QR Codes

suivez les points, sans vous perdre

Sébastien Chédor

onepoint.
beyond the obvious

Benoît Masson

OVHcloud

Breizh C@mp 2024
La conférence à l'Ouest

Qui sommes-nous

Sébastien Chédor

- développeur Typescript
- fou du typage
- équipe Wagyz

onepoint.
beyond the obvious

Benoît Masson

benoitmasson

- développeur Go
- software crafter
- noms de domaines

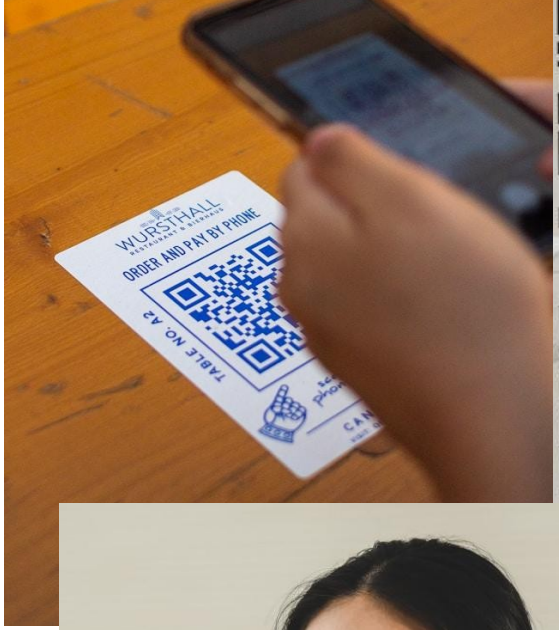
OVHcloud

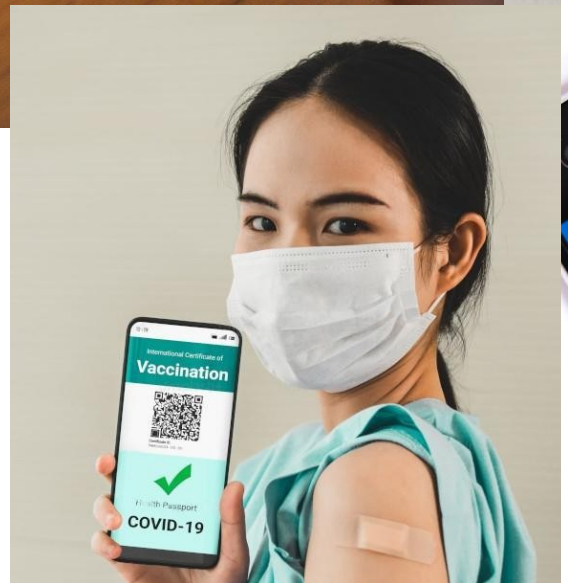
Motivation et objectifs

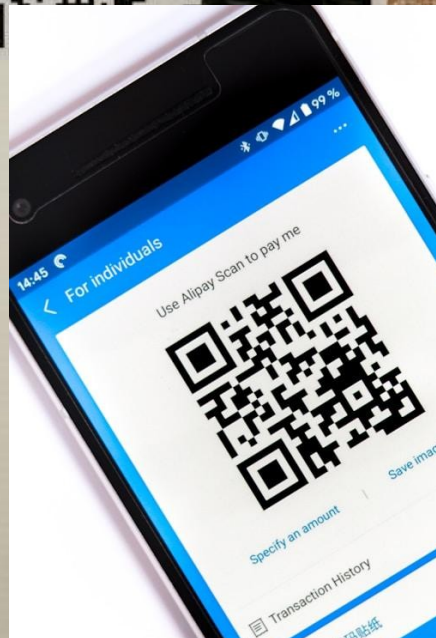
- Ils sont partout !

- Moyen original et robuste d’encoder des données

- Plusieurs concepts abordés :
  - vision par ordinateur
  - distribution de l’information
  - correction d’erreur

