalibruj program postara się, podobnie jak mikroprocesor, ustalić próg MaxY i MinY. Jeśli na zdjęciu nie będzie ciągłej linii program może wysnuć błędne wnioski, dlatego pierwsze zdjęcie powinno zawierać wyraźną linię. Po wykalibrowaniu, program przechowuje ustawione progi i stosuje je dla każdego kolejnego zdjęcia. Kolejnym krokiem symulacji jest Grupowanie. Program kwantuje i segmentuje zdjęcie, a następnie dokonuje jego grupowania. Ponieważ pierwsze dwa procesy zostały już dobrze przetestowane w poprzednich programach, na ekranie w formie listy wyświetlane są tylko wyniki grupowania. Obraz przetworzony zostaje wyświetlony w miejscu oryginalnego. Na powyższym zdjęciu widać przykład tego działania. Dwa obiekty jasne zostały uznane za fragmenty białej linii, a odblaski z tła zostały zignorowane gdyż nie zawierały pikseli jaśniejszych niż MaxY. Po wciśnięciu klawisza Analiza program przeprowadza analizę wstępnie przetworzonego zdjęcia, a dokładniej listy bloków. Wynikiem analizy jest podanie przez suwak górnej krawędzi wykrytej białej linii i położenia przerwy (kropka na pasku pod zdjęciem). Identyfikacja fragmentów linii przeprowadzana jest na podstawie podanego położenia ostatniej dobrej linii (w tym wypadku jest ona wyzerowana – 350) i marginesów (Min X=20, Max X=150).

Poza wykryciem większości drobnych błędów w algorytmie, program Test_zapisu wykazał konieczność dopracowania systemu decyzyjnego. Regulacja marginesów i dopracowanie rozróżniania konieczności celowania od atakowania, ze względu na szerokość i położenie przerwy, jest niezbędna do prawidłowego działania systemu wizyjnego. W obecnej testowej postaci nie zawsze decyzje podejmowane przez system były słuszne. Kwestia regulacji zostaje jednak pominięta gdyż jest zależna od parametrów użytego robota. Robot bardzo szybki powinien dokładnie celować nawet, gdy przeciwnik jest daleko, podczas gdy robot wolny nie musi się przejmować celowaniem z daleka, gdyż zanim dojedzie do przeciwnika sytuacja na ringu na pewno ulegnie znacznej zmianie. Analogicznie sprawa ma się dla różnych szerokości i zasięgów widzenia kamery. Im mniejsze pole widzenia, tym ważniejsze jest szybkie podejmowanie decyzji.

8. Porównanie z istniejącymi systemami

8.1. Sonar

Czujniki ultradźwiękowe, zwane od ich najpopularniejszej odmiany – sonarami, służą do wykrywania obiektów na średnich odległościach. Poza wykrywaniem przeszkód możliwy jest również pomiar odległości. Maksymalny zasięg czujników ultradźwiękowych jest uzależniony od prędkości rozchodzenia się dźwięku w danym ośrodku i współczynnika jego rozpraszania. Największa skuteczność jest w ośrodkach o dużej gęstości i małej sprężystości.

Układy czujników ultradźwiękowych można podzielić ze względu na złożoność na:

- Pojedyncze czujniki zbliżeniowe
- Układy czujników zbliżeniowych
- Skanery powierzchni – sonary

Czujniki zbliżeniowe do pomiaru odległości wykorzystują pomiar czasu między wysłaniem paczki sygnału dźwiękowego, o ściśle określonej częstotliwości, a jego odbiorem. W warunkach normalnych prędkość dźwięku jest stała i wynosi dla powietrza ok. 344 m/s. Iloczyn prędkości dźwięku i czasu przebytego przez wysłany sygnał daje w wyniku odległość.

Jakość odebranego sygnału, czyli amplituda, zależy od faktury powierzchni, wielkości i w dużej mierze od jej wypadkowego nachylenia względem czujnika. Fale dźwiękowe podobnie jak świetlne ulegają zjawiskom rozproszenia i odbicia, zatem najlepiej widoczne są powierzchnie gładkie i równe, na które pada wiązka prostopadle. Dlatego pojedyncze czujniki nadają się tylko do pomiarów na wprost.

Układy czujników zbliżeniowych składają się na ogół z kilku odpowiednio połączonych pojedynczych czujników. Rozróżnia się zasadniczo dwa sposoby działania takich układów: sekwencyjny i zbiorczy. W pierwszym przypadku każdy z czujników kolejno dokonuje pomiaru. Obraz wynikowy jest zbiorem pojedynczych wyników obarczonych opisanymi wcześniej problemami. Drugi sposób przewiduje wysłanie sygnału przez wszystkie czujniki naraz i odbiór symultaniczny. W tym przypadku każdy czujnik odbiera poza swoim sygnałem również sygnały sąsiadów, które zostały rozproszone przez przeszkody. Daje to lepszy obraz gdyż, przy odpowiednim ustawieniu odbiorników, wykrywane są powierzchnie o różnych nachyleniach, a nie tylko prostopadłe.

Zwiększenie liczby czujników i ustawienie ich pod wszelkimi możliwymi kątami, najlepiej w znacznej odległości od siebie umożliwia w miarę dokładnie skanowanie powierzchni. W sonarach na ogół stosuje się jeden silny nadajnik i wiele odbiorników. Dzięki

zastosowaniu komputera możliwe jest zestawienie wyników pomiarów wszystkich odbiorników w obraz trójwymiarowy lub dwuwymiarowy w przypadku skanerów liniowych.

W zawodach sumo robotów zwykle stosuje się małe zestawy nadajnik-odbiornik pracujące przy częstotliwości 40 kHz. Jednym z popularniejszych jest SRF10 firmy Robot-Electronics. Układ ten sterowany jest przez I2C i przesyła na żądanie wynik pomiaru odległości w wybranych jednostkach, na przykład w centymetrach. Przy maksymalnym wzmocnieniu sygnału 700x i zwłoce 65 ms wykrywa niewielkie obiekty z odległości 6 m. Jego minimalny zasięg wynosi 3 cm. Według producenta, teoretycznie zasięg maksymalny może wynosić nawet 11 m, jednak w praktyce największą dokładność układ osiąga dla odległości 1 m. Różne odmiany tego czujnika różnią się zbieżnością wiązki nadawczej, która wacha się od 15º do 75º. Na ogół stosuje się dwa takie układy umieszczone obok siebie tak, aby różnica w podawanych przez nie odległościach wskazywała lokalizację przeszkody.

Z uwagi na wygórowaną cenę tego układu (62$) na potrzeby poprzednich projektów został zaprojektowany, przez autorów tej pracy, układ podobny. Składa się z nadajnika 40ST-16 i odbiornika 40SR-16, wzmacniacza nadawczego MAX232 i odbiorczego TS272. Układ nadawczy zamienia sygnał prostokątny 0-5 V pochodzący z zewnętrznego generatora na przebieg ok.12 VAC. Układ odbiorczy wzmacnia odebrany sygnał 800 razy i wysyła go do zewnętrznego komparatora. Moduł współpracuje z układem sterującym wyposażonym w jeden mikroprocesor AT90S2313 generujący przebieg prostokątny 40 kHz i odbierający sygnał od komparatorów, które zamieniają sygnał analogowy na binarny. Dzięki zastosowaniu programowalnego ograniczenia odległości gotowy układ wykrywa obiekty z odległości nie większej niż 92 cm i unika fałszywych silnych odbić od ścian położonych nie dalej niż 3 m od niego. Tak dobrane ograniczenie pozwala uniknąć fałszywy odczytów w warunkach zawodów sumo robotów przy zachowaniu szybkości pracy.

8.2. Podczerwień

Czujniki podczerwieni są często wykorzystywane w systemach wykrywania obiektów na niewielkich odległościach. Wśród układów nadawczo-odbiorczych podczerwieni wykorzystywanych do detekcji obiektów można wyróżnić:

- Układy bez modulacji i kodowania w oparciu o fotodiody i fototranzystory
- Układy z modulacją i kodowaniem sygnału i wyjściem cyfrowym
- Czujniki odległości z wyjściem analogowym typu GP2D i GP2Y (Sharp)
- Czujniki przemysłowe (specjalizowane)

Układy bez modulacji i kodowania zbudowane na pojedynczych fotodetektorach bez dodatkowych filtrów są rzadko stosowane. Powodem tego jest ich duża wrażliwość na zakłócenia wywoływane przez źródła promieniowania podczerwonego (np. nadajniki od sprzętu RTV, silne promieniowanie słoneczne i promieniowanie z innych źródeł o dużej zawartości składowej podczerwonej). Układ taki zbudowany jest z nadajnika, którym jest dioda emitująca światło podczerwone oraz odbiornika w postaci fotodiody lub fototranzystora. Fotodetektor jest najczęściej włączony w obwód bazy tranzystora, co pozwala na bezpośrednie sterowanie tranzystorem a tym samym stanem sygnału na wyjściu odbiornika.

Układy z modulacją i kodowaniem o wyjściu cyfrowym są często wykorzystywane w sprzęcie RTV. Dzięki wprowadzeniu modulacji sygnału układ jest odporny na zakłócenia spowodowane przez promieniowanie podczerwone o innej częstotliwości niż sygnał generowany. Dodatkowo wprowadzone kodowanie pozwala na odbiór tylko tych sygnałów o danej częstotliwości, które nadajnik faktycznie wysłał (sygnałów o określonym kodzie). Ramka kodu RC5 składa się z 14 bitów, których wysłanie zajmuje w przybliżeniu 25 ms. Sześć ostatnich bitów w ramce to numer rozkazu a pięć wcześniejszych to adres urządzenia. Taki system kodowania pozwala na uzyskanie do 2048 (2^{11}) różnych poleceń. Poza protokołem RC5 stosowane są podobne protokoły transmisji w podczerwieni innych producentów, m.in. Nokia NRC17, JVC Protocol, Sharp Protocol, czy Sony SIRC.

