AKADEMIA GÓRNICZO-HUTNICZA
im. St. Staszica w Krakowie
WEAIiE, Katedra Automatyki
Laboratorium Biocybernetyki

Przedmiot: **Systemy rekonfigurowalne.**

Pr12sr512

Temat projektu:

Równoległa interpolacja obrazu barwnego z kamery cyfrowej

Wykonali:

Bartłomiej Bułat
Tomasz Drzewiecki
Informatyka Stosowana

Rok I, st. II Semestr II

V 1.0

Kraków, styczeń, 2013

1. Abstrakt

Celem projektu było napisanie biblioteki do interpolacji obrazu barwnego na podstawie zapisu z kamery z matrycą Bayera. Matryca ta charakteryzuje się specjalnym ułożeniem pikseli, które ma za zadanie odwzorowywać ludzką percepcje obrazu, gdyż znajduje się na niej ponad dwukrotnie więcej receptorów koloru zielonego od pozostałych barw. Biblioteka ma wykorzystywać wspomaganie obliczeń na GPU z wykorzystaniem technologii OpenCL. Interpolacja polega na wyliczeniu średnich wartości odpowiednich pikseli z trzech różnych masek z uwzględnieniem wzmocnienia kolorów. Aplikacja oparta o stworzoną bibliotekę ma w czasie rzeczywistym interpolować obraz z kamery HD. Wynikiem badań jest działająca biblioteka do pojedynczej i sekwencyjnej interpolacji obrazów, jak również aplikacja do interpolacji obrazów z kamery HD (JAI BB-500GE).

Udało się pokazać znaczne przyśpieszenie algorytmu zaimplementowanego na karcie graficznej w porównaniu do referencyjnego algorytmu wykonywanego na CPU.

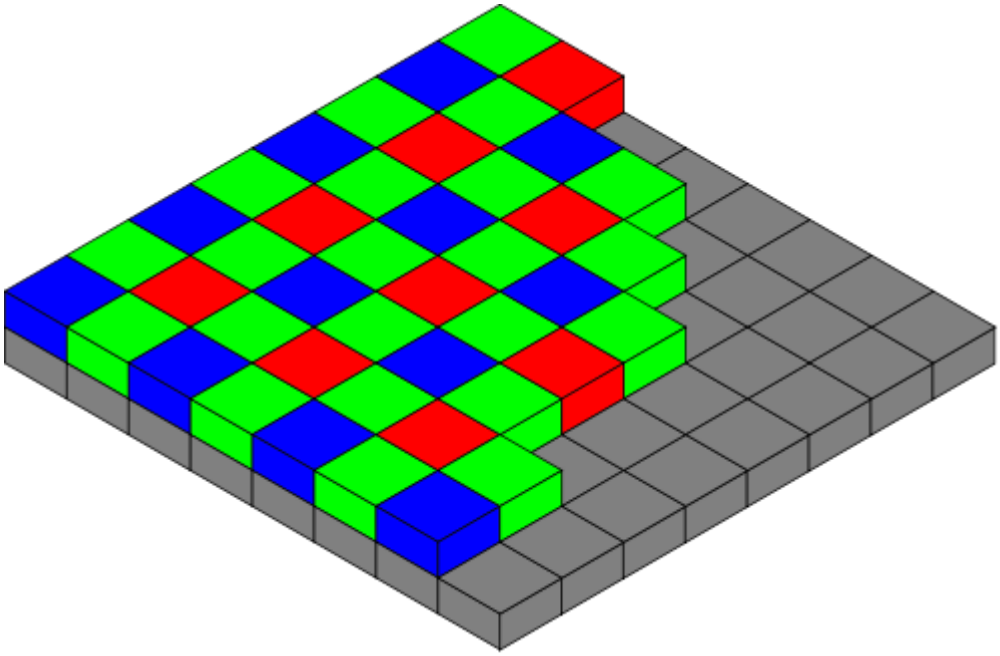
Słowa kluczowe: interpolacja barwna, matryca Bayera, OpenCL, GPU, Kamera HD, JAI BB-500GE.

2. Wstęp

Badanym problemem była równoległa interpolacja obrazy barwnego otrzymanego z czujnika wizyjnego wyposażonego w matrycę CFA. Pracę nad problemem można podzielić na 3 etapy: opracownie algorytmu, implementacja algorytmu w OpenCL[1] oraz akwizycja obrazu z kamery HD. Obraz z kamery należało podać na wejście algorytmu, aby otrzymać poprawnie interpolowany obraz barwny.