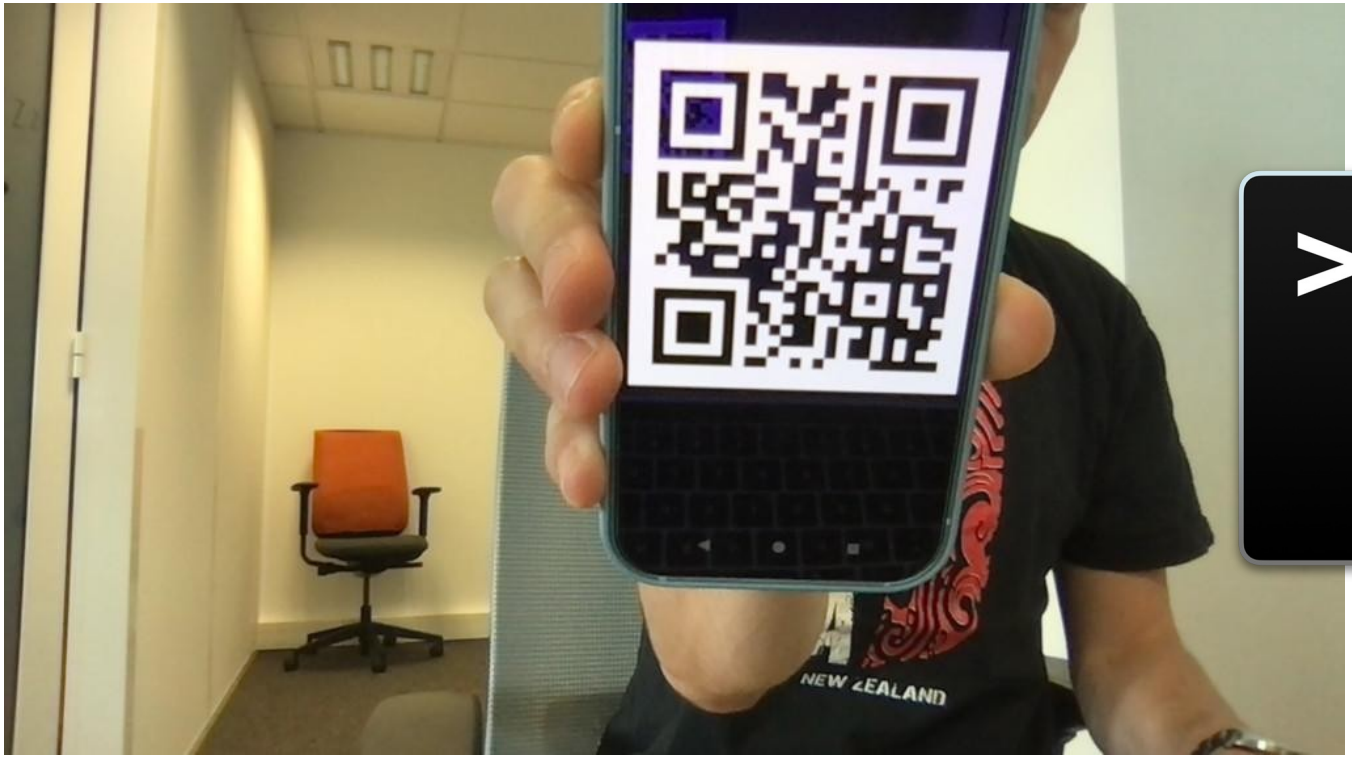


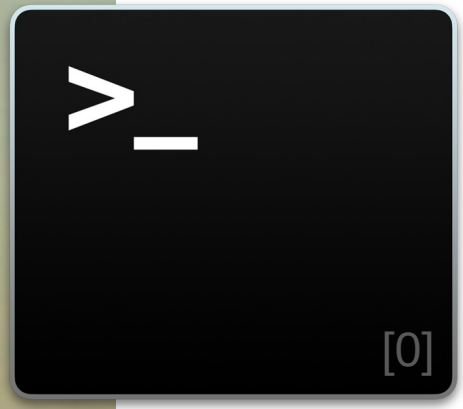


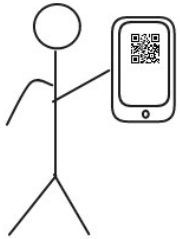


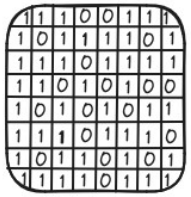
Crédits photos : unsplash.com

Détection du QR-code - live coding









0110010101110001101001001

Détection

Traitement de l'image

- Localisation
- Redressage
- Amélioration
- Lecture des points

Extraction

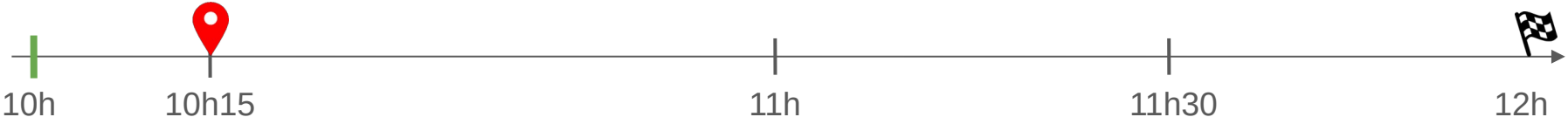
Structure d'un QR code

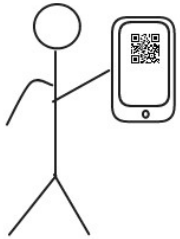
- Métadonnées
- Masque
- Eléments de structure
- Lecture de bits

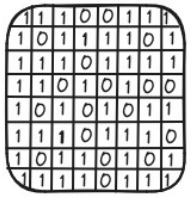
Décodage

décodage du message

- Correction d'erreur
- En-tête
- Lecture du message







0110010101110001101001001

Détection

Traitement de l'image

Localisation

Extraction

Structure d'un QR code

Décodage

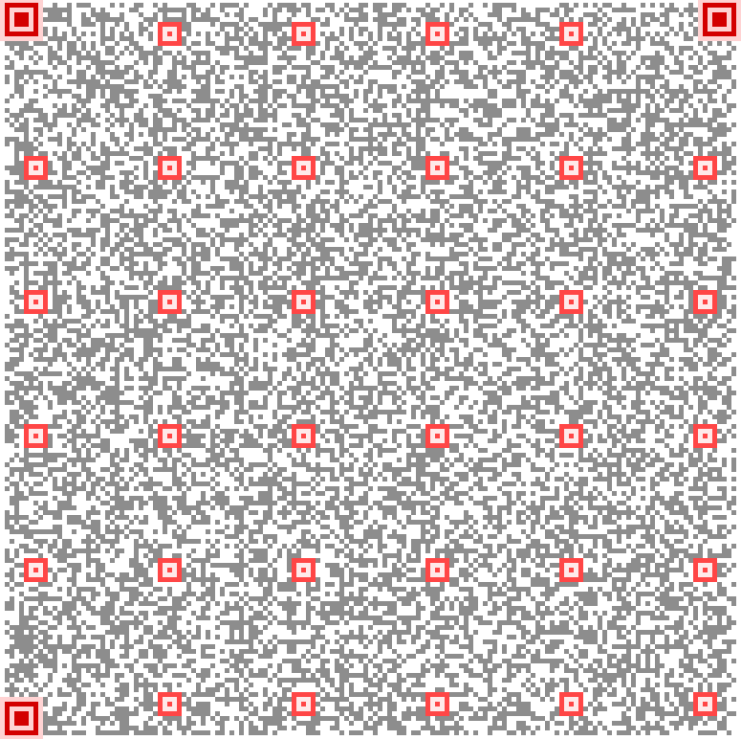
décodage du message

Détection : localisation

Objectif : Détecter les quatre coins d’un QR code dans une image

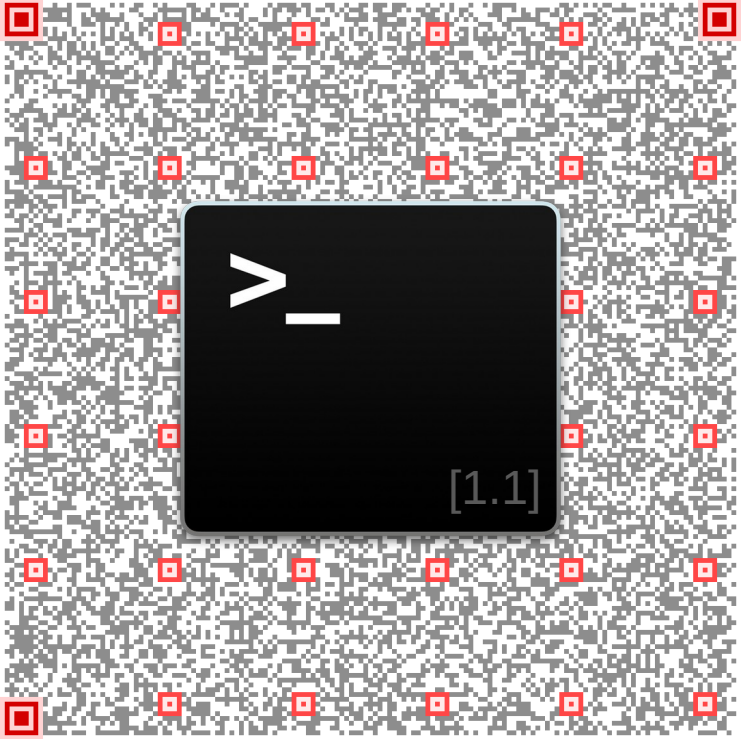
Détection : localisation

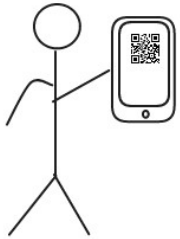
Objectif : Détecter les quatre coins d’un QR code dans une image

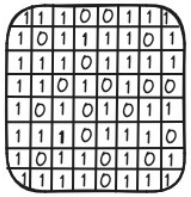


Détection : localisation

Objectif : Détecter les quatre coins d’un QR code dans une image







0110010101110001101001001

Détection

Traitement de l'image

Localisation

Redressage

Extraction

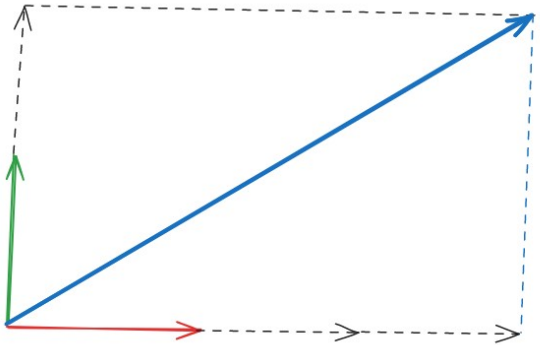
Structure d'un QR code

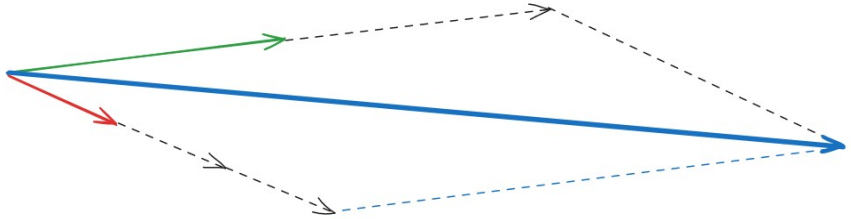
Décodage

décodage du message

Détection : redressage (transformations)

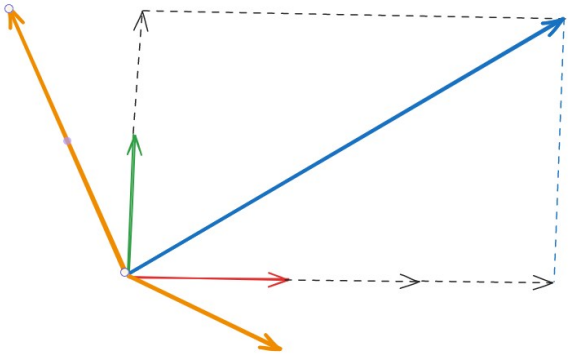
Objectif : Transformer une image en une autre