Istnieje ponadto protokół ITT, który nie wykorzystuje modulacji a jedynie kodowanie poprzez wysyłanie impulsów w odpowiednich odstępach czasowych (np. 100 μs dla stanu 0, 200 μs dla stanu 1). Pomimo dużej szybkości tego protokołu (1,7÷ 2,7 ms na przesłanie rozkazu) jest on rzadko stosowany ze względu na małą odporność na zakłócenia.

Czujniki odległości z wyjściem analogowym firmy Sharp (typu GP2D12, GP2Y0A02YK) są kompletnymi modułami zawierającymi w jednej strukturze nadajnik odbiornik oraz układy odpowiedzialne za kondycjonowanie sygnału. Na wyjściu takiego czujnika jest sygnał napięciowy proporcjonalny do odległości. W deklarowanym przez producenta zakresie pomiarowym wynoszącym 10÷80 cm dla GP2D12 (20÷150 cm dla GP2Y0A02YK) napięcie na wyjściu zmienia się w zakresie 2,5 V ÷ 0,4 V. Nie jest to jednak zależność liniowa [29]. Zastosowany przetwornik wielopunktowy w części odbiorczej oraz pomiar kąta promieniowania padającego pozwalają na uzyskanie dużej czułości i dokładności. Według producenta czujniki te są mało wrażliwe na zmiany koloru i refleksyjności obiektów, co teoretycznie jest ich przewaga nad konstrukcjami amatorskimi. Czujniki te nie były jednak badane.

Przemysłowe czujniki podczerwieni wykonywane są najczęściej jako:

- Bariery jednokierunkowe
- Bariery refleksyjne
- Czujniki odbiciowe
- Czujniki odbiciowe z eliminacją tła
- Czujniki odległości

Czujniki te charakteryzują duża odpornością na zakłócenia i są przystosowane do pracy w trudnych warunkach środowiskowych (przemysł). Ich wysoka niezawodność ma wpływ na wyższą cenę w porównaniu do czujników stosowanych w sprzęcie RTV. Posiadają dodatkowe układy kondycjonowania sygnału w swojej strukturze i dużą rozdzielczość (dokładność) pomiaru (dziesiętne i setne części mm). Napięcia zasilania są charakterystyczne dla automatyki przemysłowej i najczęściej wynoszą 10÷30 V DC lub 20÷250V AC. Duża rozdzielczość pomiaru i często większy zakres pomiarowy przekładają się na większy pobór energii (średnio powyżej 100 mA dla napięcia zasilania 10÷30 V). W robotyce mobilnej, jeżeli stosowane są takie czujniki to są to najczęściej czujniki odległości.

W celu porównania z system wizyjnym opisanym w tej pracy został wybrany układ z modulacją i kodowaniem sygnału i wyjściem cyfrowym. Podobnie jak omawiany powyżej sonar, czujnik podczerwieni został specjalnie zaprojektowany i zbudowany na potrzeby tej pracy przez jej autorów. Układ pracuje w standardzie transmisji RC5 (Philips). Składa się z dwóch par nadajnik-odbiornik pozwalających wykryć obiekt w trzech położeniach: na wprost, po prawej stronie i po lewej stronie robota. Pojedynczy układ nadajnika stanowią dwie diody podczerwone kluczowane z tranzystora, na którego bazę podawany jest przebieg zmodulowany z generatora 36 kHz. Jako generator został tutaj użyty mikrokontroler AT90S2313, który pełni również funkcję kodera w kodzie RC5. Moc nadajnika może być regulowana przez potencjometr włączony szeregowo z diodami podczerwieni. Jako odbiornik podczerwieni użyto zintegrowany układu TSOP 1736 (SFH 506-36). Zawiera on w swojej strukturze m.in. fotodetektor (dioda PIN), filtr środkowo przepustowy i demodulator. Filtr środkowo przepustowy przenosi bez tłumienia tylko sygnały o częstotliwości 36 kHz. Po przejściu sygnału przez demodulator i komparator na wyjściu układu TSOP 1736 pojawia się sygnał dyskretny, który podawany jest na wejście mikrokontrolera Atmega8, pełniącego funkcję dekodera rozkazów w kodzie RC5. Układ został zaprogramowany tak, że w danej chwili sygnał jest wysyłany i odbierany tylko przez jeden moduł czujnika. Każdy moduł wysyła sygnał o innym kodzie, po którym mikrokontroler identyfikuje, który czujnik

podczerwieni zadziałał. Jest to kodowanie stałe, bo dany czujnik podczas transmisji zawsze wysyła ten sam kod.

8.3. Testy

Wszystkie rodzaje czujników mają swoje słabe strony. Istnieją nietypowe układy przeszkód, które są niewykrywalne oraz obiekty, które będąc w tle zakłócają pracę czujników. W ramach oceny niezawodności Autonomicznego systemu wizyjnego robota mobilnego, został wybrany zestaw testów składający się z takich specyficznych przypadków dla omówionych powyżej dwóch systemów i wizji.

Ultradźwięki

- Równoległa ściana – W naszym otoczeniu występuje wiele obiektów stanowiących „zwierciadła" dla dźwięków. Wiązka dźwiękowa w powietrzu ulega znacznemu rozproszeniu, więc pomiar na dużą odległość jest prawie niemożliwy jednak duże, równoległe do nadajnika płaszczyzny świetnie odbijają sygnał i pomimo dużej odległości są „widziane" równie dobrze jak małe bliskie obiekty. W normalnych warunkach sytuacje, gdy czujnik jest dokładnie naprzeciw ściany są rzadkie, jednak ich wystąpienie zakłóca pracę czujnika przystosowanego do pomiaru tylko małych odległości.
- Nachylenie 45º – Aby sygnał wysłany został odebrany musi po odbiciu wrócić do nadajnika. Płaska, gładka powierzchnia stanowi lustro akustyczne. Jej nachylenie pod kątem większym niż 45º powoduje, że większość sygnału wysłanego nie wraca do nadajnika czyniąc przeszkodę niewykrywalną.
- Kąt prosty – Skrajnym przypadkiem nachylenia są dwie powierzchnie ustawione do siebie pod kątem prostym na przykład równoległobok. W momencie, gdy zestaw czujników wycelowany jest w grzbiet takiej bryły część wiązki odbija się zawsze pod kątem większym niż 45º. W skrajnych przypadkach, przy zastosowaniu tylko dwóch czujników, zwykłe pudełko może w zależności od kąta być albo nie widzialne albo odbijając tylko jedną wiązkę udawać, iż znajduje się z boku będą dokładnie na wprost.

Podczerwień

- Kolor czarny – Teoretycznie współczynnik odbicia światła zależy od koloru. Kolor czarny odbija najmniej światła ze wszystkich kolorów. Niektóre czernie

nie odbijają nawet podczerwieni czyniąc pomalowaną powierzchnię trudną do wykrycia.

- Podobny system – Wiązka odbita ma znacznie mniejszą energię niż wysłana, gdyż ulega po drodze rozproszeniu. Inny system działający na tej samej zasadzie może oślepić swoją wiązką czujniki. W skrajnym przypadku, gdy dwa układy używające tego samego nośnika i sposobu transmisji mogą wzajemnie uniemożliwić sobie pracę.
- Światło – Silne źródło światła może oślepić czujniki optyczne. Czasem modulacja nie chroni przed takim zakłóceniem pracy układu.

Wizja

- Paski – Ponieważ omawiany w tej pracy system wizyjny poszukuje przerwy w białej linii, odpowiednio dobrane malowanie powierzchni może go zmylić. Możliwe jest nawet zablokowanie pracy systemu przez podanie zbyt dużej ilości potencjalnych obiektów, gdyż system ma ograniczoną ilość odczytywanych bloków.
- Niski – Zbyt niski przeciwnik w pewnym ustawieniu może znaleźć się poniżej minimalnej linii wykrycia. Taka sytuacja dotyczy wszystkich typów czujników, nazywana jest kątem widzenia pionowego.
- Fałszywa linia – Na zawodach sumo robotów kilku zawodników próbowało oszukać czujniki przeciwnika malując na zderzaku fałszywą linię. Teoretycznie robot po wykryciu tej fałszywej linii zacznie uciekać przed przeciwnikiem zamiast go atakować.
- Jasne tło – Omawiany w tej pracy system wizyjny poszukuje przerwy w białej linii. Aby biała linia została wykryta musi ona odróżniać się jasnością od tła. Przy pewnym nachyleniu kamery tłem dla linii zamiast podłogi staje się ściana, która często bywa biała.

Poza przypadkami szczególnymi zbadane zostały również podstawowe cechy czujników, takie jak zasięg maksymalny i minimalny oraz kąt widzenia w poziomie.

Wszystkie testowane czujniki pochodzą z robota autorów tej pracy i zostały przez nich zaprojektowane i wykonane, dlatego ich porównanie z autorskim systemem wizyjnym jest bardziej wiarygodne od porównania z profesjonalnymi czujnikami.

8.4. Wyniki porównania

Poniższa tabela przedstawia zestawienie wyników testów. Wszystkie wartości podane są w centymetrach. Maksymalna praktyczna odległość między przeciwnikami na ringu sumo może wynieść 92 cm, dlatego w tabeli nie występują większe liczby. Wartości minimalne są mierzone od czujnika do przeciwnika. Wartość h oznacza wysokość, na jakiej zamontowany jest czujnik, a – oznacza brak prawidłowego odczytu lub odczyt niepewny.

