JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 9 (2018) 311-317
Wysłane: 2018-06-25
Przyjęte: 2018-07-01

Metoda zwiększenia pojemności kodów QR – Hexa QR Code

Daniel Janowski
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W artykule zaprezentowano autorską metodę zwiększenia pojemności kodów QR poprzez wykorzystanie ich właściwości do odczytu z dowolnej perspektywy. Przedstawione zostały proces kodowania i dekodowania symbolu oraz wyniki testów. W artykule został również zawarty przegląd literatury z dziedziny metod zwiększania pojemności kodów QR.

Słowa kluczowe: kod QR; Hexa QR Code; kody 3D; zwiększenie pojemności

Adres e-mail: daniel.janowski@pollub.edu.pl

Method of increasing the QR code capacity – Hexa QR Code

Daniel Janowski
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The article presents the author's method of increasing the QR code capacity through using their property of being readable from any perspective. The symbol encoding and decoding process and test results are presented. The article also includes a literature review in the field of method of increasing the QR code capacity.

Keywords: QR code; Hexa QR Code; 3D codes; increasing the capacity

E-mail address: daniel.janowski@pollub.edu.pl

1. Wstęp

Kody kreskowe powstały w celu ułatwienia i przyspieszenia identyfikacji towarów i usług. Ilość danych jaka dzięki nim mogła być przechowywana była niewielka, jednak prostota oraz szybkość i łatwość odczytywania, wpłynęły na ich popularność. Niestety, ograniczeniem w ich odczytywaniu, było używanie specjalnych czytników, dane w nich przechowywane mogły być zrozumiałe w ramach danego systemu, a ich niewystarczająca pojemność względem danych, jakie producenci chcieli by w nich zaszywać, spowodowała, że rozpoczęto poszukiwania alternatywnych rozwiązań.

W latach 90-tych zaczęto opracowywać tak zwane kody 2D, czyli takie, w których dane ułożone są nie tylko w poziomie, ale i w pionie. Dzięki temu rozwiązaniu, ilość przechowywanych danych oraz szybkość ich detekcji przez urządzenia odczytujące wielokrotnie wzrosła. Mogą one pomieścić od kilku do kilku tysięcy znaków, co przekłada się na ich uniwersalność.

Do tej pory powstało wiele standardów, jednak ze względu na swoje wady i zalety oraz wymagania jakie są przed nimi stawiane, do najbardziej popularnych należą: Data Matrix, PDF417 oraz kod QR (ang. Quick Response code). Każdy z tych typów ma swoje zastosowanie w konkretnych segmentach przemysłu. Pierwotnie kody 2D wykorzystywane były w logistyce, transporcie i handlu, ale wraz z rozwojem technologii cyfrowych i internetowych, zaczęto wykorzystywać je również w reklamie i celach rozrywkowych, takich jak technologia wirtualnej rzeczywistości Niestety dane jakie próbowano w nich zamieszczać, były zbyt duże jak na ich możliwości, dlatego zaczęto modyfikować je tak, aby mogły one pomieścić coraz więcej informacji. W toku tych działań powstały kody 3D, których trzecim wymiarem był kolor. W niniejszym artykule zostały przedstawione i porównane różne technologie zwiększenia pojemności kodu QR. Została również przedstawiona implementacja nowego, unikalnego algorytmu.

2. Przegląd literatury

2.1. QR code

Rys. 1. Przykład kodu QR w wersji 2

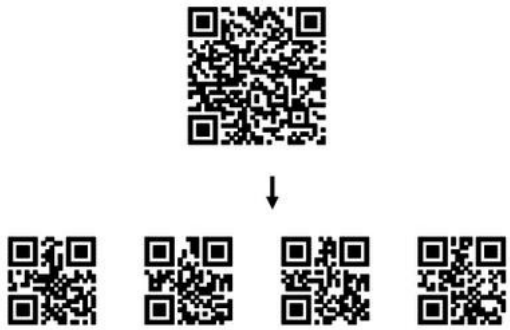
Kod QR (Rys. 1.) został opracowany w 1994 roku przez japońską firmę Denso-Wave, w celu przyspieszenia identyfikacji przez czytniki, przedmiotów szybko poruszających się względem nich. Początkowo używany w przemyśle transportowym i logistyce, szybko został zaadoptowany przez inne gałęzie gospodarki. Stało się to za sprawą upowszechnienia się smartfonów i tabletów, zniesienia licencji na jego stosowanie oraz dużej ilości oprogramowania i bibliotek do jego generowania.