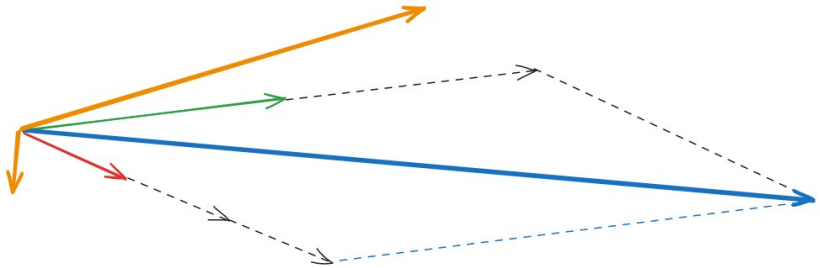




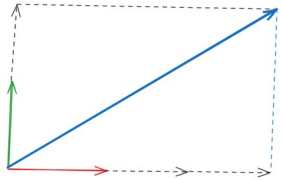
Détection : redressage (transformations)

Objectif : Transformer une image en une autre



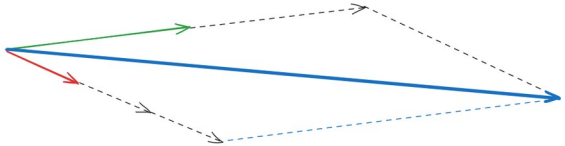


Détection : redressage (transformations)



blue= 3 red + 2 green

newRouge = 0.7 red – 0.5 green
newGreen = 2 red + 0.2 green



newBlue = 3 newRed + 2 newGreen
= 3 (0.7 red – 0.5 green) + 2(2 red + 0.2 green)
= 6.1 red – 1.1 green

Détection : redressage (transformations)

Objectif : Transformer une image en une autre

En deux dimensions => Multiplication par une matrice 2x2

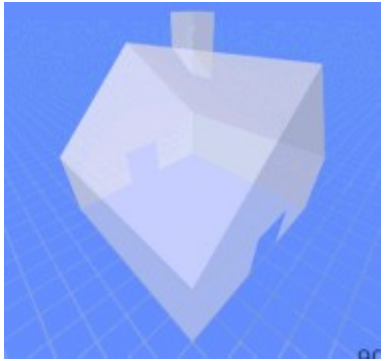
Ici, perspective, donc trois dimensions => Multiplication par une matrice 3x3

Détection : redressage (transformations)

Objectif : Transformer une image en une autre

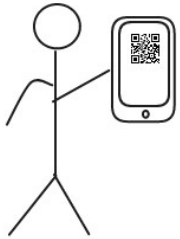
En deux dimensions => Multiplication par une matrice 2x2

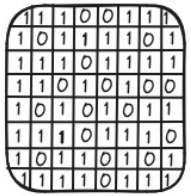
Ici, perspective, donc trois dimensions => Multiplication par une matrice 3x3



Détection : redressage

Objectif : Obtenir une image du QR Code isolé et redressé





0110010101110001101001001

Détection

Traitement de l'image

Localisation

Redressage

Amélioration

Extraction

Structure d'un QR code

Décodage

décodage du message

Détection : amélioration de l’image

Objectif : Obtenir une image en noir et blanc, sans bruit

- Élimination du bruit :
  - Érosion : Tous les pixels autour sont blancs *(max)*
  - Dilatation : Au moins un pixel est blanc *(min)*
  - Ouverture : Érosion, puis dilatation
  - Fermeture : Dilatation, puis érosion

:= Érosion

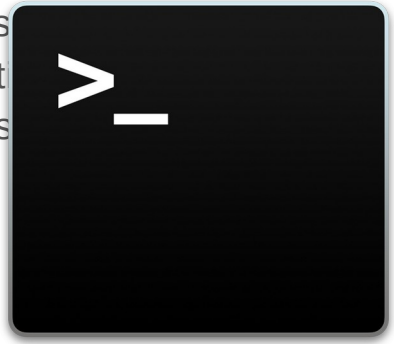
Dilation =>

Détection : amélioration de l’image

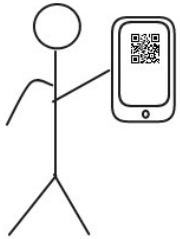
Objectif : Obtenir une image en noir et blanc, sans bruit

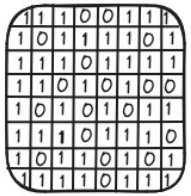
- Élimination du bruit :
  - Érosion : Tous les pixels autour sont blancs *(max)*
  - Dilatation : Au moins un pixel es
  - Ouverture : Érosion, puis dilatat
  - Fermeture : Dilatation, puis éros

:= Érosion



Dilation =>





0110010101110001101001001

Détection

Traitement de l'image

- Localisation
- Redressage
- Amélioration
- Lecture des points

Extraction

Structure d'un QR code

Décodage

décodage du message

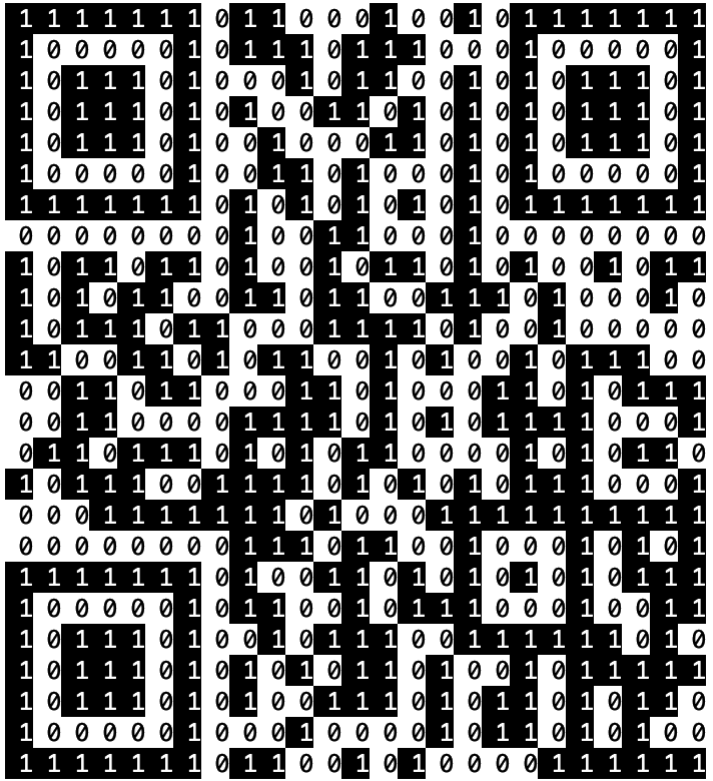
Détection : lecture des points

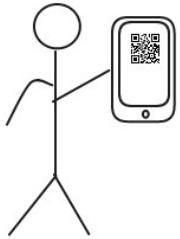
Objectif : Récupérer une grille de points

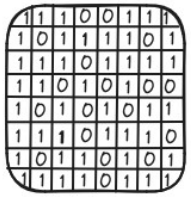
Détection : lecture des points

Objectif : Récupérer une grille de points

- Calculer les coordonnées des points à lire
- Lire des coordonnées







0110010101110001101001001

Détection

Traitement de l'image

- Localisation
- Redressage
- Amélioration
- Lecture des points

Extraction

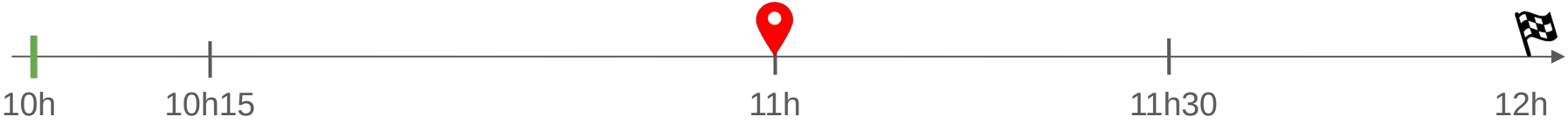
Structure d'un QR code

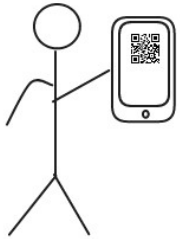
- Métadonnées
- Masque
- Eléments de structure
- Lecture de bits

