Vol.2022-MPS-141 No.7
2022/12/12

ICBI を用いた倍率可変リアルタイム超解像システム

近藤真由† 松本尚†

概要：本稿では，超解像技術を用いたリアルタイム超解像システムの防犯用途や監視用途への実用化を目指して，倍率を変えながら動画を確認できる機能を開発したことを報告する．本研究において，ベースとして用いたリアルタイム超解像システムは，カメラから得た画像を ICBI アルゴリズムによって縦横 2 倍に画素補間を行い，ディスプレイに出力するものである．実用化を目指すに当たり，任意の領域をさらに拡大表示する機能を研究開発した．ICBI の補間パイプラインの段数を倍に増やせば，さらに縦横 2 倍に画素補間を行い， 動画の拡大が可能である．ただし，この方式では，パイプライン段数が増えて，ハードウェアの資源制約および消費電力増加等の問題を生じる．そこで，防犯用途や監視用途の動画に 1 秒間に 50 フレームは不要だと考え，縦横 4 倍の動画では 1 秒間に 25 フレーム表示，縦横 8 倍の動画では 1 秒間に 16.7 フレーム表示とすることにより，ICBI の補間パイプラインを時分割で使い回すことで，必要なハードウェア量をほとんど増やすことなく，倍率可変リアルタイム超解像システムを作成した．
キーワード：FPGA，ICBI，超解像

ICBI-Based Real-Time Super-Resolution System with Variable Magnification

MAYU KONDO† TAKASHI MATSUMOTO†

Abstract: This paper reports on the development of a function that allows users to verify video while changing the magnification, with the aim of practical application of a real-time super-resolution system using super-resolution technology for crime prevention and surveillance applications. In this research, the real-time super-resolution system used as a base is to output images obtained from the camera to the display after pixel interpolation by doubling the height and width using the ICBI algorithm. For practical use, we have researched and developed a function to enlarge an arbitrary area by doubling the number of interpolation pipeline stages of ICBI, which can further enlarge the video image by interpolating pixels twice as much as the vertical and horizontal. However, this method increases the number of pipeline stages, resulting in hardware resource constraints, increased power consumption, etc. Therefore, we thought that 50 frames per second was unnecessary for security and surveillance applications, and created a real-time super-resolution system with variable magnification that displays 25 frames per second for 4x video and 16.7 frames per second for 8x video, using the interpolation pipeline of ICBI in a time-division manner, with almost no increase in hardware requirements.

Keywords: FPGA, ICBI, Super-resolution

1. はじめに

近年パソコンやテレビのディスプレイなど表示デバイスの解像度が向上している一方で，小型カメラや古い映像など解像度の低いビデオデータが多く存在している．これらの入力デバイスの解像度が不足していることが問題になっており，解像度の低い入力データを解像度の高い表示デバイスで表示すると，粗くぼやけて表示されてしまう．この差を補うために，解像度の低い画像に画素を補間して解像度の高い画像を生成する超解像技術が誕生し，ニアレストネイバー法，バイリニア法，バイキュービック法といった，様々な種類の補間アルゴリズムが考案されている[1]．
ICBI(Interactive Curvature Based Interpolation)はそのような補間アルゴリズムの 1 つで，画素のエッジに着目することで，少ない計算量で人間の目に見やすく補間することを特徴としている[2]．しかし，CPU によるソフトウェア実装では，ICBI アルゴリズムを使用しても画像の大きさ次第で 1 枚の画像に超解像処理を施す際に数十秒単位の時間を要してしまい，リアルタイムでの超解像は困難であった．

我々は ICBI アルゴリズムの実行時間を短縮するため，FPGA（Field Programmable Gate Array）でハードウェア実装を行い，リアルタイムで超解像処理を可能にするシステムを開発した[3-6]．このシステムは ICBI アルゴリズムを FPGA によってハードウェア実装可能であることを実証することに重点が置かれていたため，実用化に至るには機能が不十分であった．本研究では，このシステムを改良し，社会における実使用を見据えた高機能化を図る．本研究では主に，システムの機能性の充実及び出力動画の倍率可変性の向上についての改良を行った．

