Fractal Codes: 自己相似的に配置される二次元コード

Fractal Codes: a Self-Similar Layout for 2D Code Complex

綾塚 祐二[*]

概要. カメラを備えたモバイルデバイスが広く使われるようになり，印刷された二次元コードを読み取らせるようなアプリケーションも普及している．既存の二次元コードの制約の一つは，カメラの画角や解像度，画質などで決まる，読み取り可能な距離の範囲の狭さである．この制約を軽減するために，系統立てられた ID を割り振られた複数のコードが，自己相似性を持つレイアウトで配置される複合コード，フラクタルコードを提案する．大きなコードの内側に小さなコードが再帰的に敷き詰められ，カメラをコードに近づけても，どれかのコードが認識できる．複合コード内での各コードの相対的な位置は，各コードの ID から引き出せるようになっており，どのコードからでも同じ座標系を取得することができる．

1 はじめに

カメラデバイスが安価になり，また小型のプロセッサの処理能力も向上してきたことを受け，二次元コードを用いたアプリケーションは広く使われるようになってきている．既存のアプリケーションとしてはカメラ付き携帯電話で QR コードを認識してウェブにアクセスするというものが多いが，拡張現実感 (Augmented Reality, AR) を実現し，二次元コードを印刷したカード上にモンスターの CG (Computer Graphics) を表示するようなゲームも市場に登場しつつある [5]．拡張現実感を実現するためには他にジャイロや磁気トラッカーなどを用いる方法もある．しかし一般ユーザ向けの携帯電話やゲーム機には高価，もしくは特殊なデバイスは搭載しづらいので，一般的なカメラデバイスのみで実現可能な二次元コードを用いる方法は，これらの用途に適している．

既存の二次元コードには弱点もいくつかある．その一つが，認識可能な距離の範囲 (レンジ) である．コードがカメラの画角からはみ出してしまうような近距離からでは当然認識ができないし，コードが小さくつぶれてしまう距離まで遠ざかっても認識ができない．そのため，ゲームが行われる CG のフィールド全体を俯瞰するような位置からその上の個々の CG キャラクタに寄っていってその足許までを連続的に眺めるようなことはできない．コードを認識した後に，オプティカルフローなどを用いてカメラの動きを追うという手法も考えられるが，いきなりコードがカメラの画角からはみ出すほどの近くを認識させることはできないし，画像処理のコストも高い．ジャイロなどを使ってカメラの動きを追うとしても，いきなり近いところが認識できないという問題は同じであるし，カメラ以外のデバイスを必要とする，すなわちコストが上昇するという欠点がある．

我々は，この弱点のない，広いレンジで座標も含めて認識可能な二次元コードとして，体系づけられた ID を持つ複数の二次元コードを自己相似性のあるレイアウトで配置した複合的な二次元コード,「フラクタルコード」(Fractal Codes) を提案する．既存の二次元コードのデザインを工夫することで，自己相似性を持つレイアウトが可能で，一つのコードの内側に小さなコードを敷き詰めることができる．また，体系づけられた ID により，ID から他のコードに対する相対的な位置を計算し，敷き詰められたどのコードからでも同一の座標系を取得することが可能である．これにより，遠くからはより大きなコードが，近くではその内側に敷き詰められたコードのどれかが認識され，距離を変えても途切れずに認識ができるようになる．個々のコードを認識するアルゴリズムは既存の二次元コードと変わらず，計算コストには影響を与えずに広いレンジで認識可能な二次元コードが実現できる．

以下の節では，フラクタルコードのレイアウトについて説明し，続いて ID 体系について説明する．その後，実装例を紹介し，現状での問題点や考察などを述べる．

2 自己相似性を持つ二次元コード

フラクタルコードを設計するにあたり，まずは，コード情報と同時にコードの座標や向きも認識可能な二次元コードである CyberCode[4] をベースとして，コードの内側に小さなコードを配置することを考える．CyberCode はコードの位置を特定するための，バーと四隅の点からなるガイド部分と，その中の相対的な座標で規定されるデータ部分から成る (図 1)．データ部分の配置は自由であり，オリジナルの CyberCode のように密でなくともよいので，配置を工夫し余白を持たせて，その余白に小さなコー