	IR		Ultradźwięki		Wizja	
	Min	Max	Min	Max	Min	Max
Kolor czarny	0	92	0	92	9	92
Nachylenie 45º	0	h	0	h	9/-	92/-
Identyczny system	-	-	0/-	92/-	9	92
Światło	0	92	0	92	-	-
Ściana blisko	170		92		92	
Kąt prosty	0	92	-/0	-/92	9	92
Paski	0	92	0	92	9	92
Niski przeciwnik	h	92	h	92	42/9	92/40
Inna linia	0	92	0	92	9/-	92/-
Białe tło	0	92	0	92	9/-	92/-

	40 cm	80 cm	40 cm	80 cm	40 cm	80 cm
Kąt widzenia	55º	25º	70º	35º	70º	70º

Tabela 8.1 Zestawienie wyników testów

8.4.1. Podczerwień

Testowany czujnik podczerwieni okazał się nieczuły na zmianę kolorów, tzn. wszystkie testowane kolory powierzchni widział tak samo dobrze. Wyjątkiem okazało się lustro nachylone pod kątem 45º. W tym przypadku czujnik zauważył je dopiero, gdy było w odległości mniejszej niż wynosiła wysokość zamontowania czujnika. W pozostałych wypadkach cała wiązka była odbijana w górę i nie wracała.

System oparty na kodzie RC5 okazał się być wrażliwy na podobne sygnały pochodzące zwłaszcza od silniejszych nadajników. W przypadku zastosowania tego samego kodu przez inny nadajnik czujnik praktycznie oślepł. Natomiast sygnał ciągły nie wpływał na pracę odbiornika nawet, gdy dioda podczerwona świeciła wprost na niego.

Ustawienie przeciwnika względem czujników nie miało wpływu na pracę, z uwagi na długość fali świetlnej odbicie było wyraźnie nawet od ostrych kątów. Dopiero niski przeciwnik okazał się czasem kłopotliwy. Ponieważ czujnik wykrywał głównie odbicie od części frontowej przeciwnika, stawał on się niewidzialny, gdy odległość była mniejsza niż wysokość zamontowania układu.

Z powodu braku regulacji maksymalnej odległości spory kłopot sprawiła równoległa, biała ściana. Maksymalny zasięg czujnika wyniósł 170 cm i z tej odległości ściana była dobrze wykrywana. Ponieważ zgodnie z regulaminem odległość najbliższej przeszkody (w tym ściany) od ringu musi być większa niż metr, teoretycznie 70 cm od krawędzi ringu możliwe jest odebranie sygnału od ściany.

8.4.2. Ultradźwięki

System ultradźwiękowy jest zupełnie niewrażliwy na kolory. Teoretycznie wpływ na odczyt mogłaby mieć faktura powierzchni, w praktyce jednak nie udało się zaobserwować znaczących różnic.

Znaczny wpływ na pracę miał kąt odbicia sygnału. Fale dźwiękowe doskonale odbijają się od płaskich powierzchni o rozmiarach robota sumo (20 cm x 20 cm). Jeśli kąt między nadajnikiem, a linią normalną do przeszkody wynosił więcej niż 45º, fala dźwiękowa nie wracała do odbiornika. Pudełko o ostrych kątach ustawione na wprost czujników dawało bardzo dziwne echo. W przypadku, gdy wiązki z obu nadajników zostały odbite na zewnątrz, system niczego nie wykrywał. Gdy jedna z wiązek wracała, notowane były różne, głównie sprzeczne z prawdą wyniki. Rozpiętość błędnych odczytów wynosiła ok. ±30º, przy większych kątach jeden z boków był już dobrze wykrywany.

Silny sygnał ultradźwiękowy wywoływał różne reakcje. Ponieważ dźwięk się dobrze obija i co za tym idzie rozprasza w pomieszczeniach, wpływ na pracę miały głównie wiązki skierowane wprost na czujniki. Układ czujników ultradźwiękowych mierzy czas od wysłania sygnału do jego powrotu. Zależnie od momentu, w którym sygnał dotarł do odbiornika był on uznawany za dużą lub małą odległość. W większości przypadków oba odbiorniki były zgodne, co do kierunku, czyli były wykryte równe odległości, gdy nadajnik był mniej więcej na wprost. W innych sytuacjach reagował zwykle czujnik od strony nadawanego sygnału, więc wskazywał dobry kierunek.

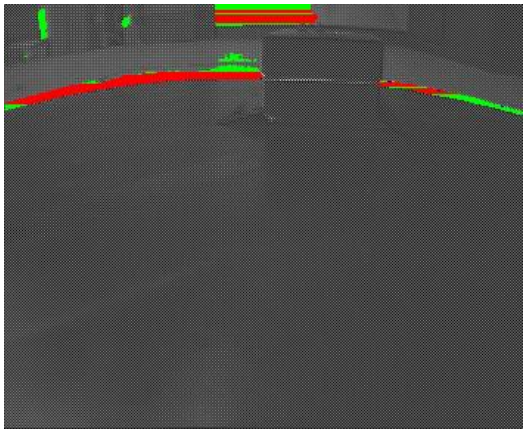
Niski przeciwnik wywoływał podobne reakcje ultradźwięków jak i podczerwieni. Wykrywany był tylko przód przeciwnika, więc minimalna odległość była równa wysokości zamontowania czujnika.

Badany system ultradźwiękowy został zbudowany specjalnie na potrzeby zawodów sumo, dlatego jego zasięg maksymalny jest ściśle ograniczony do 92 cm. Ściana położona w odległości większej niż 92 cm od krawędzi ringu nie ma żadnego wpływu na pracę czujnika.

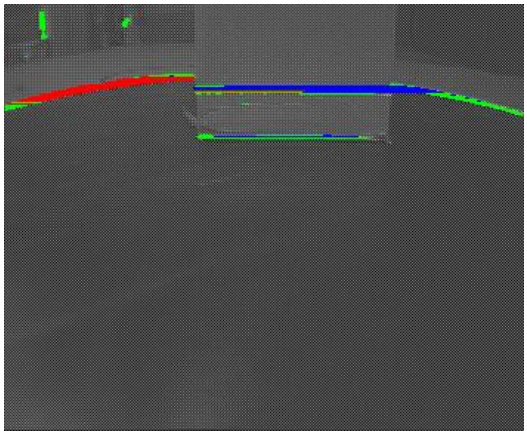
8.4.3. Wizja

System wizyjny z uwagi na sposób działania potencjalnie jest wrażliwy na kolor. Dzięki dopracowaniu kontrastu i ustawieniu balansu bieli, przy oświetleniu górnym, autonomiczny system wizyjny poradził sobie z wykryciem większości przeszkód.

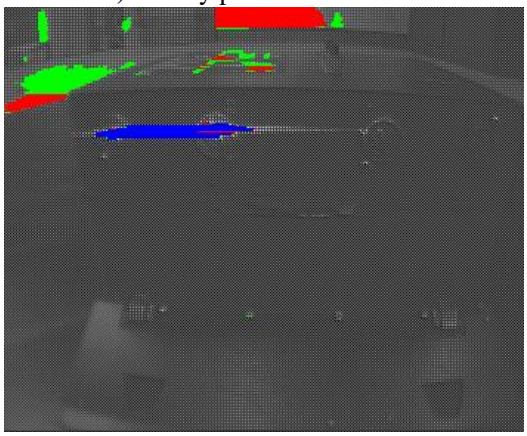
Przeciwnik w kolorze czarnym, bądź dowolnym innym znacznie ciemniejszym od białej linii, jest wykrywany tylko gdy przetnie linię krańcową. Przy wyliczonym w punkcie 4.2 położeniu kamery, wykrywany był, w całym zakresie działania (rysunek a) i c)), każdy przeciwnik wyższy nie niższy niż 11cm (wysokość robota, na którym zamontowana była kamera). Jeśli kolor był optycznie równie jasny jak dolny próg ustawiony dla linii zawsze była widoczna na nim linia światłą laserowego b).



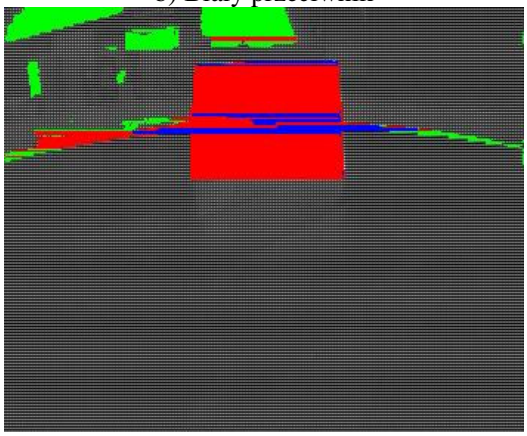
a) Czarny przeciwnik daleko



b) Biały przeciwnik



c) Przeciwnik bardzo blisko



d) Lustro nachylone

Rysunek 8.1 Zespolone zdjęcia testowe

Powierzchnie nachylone pod kątem 45º w większości wypadków nie stanowiły problemu. Znaczne trudności sprawiały natomiast te powierzchnie, które były odblaskowe. Nachylone lustro d), z uwagi na zastosowanie górnego oświetlenia okazało się poważną przeszkodą. Dopiero po ręcznym dostosowaniu parametrów obrazu udało się na jasnym odbiciu światła zauważyć laser (linia niebieska na obrazie po kwantyzacji). Położenie lasera

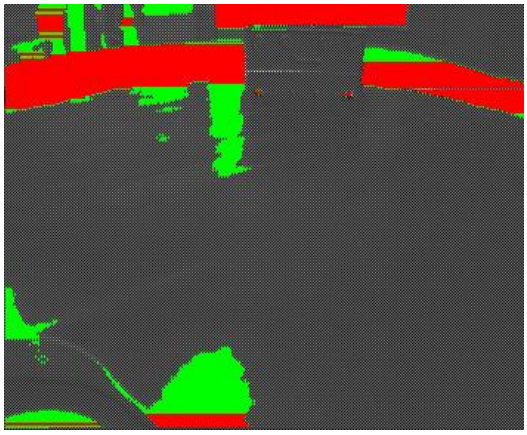
na zdjęciach jest trochę zakłamane z uwagi na to, że system zapamiętuje tylko pierwsze i ostatnie wystąpienie koloru czerwonego.