Kod QR został włączony do norm ISO[1] w 2000 roku jako standard ISO/IEC 18004, który definiuje elastyczne rozwiązanie pośród innych kodów 2D. Oferuje on dużą pojemność (do 7 tys. znaków), wsparcie obsługi różnych zestawów znaków (numeryczny (*ang. numeric*), alfanumeryczny (*ang. alphanumeric*), binarny (*ang. byte data*) i Kanji), funkcje zapewniające szybkie i niezawodne wykrycie pod wpływem zmiennego oświetlenia, zmian położenia i orientacji oraz wbudowaną czteropoziomową korekcję błędów algorytmem Reeda-Solomona, którą można przywrócić od ok. 7% do ok. 30% uszkodzonych danych. Charakteryzuje się modularną i stało wymiarową strukturą. Oznacza to, że składa się z modułów będących źródłem niesionej informacji, a liczba modułów w poziomie i w pionie jest stała w obrębie danej wersji. Moduł jest najmniejszą jednostką informacji przechowywanej w kodzie QR, gdzie ciemny moduł oznacza binarną jedynkę, a jasny zero. Kod QR występuje w 40-stu wersjach, które determinują liczbę modułów, co za tym idzie ilość danych możliwych do przechowania [1, 2].

2.2. Tryb „Structured append"

Jedną z funkcjonalności standardowego kodu QR, jest tryb „structured append", umożliwiający połączenie w sekwencji do 16-stu symboli kodu QR lub podzielenie jednego kodu QR na nawet 16 mniejszych kodów (Rys. 2.).

W tym trybie kody mogą być skanowane w dowolnej kolejności, jednak wymagane jest odczytanie ich wszystkich, żeby poprawnie zrekonstruować zawartą w nich wiadomość. Jest to spowodowane nagłówkami zawartymi w poszczególnych symbolach, zawierającymi informację o długości całej sekwencji i pozycji kodu w sekwencji oraz służą do weryfikacji czy symbole należą do jednej i tej samej sekwencji. [1]



Rys. 2. Przykład użycia trybu "Structured append" do podzielenia kodu QR w wersji 4. na cztery kody w wersji 1. [1]

2.3. Kompresja danych

Maksymalna pojemność kodu QR w wersji 40. wynosi ponad 7 tys. znaków, ale jednocześnie taki kod może być nieczytelny lub za duży do umieszczenia go w pewnych miejscach. Najlepszym w takim przypadku rozwiązaniem jest zmniejszenie ilości danych i użycie niższej wersji kodu, jednak aby to osiągnąć bez ich utraty, trzeba użyć algorytmów kompresujących. Jest to bardzo dobra metoda, aby „zapakować" nawet megabajty danych. Niestety, czytniki kodów QR domyślnie nie wspierają kompresji danych, co ogranicza stosowanie tej techniki tylko dla dedykowanych czytników. Aby przyspieszyć wybór odpowiedniego algorytmu i jego interpretację przez czytnik, można użyć wbudowanej w kod QR funkcjonalności – Extended Channel Interpretation (ECI). Służy ona do informowania systemu odbiorczego, że odczytane dane wymagają pewnego określonego dekodowania, dekompresowania i/lub deszyfrowania, zanim będzie można skorzystać z ich zawartości [3].

2.4. Kody 3D

Nazywa się tak kody, które oprócz dwóch wymiarów odpowiadającym zapisywaniu informacji w poziomie i w pionie, posiadają trzeci wymiar, którym jest kolor. Każdemu kolorowi, mogącemu wystąpić w danym kodzie, przypisana jest inna wartość, odpowiadająca ciągowi kilku bitów, w przeciwieństwie do jedno-bitowego kodu QR.

Używa się w nich zazwyczaj 8-bitowej przestrzeni kolorów RGB, która oferuje 16 777 216 różnych kolorów. Jest to uwarunkowane tym, że aparaty cyfrowe głównie w tej przestrzeni pracują oraz jest to podstawowa przestrzeń dla plików graficznych.

$$16\,777\,216 = 256 \times 256 \times 256 = 2^8 \times 2^8 \times 2^8 = 2^{24} = 2^n \quad (1)$$

$$n = \log_2(\textit{liczba użytych kolorów}) \quad (2)$$

gdzie: n – liczba kodów QR możliwych do połączenia

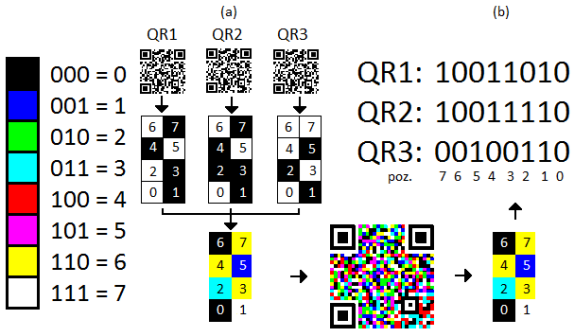
Każdy kanał tej przestrzeni oferuje po 256 różnych odcieni, co równa się możliwości zwiększenia pojemności kodu QR 24-krotnie (1). Na podstawie równania (2), można stwierdzić, że 2^n niezależnych kolorów jest potrzebnych do zakodowania n liczby kodów QR, co za tym idzie dla każdego odrębnego wzoru binarnego jest przypisana odrębna kombinacja wag kanałów R, G, B [4].