2.1. Barwny czujnik wizyjny

Większość kolorowych kamer posiada matryce czujnika zorganizowane w postaci mozaiki kolorowych filtrów (CFA, Color Filter Array). Jedną z najbardziej powszechnych układów filtrów jest topologia matrycy Bayer-a, w której filtry koloru czerwonego, niebieskiego i zielonego ułożone są naprzemiennie, z przewagą filtrów zielonego. Taki układ filtrów pozwala odwzorować sposób postrzegania kolorów przez ludzkie oko, które jest bardziej wrażliwe na światło widzialne w paśmie koloru zielonego. Układ filtrów na matrycy Bayer-a przedstawiono na rysunku 1:



Rysunek 1: Układ filtrów na matrycy Bayer-a (źródło Wikipedia.org)

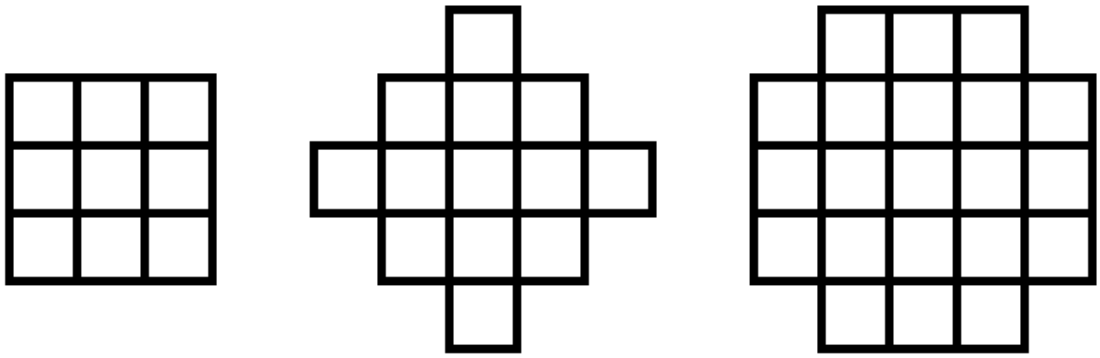
Podstawowym formatem zapisu obrazu kolorowego jest format RGB, gdzie każdy piksel

1 Biblioteka programistyczna pozwalająca wykorzystać zasoby karty graficznej lub wielordzeniowego procesowa do zrównoleglania obliczeń. Więcej na https://www.khronos.org/opencl/

zapisany jest jako trzy wartości intensywności koloru czerwonego, zielonego i niebieskiego. Aby przedstawić obraz z kamery wyposażonej w filtr mozaikowy należy dla każdego piksela zastosować interpolację każdej składowej na podstawie wartości z otoczenia.

2.2 Interpolacja pikseli kolorowych

Interpolacja musi wykorzystywać informacje zawarte w sąsiednich pikselach. Według [1] maska interpolacji o rozmiarze 3 na 3 piksele daje wystarczające dobre rezultaty wizualne. W wymienionej publikacji rozmiar maski był podyktowany wymaganiem budowy linii opóźniającej, w naszym przypadku nie ma to znaczenia, dlatego możemy zastosować dowolny kształt maski (ograniczony jednak kształtem piksela). Rysunek 2 przedstawia wykorzystane kształty masek.



Rysunek 2: Maski wykorzystane do interpolacji kolorów

Przy wykorzystaniu maski pierwszej, gdy pikselem środkowym jest czerwony lub niebieski, jego wartość jest brana tylko i wyłącznie z tego piksela. Przy użyciu maski drugiej dodatkowo brane są pod uwagę 4 piksele o barwie takiej samej jak piksel środkowy. Maska trzecia w porównaniu do maski drugiej używa większej liczby pikseli koloru czerwonego i niebieskiego przy interpolacji, gdy pikselem środkowym jest punkt koloru zielonego.
W rozdziale [5] zostanie pokazane porównanie obrazów wynikowych po interpolacji z wykorzystaniem różnych masek.

