Dependable Coding of Fiducial Tags

Andrew Rice
Lab For Communication Engineering
University of Cambridge

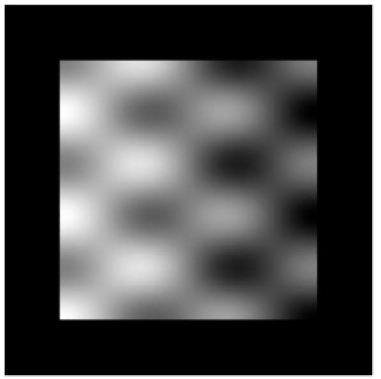
Introduction

- Many properties of a machine vision system are determined by the coding scheme for identifying objects
  - reliability, address space, performance
- Current schemes do not offer quantifiable error protection
- Propose and evaluate new coding schemes with analysable characteristics

ARToolKit

Template Tags

- Print an image (template) on the tag
- When sighted
  - perform perspective correction
  - compare against all issued images
  - return a close match



✔ Users can easily distinguish between tags

✗ Must guarantee a large separation between templates

✗ What happens when there is noise in the image?

Symbolic Tags

- Store data on the tag by dividing payload area into data cells

✔ Quantifiable address space

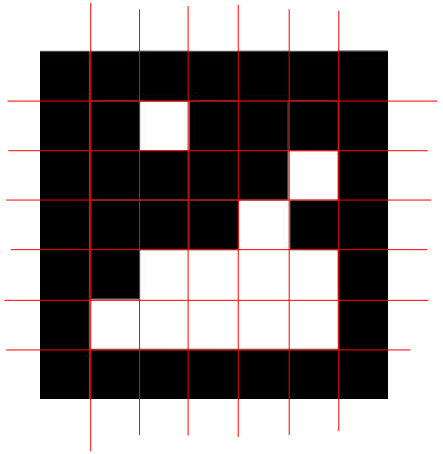
✔ Image noise translates to bit errors

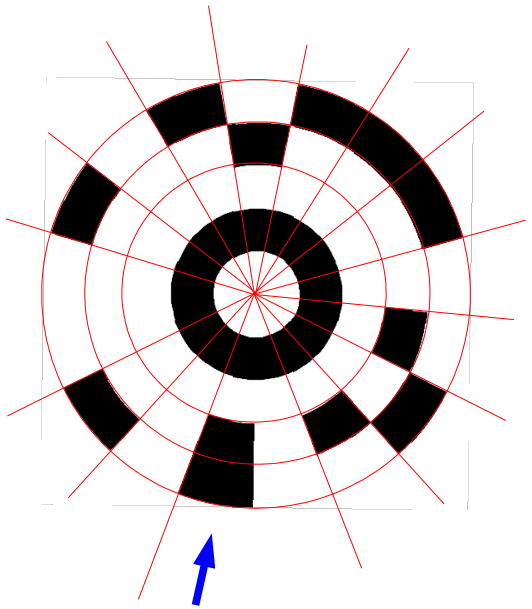
  - we can model this

✗ No longer possible for people to “decode” the tag by eye

Examples

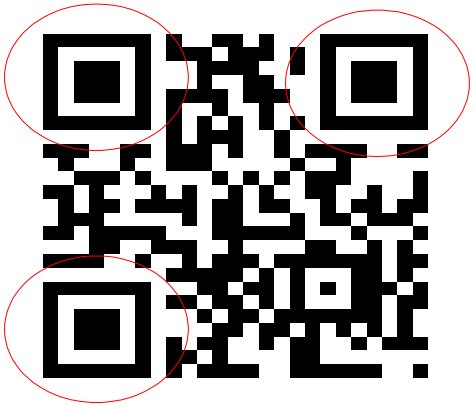
- Matrix - Square tag with 5x5 payload containing a CRC
- ✗ Rotated tag gives a permutation of the original code – breaks CRC





- TRIP - Circular tag with 2 rings x 17 sectors.
- ✗ 2 bit errors could confuse us into choosing the wrong synchronization sector