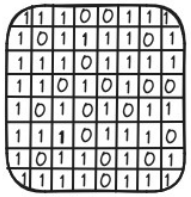
Décodage

décodage du message

- Correction d'erreur
- En-tête
- Lecture du message







0110010101110001101001001

Détection

Traitement de l'image

- Localisation
- Redressage
- Amélioration
- Lecture des points

Extraction

Structure d'un QR code

- Métadonnées

Décodage

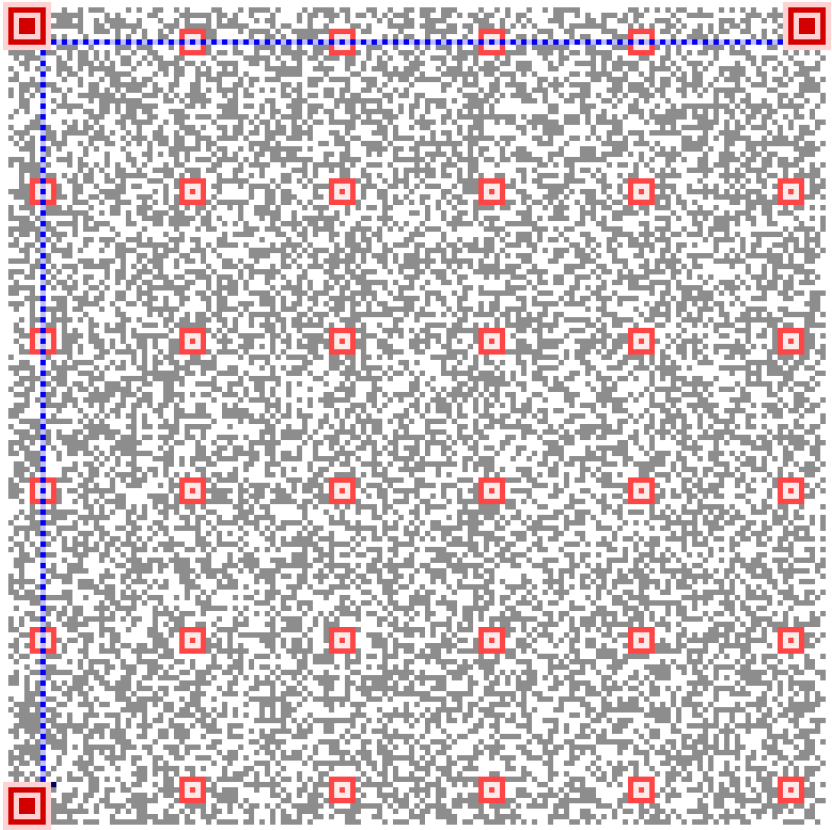
décodage du message

Extraction : métadonnées

Objectif : métadonnées
du code (version)

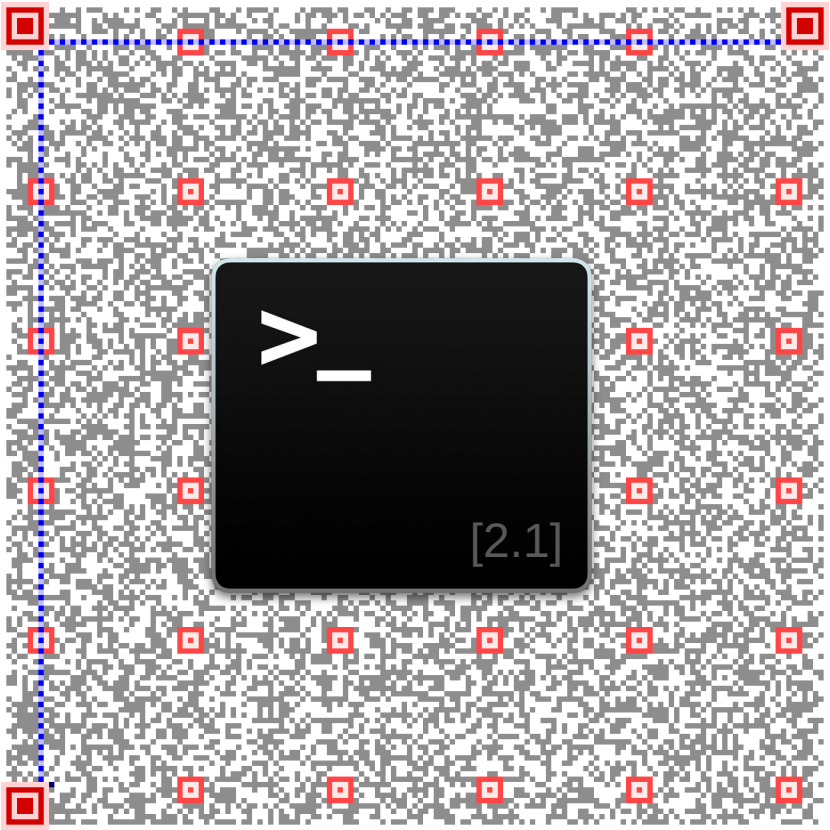
Extraction : métadonnées

Objectif : métadonnées du code (version)



Extraction : métadonnées

Objectif : métadonnées du code (version)

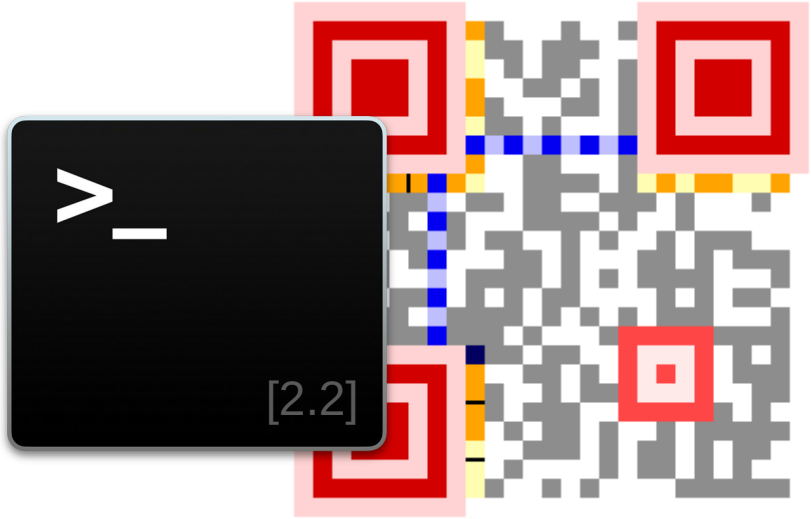


Extraction : métadonnées

Objectif : métadonnées
du code (format)

Extraction : métadonnées

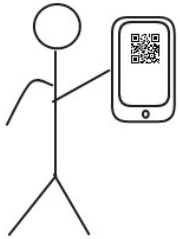
Objectif : métadonnées
du code (format)