Konkurencyjny system wizyjny lub czujnik podczerwieni nie ma żadnego wpływu na pracę. Niestety ostre źródło światła, w tym i laser skierowany wprost na kamerę zupełnie ją oślepia. Na szczęście taki zabieg jest zakazany przez regulamin z uwagi na to, że jest to celowe zakłócenie pracy czujników. Błysk lampy błyskowej z uwagi na krótki czas trwania ma niewielki wpływ, powoduje zafałszowanie najwyżej jednej klatki. Ponieważ system wykonuje kilka zdjęć na sekundę, ten problem można pominąć.

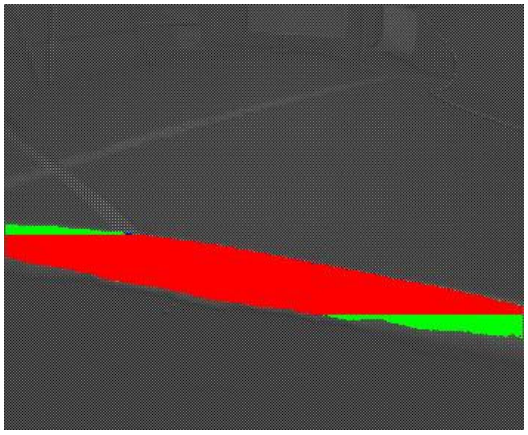
Kształt przeciwnika, w tym wszelkie ostre krawędzie, nie mają wpływu na odczyt, pod warunkiem, że przynajmniej jeden fragment obudowy wystaje ponad założony pułap 11cm. Przeciwnik niższy jest wykrywany tylko wtedy, gdy znajduje się na tyle blisko linii by optycznie ją zasłonić. W przypadku największego oddalenie kamery od linii końcowej niski przeciwnik wykrywany jest dopiero z odległości 42 cm, ze środka ringu praktycznie dowolny robot jest dobrze widoczny. Również wszelkie paski mające zmylić czujniki nie mają większego wpływu na pracę systemu wizyjnego. Teoretycznie duża ilość jasnych obiektów mogłaby przepełnić pamięć, w praktyce z uwagi na ograniczoną rozdzielczość, paski zbyt liczne musiałyby być tak cienkie, iż system w większości nawet by ich nie wykrył. Fałszywa linia czasem zakłóca pracę systemu wizyjnego. Jednak zdarza się to tylko dla pewny specyficznych wzajemnych położeń kamery, linii i przeciwnika. Po niewielki poruszeniu się jednego z robotów wizja ponownie bezbłędnie identyfikuje przeciwnika.

Kolor ściany nie spowodował zakłamania w odczytach. Natomiast system wizyjny jako jedyny zareagował na podłogę za ringiem. W przypadku, gdy podłoga była zbyt jasna, lub jak to miało miejsce w laboratorium, zbyt błyszcząca e), przy błędnej kalibracji zdarzały się przypadki, gdy przeciwnik nie przecinał linii, gdyż przez system podłoga też była uznana za linię. Ponownie regulamin zawodów sumo niweluje ten problem, gdyż zgodnie z nim, podłoga powinna być znacznie ciemniejsza od linii. Ponadto po dojechaniu do krawędzi przy zachowaniu tamtego, błędnego wykalibrowania, podłoga już nie była wykrywana f), gdyż przy różnym kącie widzenia, był też różny stopień odbicia światła górnego.

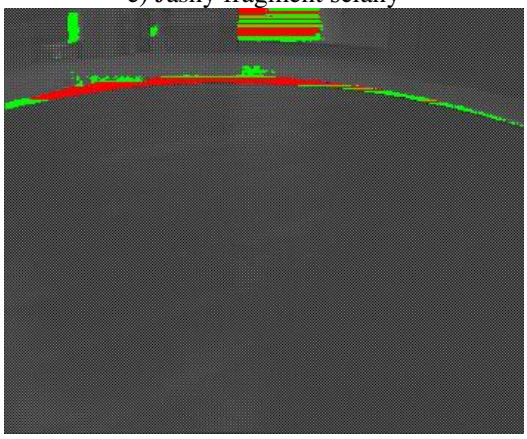
Znacznym problem okazał się automatyczny balans bieli i balans ekspozycji. Ten ostatni po zmianie napięcia odniesienia, udało się naprawić. Balans bieli natomiast działał dobrze tylko dla formatu RGB, w formacie YCrCb zdarzały się notorycznie przekłamania w kolorach. Kolor biały wahał się od barwy zielonej, przez białą aż do niebieskiej. Przy okazji zahaczając o kolor żółty i czerwony, co powodowało mylne wykrycie lasera (h. W celu zapewnienia niezawodnej pracy systemu wizyjnego należy wyposażyć go we własny program do ustawiania balansu bieli na podstawie wybranego, znanego markera.



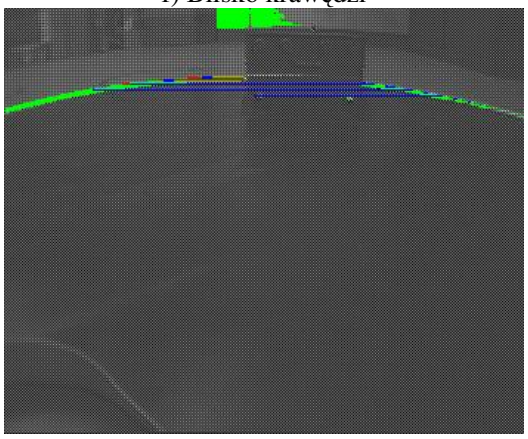
e) Jasny fragment ściany



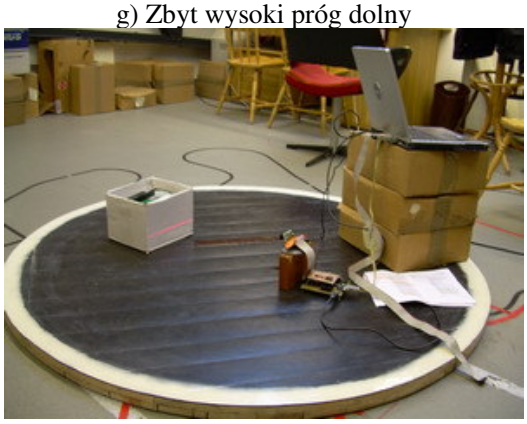
f) Blisko krawędzi



g) Zbyt wysoki próg dolny



h) Za niski próg górny i zły balans bieli



i) Ustawienie do testów z jasnym obiektem

j) Testy z prawdziwym robotem

Rysunek 8.2 Zdjęcia i stanowisko testowe.

Widoczne powyżej stanowisko testowe Autonomicznego systemu wizyjnego było wyposażone w komputer tylko w celu pobrania zdjęć w formacie BMP. Wszystkie wyniki zostały uzyskane na podstawie wskazań diod umieszczonych na płycie testowej.

9. Podsumowanie i wnioski

9.1. Wnioski

Podstawowym celem pracy było zbudowanie od podstaw autonomicznego systemu wizyjnego dla robota sumo. System ten miał być konkurencyjny wobec czujników podczerwieni i ultradźwiękowych, które są obecnie stosowane. Testy przeprowadzone w poprzednim rozdziale potwierdziły większość ze znanych problemów dotyczących tych czujników. Wykazały też, iż jeden rodzaj czujnika da się bardzo łatwo oszukać, natomiast dwa różne systemy wzajemnie się uzupełniają.

Głównym problemem dotychczas stosowanych układów wykrywających była wrażliwość na całkowite odbicie wiązki nadawczej. Obiekty o nachyleniu płyty czołowej powyżej 45º, przy zachowaniu wysokiego współczynnika odbicia, są praktycznie niewykrywalne. Zjawisko to jest powszechnie znane z konstrukcji Stealth. Ponieważ system wizyjny wykrywa przeciwnika pośrednio, na podstawie obserwacji białej linii za nim, jest praktycznie nieczuły na nachylenia powierzchni. Problem ten został więc rozwiązany.

Drugim problemem układów aktywnych jest ich wrażliwość na obcą wiązkę nadawczą. Ponownie system wizyjny okazał się lepszy, gdyż z uwagi na pasywny charakter pracy, obce systemy nie zakłócają wykrycia.

Niestety nowy system oznacza nowe problemy. Wizja, z uwagi na zastosowany algorytm wykrywania, jest wrażliwa na jasność tła za linią i bezpośrednie oświetlenie kamery.

Ponadto nie udało się rozwiązać do końca problemu wykrywania niskiego przeciwnika. Dużym postępem jest uniknięcie znikania przeciwnika przy małych odległościach. Kwestia ta powodowała inicjowanie poszukiwanie przeciwnika, gdy ten był tuż na wprost. Jest to osiągnięte kosztem ograniczenia maksymalnego zasięgu widzenia niskiego przeciwnika. Analogiczne wyniki można osiągnąć przy zastosowaniu kilku czujników konwencjonalnych na różnych wysokościach.

Największą przewagą systemu wizyjnego nad jej poprzednikami jest kąt widzenia. Z uwagi na zbieżność wiązki nadawczej praktyczny kat widzenia poziomego zawęża się wraz ze wzrostem odległości. System wizyjny tak samo dobrze lokalizuje z małych jak i dużych odległości. Daje to dużą przewagę, gdyż można dokładnie wycelować w przeciwnika, podczas gdy on jeszcze nie jest w stanie nawet zauważyć zagrożenia.

Celem pobocznym niniejszej pracy było udowodnienie przydatności taniego i ogólnie dostępnego mikroprocesora do celów analizy obrazu na żywo. W połączeniu z kamerą, której

cena nie przekraczała ceny gotowego czujnika ultradźwiękowego, miał on być w stanie wykryć szukany obiekt z rozsądną prędkością, co najmniej 1 zdjęcia na sekundę.