Asymmetric Tags



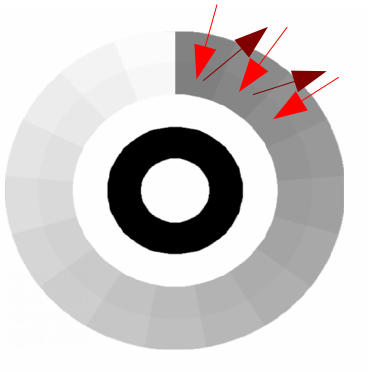
- QR-Code
- Uses 3 corner markers to orient the code

✔ Code remains robust when the tag is rotated

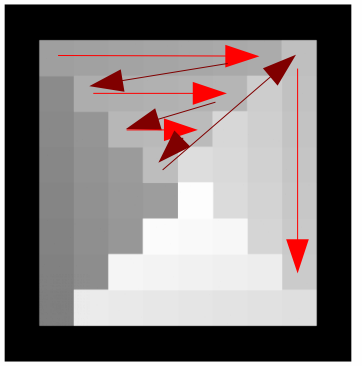
  - assuming that the features can be detected

✗ How do we know that we have made the features the optimum size?

  - too big implies wasted payload space
  - too small implies less resistant to noise than coding scheme



Rotational Invariance



- Structure data on the tag so that a rotation of the tag is a rotation of the codeword
- Symbol Size – number of bits to store each symbol.
- Payload Size – number of symbols per tag
- Every rotation of a codeword in a cyclic code is also a valid code word

✔ We can be (mathematically) sure that rotating the tag does not reduce the Hamming distance of our code

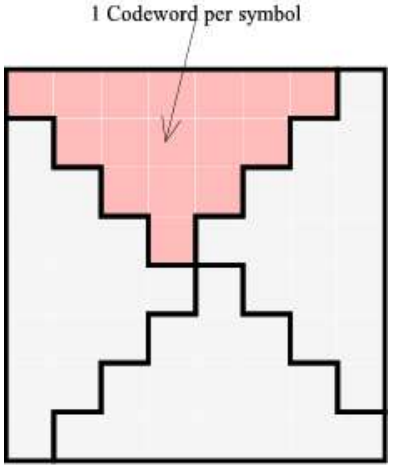
Simple Parity Code

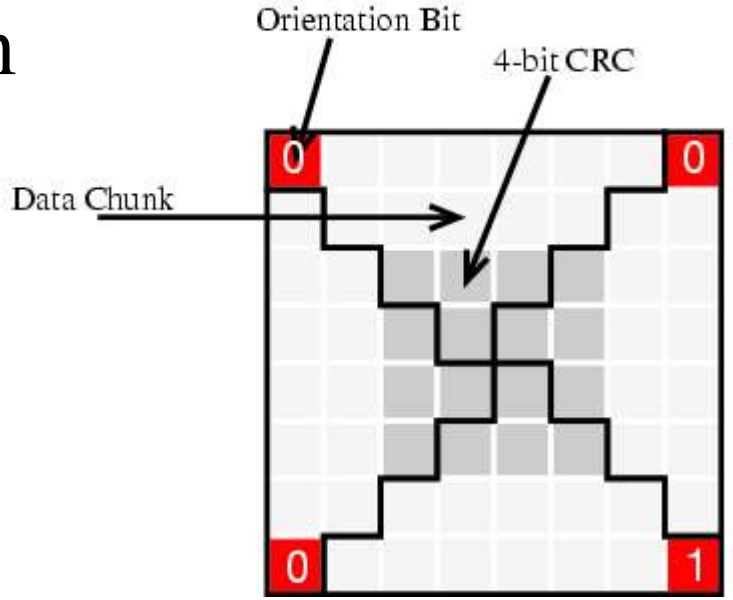
- Store a sequence of bits with a parity bit appended
  - Hamming distance is only 2 bits
  - This is as good as the TRIP coding scheme!
- Rotating the code to the smallest value is a deterministic way to produce a unique reading