[*] Yuji AYATSUKA (株) ソニーコンピュータサイエンス研究所 インタラクションラボラトリー

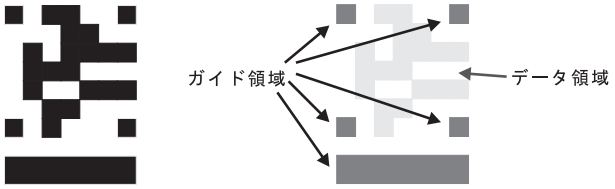


図 1. CyberCode

ドを埋め込むという方針を採る．

以後の説明のために，まずは，いくつかの用語を定義しておく．自己相似性をもつレイアウトで配置されたひとまとまりの二次元コード群を「複合コード (code complex)」と呼ぶ．ある複合コードの中の，もっとも大きな二次元コードを「第一層のコード」と呼ぶ．その次の大きさのものを「第二層のコード」，さらにその次の大きさのものを「第三層のコード」というように呼ぶ．第二層のコードに対し，第一層のコードを「ひとつ上の層のコード」，第三層のコードを「ひとつ下の層のコード」と呼ぶ．ある層とひとつ下の層との間でコードの大きさの比 (ひとつ下の層のコードの一辺を，ある層のそれで割った値) を層間倍率と呼ぶ．

2.1 レイアウトの制約条件

複合コードのレイアウトのデザインには，いくつかの制約がある．その中でも重要な二つの制約は，カメラの動きに対して途切れずにコードが認識できるようにするための制約であり，複合コードに近づく・離れる動き (これを前後の動きと呼ぶ) に対応するための制約と，複合コードに平行な動き (これを横方向の動きと呼ぶ) に対応するための制約である．

複合コードの中のあるコードをカメラで認識している状態から，コードに近づいていってひとつ下の層のコードを認識できるようになる，あるいは離れていってひとつ上の層のコードを認識できるようになるまで現在のコードが認識し続けられなければ，前後の動きに対して途切れずに認識することはできない．層間倍率が小さすぎる，すなわちコードの大きさのギャップが大きすぎると，これは達成できない．許容される層間倍率はカメラの性能や取り込む解像度などによる．市販のウェブカメラで VGA サイズ (640×480) の画像を取り込み認識させる場合，我々のテストした CyberCode のバリエーションを用いると，およそ 1/3 以上の層間倍率であれば安定して途切れずに認識が可能であることが判った．よって，層間倍率を 1/3 以上に保つことが第一の制約である．

ある層のコードを認識している状態でカメラを横方向に動かして途切れずに認識が続くようにするためには，ある層のコードは，ひとつ上の層のコードの中 (あるいはコードとコードの狭間) にできるだけ

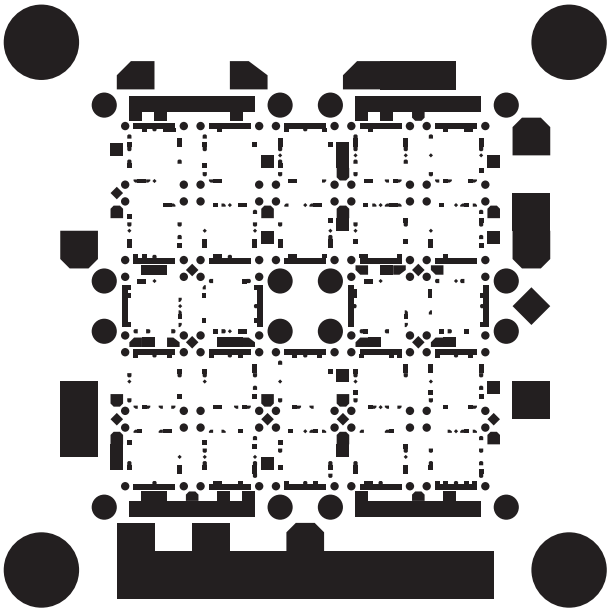


図 2. フラクタルコード (第三層まで)