2.3 Zrównoleglanie algorytmów w OpenCL

Algorytmy splotowe (takie jak ten) na obrazach bardzo dobrze nadają się do zrównoleglania. Od kilku lat producenci sprzętu komputerowego udostępnią możliwość wykonywania obliczeń równoległych na kartach graficznych. Istnieją dwa wiodące rozwiązania: CUDA, pozwalające wykonywać obliczenia z wykorzystaniem procesora graficznego (GPU) nVidia oraz OpenCL,

otwarty standard programowania współbieżnego na GPU jak i na wielordzeniowych CPU. Za wyborem drugiego rozwiązania przemawia jego dostępność i wieloplatformowość.
Programowanie w OpenCL polega na przygotowaniu jednostki algorytmu (kernela) i uruchomieniu go na wcześniej wgranych danych w środowisku uruchomieniowym dostarczonym przez producenta sprzętu.
Ta biblioteka posiada bardzo dobrą dokumentację , która w nieoceniony sposób przydała się przy implementacji algorytmu. Pomogła dobrze zrozumieć ideę programowania równoległego realizowanego na karcie graficznej.

2.4 Pobieranie obrazu z kamery HD JAI BB-500GE

Kamera udostępnia interfejs poprzez kabel typu Ethernet. Został on użyty podczas implementacji. Dzięki zastosowaniu dołączonej do kamery biblioteki ten punkt był łatwy do zrealizowania. Głównym celem podczas implementacji pobierania obrazu z kamery było nie dopuszczenie do sytuacji, by ta część blokowała wykonanie innych. Zadanie to okazało się proste, ponieważ architektura biblioteki dołączonej do kamery opierała się na rozwiązaniu wielowątkowym, nieblokującym wykonania innych części programu. Wykorzystywana jest kolejka, do której zapisywane są obrazy przychodzące z kamery. Z tej samej kolejki w dowolnym momencie użytkownik może pobrać obraz.

3. Koncepcja proponowanego rozwiązania

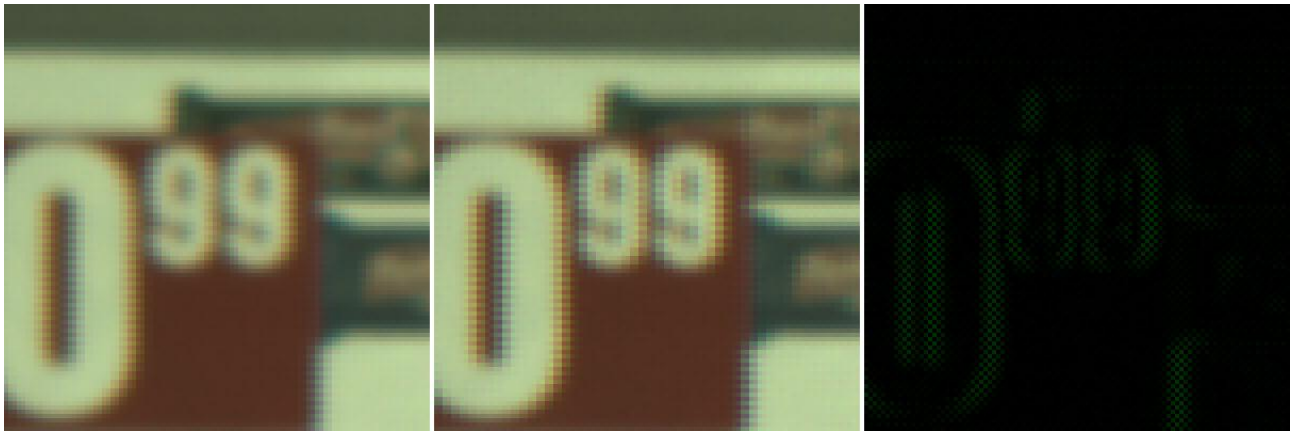
Biblioteka została podzielona na moduły: moduł OpenCL oraz moduł obsługi kamery. Oba moduły są od siebie niezależne, kompilują się do plików DLL i mogą zostać użyte w dowolnych innych projektach.