Na podstawie technik przedstawionych w dalszej części rozdziału, można wydzielić dwa rodzaje wyznaczania kolorów dla pojedynczych modułów:

- Pozycyjne – polegające na łączeniu kilku kodów QR w jeden kolorowy kod. Liczba łączonych kodów, determinuje liczbę bitów potrzebnych na zakodowanie informacji z każdego kodu. Każdej kombinacji bitów przypisuje się kolor, a pozycja bitu wyznacza kod, z którego pochodzi informacja. Warunkiem jest użycie tej samej wersji kodu we wszystkich wystąpieniach.
  Kodowanie odbywa się poprzez użycie operacji logicznej OR na każdym module z danej pozycji ze wszystkich łączonych kodów, a następnie wybranie koloru odpowiadającemu przeliczonej wartości i wstawienie go na tej samej pozycji w generowanym kodzie. Podczas dekodowania wykrywany jest kolor modułu z danej pozycji, a następnie wybierana jest wartość do niego

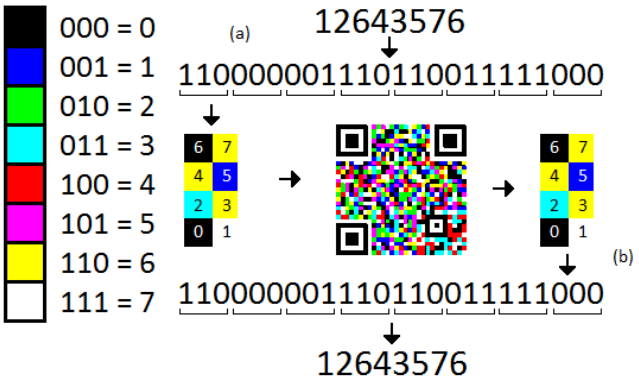
[1] Międzynarodowa Organizacja Normalizacyjna (ang. *International Organization for Standardization*)

przypisana, z której każda kolejna wartość bitu przypisywana jest do kolejnych kodów, które tworzyły kod (Rys. 3.).



Rys. 3. Przykład kodowania trzech czarno-białych słów kodowych do jednego kolorowego słowa kodowego (a) i dekodowania kolorowego słowa kodowego do trzech wartości binarnych (b), w kolorowym kodzie QR

- Wartościowe – polegające na tworzeniu kodu w sposób analogiczny do kodu QR. Liczba kolorów wybranych do zakodowania informacji, determinuje liczbę bitów potrzebnych na zakodowanie jednego modułu. Każdemu kolorowi przypisuje się wartość jaką reprezentuje.
  Kodowanie odbywa się poprzez wyodrębnienie kolejnych ciągów bitów, wybieranie koloru im odpowiadającego i wstawianie go na odpowiedniej pozycji w generowanym kodzie. Podczas dekodowania wykrywany jest kolor modułu z danej pozycji, a następnie wybierana jest wartość do niego przypisana, która jest dołączana do dekodowanej informacji (Rys. 4.).



Rys. 4. Przykład kodowania ciągu znaków do kolorowego słowa kodowego, poprzez wyznaczenie poszczególnych wartości z ciągu bitów (a) i dekodowania kolorowego słowa kodowego do ciągu bitów reprezentujących dane (b)

Poniżej zostały przedstawione po dwie techniki „pozycyjne” i „wartościowe”.

1) Color QR code

Jest to technika „pozycyjna”, polegająca na łączeniu trzech standardowych (czarno-białych) kodów QR w jeden kolorowy, wykorzystując kanały przestrzeni barw CMY do kodowania i RGB do dekodowania. Wzory wyrównania i synchronizacji pozostają czarno-białe, natomiast wzorom detekcji przypisuje się podstawowe barwy przestrzeni barw CMY i RGB (Rys. 5.). Dzięki tej operacji czytnik może synchronizować kolory zeskanowane z oczekiwanymi [5, 6, 7].

