Temat: *System wspomagający grę w darta z wykorzystaniem analizy obrazu w czasie rzeczywistym*.

1. Wstęp.

We wstępie opisane zostaną ogólne cele pracy oraz powody, dla których zajęto się taką tematyką. Podkreślone zostanie istniejące zapotrzebowanie na tego typu system i niewielka liczba podobnych rozwiązań.

W skrócie:

- cele pracy
- motywacja, zapotrzebowanie, nisza

2. Analiza problemu.

Na początek pokrótce omówione zostaną najważniejsze zasady gry w darta, budowa tarczy i sposób liczenia punktów. Następnie sformalizowany zostanie główny problem, polegający na wykryciu pozycji rzutki na tarczy. Należy zaznaczyć również ograniczenia czasowe i problemy z nich wynikające.

W skrócie:

- sformalizowanie problemu
- opis tarczy, gry
- konieczność szybkiej reakcji na rzut

3. Teoretyczne rozwiązanie problemu.

To będzie najważniejszy rozdział pracy, w którym w pełni zostanie przedstawione teoretyczne rozwiązanie problemu. Najpierw nakreślony zostanie ogólny pomysł na rozwiązanie i jego podział na kilka etapów, które później będą szczegółowo opisane. Aby umożliwić łatwiejsze zrozumienie, rozdział powinien zawierać fragment wprowadzający do zagadnienia triangulacji (z przykładami). Następnie, krok po kroku, zaprezentowane zostaną wszystkie etapy składające się na rozwiązanie - służyć temu będą przede wszystkim diagramy i obliczenia, prowadzące od pozycji rzutki na zdjęciu jednej i drugiej kamery do uzyskania segmentu punktowego, w jaki trafiła rzutka. Ważnym elementem będą także obliczenia i działania, które zostały wykonane w celu wykonania stelaża, do którego zamocowane zostały tarcza i kamery.

W skrócie:

- pomysł na rozwiązanie, dlaczego dwie kamery, a nie jedna
- ogólnie o triangulacji
- rysunek/szkic stelażu z kamerami i tarczą
- obliczenia:
    - wymiary stelaża, odległości kamer od tarczy
    - analiza rzutu:
        - uzyskanie kąta ze współrzędnych piksela odpowiadającego punktowi wbicia rzutki (proporcja, interpolacja)
        - umiejscowienie kąta w układzie związanym z kamerami, sformułowanie trójkąta kamera-kamera-rzutka
        - obliczenie nieznanych kątów i boków z tw. sinusów

    - obliczenie pozycji (x,y) rzutki, potem zamiana układu odniesienia na taki, gdzie (0, 0) to lewy górny róg stelaża
    - zamiana współrzędnych kartezjańskich na biegunowe
    - mapowanie (r, φ) na pole punktowe (segment) na tarczy

4. Projekt systemu.

Pomiędzy prezentacją teoretycznego rozwiązania głównego problemu, a szczegółami technicznymi, należy opisać projekt systemu, jego składowe i powiązania między nimi. Pomogą w tym celu diagramy UML, przede wszystkim diagram komponentów, sekwencji oraz czynności. Warto w tym miejscu przypomnieć i rozwinąć niektóre z założeń funkcjonalnych przedstawionych we wstępie, opisać główny przypadek użycia. Ważnym punktem będzie również krótki opis algorytmów z biblioteki OpenCV, tak by zwrócić uwagę, że niewiele części systemu pełni rolę "czarnych skrzynek", a autor (przynajmniej na pewnym poziomie abstrakcji) rozumie algorytmy, z których korzysta, mimo iż nie zostały przez niego zaimplementowane.

W skrócie:

- wykaz komponentów
- diagramy:
    - komponentów
    - sekwencji
    - czynności
    - ewentualnie przypadków użycia
- krótki opis aplikacji mobilnej, jej cel
- opis używanych algorytmów (również tych związanych z przetwarzaniem obrazu z OpenCV)

5. Implementacja systemu.

Po zrozumieniu sposobu rozwiązania i planu na jego wcielenie w życie, można omówić szczegóły implementacyjne. Nie jest jednak celem dokładne omówienie każdej z używanych technologii (jej historii itd.) oraz każdej linii kodu, bardziej ogólny przegląd, który pozwala wyobrazić sobie strukturę i najważniejsze punkty aplikacji. Warto także rozwinąć opis komponentów (Raspberry Pi, kamery). Wymienione zostaną także wszystkie problemy, które pojawiły się podczas tworzenia systemu. Krótko można wspomnieć o sposobie, w jakim prowadzony był projekt (kontrola wersji, tablica z zadaniami). Jednym z ostatnich elementów będą testy dokładności, które pokażą, jak precyzyjnie system wykrywa rzutkę.

W skrócie:

- pobieżny opis kodu (systemu wbudowanego i aplikacji mobilnej)
- języki programowania, zależności
- struktura projektu
- krótki opis komponentów
- problemy
- krótko o OpenCV
- GitHub, Kanban board
- testy dokładności

6. Instalacja i wdrożenie.

Ze względu na to, że mamy do czynienia z systemem wbudowanym, w dodatku wymagającym wykonania pewnych technicznych elementów (stelaż), podłączenia zewnętrznych komponentów, trudno o prosty sposób wdrożenia systemu. W tej części pokazane zostaną przede wszystkim kroki potrzebne do uruchomienia aplikacji - konieczność instalacji wymaganych bibliotek (przede wszystkim OpenCV), interpreterów itd. Z pewnością dobrze będzie wspomnieć o autorskim rozwiązaniu autora, które pozwala na łatwiejszą pracę - skrypt `sync.sh` synchronizujący folder z kodem źródłowym pomiędzy Raspberry Pi a komputerem.

W skrócie:

- wymagane biblioteki, instalacja OpenCV
- skrypt `sync.sh`

7. Podsumowanie.

Na koniec należy podsumować, które elementy planu udało się zrealizować, które nie, a które można jeszcze dopracować. Przedstawione zostaną również usprawnienia pozwalające na zwiększenie dokładności (np. inny sposób wyliczania kąta pomiędzy kamerą a tarczą) i funkcjonalności, które według autora warto wprowadzić w przyszłości.

W skrócie:

- rozliczenie z zadeklarowanych funkcjonalności
- usprawnienia na przyszłość