Moduł algorytmów OpenCL

Moduł ten oparty jest o abstrakcyjną klasę algorytmu wykonywanego w OpenCL. Na bazie tej klasy zaimplementowane zostały algorytmy konwersji typów danych ze całkowitoliczbowych na zmiennoprzecinkowe oraz algorytm interpolacji matrycy Bayera z wykorzystaniem struktury płaskiej (jednowymiarowej tablicy wartości) oraz struktury obrazu 2D, którą udostępnia OpenCL. Struktura ta pozwala na automatyczną interpolacje piksela poza obszarem obrazu (lustrzane odbicie, piksel ramki, etc). Dodatkowo istnieją klasy pomocnicze definiujące urządzenie OpenCL, zapewniające obsługę wyjątków oraz funkcje pozwalające uzyskać dodatkowe informacje o implementacji OpenCL na komputerze. Do przetwarzania obrazu z kamery została użyta klasa strumienia algorytmów, który definiuje listę algorytmów, którymi kolejno przetwarza zadany obraz.

Moduł obsługi kamery

Moduł obsługi kamery składa się z kamery oraz klasy imitującej kamerę. Obsługa kamery została maksymalnie uproszczona i zaimplementowana z użyciem biblioteki dołączonej do kamery. Klasa imitująca kamerę pobiera kolejno obrazy z określonego folderu udostępniając je, dzięki czemu można było przeprowadzać testy bez konieczności posiadania kamery.



Rysunek 3: Fragmentu wynikowego obrazu z (kolejno od lewej) implementacji w OpenCL, implementacji w OpenCV oraz róznicy obrazów.

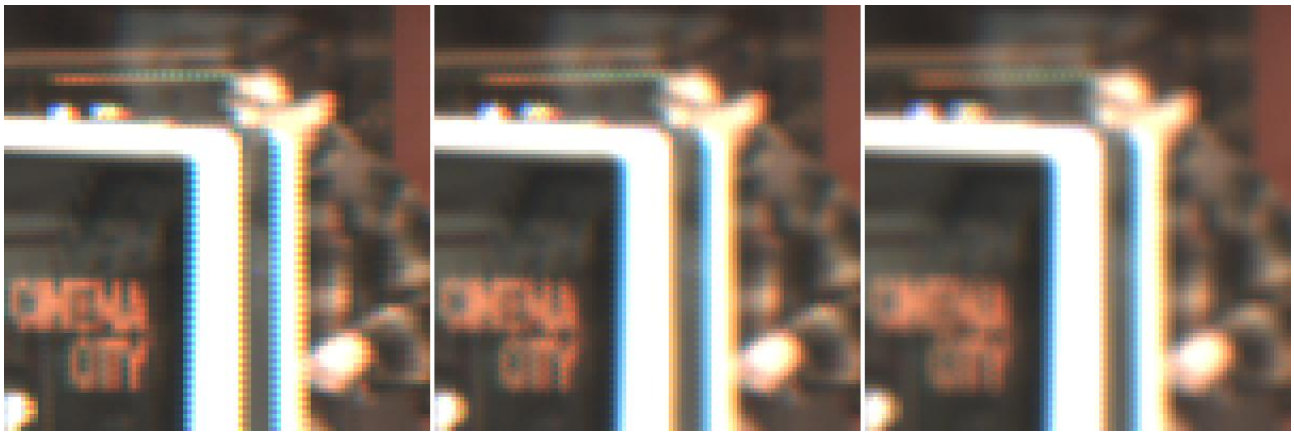
4. Symulacja i Testowanie

Testowanie i weryfikacja poprawności odbyło się za pomocą porównania stworzonej implementacji do już istniejącej. Do porównania wybrano realizację interpolacji z biblioteki OpenCV. Na rysunku 3 znajdują się (kolejno od lewej) fragmenty obrazu wynikowego z algorytmu zaimplementowanego w OpenCL, algorytmu z OpenCV oraz różnica wartości tych obrazów.
Porównując wyniki obu implementacji nie widać większych różnic. Jeśli się pojawiają jest to wynik różnych zaokrągleń oraz przyjętego balansu bieli. Na powyższym rysunku widać różnica występuje w zielonej składowej, jej możliwym powodem jest inny sposób interpolacji koloru zielonego (np. średnia 5 zielonych pikseli zamiast jednego, środkowego).

5. Analiza jakościowa algorytmów

Zostały zaimplementowane 3 różne maski do interpolacji kolorów w przestrzeni RGB z obrazu pobranego z matrycy Bayera. Im rozmiar maski jest większy tym mniej poszarpane są krawędzie na obrazach i zmniejsza się efekt szumu RGB. Im rozmiar maski jest mniejszy tym ostrzejszy jest obraz wynikowy, operacja interpolacji przebiega szybciej. Na rysunku 4 przedstawiono (kolejno od lewej) fragmenty obrazów interpolowanych z maską kwadratową 3x3, maskę okrągłą i maską w kształcie krzyża. Widać na nich jak wraz ze zwiększeniem rozmiaru maski znikają poszarpane

krawędzie, ale również zmniejsza się czytelność obrazka (np. napis „Cinema City” na ostatnim obrazku jest prawie nieczytelny).