Rys. 5. Przykład Color QR code [6]

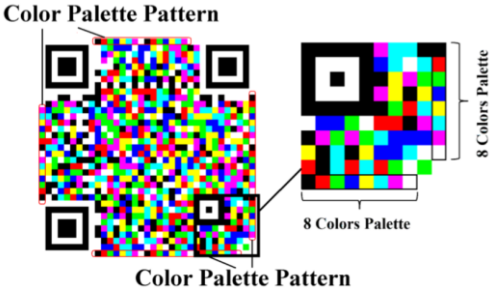
2) Paper Memory Code (PM Code)

Jest to jeden z pierwszych zaprezentowanych kodów 3D bazujących na kodzie QR. Kodowanie danych odbywa się w identyczny sposób jak w poprzedniej metodzie, jednak w tym przypadku pracuje na całej gamie przestrzeni kolorów RGB, dając możliwość połączenia od 3 do 24 czarno-białych kodów QR. Jest to jednak liczba zarezerwowana do użytku lokalnego lub w Internecie. Do zastosowań mobilnych, Wynalazca PM Code sugeruje łączenie od 3 do 8 kodów QR. W PM Code, kolory wzorów detekcji (*ang. finder pattern*), wyrównania (*ang. alignment patterns*) i synchronizacji (*ang. timing patterns*) oraz strefy ciszy (*ang. quiet zone*) są odwrócone w porównaniu do QR code (Rys. 6.) [8, 9, 10].

Rys. 6. Przykład PM Code [10]

3) High Capacity Colored Two Dimensional Code (HCC2D)

Jest to technika „wartościowa”, dzięki temu wymiar kodu HCC2D dostosowywany jest do rzeczywistego rozmiaru danych wejściowych. Wszystkie stałe elementy kodu QR wraz z informacją o wersji i formacie pozostają w tym kodzie niezmienne. Poza tym, udostępnia wszystkie funkcjonalności standardowego kodu QR oraz jest z nim w 100% kompatybilny, czyli czarno-biały kod HCC2D, zostanie bezproblemowo zdekodowany przez czytniki kodów QR. Jeśli kod składa się z co najmniej czterech kolorów. Dochodzi dodatkowy element kodu – paleta użytych barw (*ang. color palette pattern*) – zawierająca posortowane według jasności kolory, które determinują kombinacje bitów do nich przypisane oraz pozwala na lepsze dekodowanie, w przypadku powstania zmiany zabarwienia kodu pod wpływem światła (Rys. 7) [4, 11, 12].



Rys. 7. Przykład HCC2D z uwzględnieniem palety użytych barw [12]

4) Colored Quick-Response Code (CQRcode)

Jest to również technika „wartościowa”, jednak ogranicza się ona do czterech kolorów – trzech podstawowych kolorów przestrzeni RGB i białego. CQR Code ma tylko jeden rozmiar wersji, składający się z 49x49 modułów (wersja 8. kodu QR). W porównaniu do standardowego kodu QR w tej wersji, CQR Code wspiera tylko kodowanie danych binarnych i korekcję błędów opartą o algorytm Reeda-Solomona, natomiast nie występują wzory wyrównania, synchronizacji oraz informacje o wersji (Rys. 8) [13].

Rys. 8. Budowa i przykład CQRcode [13]

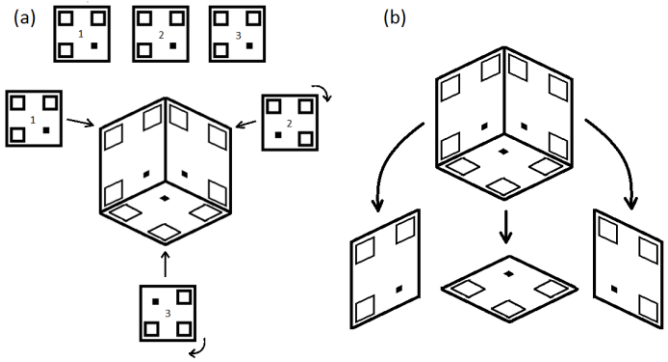
3. Propozycja metody zwiększenia pojemności kodu QR - Hexa QR Code

W przeciwieństwie do zaprezentowanych w poprzednim rozdziale technik, zaproponowana metoda nie generuje ani nie modyfikuje w żaden sposób generowania „ciała” kodu QR. Służy ona do przygotowania miejsca, w którym dane symbole mają być umieszczone. Zostały w niej wykorzystane dwie główne jego zalety – możliwość odczytu z dowolnej perspektywy. Kolejnymi ważnymi cechami były stały wymiar oraz forma kwadratu. Termin kod 3D został potraktowany w tym przypadku bardziej dosłownie, ponieważ do jego generowania wykorzystano geometrię przestrzenną i technologię 3D.

Rys. 9. Przykład Hexa QR Code w wersji 2.

Generowanie kodu odbywa się poprzez wykorzystanie sześcianu, który rzutowany jest izometrycznie na jeden z jego wierzchołków pod kątem 45º. Wierzchołek ten, wyznacza środek tak powstałego na rzutowanej płaszczyźnie sześciokąta foremnego, zaś osie wyznaczają krawędzie oraz ściany sześcianu wykorzystywane w kolejnych krokach. Nic jednak nie stoi na przeszkodzie, żeby użyć przeciwnych krawędzi. Powstanie w ten sposób odwrócona o 180º wersja tej metody.