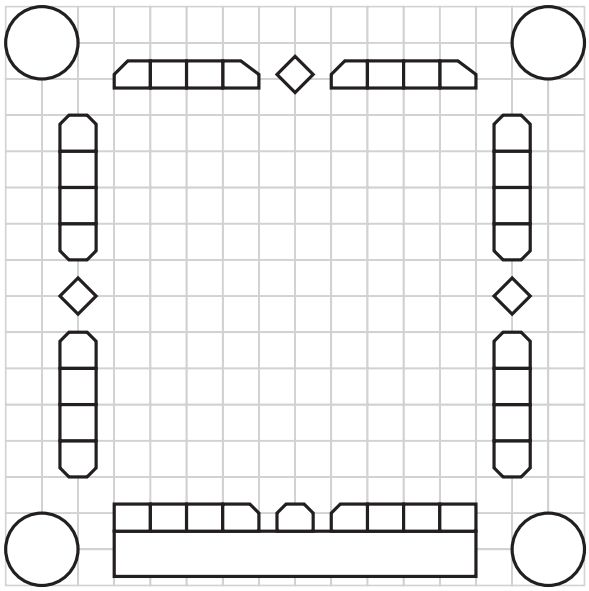
稠密に配置される必要がある．CyberCode のように有限の大きさの点をベースにしたコードであるかぎり，層をいくつも重ねれば，コードの大きさに対して相対的に大きなギャップが何カ所かに現れるざるを得ない．たとえば，下のほうの層では上のほうのコードのガイド部分やデータ部分自体がギャップになる．それを最小限にするのが，第二の制約である．

これらに加え，以下のような制約ももちろん課せられる．まず，データ部分は充分なビット数 (少なくとも数層目まで敷き詰められたひとつの複合コードに含まれるコード数に対応した分) を持たせなければならない．また，単体のコードとして，なるべく認識しやすくなければならない (ガイド部分，データ部分ともにあまり小さくすることはできない)．

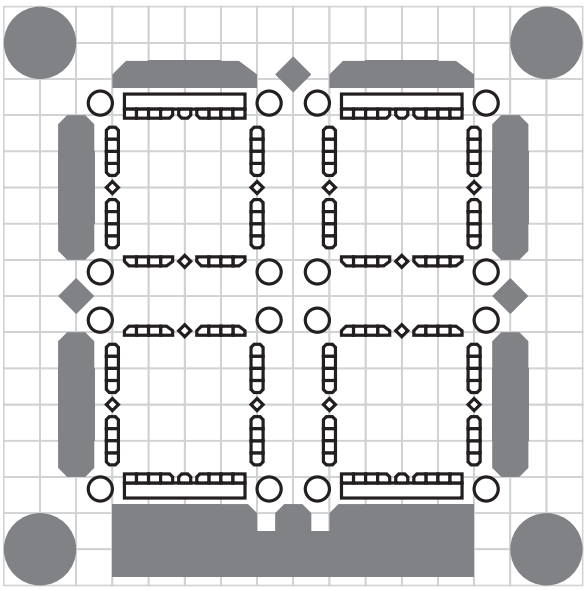
2.2 制約条件を満たすレイアウト

上述の条件を満たすものとして，試行錯誤の末に我々がデザインしたのが，図 2 のようなコード (複合コード) である．単体のコードとしては図 3(a) のような構造になっており，ガイド部分の四隅を繋ぐ線のやや内側にデータ部分が並んでいる．ガイドバーに近い部分ではデータ部分とガイドバーが連続するようになっている (このようになっていても，ガイドバーは問題なく認識できる)．36 個のデータ部分のセルを持ち，24 ビットのデータ (ID) をエンコードするようになっている．