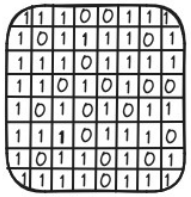


Extraction : métadonnées

Objectif : métadonnées
du code

- Version (= *taille*)
- Niveau de la correction d’erreur
- Masque





0110010101110001101001001

Détection

Traitement de l'image

- Localisation
- Redressage
- Amélioration
- Lecture des points

Extraction

Structure d'un QR code

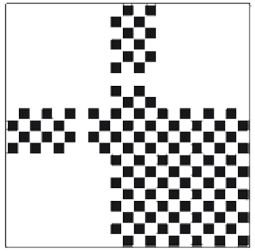
- Métadonnées
- Masque

Décodage

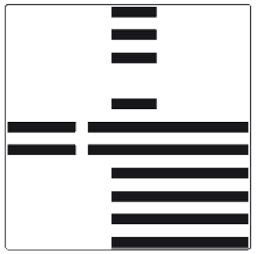
décodage du message

Masques

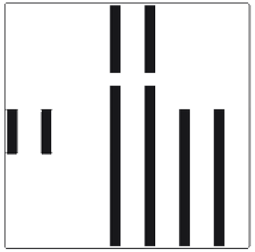
0 0 0



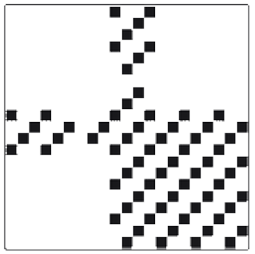
0 0 1



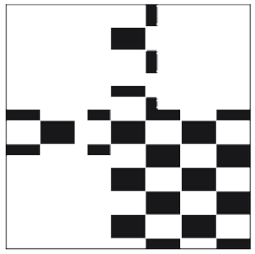
0 1 0



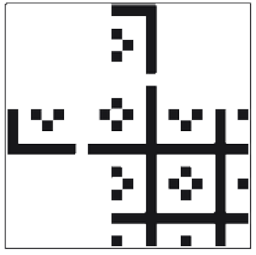
0 1 1



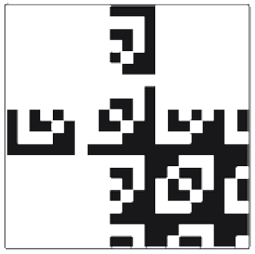
1 0 0



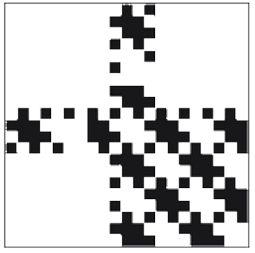
1 0 1



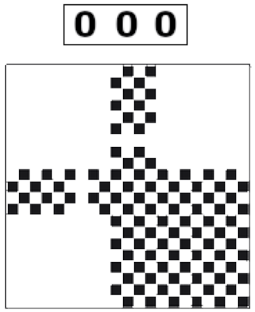
1 1 0

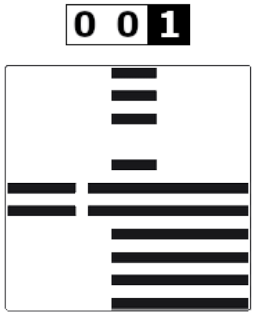


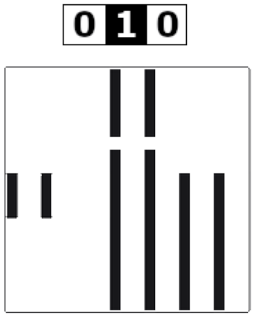
1 1 1

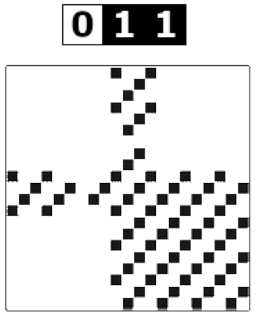


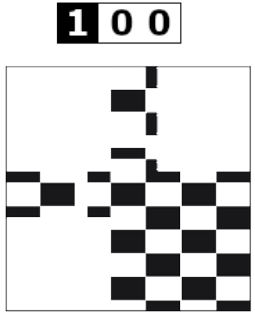
Masques

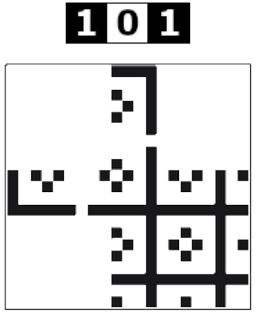


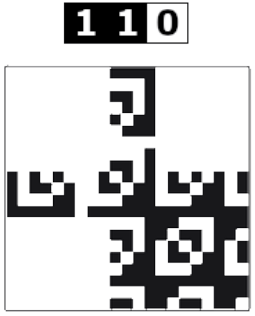


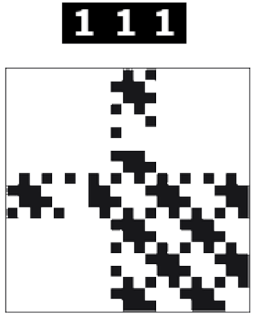


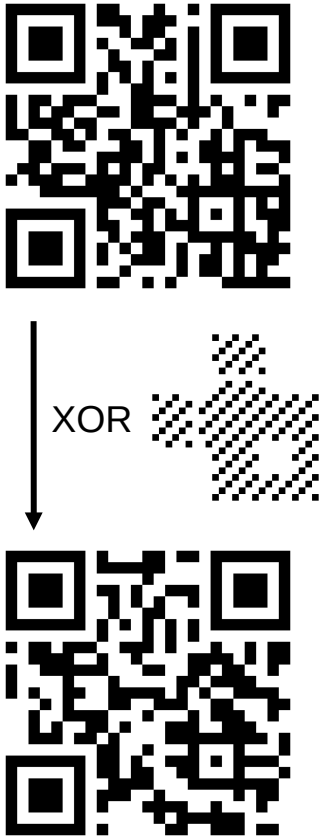