Do wypełnienia wyznaczonych ścian, mogą zostać użyte trzy niezależne kody QR lub zależne od siebie kody QR utworzone przy użyciu trybu „structured append”. Możliwe jest również użycie kodów 3D, wspierających wyżej wymienione wymagania, aby dodatkowo zwiększyć ich potencjał. Symbol „lewy” wypełnia ścianę bez modyfikacji. Natomiast symbole „prawy” i „dolny”, podczas wypełniania tych ścian, muszą zostać obrócone odpowiednio o 90º i o 180º zgodnie z ruchem wskazówek zegara (Rys. 10.). Dzięki tej operacji każdy z kątów sześciokąta posiada po dwa wzorniki pozycji, a kody zwrócone są ku środkowi figury. Ważnym elementem jest pozostawienie oryginalnej lub nawet nieco większej strefy ciszy każdemu kodowi w trakcie wypełniania, co umożliwi odczytanie ich również zwykłym czytnikiem.



Rys. 10. Schemat kodowania (a) i dekodowania (b) Hexa QR Code

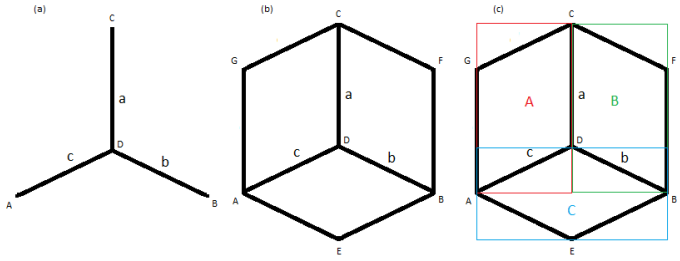
Odczytywanie kodu polega na podzieleniu go na trzy części wzdłuż linii, które wyznaczały poprzednio używane krawędzie. Następnie każdy z tak utworzonych obrazków należy wczytywać do dekodera kodów QR (Rys. 10.).

3.1. Implementacja

Ze względu na wykorzystane w algorytmie kodowania bryły i figury geometryczne, najprostszym sposobem na jego implementację było wykorzystanie grafiki 3D. Oferuje ona wszystkie przedstawione w założeniach operacje i przekształcenia. Dodatkowo algorytm ten może być zaimplementowany nie tylko w językach programowania wysokiego lub niskiego poziomu, ale również w programach do obróbki grafiki trójwymiarowej. Zależnie od wybranej techniki, tworzenie i renderowanie grafiki 3D, odbywa się w inny sposób i przy użyciu innych metod, jednak poniższe instrukcje są uniwersalne.

Podstawowym krokiem jest stworzenie wirtualnej sceny, do której dodawane będą kolejne wymagane obiekty. Pierwszym z nich jest wirtualna kamera, rzutująca izometrycznie na obiekty ustawione w jej polu „widzenia”. To ona wizualizuje i generuje zdjęcie przygotowanej sceny. Wizualizacja bazuje jednak na świetle, dlatego jego utworzenie jest kolejnym krokiem przy tworzeniu sceny. Najlepszym w tym przypadku wyborem, było użycie tak zwanego światła rozproszonego (*ang. ambient light*) oraz przypisanie go do obiektu 3D. Dzięki temu obiekt generuje światło na scenie i zapewnia mu równomierne oświetlenie z każdej strony. Wspomnianym obiektem jest sześcian o wymiarach 1:1:1, ustawiony w środku sceny. Analogicznie do założeń, kamera została ustawiona tak, aby rzutowała pod kątem 45º na jeden z wierzchołków bryły. Ściany sześcianu zostały tak przygotowane, aby materiał, czyli obrazy do nich

przypisywane, nie duplikowały się oraz wypełniały je odpowiednio obrócone.



Rys. 11. Schemat wydzielania obszarów dekodowania w Hexa QR Code

Ze względu na operacje na zdjęciach i obrazach, w algorytmie dekodowania została wykorzystana grafika 2D oraz rysowanie na płaszczyźnie.

Najważniejszym elementem jest wyznaczenie trzech jednakowej długości odcinków, których jeden punkt jest wspólny dla wszystkich, natomiast odcinki były odchylone od siebie pod kątem 120º, tak aby ich punkty tworzyły wierzchołki trójkąta równobocznego. Dzięki temu możliwe jest wykorzystanie ich w kolejnych krokach.