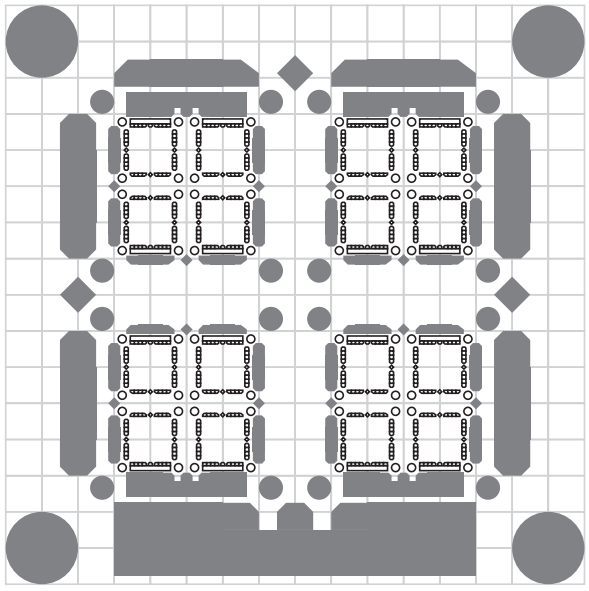
図 3(a) のコードを第一層のコードとすると，第二層のコードは図 3(b) のように配置される．層間倍率は 1/3 である．第三層のコードは，まず，自己相似性を持つので第二層のコードの内側に，第一層に対する第二層と同じように配置される (図 3(c))．



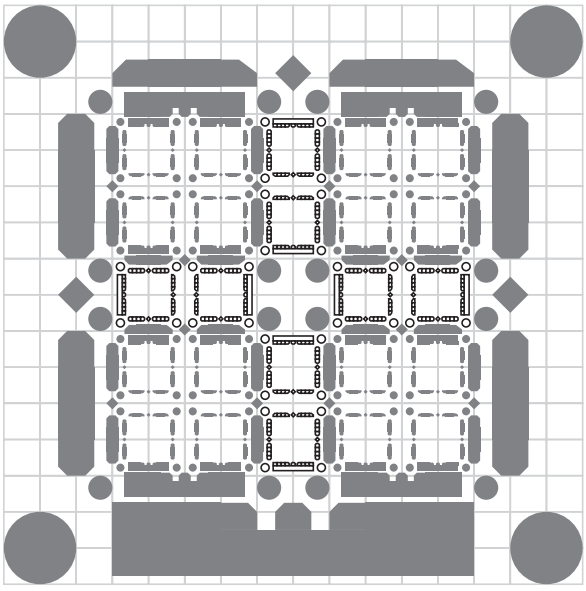
(a) 単体のコードの構造



(b) 第二層のコードの配置



(c) 第三層のコードの配置(1)



(d) 第三層のコードの配置(2)

図 3. フラクタルコードのレイアウト

加えて，第二層のコードとコードの間にも配置する (図 3(d))．これにより，第一層のコードの中央部と外縁部以外にはほぼ均一に第三層のコードが敷き詰められる．図は省略するが，第四層には，このプロセスの繰り返しに加え，中央部のスペースにも十字型に 5 つ，コードを配置する．以下，同様の手順の繰り返しにより，無限の階層のコードをひとつの複合コードの中にレイアウトすることができる[1]．

このレイアウトでは，1 つの第一層のコードの内側に，第二層には 4 つ，第三層には 24 個，第四層には 141 個のコードが配置される．第五層では 808 個ものコードが配置されることになる．第五層のコードは大きさが第一層のコードの $1/3^4 = 1/81$ であり，複合コード全体を A4 サイズの用紙に印刷したとしても，コード内の点やバーがつぶれてしまう可能性が高い．実用的に必要，あるいは充分な層の数については後の節で議論する．