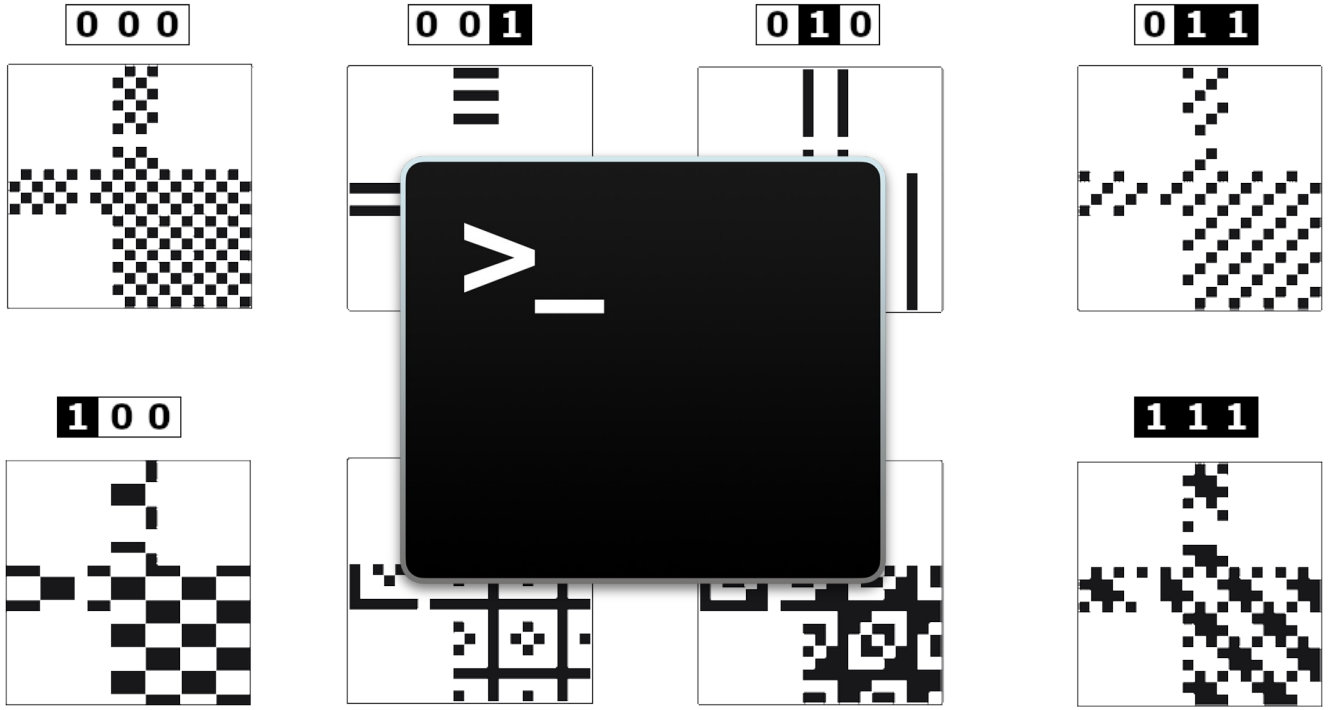


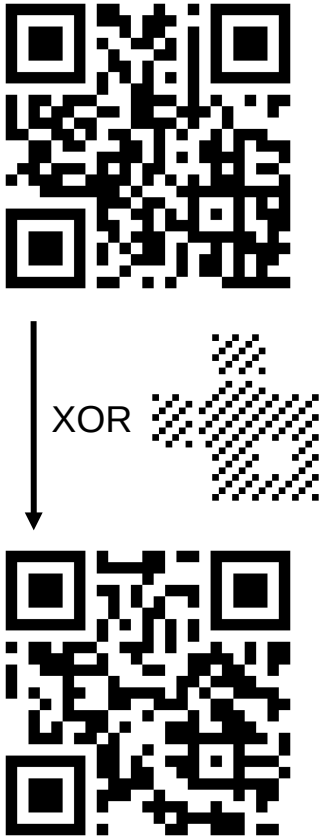


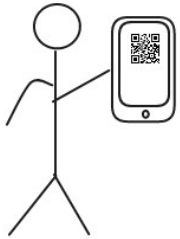


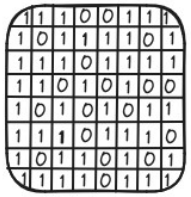


Masques









0110010101110001101001001

Détection

Traitement de l'image

- Localisation
- Redressage
- Amélioration
- Lecture des points

Extraction

Structure d'un QR code

- Métadonnées
- Masque
- Eléments de structure

Décodage

décodage du message

Extraction : éléments de structure

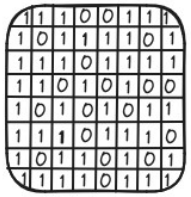
Objectif : Éliminer les cases qui ne portent pas d’information

- Elimination des patterns d’alignement
- Elimination des éléments de format

Extraction : éléments de structure

Objectif : Éliminer les cases qui ne portent pas d’information

- Elimination des patterns d’alignement
- Elimination des éléments de format



0110010101110001101001001

Détection

Traitement de l'image

- Localisation
- Redressage
- Amélioration
- Lecture des points

Extraction

Structure d'un QR code

- Métadonnées
- Masque
- Eléments de structure
- Lecture de bits

Décodage

décodage du message

Extraction : lecture des bits

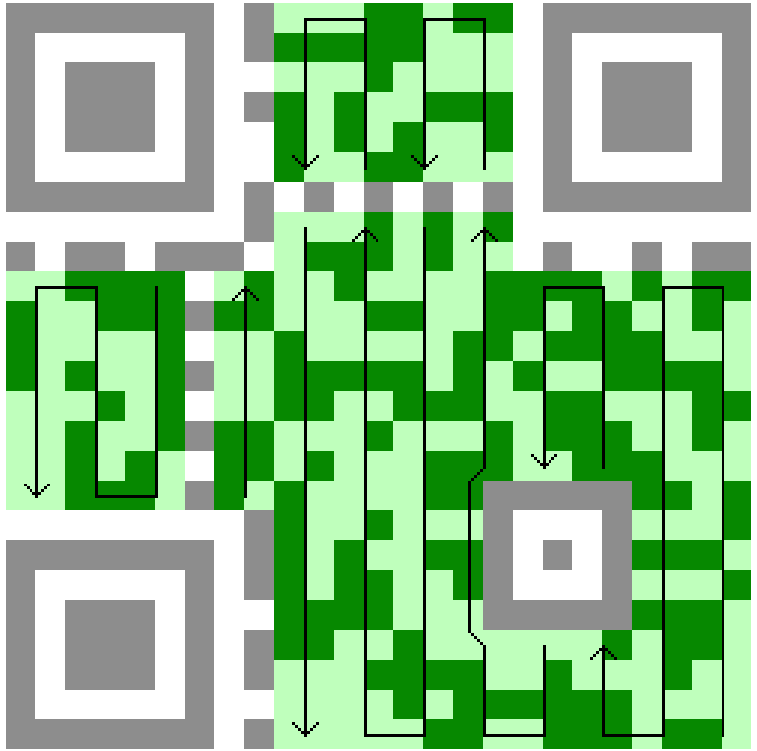
Objectif : Lire les bits dans le bon ordre