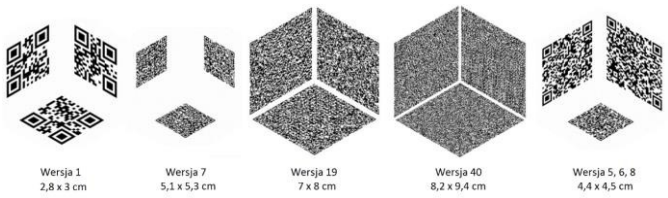
Na rysunku 11. zostały przedstawione trzy części A, B i C, wycinane z wczytanego obrazu, stanowiące obszary, w których dekoder powinien szukać kodów QR. Przed przystąpieniem do wycinania, następuje wyznaczenie trzech dodatkowych punktów, tworzących wraz z innymi punktami, sześciokąt foremny. Jednocześnie tworzone są równoległoboki z odcinkami a, b i c jak przedstawiono na rysunku, wyznaczające miejsca w których powinien znaleźć się kod. Wyznaczenie granic dla każdej części, odbywa się poprzez wyznaczenie najmniejszych i największych wartości x i y, z pośród wierzchołków danych figur. Następnie na ich podstawie wycina się fragmenty z wejściowego obrazu i maskuje, czyli zamalowuje się jednolitym kolorem, miejsca poza granicami poprzednio wyznaczonych równoległoboków. Ma to na celu zapobieganie powstawania przekłamań podczas wykrywania kodu QR, gdyż fragmenty sąsiednich kodów mogły by znaleźć się w wyciętym obrazie. Aby przyspieszyć ten proces, każda część powinna być przetwarzana równolegle.

4. Testy zaimplementowanego algorytmu

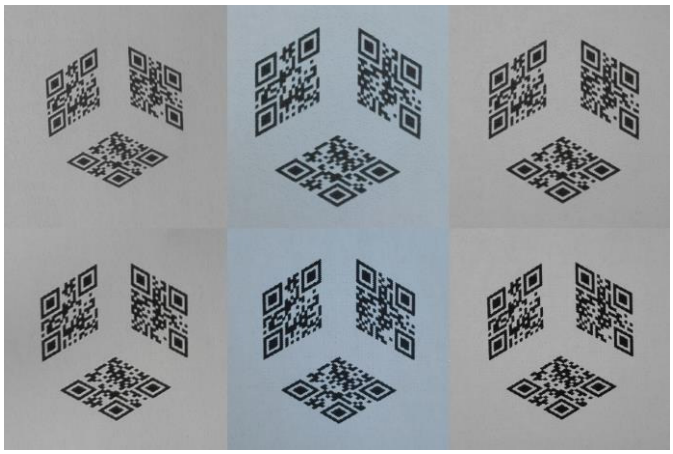
W celu implementacji i przeprowadzenia testów powyższych algorytmów, została przygotowana aplikacja na komputer stacjonarny w technologii WPF, generująca i dekodująca Hexa QR Code, przy użyciu biblioteki Zxing.NET [14]. Testy zostały podzielone na dwa etapy, oba przeprowadzone na jednym komputerze stacjonarnym wyposażonym w procesor Intel Core i5 1,6 GHz, 8 GB pamięci RAM i kartę graficzną Nvidia GeForce GT 2 GB.

W pierwszym etapie, wygenerowane symbole były od razu wczytywane do dekodera, aby sprawdzić jego możliwości, oraz to jaka jakość jest potrzebna, aby poprawnie odczytać nawet najwyższe wersje kodu QR. Drugi etap miał na celu przetestowanie Hexa QR Code'u w bardziej „naturalnych" warunkach.

Do testów zostało wygenerowanych pięć Hexa QR code'ów w rozdzielczości 1000x1000 px (Rys. 12.). Cztery w danym zestawie składały się z kodów QR o takiej samej wersji (1., 7., 20. i 40.), natomiast jeden składał się z kombinacji trzech różnych wersji (5., 6. i 8.). Zostały one wydrukowane drukarką atramentową Canon MG3053 i laserową Brother HL-2250DN, a jako próbki posłużyły ich zdjęcia, zrobione w świetle lampki biurowej oraz w świetle dziennym w nasłonecznionym i zacienionym miejscu, wbudowanym aparatem cyfrowym 8 Mpx w Huawei P8 Lite (Rys. 13). Symbole zostały wydrukowane w różnych wielkościach, ponieważ w przeciwnym wypadku były by całkowicie nieczytelne. Aby zostać poprawnie odczytane przez dekoder, musiały być uprzednio przygotowane. Należało z każdego zdjęcia wyciąć fragment w kształcie kwadratu tak, aby jego środek pokrywał się orientacyjnie ze środkiem symbolu.