2.3 ID の割り付け

複合コード内のコードの ID の割り付けに関する基本的なアイディアは，第 n 層での上の層に対する相対的なコード位置を，n 番目の桁の数字 (より一般的には n に一対一対応する桁の数字) で表す，ということである．たとえば，五層からなる複合コードのためには，5 桁の数字が必要となる[2]．基数は，すべてのコードに不足なく割り当てられるものであれば 8 や 16 などであっても構わないが，以下，10 を用いるとして説明する．

第一層に一の位，第二層に十の位，と割り当てたとすると，第一層のコードの ID をたとえば 1 とし，第二層は図 3(b) の左上から順に 1, 2, 3, 4 の数字を対応させ，ID はそれぞれ 11, 21, 31, 41 となる．第二層のコードの狭間に置かれる第三層のコードには 101, 201 などの ID を割り当てる．第二層のコードの内側の第三層のコードは，111, 211 などである．層の数が予め規定されている場合は，この順序を逆にして，第一層を一万の位に，第二層を千の位に，と割り当てることもできる．この場合には上に出てきた例の ID はそれぞれ 10000 (第一層)，11000, 12000, 13000, 14000 (第二層)，10100, 10200, 11100, 11200 (第三層) となる．どちらの方法でも，コードの ID のビット数は有限であるので，実現できる層の数には限界がある．

この割り当て方により，認識したコードのひとつ上の層のコードに対する位置，そして複合コード全体に対する位置が ID から再帰的に計算できるようになる．たとえば図 3(b) の左上の第二層のコードであれば，座標系を 180 度回転し，3 倍して，ガイド部の四隅の点の作る正方形の一辺を 1 として (3/7, 3/7) だけずらせば，第一層のコードの座標系と重なる．それゆえ，複合コード中のどのコードが認識されても，複合コード全体の座標系が認識可能で，そこに合わせた CG などを表示することができる．

ID はもちろん，複合コード自体を識別するのにも使う．上の例でも第一層のコードが 10 種類識別できるが，上の方法で用いる桁以外の桁を，そのために割り当てて，ID のビット数の許す限りの桁数分の複合コードが識別できる．複合コードの数と，各複合コードの階層数がトレードオフになる．

3 フラクタルコードの実装

この節では，フラクタルコードの実装と，試作したアプリケーションについて説明し，それらから得られた知見について議論する．

3.1 実装と試作アプリケーション

フラクタルコードの実装のうち，コード認識部分は，ほとんどがベースとなった CyberCode のものと同じであり，いくつかのパラメータが異なるのみである．よって，認識速度も CyberCode とほぼ同じである．CyberCode と同様に，ID とその座標が得られた後，2.3 節で説明した ID の解析と座標変換を行い，三次元 CG などを表示する．

図 4 は試作したアプリケーションの実行例である．Pentium 4 を搭載した PC 上で，640×480 ピクセルの画像を取り込める USB カメラを用い実行している．左から右へ，カメラをだんだんとフラクタルコードの印刷された紙へと近づけていっている．上段の写真では，白い枠で認識している ID の位置を示している．左の二枚では第一層のコードが認識されているが，三枚目では第二層のコードが，四枚目では第三層のコードが認識されているのが判る．下段の写真は，上段のものと同じ位置で，三次元 CG を表示させたものである．最初は丸い外形しか判らない中世ヨーロッパ風の要塞だが，近づいていくと砲門の跳ね扉の鎖がはっきりと判るようなところまで寄ることができる．