Wstępne obliczenia wykazywały niemożliwość realizacji tego założenia z uwagi na zbyt dużą ilość danych do przechowywania i za małą moc obliczeniową. Po dokonaniu kompresji obrazu i zoptymalizowaniu większości funkcji okazało się, iż możliwe jest wykonanie teoretycznie nawet 8 zdjęć na sekundę. Praktyczne testy zakończyły się przy 4 zdjęciach na sekundę, z uwagi na niestabilność pracy kamery.

Pomimo udowodnienia, że połączenie powszechnie dostępnego mikroprocesora z tanią kamerą jest możliwe i działa, należy zastanowić się nad rozbudowaniem systemu o kolejne elementy. Użyta kamera CMOS równocześnie naświetla obraz i go wysyła. Oznacza to, że pierwsza linijka zdjęcia jest znacznie starsza od ostatniej równocześnie będą niewiele młodszą od ostatniej linijki poprzedniego zdjęcia. Na obrazie statycznym jest to niezauważalne jednak w czasie ruchu wyraźnie widać odchylenie obrazu od kierunku ruchu. Rozwiązaniem tego problemu może być zastosowanie kamery z mechaniczną przesłoną lub użycie szybszej kamery z pośredniczącym buforem FIFO albo Framegrabberem.

9.2. Podsumowanie

Niniejsza praca obejmuje swoją tematyką wiele zagadnień, począwszy od problematyki zasilania, po przez projektowanie układów elektronicznych i programowanie zoptymalizowane aż po optykę. Z uwagi na tą rozpiętość materiału nie było możliwym dopracowanie do perfekcji wszystkich jej części. Kolejne etapy pracy wpływały zarówno na poprzednie jak i na następne ujawniając inne, lepsze rozwiązania. Na podstawie wyników tej pracy można śmiało pokusić się o poprawienie każdej z jej części.

Praca miała charakter badawczy, pozwoliła autorom znacznie poszerzyć wiedzę teoretyczną z zajęć o aspekty praktyczne. Udało się odkryć sporą rozbieżność między opisem działania, a problemami związanymi z użytkowaniem. Chociaż braki w instrukcjach obsługi utrudniły znacznie pracę, samodzielne poznawanie zasad działania układów scalonych okazało się bardzo ciekawe pomimo nie zawsze satysfakcjonujących wyników.

Końcowym efektem pracy jest działający, ale jeszcze nie gotowy do użycia system wizyjny dla robota sumo. Intencją autorów jest by ten projekt kontynuować w ramach pracy doktorskiej, bądź by został on wykorzystany w pracy nad podobnym, bądź zupełnie innym układem wykrywającym dla robota mobilnego.

Bibliografia

[1] ARM DDI 0029G – Technical Reference Manual
http://www.arm.com/pdfs/DDI0029G_7TDMI_R3_trm.pdf

[2] Atmel, AT91SAM7S Series Preliminary Complete, http://www.atmel.com

[3] Averlogic Technologies Inc., AL440C Data Sheet (version 1.0),
http://semiconductorstore.com/pdf/AL440C-20%20%20%20%20%20%20%20.pdf

[4] Bomhof T., Cebe A., Ewerlin C., i inni, VisiRobs, Die Projektgruppe 472,
Universität Dortmund 1. März 2006

[5] Brodin A., Classon J., Melander J., i inni, Giraffen KTH,
Royal Institute of Technology 15th June 2005

[6] Bruce J., The CORAL Group's Color Machine Vision Project,
Carnegie Mellon University Pittsburgh, http://www.cs.cmu.edu/~jbruce/cmvision/

[7] Bruce J., Realtime Machine Vision Perception and Prediction,
Carnegie Mellon University Pittsburgh May 2000

[8] Carnegie Mellon University Pittsburgh, CMUcam2 Vision Sensor.User Guide,
http://www.cs.cmu.edu/~cmucam/cmucam2/CMUcam2_manual.pdf

[9] Chmielnicki E., Lokalizacja i śledzenie ruchu obiektów za pomocą robota
wyposażonego w kamerę. Politechnika Warszawska Maj 2005

[10] Duchinski M., Współpraca manipulatora z systemem wizyjnym,
Politechnika Wrocławska 2003

[11] FUJISOFT ABC Inc., Regulamin zawodów sumo robotów-edycja japońska,
http://www.fsi.co.jp/sumo-e/out/outa0000.html

[12] Hofman Y., License Plate Recognition Hi-Tech Solutions May 2, 2004
http://www.licenseplaterecognition.com/

[13] Król P., Dzioch M., Internetowy Kurs Telewizji Dozorowej, http://www.dipol.com.pl

[14] Lukin A., Kubasov D., An Improved Demosaicing Algorithm,
State University of Moscow 2004

[15] Majchrzak D., Identyfikacja ruchu postaci. Analiza możliwości, metody, algorytmy.
Politechnika Wrocławska 2005

[16] Marszałek M., SkySpy Politechnika Warszawska 2005

[17] Micron Technology Inc., Przetwornik CMOS o wysokiej rozdzielczości,
http://download.micron.com/pdf/flyers/MT9E001_flyer.pdf

[18] Oiza, „Digital Camera Interface”, 2004,
http://www.robozes.com/inaki/dproject/report.pdf

[19] OmniVision Technologies, Specyfikacja interfejsu SCCB,
http://www.ovt.com/products/SCCBSpec_AN_2_1.pdf

[20] OmniVision Technologies, Przetwornik CMOS kamery C3088,
OV6620 Advanced Information Preliminary (version 1.4, 13 May 2000),
http://www.cs.cmu.edu/~cmucam/Downloads/ov6620DSLF.PDF

[21] Philips, Specyfikacja interfejsu I2C,
http://www.nxp.com/acrobat_download/literature/9398/39340011.pdf

[22] Pietkiewicz W., Wykorzystanie obrazów barwnych do lokalizacji klasyfikacji obiektów. Wrocław 2002

[23] Politechnika Poznańska, Regulamin zawodów sumo robotów w Poznaniu, http://www.sumo.put.poznan.pl/?pl/sumo/regulamin

[24] Propox, Programator mikrokontrolerów z rdzeniem ARM7, http://www.propox.com/products/t_122.html

[25] Propox, Instrukcja do minimodułu MMsam7s, http://www.propox.com/download/docs/MMsam7s_rev2_sch.pdf

[26] Ricarte C., Cook J., Kim J., i inni, Mobile Image Capture And Retrieval System (MICARS), California State University Long Beach (CSULB) 2005

[27] Rowe A., Rosenberg C., Nourbakhsh I., A Second Generation Low Cost Embedded Color Vision System, IEEE Computer Society Conference, 2005. Identifier 10.1109/CVPR.2005.396

[28] Schurter, Bezpiecznik polimerowy PFRA, http://www.schurter.ch/pdf/english/typ_PFRA.pdf

[29] Sharp, Czujnik odległości GP2Y0A02YK, http://document.sharpsma.com/files/GP2D12-DATA-SHEET.PDF, http://www.robotshop.ca/PDF/Sharp_GP2Y0A02YK_Ranger.pdf

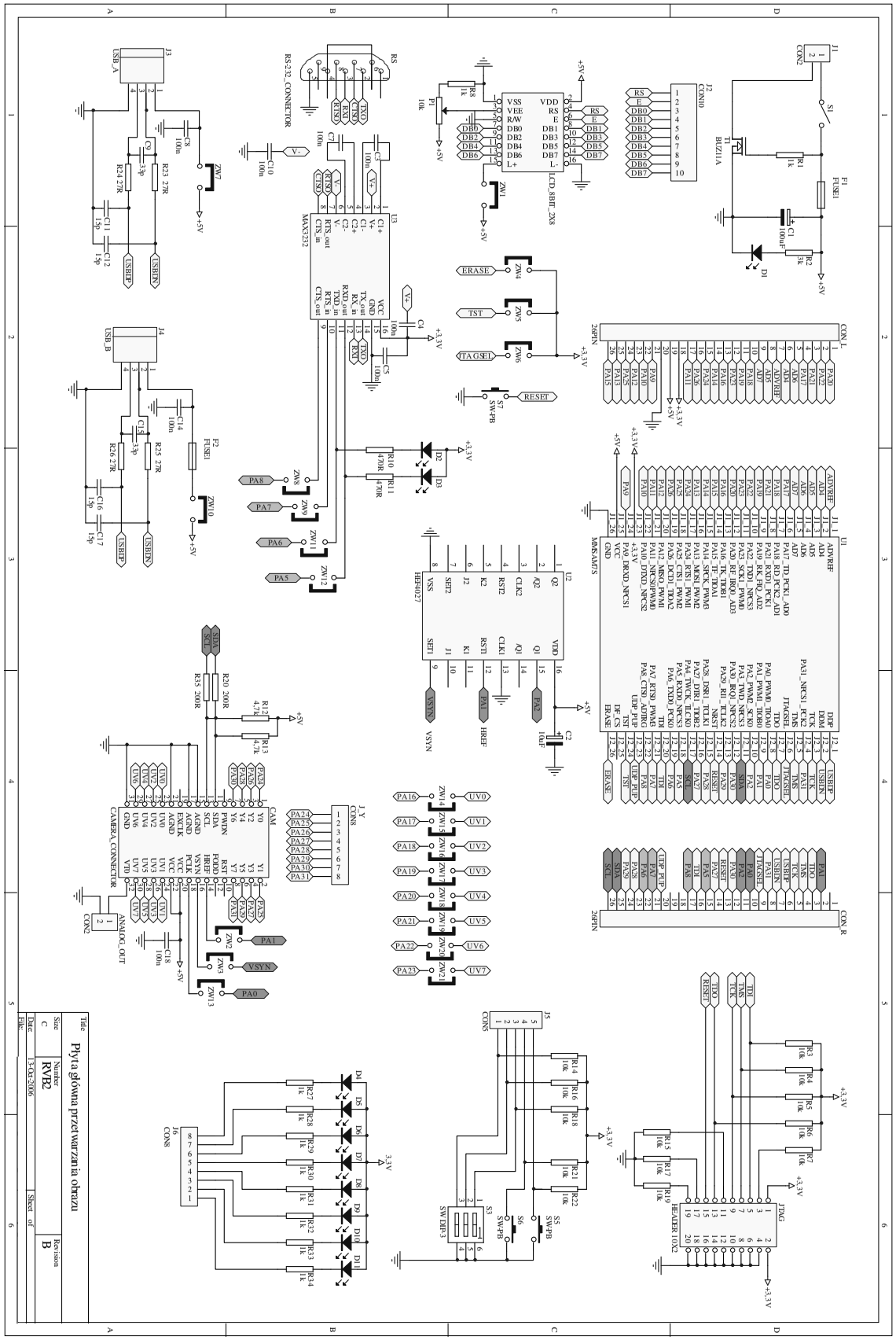
[30] TIA, Specyfikacja interfejsu RS-232 (EIA/TIA-232-F), http://www.tiaonline.org, http://focus.ti.com/lit/an/slla037a/slla037a.pdf