Rysunek 4: Fragmenty obrazów interpolowanych z różnymi maskami. Kolejno od lewej: maska 3x3, maska okrągła, maska w kształcie krzyża.

6. Analiza czasowa algorytmu

Porównano czas wykonania obu implementacji: stworzonej oraz dostępnej w bibliotece OpenCV. Dla własnej realizacji porównano wynik wszystkich trzech użytych masek. W OpenCV dostępna jest tylko maska kwadratowa 3x3 piksele. Test był przeprowadzany przy użyciu 79 obrazów o rozmiarze 2456 x 2058 pobranych z kamery. Osiągnięte wyniki przedstawiono w tabeli 1. Czas obliczeń na karcie graficznej jest to czas wykonania algorytmu na karcie graficznej, bez liczenia czasu kopiowania pamięci oraz przeprowadzania koniecznych operacji pomocniczych. Testy zostały przeprowadzone z użyciem karty graficznej nVidia GeForce GT 555M oraz procesorze Intel Core i7-2670QM.

Tabela 1: Tabela porównawcza czasów wykonania przetwarzania obrazów w zaimplementowanych algorytmach

	OpenCV	Maska 3x3	Maska krzyż	Maska okrągła
Czas całości obliczeń	2,548s	2,139s	3,107s	2,309s
Średni czas na jeden obraz	0,032s	0,027s	0,039s	0,029s
Czas obliczeń na karcie graficznej		1,467s	2,462s	1,571s
Średni czas obliczeń na karcie graficzne na jeden obraz		0,018s	0,031s	0,020s

7. Rezultaty i wnioski

Na podstawie wyników testów można zauważyć, że różnice w czasie całego przetwarzania obrazów są niewielkie. Jednak jeśli porówna się czas samych obliczeń (przy realizacji OpenCV nie ma potrzeby kopiowania danych pomiędzy pamięcią a kartą graficzną) to różnica wychodzi znacząca. W celu wykorzystania w pełni możliwości karty graficznej można użyć takiej, która ma większy interfejs pamięci.

8. Podsumowanie

Udało się zrealizować osiągnięte cele. Zaimplementowany algorytm poprawnie interpoluje obraz barwny z kamery korzystającej z matrycy Bayera. Poprawność określono na podstawie prawie identycznych wyników (z dokładnością do błędów obliczeń numerycznych oraz balansu bieli) w porównaniu do sprawdzonej implementacji na procesorze.

Uzyskane wyniki mogą nie być satysfakcjonujące. Należy jednak pamiętać, że do realizacji na karcie graficznej jest potrzebna przeprowadzenia pewnych operacji, które przy implementacji na procesorze nie są konieczne. Taką operacją jest kopiowanie danych wejściowych i wyjściowych pomiędzy pamięcią operacyjną a pamięcią karty graficznej. Dodatkowo testy były przeprowadzane na karcie graficznej z małym interfejsem pamięci.

W kolejnych modyfikacjach można spróbować zidentyfikować główne punkty opóźnień i zoptymalizować kernele, które zostały stworzone. Dodatkowo można rozwinąć bibliotekę i dodać

kolejne algorytmy, które wykorzystywałyby interpolowany obraz. Dzięki strukturze strumienia nakład prac nad kolejnymi algorytmami byłby mniejszy niż realizacja od początku. Również przy dodaniu nowych algorytmów do jednego strumienia ograniczona będzie liczba operacji kopiowania pomiędzy pamięciami procesora i karty graficznej.