✗ Address space is non-contiguous

Independent Chunk Code

- If the symbol size is large then we can consider each symbol to be a separate codeword
- We can store arbitrary data if we include orientation information
  - protected within codeword





Structured Cyclic Code

- A cyclic code with additional structure to record the rotation
  - simple example: 0123456789
  - whatever rotation of this codeword we have we can recover the original codeword if we rotate by the number of places given by the first digit
- We can do this and spread the rotation information over the entire code
  - it will be equally error resistant as the data itself
  - we can store arbitrary data

SCC Parameters

- Storing the rotational information reduces the data payload by one symbol
  - requires less “space” than TRIP's synchronization sector
- On a 31 sector, 5 ring tag we can use a Reed-Solomon based cyclic code
  - 11 symbols Hamming Distance
  - Detect an error even if 30% of symbols are wrong
  - Can store 99 bits out of 155

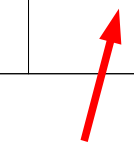
SCC Parameters(2)

- SCC stores 99 bits out of 155
  - 64% utilisation
- QR-Code with level 'M' error correction can correct approximately 15% of symbols
  - comparable protection to this instance of SCC code
  - only gets 59.5% utilization
- Cannot engineer a perfect tradeoff between asymmetric feature size and payload space

Results

- Error resistance
  - 1000 samples
  - Artificial noise added uniformly across the image

	Message Size	Hamming Distance	Correct Read (%)	Failed Read (%)	False Read (%)
TRIP	139	2 symbols	24	74	2
SPC	154	2 bits	31	46	23
ICC	93	2 bits per symbol	27	73	0.3
SCC-1	141	3 symbols	56	6	38
SCC-2	101	11 symbols	94	6	<0.1



No data

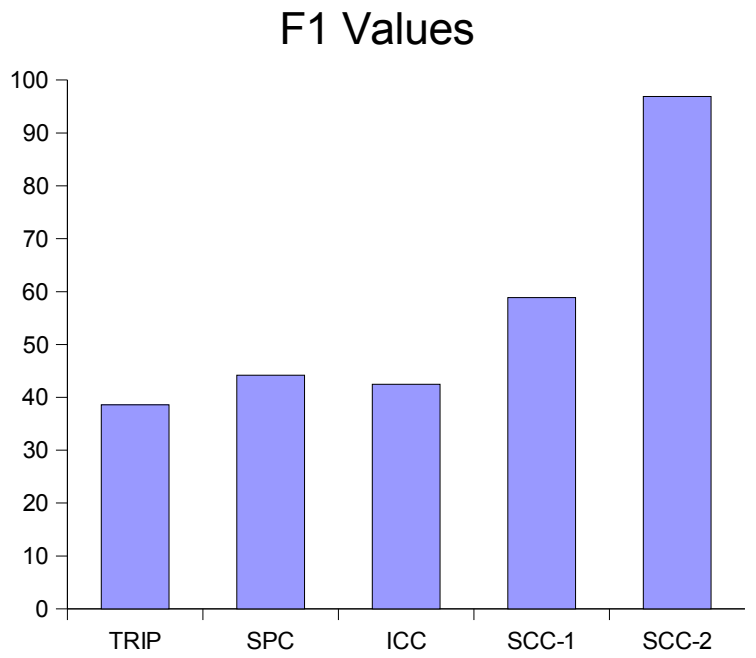
Invalid data

Comparison

- Computed F_1 value for each code

$$F_1 = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}$$

$$= \frac{2 \times \text{correct read} \times (100\% - \text{false read})}{\text{correct read} + 100\% - \text{false read}}$$



Conclusion

- Situations exist where tags might appear in any orientation
- Current coding schemes on symmetric tags are not robust to this
- Presented a new coding scheme that is quantifiably robust under these conditions
- Aysmmetric features cannot offer optimal performance