3.2 議論

この試作アプリケーションを用い，性能評価や，問題点の洗いだしなどを行った．以下，それらについて議論する．

認識のレンジ まず，A4 サイズの紙に印刷した三層の複合コードで実験した．第一層のコードのサイズは 168mm，第三層で約 19mm である．照明の条件は日中の筆者らのオフィスの照明 (直射の入らない窓があり，蛍光灯とハロゲン電球がついている) である．このとき，安定して認識できる最短距離は 4cm 程度，最長距離は 170cm 程度で，約 42 倍のレンジ

[1] レイアウト上は可能であるが，実際には後述する ID の制約により無限階層のフラクタルコードは実現できない．

[2] 第一層のコードが 1 つで，複合コードを識別する必要が無い場合には，一桁少なくともよい．

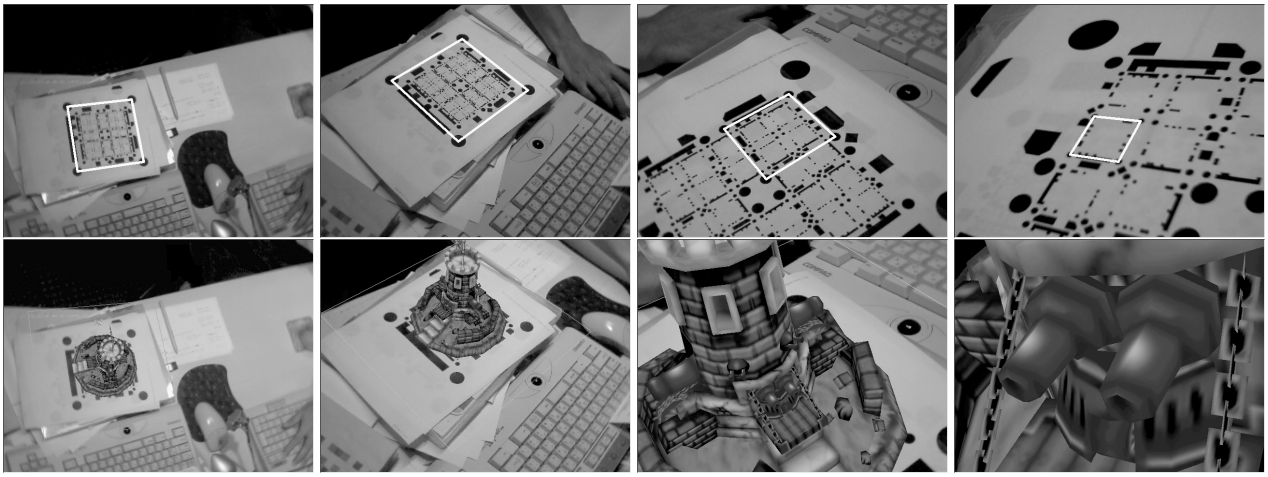


図 4. アプリケーション実行例: (上段) 認識しているコードを白枠で表示，(下段) コードに三次元 CG を重ね合わせる

があった．次に A3 サイズの紙に印刷した四層の複合コード (第一層のコードのサイズは 254mm，第四層で約 9mm) で実験した．最短距離は 2cm 程度，最長距離は 250cm 程度で，約 125 倍のレンジがあった．もちろん，この中間の距離でもすべて認識が可能であった (ただし，カメラを動かすことで画像がぶれたりして認識が途切れることはある).

旧来の CyberCode を用い，同じ環境で実験を行うと，22mm×30mm のコードで 10cm 程度の距離から 60cm 程度の距離まで認識ができ，約 6 倍のレンジである．フラクタルコードを用いると，レンジが大きく改善していることが確認できる．一方，コードのサイズと最長距離の比を見ると，CyberCode が約 20 倍 (30mm に対し 60cm) なのに対し，フラクタルコードでは約 10 倍 (168mm に対し 170cm) に留まっている．これは，フラクタルコードの場合，単体のコードとしてはデータ部分のセルの大きさが CyberCode よりもかなり小さいこと，内部に他のコードがある分，撹乱されやすいことが原因である．