9. Literatura

[1]: Mirosław Jabłoński, Metodyka zrównoleglania algorytmów przetwarzania i analizy obrazów w systemach przepływowych, 2009

[2]: Khronos, Dokumentacja OpenCL, 2012,
http://www.khronos.org/registry/cl/sdk/1.1/docs/man/xhtml/

[3]: Google, GTest, 2012, http://code.google.com/p/googletest/

10. DODATEK A: Szczegółowy opis zadania

Specyfikacja projektu

Typowa kamera kolorowa wyposażona w czujnik z filtrem mozaikowym typu Bayer dostarcza strumień wizyjny w postaci poziomów szarości. Celem projektu jest implementacja algorytmu interpolacji składowych R, G i B obrazu z kamery kolorowej wysokiej rozdzielczości w czasie rzeczywistym z wykorzystaniem równoległych zasobów karty graficznej. Algorytm należy zakodować w postaci programu OpenCL (ang. kernel). W ramach projektu należy przeprowadzić testy wydajności poprzez uruchomienie algorytmu na GPU oraz CPU oraz wyznaczyć przyspieszenie względem wersji programowej (np. OpenCV). Oczekiwanym rezultatem jest biblioteka DLL realizująca interpolację z możliwością parametryzacji początkowych współrzędnych matrycy oraz współczynników wzmocnienia kanałów R i B. Oprogramowanie powinno pracę w różnych formatach danych: 8, 10 i 12 bitów. Testy powinny obejmować dwa rodzaje kerneli: wykorzystujące 1-wymiarową strukturę danych oraz 2-wymiarowe obrazy.

11. DODATEK B: Dokumentacja techniczna

Do tworzenia implementacji w systemie Windows użyto programu Visual Studio 2012. Wersja 2012 jest konieczna do kompilacji w systemie Windows z uwagi na użycie niektórych funkcji standardu C++11, które zostały wprowadzone w tej wersji.
W systemie Linux można użyć kompilatora GCC w wersji 4.7.2.
Stworzone testy jednostkowe są zrealizowane w oparciu o GTest [3].

Procedura symulacji i testowania:

Wraz z biblioteką dostarczony jest przykładowy program wykorzystujący możliwości biblioteki (BayerFilter.exe). Pozwala on na przeprowadzenie interpolacji barwnej z obrazów, które są zapisane w odpowiedniej postaci. Dostępne opcje:

- -c – pobieranie obrazów z kamery JAI. Jest to również opcja wybierana w przypadku nie podania żadnej.
- -d katalog prefiks – pobieranie obrazów z katalogu. Obrazy muszą mieć wspólny prefiks, następnie trzycyfrowy kolejny numer (zaczynając od 000), oraz rozszerzenie bmp. Dodatkowa opcja to --openCV, która zmienia realizację z karty graficznej na OpenCV.
- -f plik_wejsciowy –o plik_wyjsciowy – przetworzenie jednego obrazu i zapisanie wyniku w podanym miejscu. Dodatkowa opcja to --openCV, która zmienia realizację z karty graficznej na OpenCV.

- -h – wyświetla pomoc.
- --wb r g b – pozwala ustalić balans bieli. Należy podać kolejno liczby odpowiadające kolorowi czerwonemu, zielonemu i niebieskiemu. Zalecane wartości to od 0.0 do 1.0. Ta opcja nie ma wpływu na wynik przy użyciu opcji --openCV.

Dodatkowo podczas uruchomienia, jeśli jest potrzeba (bez opcji --openCV) program pyta o wybór urządzenia, które należy zastosować do obliczeń z użyciem OpenCL.

Dokumentacja klas

Przygotowany kod był dokumentowany w komentarzach, na podstawie których został wygenerowany podręcznik klas i funkcji który znajduje się w pliku `reference_manual.pdf`. Dokumentacja ta została wygenerowana w programie doxygen.

12. DODATEK D: Spis zawartości dołączonego nośnika (płyta CD ROM)

W poszczególnych folderach nośnika, w zależności od rodzaju projektu, znajdują się:

- **SRC** - Kompletne źródła projektu wraz z plikami VisualStudio 2012 i plikami CMake,
- **EXE** - postać binarna skompilowana na systemie Windows 7 64bit,
- **TEST** - data – obrazki o rozmiarze 1024x1024, data2 – seria obrazków o rozmiarze 2456x2058.
- **DOC** - ten dokument w wersji PDF i MS WORD. Dokumentacja klas w wersij PDF.

13. DODATEK E: Historia zmian

Tabela 2 Historia zmian.

Autor	Data	Opis zmian	Wersja
Bartłomiej Bułat, Tomasz Drzewiecki	2013-01-09	Pierwsza wersja dokumentu	1.0

pr12sr512			**Karta oceny projektu**
Data	Ocena	Podpis	Uwagi