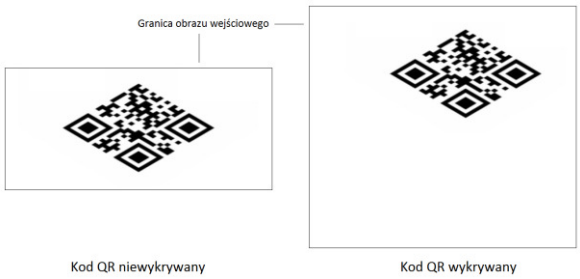
Rys. 12. Wygenerowane symbole użyte do testów



Rys. 13. Zdjęcia symboli w wersji 1. wydrukowane przy użyciu drukarki atramentowej (wiersz 1.) i drukarki laserowej (wiersz 2.), wykonane w świetle dziennym w nasłonecznionym (kolumna 1.) i zacienionym (kolumna 2.) miejscu oraz w świetle lampki biurowej (kolumna 3.)

Niestety możliwości odczytu Hexa QR Code, są uzależnione od możliwości odczytu biblioteki dekodującej. W niniejszej pracy została wykorzystana biblioteka ZXing.NET, która jest bardzo dobrym wyborem w przypadku odczytywania kodów QR, ustawionych na wprost z niewielkim marginesem błędu. Zniekształcenia wynikające z odmiennego sposobu odczytywania symbolu, powodują w tym przypadku niestabilne zachowanie się dekodera, a wykrycie kodu QR zaszytego w Hexa QR Code, jest często losowe. Trudności z dekodowaniem kodu QR objawiały się, gdy pochylony symbol był obrócony o kąt 150º, 240º lub 330º

(±10º – 15º) względem środka, a po przeprowadzeniu kilku dodatkowych testów przy użyciu tej biblioteki oraz biblioteki ZXing [15], okazało się, że wpływ na wykrycie takiego kodu ma również jego położenie i rozmiar na obrazie wejściowym (Rys. 14).



Rys. 14. Wpływ położenia pochylonego symbolu i rozmiaru obszaru roboczego, na możliwość jego wykrycia przez bibliotekę ZXing.Net i ZXing

5. Wnioski

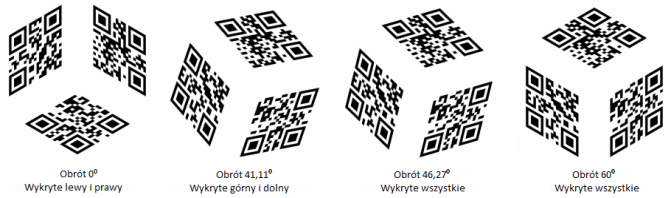
Przedstawiona w niniejszej pracy metoda zwiększenia pojemności kodu QR jest innowacyjna, jednak testy wykazały, że nie jest pozbawiona wad.

Zaproponowany algorytm kodowania jest prosty i w dużej mierze korzysta z możliwości grafiki 3D, co wspomaga jego przenośność pomiędzy technologiami. Dlatego może być zaimplementowany nie tylko w językach programowania wysokiego lub niskiego poziomu, ale jest to również możliwe w programach do obróbki grafiki trójwymiarowej. Takie rozwiązanie może pomóc w ograniczeniu kosztów jego implementacji, na przykład poprzez używanie bezpłatnego oprogramowania.

Tabela 1. Porównanie wyników dekodowania Hexa QR code, otrzymanych podczas testów

Typ zdjęć	Warunki	Obrót	Wersja 1	Wersja 7	Wersja 19	Wersja 40	Wersja 5,6,8
Wygenerowane	-	0º	2/3	3/3	X	X	2/3
		13º	3/3	2/3	2/3	X	2/3
		46º	3/3	3/3	1/3	X	3/3
Wydruk atramentowy	Dzienne - nasłonecznione	0º	1/3	3/3	X	X	2/3
		13º	2/3	1/3	X	X	2/3
		46º	3/3	2/3	X	X	2/3
	Dzienne - zacienione	0º	1/3	3/3	1/3	X	1/3
		13º	1/3	1/3	X	X	1/3
		46º	1/3	2/3	X	X	1/3
	Sztuczne - lampka biurowa	0º	3/3	2/3	1/3	X	2/3
		13º	2/3	2/3	1/3	X	2/3
		46º	3/3	2/3	X	X	2/3
Wydruk laserowy	Dzienne - nasłonecznione	0º	3/3	X	X	X	2/3
		13º	2/3	X	X	X	2/3
		46º	3/3	X	X	X	2/3
	Dzienne - zacienione	0º	2/3	1/3	X	X	1/3
		13º	3/3	X	X	X	1/3
		46º	3/3	X	X	X	X
	Sztuczne - lampka biurowa	0º	3/3	X	X	X	1/3
		13º	2/3	X	X	X	2/3
		46º	3/3	X	X	X	1/3

Takie ograniczenie, miało szczególny wpływ na dolny obszar wykrywania, ponieważ kod QR w nim zawarty nie zawsze był wykrywany przez dekoder. Wczytanie obrazu z Hexa QR Code obróconym pod odpowiednim kątem na szczęście pomagało (Rys. 15.), dlatego w drugim etapie, wszystkie przygotowane zdjęcia zostały dodatkowo obrócone o kąt 13º oraz 46º. W tabeli 1. zostały zestawione wyniki testów wykrycia symboli dla poszczególnych konfiguracji.