- 359 bits (44 octets) de données

Extraction : lecture des bits

Objectif : Lire les bits dans le bon ordre

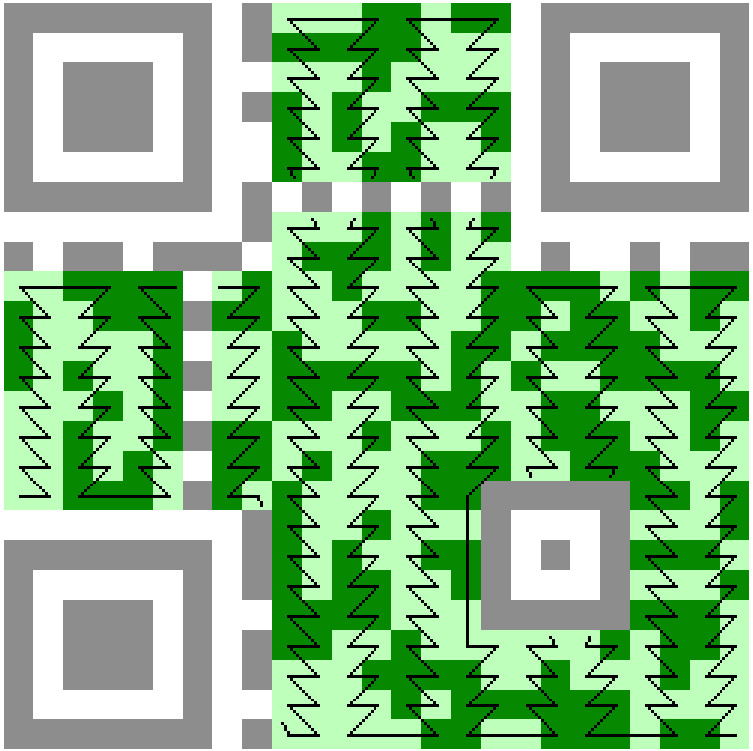
- 359 bits (44 octets) de données



Extraction : lecture des bits

Objectif : Lire les bits dans le bon ordre

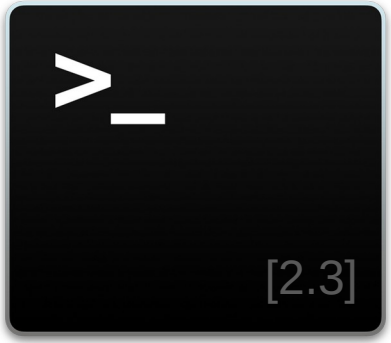
- 359 bits (44 octets) de données

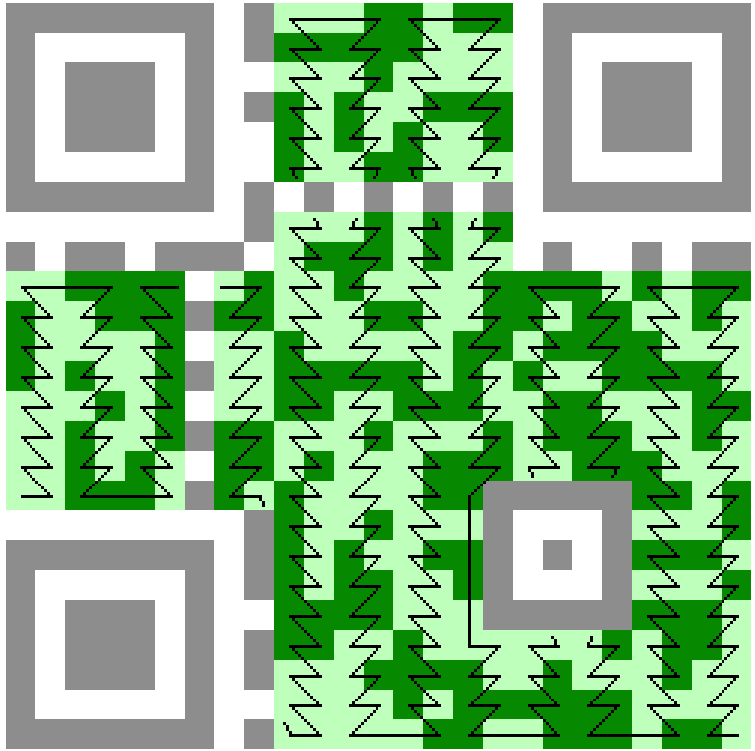


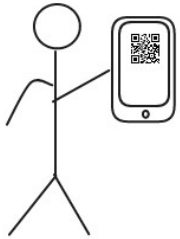
Extraction : lecture des bits

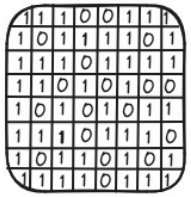
Objectif : Lire les bits dans le bon ordre

- 359 bits (44 octets) de données









0110010101110001101001001

Détection

Traitement de l'image

- Localisation
- Redressage
- Amélioration
- Lecture des points

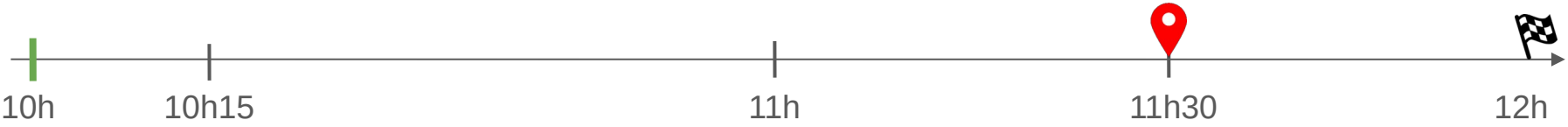
Extraction

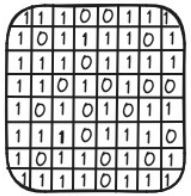
Structure d'un QR code

- Métadonnées
- Masque
- Eléments de structure
- Lecture de bits

Décodage

décodage du message





0110010101110001101001001

Détection

Traitement de l'image

- Localisation
- Redressage
- Amélioration
- Lecture des points

Extraction

Structure d'un QR code

- Métadonnées
- Masque
- Eléments de structure
- Lecture de bits

Décodage

décodage du message

- Correction d'erreur

Décodage : correction d’erreur (digression)

Première approche : le bit de signe

Exemple : 1001101 + 0 : 10011010

Réception : 10011110

Décodage : correction d’erreur (digression)

Première approche : le bit de signe

Exemple : 1001101 + 0 : 10011010

Réception : 10011110

Problème : on ne sait pas quel bit a changé
Même problème pour les cryptogrammes

Reste utile si on sait quel bit on a perdu (RAID5)

Décodage : correction d’erreur (digression)

Essayons de transmettre 1 bit avec 1 bit d’erreur possible

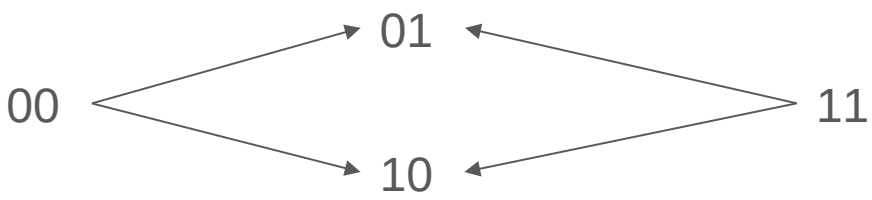
Impossible si on envoie un seul bit

Décodage : correction d’erreur (digression)

Essayons de transmettre 1 bit avec 1 bit d’erreur possible

Impossible si on envoie un seul bit

Avec deux bits :

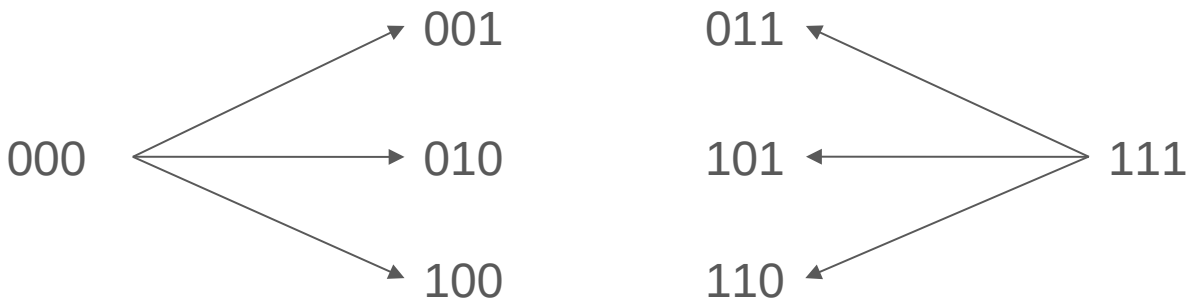


=> Ambiguïté

Décodage : correction d’erreur (digression)

Essayons de transmettre 1 bit avec 1 bit d’erreur possible

Essayons avec trois bits :



Transmettre 1 bit avec 2 erreurs possibles => 5 bits

Décodage : correction d’erreur (digression)

Transmettre 2 bits avec 1 erreur possible

xxxxxxxxxxxxx => 2^n messages possibles
n+1 mutations possibles

On ne peut pas transmettre plus de 2^n / (n+1) messages différents

3 bits => 2 messages = 1 bit
4 bits => 3,2 messages
5 bits => 5.33 messages > 2 bits
6 bits => 9.14 messages > 3 bits
7 bits => 16 messages = 4 bits

Décodage : correction d’erreur (digression)

Transmettre 2 bits avec 1 erreur possible (5 bits total)

00000	00001	00101	00100
00010	00011	00111	00110
01010	01011	01111	01110
01000	01001	01101	01100

10000	10001	10101	10100
10010	10011	10111	10110
11010	11011	11111	11110
11101	11001	11101	11100

Décodage : correction d’erreur (digression)

Transmettre 2 bits avec 1 erreur possible (5 bits total)

00000	00001	00101	00100
00010	00011	00111	00110
01010	01011	01111	01110
01000	01001	01101	01100

10000	10001	10101	10100
10010	10011	10111	10110
11010	11011	11111	11110
11101	11001	11101	11100

00000 : 00

Décodage : correction d’erreur (digression)

Transmettre 2 bits avec 1 erreur possible (5 bits total)

00000	00001	00101	00100
00010	00011	00111	00110
01010	01011	01111	01110
01000	01001	01101	01100

10000	10001	10101	10100
10010	10011	10111	10110
11010	11011	11111	11110
11101	11001	11101	11100

00000 : 00
00111 : 01

Décodage : correction d’erreur (digression)

Transmettre 2 bits avec 1 erreur possible (5 bits total)

00000	00001	00101	00100
00010	00011	00111	00110
01010	01011	01111	01110
01000	01001	01101	01100

10000	10001	10101	10100
10010	10011	10111	10110
11010	11011	11111	11110
11101	11001	11101	11100

00000 : 00
00111 : 01

11011 : 10

Décodage : correction d’erreur (digression)

Transmettre 2 bits avec 1 erreur possible (5 bits total)

00000	00001	00101	00100
00010	00011	00111	00110
01010	01011	01111	01110
01000	01001	01101	01100

10000	10001	10101	10100
10010	10011	10111	10110
11010	11011	11111	11110
11101	11001	11101	11100

00000 : 00
00111 : 01

11011 : 10
11100 : 11

Décodage : correction d’erreur (digression)