フォーカス これほどの広いレンジで認識が可能であると，オートフォーカス機能がないかぎりは，認識範囲外になるより先にフォーカスが合わなくなる．我々の使ったカメラでは手動でフォーカスが調整できたが，それもできないカメラではあまり多層にする意味がなくなってしまう．複合コードに含まれる黒い点は，CyberCode よりも細かく，数も桁違いに多くなることもあり，フォーカスはよりシビアな問題である．

層の数 複合コードには ID の制約の許す限りの数の層のコードを敷き詰めることができるが，あまり多くの層を持たせることは現実的ではない．前述のフォーカスの問題もあるが，コードの印刷の解像度の問題もある．A3 サイズの用紙に一般的なレーザープリンタで複合コードを印刷した場合，第五層のコードはつぶれてしまう．アプリケーションにも依ると思われるが，A4 程度のサイズのコードであれば三層，A0 サイズなど，巨大な複合コードを利用する場合でも五層あればほとんどの場合充分であろう．

相対的に大きな CG の表示 複合コード全体のサイズに合わせた CG を表示しようとすると，下の層のコードに対しては相対的に大きな CG を表示することになる．取り込まれた画像からコードの座標系を認識する際には，解像度や画質の制約などから必ず誤差が含まれる．コードと同じ程度のサイズの三次元 CG を表示する際にはさほど目立たない誤差でも，コードよりも数段大きな三次元 CG を表示すると，差は増幅される．これは，カメラとコードを相対的に静止させているときに特に目立つので，カルマンフィルタを用いるなど，得られる座標系の安定化の工夫がより重要になる．我々の実装では二段階，認識された二次元コードのカメラ画像内での座標と，計算の結果得られた複合コードの三次元座標系を表す行列に対してフィルタを掛けている．

レンズの収差の問題 フラクタルコードを用いている場合，複合コード上に同じ CG を表示し続けていても，認識しているコードは随時変わっているかもしれない．カメラとコードの相対位置が変化しているときにはもちろんそうであるが，静止しているときにもカメラの画角内に複数のコードが写り，ちょっとした揺らぎにより認識されるコードが変わりうる．理想的にはどのコードが認識されても同じ座標系が得られるはずだが，実際の環境では必ずしも同じにはならない．

前段と同じように，解像度や画質の制約による誤差もあるが，それよりも大きな問題となるのはカメラのレンズの歪曲収差である．たとえば我々の用

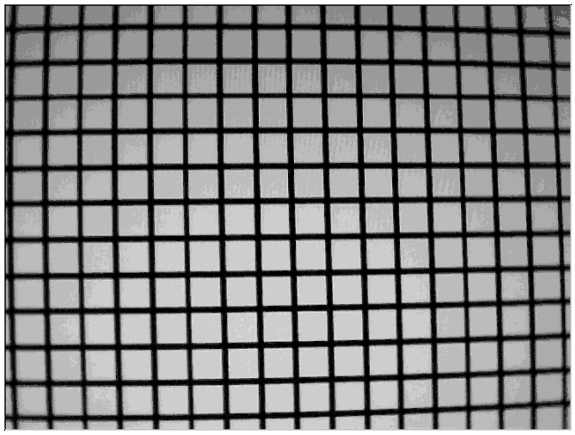


図 5. 使用したカメラの樽型収差

いたカメラのレンズには図 5 のような樽型の歪曲収差がある．これはすなわち，画像の周縁部では直線が直線として写らないということであり，かたや CyberCode やフラクタルコードでは射影変換に基づき座標を計算しているので，画面内全体として考えると必然的に誤差が出る．それぞれのコードの上にそれぞれ別の CG キャラクタを表示しているのであれば，まっすぐに並ぶはずのキャラクタが曲がって並んでもそれは背景の画像の歪曲に対応しているので，さほど違和感はない．しかし，ひとつの大きな CG を表示すると，それは認識されるコードが変わった瞬間に揺れとして認識されることになる．