[31] Tykkälä T., Construction of an Optical Tracking System, Helsinki University Of Technology 2006

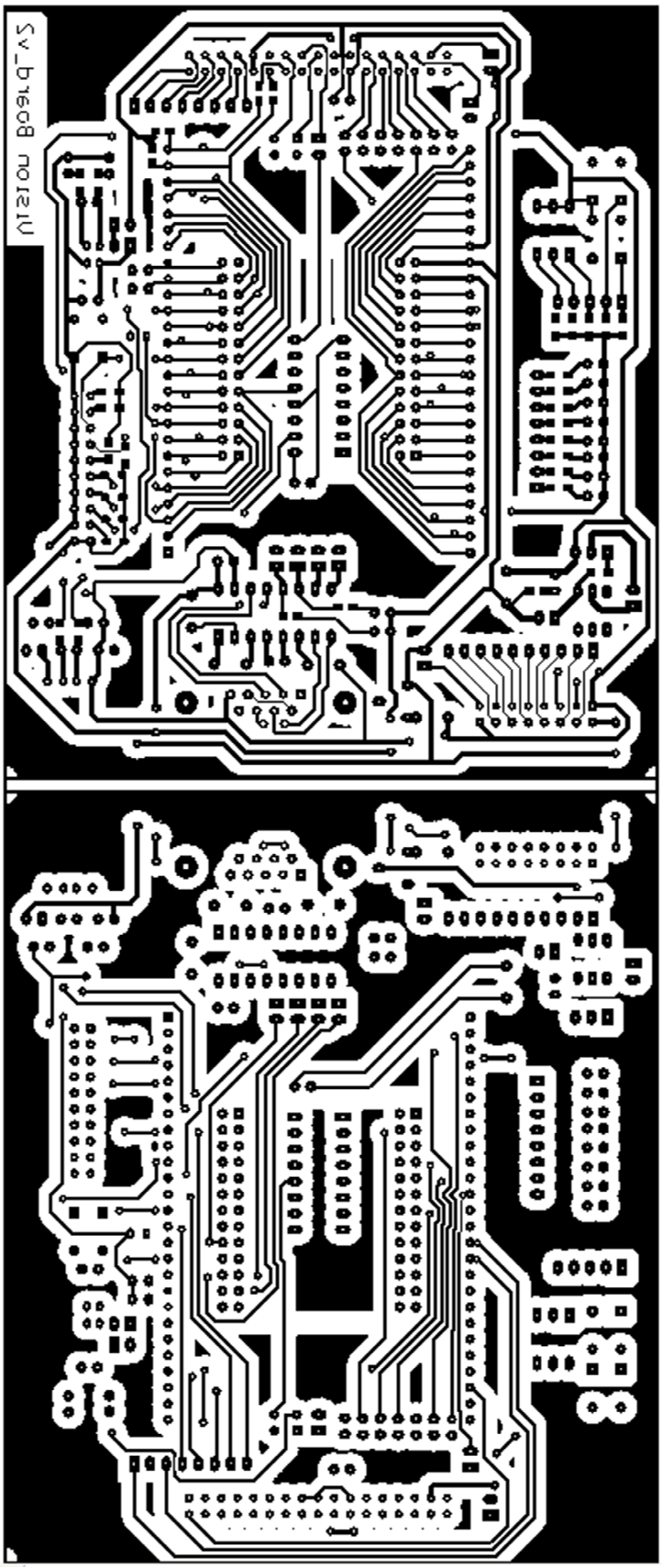
[32] Wikipedia, http://www.wikipedia.org