Transmettre 2 bits avec 1 erreur possible (5 bits total)

00000	00001	00101	00100
00010	00011	00111	00110
01010	01011	01111	01110
01000	01001	01101	01100

10000	10001	10101	10100
10010	10011	10111	10110
11010	11011	11111	11110
11000	11001	11101	11100

00000 : 00
01011 : 01

10111 : 10
11100 : 11

Décodage : correction d’erreur (digression)

Trois paramètres :

- Nombre de bits à transmettre
- Nombre d’erreurs autorisées
- Taille totale du message

On a vu :

- 1 / 1 / 3
- 2 / 1 / 5
- 4 / 2 / 7

Décodage : correction d’erreur

Objectif : Corriger les erreurs

- Algorithme de **Reed-Solomon**, propriétés (stabilité du message)
- Tout le message est décodé d’un seul bloc (entrelacement)

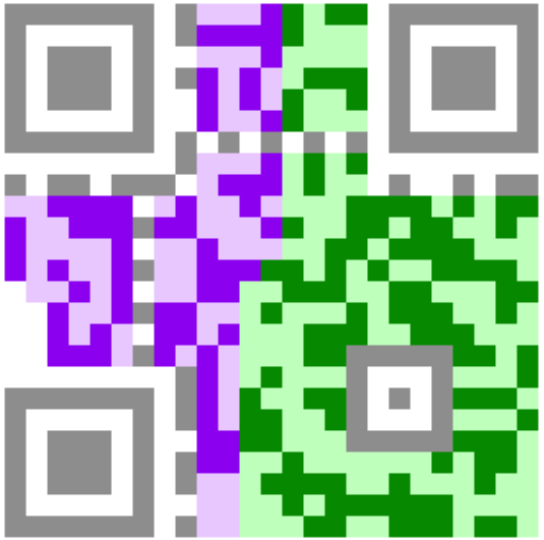
Décodage : correction d’erreur

Objectif : Corriger les erreurs

- Algorithme de **Reed-Solomon**, propriétés (stabilité du message)
- Tout le message est décodé d’un seul bloc (entrelacement)

- **L** (`01`) : 7% **Q** (`11`) : 25%
  M (`00`) : 14% **H** (`10`) : 30%

  Exemple niveau **M**, version **2** :
  44 octets au total = 28 de message + 16 ECC
  => permet de corriger 30 bits



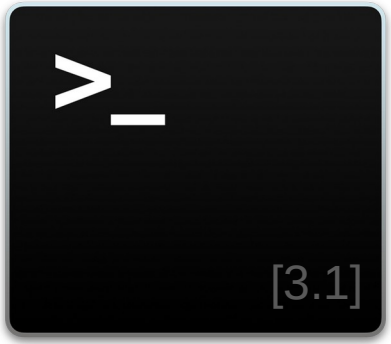
Reed-Solomon : étape de correction

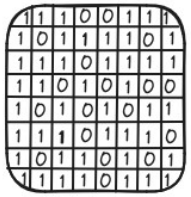
https://en.m.wikiversity.org/wiki/Reed%E2%80%93Solomon_codes_for_coders

- Calcul des “syndromes”
- Détermination du nombre d’erreurs
- Emplacement des erreurs
- Valeur des erreurs
- Correction du message

Décodage : correction d’erreur

Objectif : Corriger les erreurs





0110010101110001101001001

Détection

Traitement de l'image

- Localisation
- Redressage
- Amélioration
- Lecture des points

Extraction

Structure d'un QR code

- Métadonnées
- Masque
- Eléments de structure
- Lecture de bits

Décodage

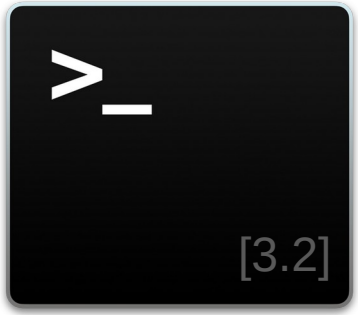
décodage du message

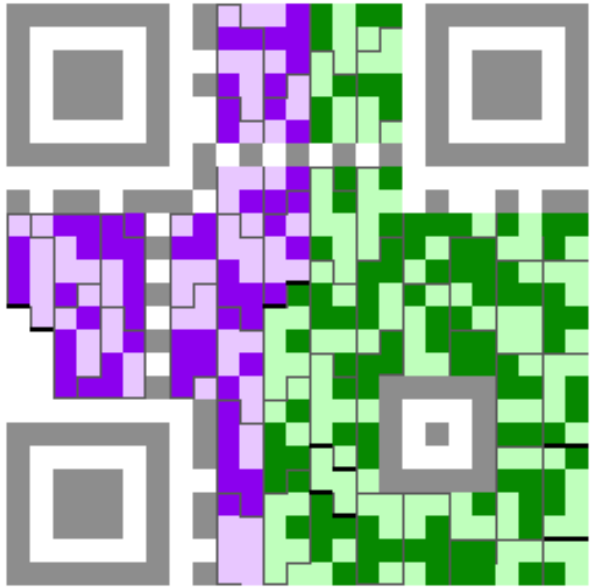
- Correction d'erreur
- En-tête

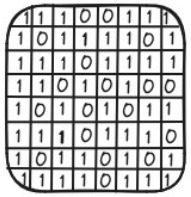
Décodage : en-tête et encodage

Objectif : Déterminer le format des données

- 4 premiers bits : mode
  - Numérique, Alphanumérique, Octets, Kanji
- 8 à 16 bits suivants : longueur du message







0110010101110001101001001

Détection

Traitement de l'image

- Localisation
- Redressage
- Amélioration
- Lecture des points

Extraction

Structure d'un QR code

- Métadonnées
- Masque
- Eléments de structure
- Lecture de bits

Décodage

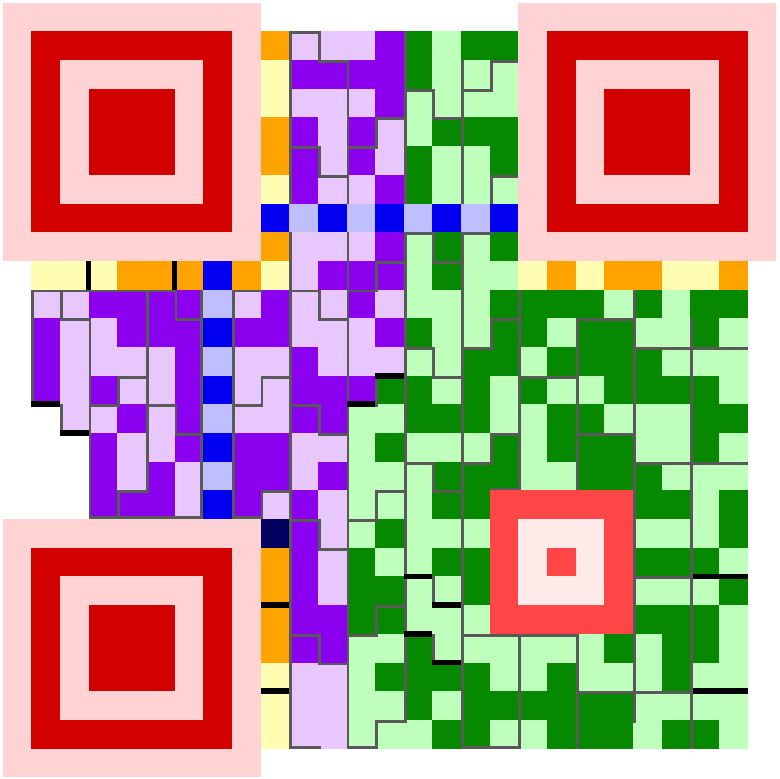
décodage du message

- Correction d'erreur
- En-tête
- Lecture du message

Décodage : lecture du message

Objectif : Récupérer le message

Récapitulatif



Merci !

That's all Folks !

Open feedbacks

Code & slides & références

https://github.com/benoitmasson/qrcode-demo