Rys. 15. Wpływ obrotu Hexa QR Code, na możliwość wykrycia poszczególnych kodów QR przez bibliotekę ZXing.Net i ZXing

Fundamentalną rolę odgrywa tu dekoder symboli, który również musi być przystosowany do pracy w warunkach, jakie przewiduje specyfikacja kodów QR – możliwości odczytywania symbolu z różnej perspektywy. Technika ta nie ogranicza się tylko do kodów QR. Służy ona do przygotowania miejsca, w którym dane symbole mają być umieszczone, zatem użycie kodu, który wspiera możliwość odczytania pochylonego lub obróconego symbolu, jest możliwe. Zalecane jest jednak, aby zarówno symbol jak i wydruk były jak najlepszej jakości, a wybrany rozmiar był adekwatny do wybranej wersji symbolu, tak aby zapewnić jego poprawne odczytanie w różnych warunkach.

W porównaniu do przedstawionych w rozdziale 2.4 metod korzystających z koloru do zwiększenia pojemności symbolu, Hexa QR Code zajmuje 2,6 razy więcej powierzchni drukowanej. Porównując go jednak z trzema standardowymi kodami QR, ułożonymi obok siebie w podobnym stylu (dwa na górze, jeden na dole), zajmuje 0,86 razy mniej miejsca. Atutem też jest wsparcie odczytania każdego z zawartych symboli przez standardowe czytniki kodów QR.

Ze względu na użycie nieprofesjonalnych drukarek oraz ograniczenie rozdzielczości generowanego symbolu do 1000x1000 px, jakość wydruków testowych była niska, dlatego najlepiej dekodowane były kody w wersjach 1. – 8.

Literatura

[1] ISO/IEC, 18004:2006 Information technology — Automatic identification and data capture techniques — QR Code 2005 bar code symbology specification, Geneva: ISO/IEC, 2006.

[2] Denso-Wave Inc., „QRcode.com", http://www.qrcode.com/en/ [01.06.2018]

[3] AIM, Inc., „AIM", https://www.aimglobal.org/ [01.06.2018]

[4] A. Grillo, A. Lentini, M. Querini i G. F. Italiano, „High Capacity Colored *Two* Dimensional Codes," *Proceedings of the IMCSIT,* tom 5, pp. 709-716, 2010.

[5] O. Bulan, G. Sharma i V. Monga, „High Capacity Color Barcodes Using Dot Orientation and Color Separability," *SPIE-IS&T Electronic Imaging,* tom 7254, nr 725417, pp. 1-7, 2009.

[6] O. Bulan, H. Blasinski i G. Sharma, „Color QR Codes: Increased Capacity Via Per-Channel Data Encoding and Interference Cancellation," *Color and Imaging Conference Final Program and Proceedings,* nr 14627-0126, pp. 156-159, 2011.

[7] H. Blasinski i S. Fok, Mobile Color Barcode Reader*,* Stanford: Stanford University, 2011-2012.

[8] E. Jenks , „QR Based PM Code", 11.01.2007, http://symblogogy.blogspot.com/2007/01/qr-based-pm-code-best-3d-symbology-ever.html [01.06.2018]

[9] H. Kato, K. T. Tan i D. Chai, Barcodes for Mobile Devices, 1 red., Nowy Jork: Cambridge University Press, 2010.

[10] Inc., PMCode, „PMCode", http://pm-code.com/eng/ [01.06.2018]

[11] M. Querini, A. Grillo, A. Lentini i G. F. Italiano, „2D Color barcodes for mobile phones," *International Journal of Computer Science and Applications,* tom 8, nr 1, pp. 136-155, 2011.

[12] M. Querini i G. F. Italiano, „Reliability and Data Density in High Capacity Color Barcodes," *Special Issue on Advances in Systems, Modeling, Languages and Agents,* tom 11, nr 4, p. 1595–1615, 2014.

[13] M. E. V. Melgar, A. Zaghetto, B. Macchiavello i A. . C. A. Nascimento1, „CQR codes: Colored quick-response codes," w *IEEE Second International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, Berlin, 2012.

[14] M. Jahn, „ZXing.Net", https://github.com/micjahn/ZXing.Net [01.06.2018]

[15] S. Owen, „ZXing", https://github.com/zxing/zxing [01.06.2018]