Spis ilustracji

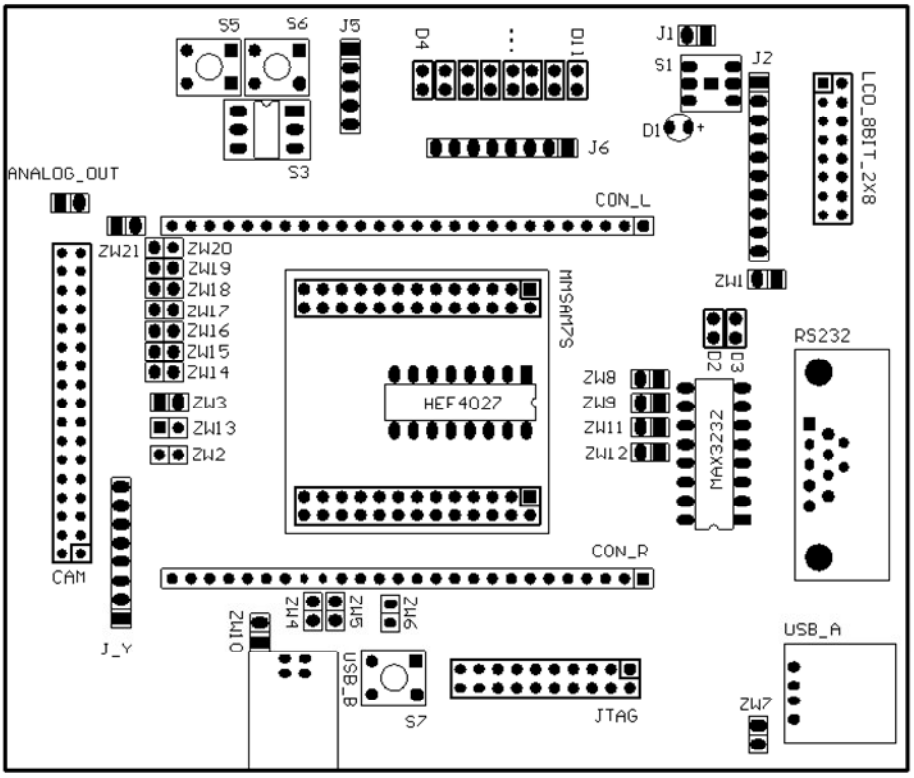
Dodatek A



Rysunek A.1: Schemat elektryczny płyty głównej przetwarzania obrazu RVB2

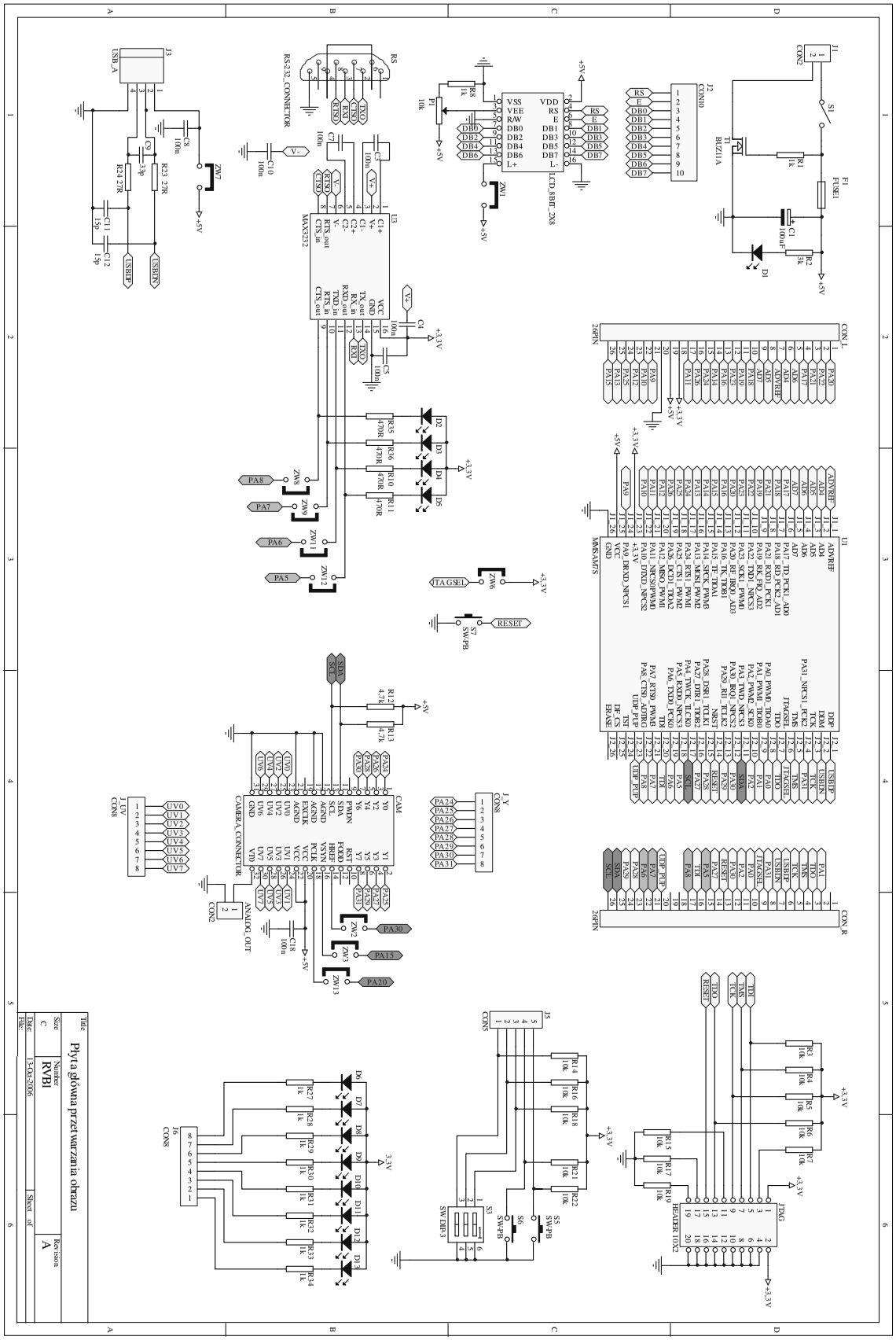


Rysunek A.2: Widok płytki drukowanej RVB2

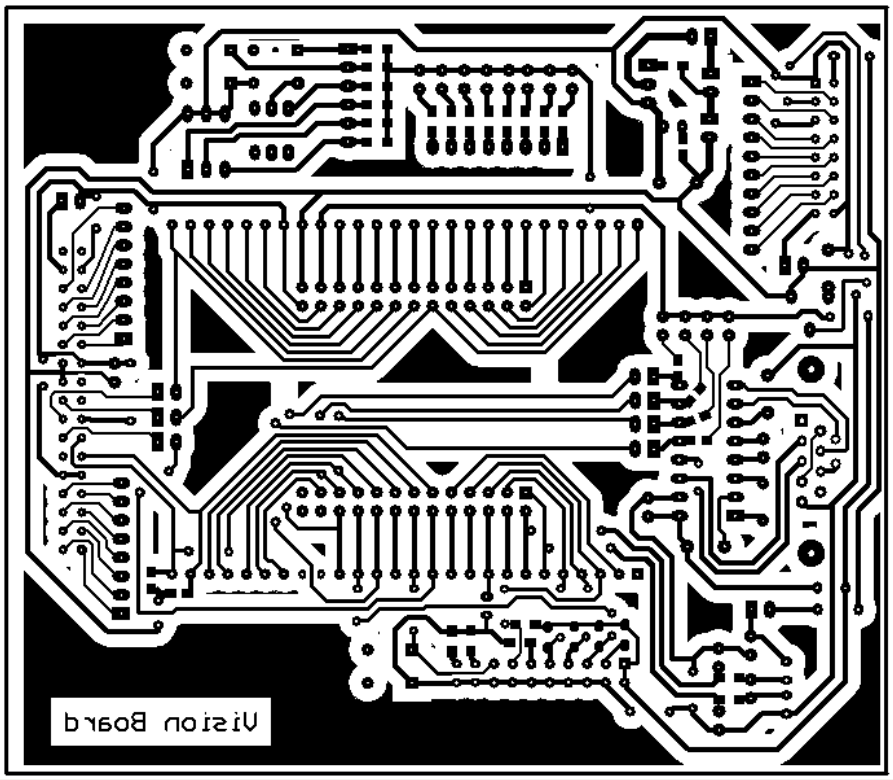


Nr. złączki	Pełniona funkcja (gdy zwarta)	Domyślnie
ZW1	Podświetlenie wyświetlacza LCD włączone	rozwarta
Synchronizacja kamery		
ZW2	HREF włączone	zwarta
ZW3	VSYN włączone	zwarta
ZW13	PCLK włączone	zwarta
Programowanie, emulacja, kasowanie		
ZW4	Skasowanie pamięci FLASH	rozwarta
ZW5	Bootloader SAM-BA aktywny(TST)	rozwarta
ZW6	Zmiana trybu emulacji z ICE na Boundary Scan	rozwarta
Zasilanie z USB		
ZW7	Płytka jest zasilana z portu USB_A	rozwarta
ZW10	Płytka jest zasilana z portu USB_B	rozwarta
Port RS232		
ZW8	CTS włączone	zwarta
ZW9	RTS włączone	zwarta
ZW11	TXD włączone	zwarta
ZW12	RXD włączone	zwarta
Port UV kamery		
ZW14 ÷ ZW21	Włączenie jednego z 8 bitów kanału UV kamery	zwarte
Inne złączki, przyciski i przełączniki		
J1	Złącze zasilania 5V DC	
S1	Włącznik zasilania	
S7	Reset mikrokontrolera	

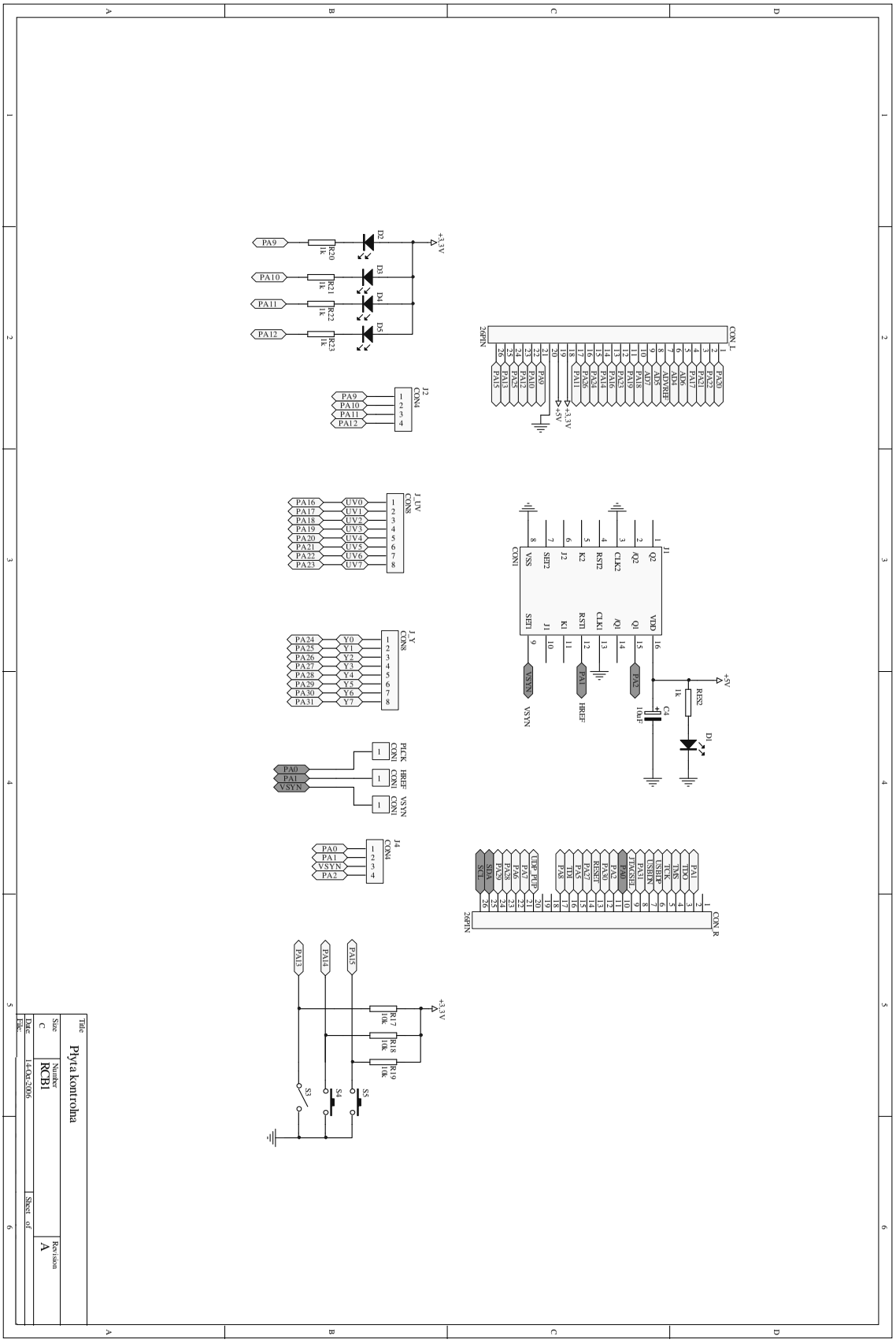
Rysunek A.3: Rozmieszczenie elementów i zworek konfiguracyjnych na płycie RVB2



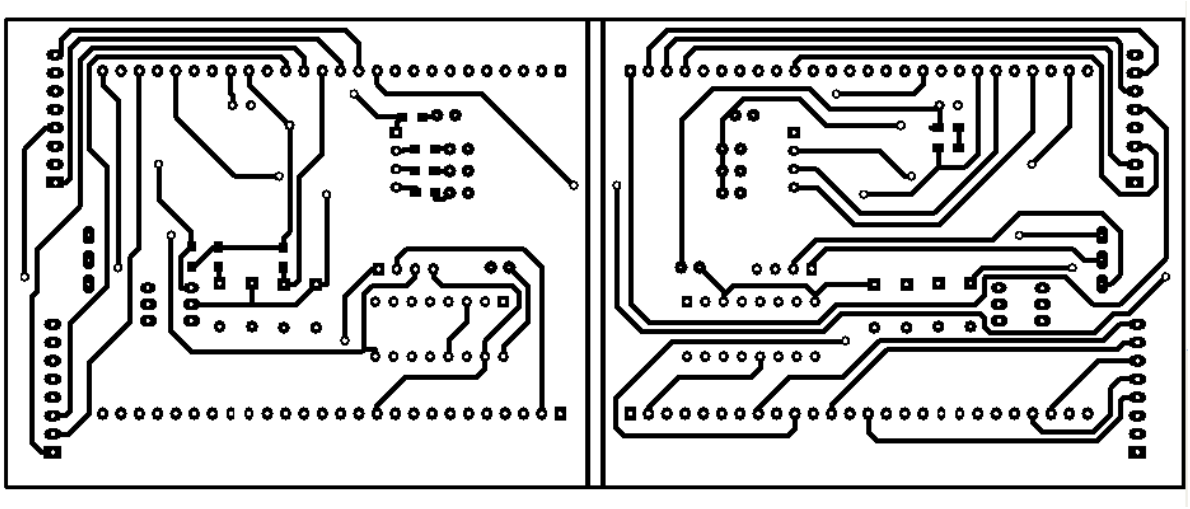
Rysunek A.4: Schemat elektryczny płyty głównej przetwarzania obrazu RVB1