この問題を取り除くためには，予め収差を計測しておき，取り込んだ画像に対しその補正を行った上で処理を行うか，歪曲収差のない (もしくは少ない) レンズを持つカメラを使う必要がある．前者は計算量的なコストが掛かり，後者はハードウェアのコストが掛かる．

4 関連研究

AnotoPen[1] は紙に小さなドットパターンを敷き詰めることにより，専用のペンがどの紙の上のどのどこをなぞっているかが判るシステムである．しかし，対応しているのは横方向の動きだけで，ペンが紙に接している場合にしか位置の認識はできない．

数本の平行な線に対し交わる直線が作る比 (cross-ratio) が射影変換で保存されることを利用したヴァーチャルスタジオシステムもある [3]．それぞれの場所で cross-ratio が独自になるようレイアウトしたグリッドパターンをブルーバックの背景のシートに敷き詰め，どこをどの角度で見ているかを認識することで，それを撮影しているカメラの位置を取得する．縦横のグリッドがそれぞれ 4 本以上ずつカメラに写っていればよいので，ある程度近寄ることはでき，また幅の 1 ピクセル以下の誤差が cross-ratio の計算に影響を与えない範囲では離れることもできるが，フラクタルコードほどのレンジの広さはない．

ブルーバックでグリッド部分を画像中から抜き出すことを前提としており，一般の環境下での使用は難しく，また画質の低いカメラでの利用には向いていない．cross-ratio を用いるという性質上，予め全 cross-ratio と場所の対応を記憶しておかねばならず，認識できる場所やシートの数を増やしていくと，精度が下がりやすくなる．一方，フラクタルコードでは，認識した ID を機械的に解析することで複合コード内の相対的な位置が判り，また精度を維持したまま 24 ビット分の複合コードと場所を識別できる．

5 まとめ

本稿では，体系づけられた ID を持つ複数の二次元コードを自己相似性のあるレイアウトで配置した複合的な二次元コード，フラクタルコード (Fractal Codes) を提案した．一つのコードの内側に小さなコードが敷き詰められており，離れた位置からでも，外側のコードがカメラの画角からはみ出してしまうような至近距離からでも，どれかのコードが認識できる．ID の一部は複合コード内での相対位置を表すようになっており，どのコードを認識しても，複合コード全体の座標系を得ることができる．これにより，離れて見るとマップ全体が俯瞰でき，近づくと個々のキャラクターの動きを見られるといった AR ゲームなどがカメラのみで実現できる．

今後は，3.2 節で述べたいくつかの問題点の解決を図るとともに，コードをより目立たなくし，自由な図案などと重ねられるように改良することを考えている．これが可能であれば，たとえば地図をカメラ越しに見るとその上に CG の情報を重ね，そこに近寄っていって見ることができるといった文献 [2] では磁気トラッカーを使って実装しているようなアプリケーションが，より低コストで構築できる．

参考文献

[1] Anoto AB. Anoto technology, http://www.anoto.com.

[2] G. W. Fitzmaurice, S. Zhai, and M. H. Chignell. Virtual reality for palmtop computers. ACM Transactions on Information Systems, Vol. 11, No. 3, pp. 197–218, July 1993.

[3] Seong-Woo Park, Yongduek Seo, and Ki-Sang Hong. Real-time camera calibration for virtual studio. In Real-Time Imaging, Vol. 6, pp. 433–448, December 2000.

[4] Jun Rekimoto and Yuji Ayatsuka. Cybercode: Designing augmented reality environments with visual tags. In DARE 2000, pp. 1–10, April 2000.

[5] ファミ通.com. プレイステーション 3 を使ったカードゲームの究極進化形がお披露目! In http://www.famitsu.com/game/event/2006/05/09/264,1147165516,52796,0,0.html. ENTERBRAIN, INC., May 2006.