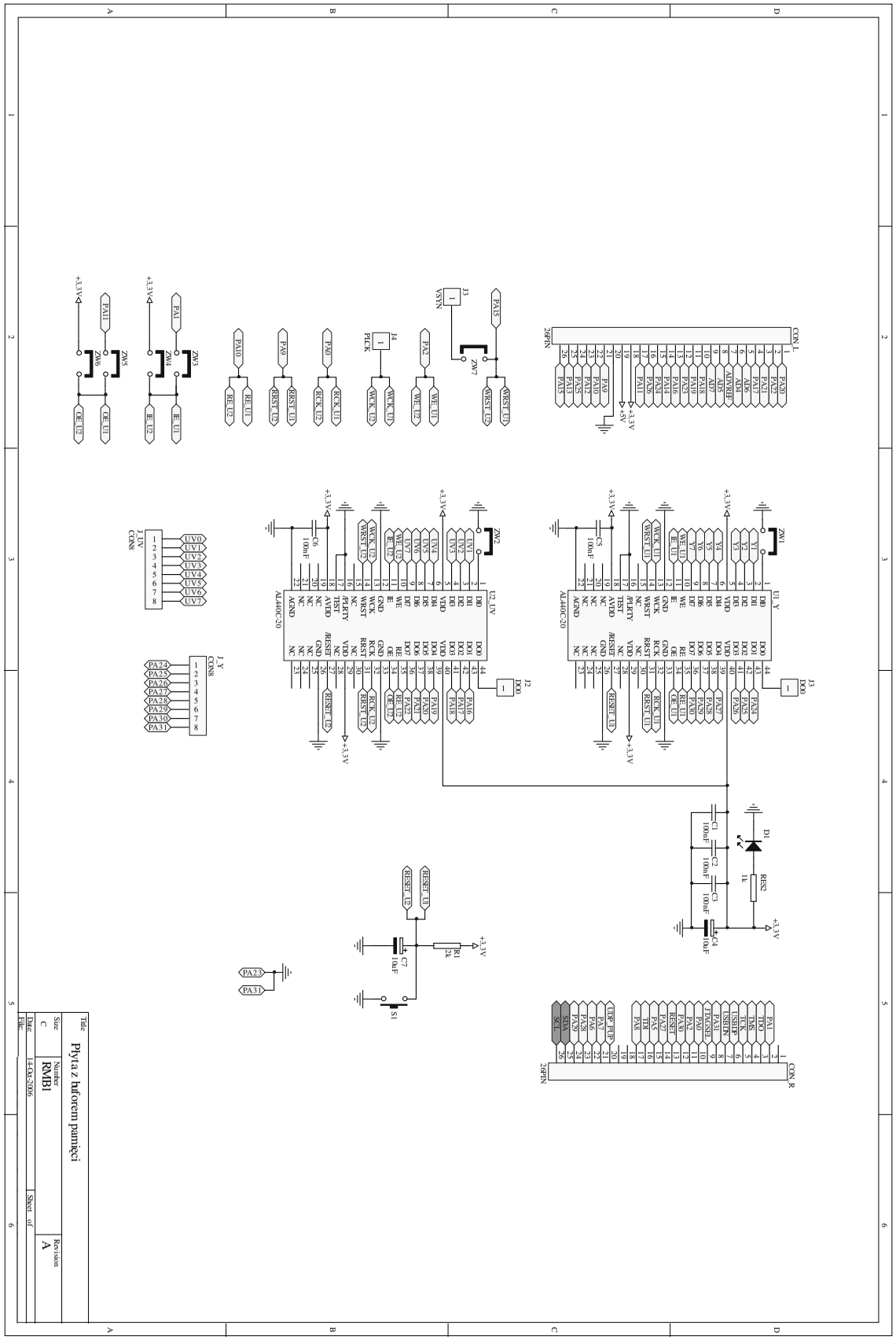
Rysunek A.5: Widok płytki drukowanej RVB1



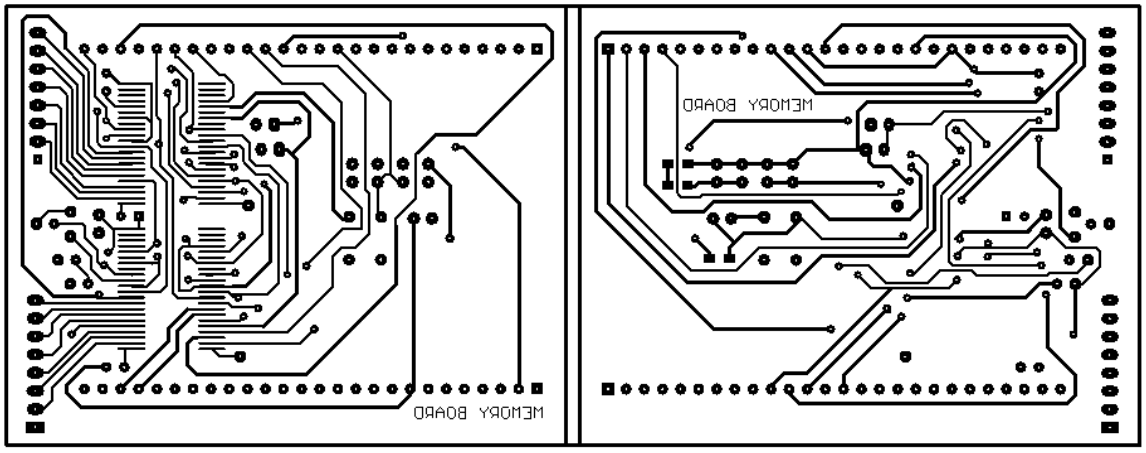
Rysunek A.6: Schemat elektryczny płyty kontrolnej RCB1



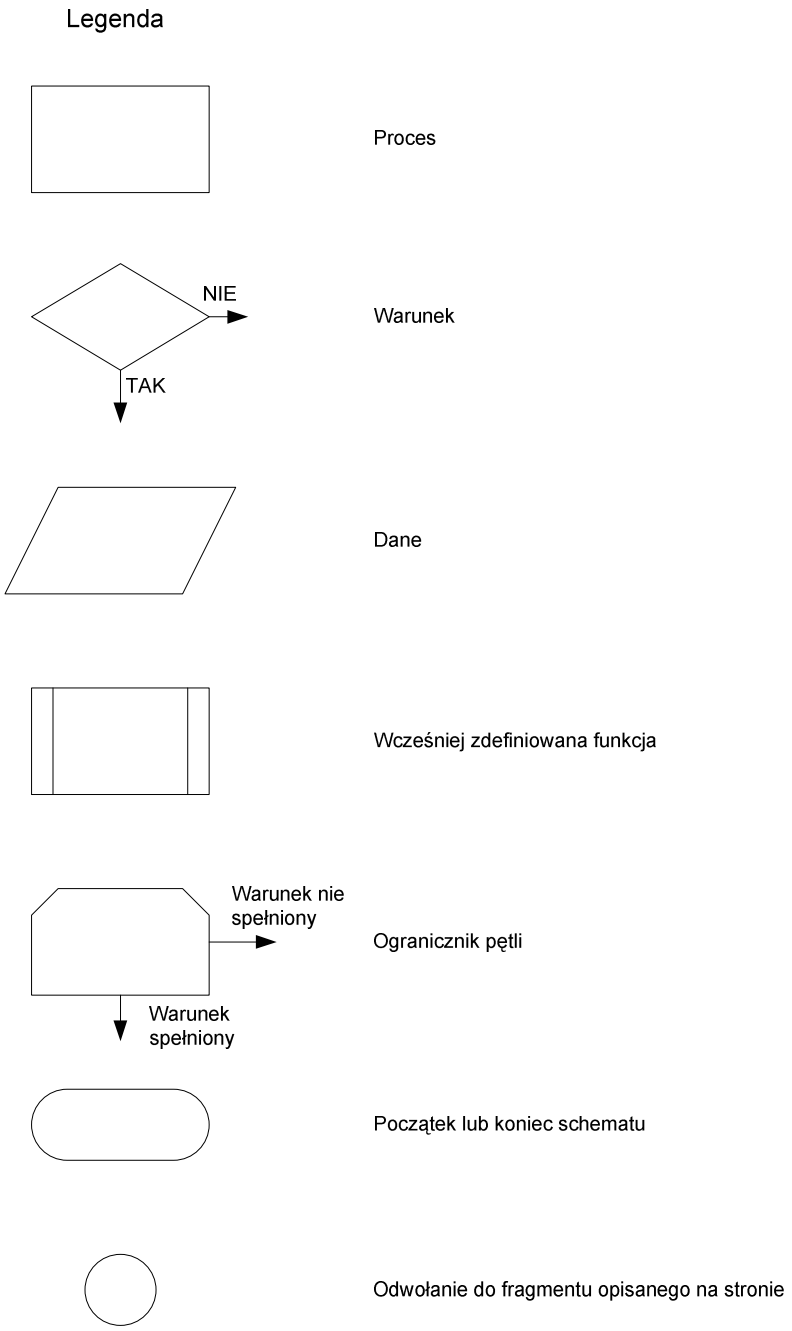
Rysunek A.7: Widok płytki drukowanej RCB1



Rysunek A.8: Schemat elektryczny płyty z buforem pamięci RMB1



Rysunek A.9: Widok płytki drukowanej RMB1



Rysunek A.10: Legenda do schematów blokowych