本論文の構成は以下の通りである．第 2 章では画素補間アルゴリズムとして用いる ICBI および FPGA でハードウ

†奈良女子大学
Nara Women's University

ェア実装したリアルタイム超解像システムについて説明する．第 3 章ではシステムの問題点や課題点を示し，第 4 章では解決方法について述べる．第 5 章ではまとめを述べる．

2. リアルタイム超解像システム

2.1 ICBI

ICBI は，FCBI(Fast Curvature Based Interpolation)アルゴリズムと曲率平滑化の 2 段階からなる．

(1) FCBI

FCBI は，画像に対して縦横 2 倍の補間を行うアルゴリズムである．FCBI は 2 段構成のアルゴリズムで，この画像の値は偶数行偶数列に格納されている．

1 段目では，奇数行奇数列に補間する値を求める．下図では，丸型の点が元の画像の画素，三角形の点がこの段で求められる画素，四角形の点がこの図の場合に計算される画素を表している．

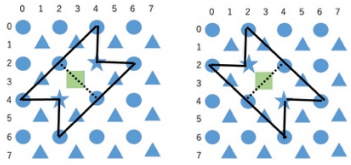

図 1　斜め方向の FCBI 補間

値を求めたい点に対し，左斜めと右斜めの 2 つの方向から計算する．まず，点線で繋がれた 2 点を比較して，エッジがあるかを判別する．この 2 点の値の間に大きな差がある場合，この間には明確なエッジがあると判断し，エッジのない斜め方向の平均値を補間する．

どちらの斜め方向にも明確なエッジが認められなかった場合，最終的にどちらの斜め方向を使って値を求めるか決めるために，エッジ判定に使用した 2 点の 2 階差分を求める．そして，その値が小さい方の平均値を四角形の点の値とする．しかし，現段階では四角形の点の値は求められていないため，この 2 階差分は計算できない．そこで，計算に用いる値の範囲を実線で結ばれた 8 点とし，丸型の点 6 個の値の合計から星型の点 2 個の値を 3 倍したものを引いて得られた値を 2 階差分の近似値とする．

2 段目では，元の値と 1 段目で求めた値を使って偶数行奇数列，奇数行偶数列の値を求める．ここでは補間時の計算対象を十字方向に設定し，残りの画素の値を求める．

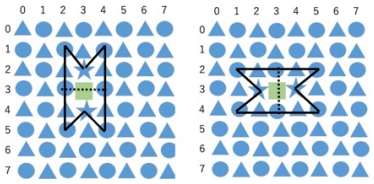

図 2　十字方向の FCBI 補間

(2) 曲率平滑化

FCBI 補間では，画素を補間することで一定精度の画素の値を設定した．ここでは，曲率平滑化処理として，補間前の入力画素以外の値の微調整を行う．

まず，奇数行奇数列の点の曲率平滑化処理には，処理対象から見て斜め方向にある点を計算に用いる．図 3 の四角形の点に曲率平滑化を行う際，斜めに 1 ピクセルずれた値の 2 階差分と比較する．実線の 2 階差分と点線の 2 階差分の差を求め，その差がなるべく小さくなるように四角形の点の値を調整する．

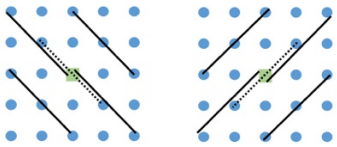

図 3　斜め方向の曲率平滑化

偶数行奇数列，奇数行偶数列の点には，計算方向を十字方向にして，同様の計算を行う．

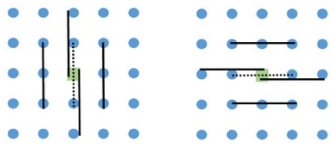

図 4　十字方向の曲率平滑化

2.2 システム構成

我々が開発したリアルタイム超解像システムは図 5 のようにディスプレイ，ZedBoard[7]，カメラモジュールで構成されている．

図 5　システム構成

ZedBoard は Xilinx 社の Zynq チップを搭載した Avnet 社の FPGA 評価ボードで，Xilinx 社の回路設計ツール Vivado[8]を用いてシステム開発を行う．Zynq は FPGA と ARM プロセッサを統合した SoC(System on a Chip)であり，ハードウェア/ソフトウェアの協調システムの開発デバイスとして用いられている[9].

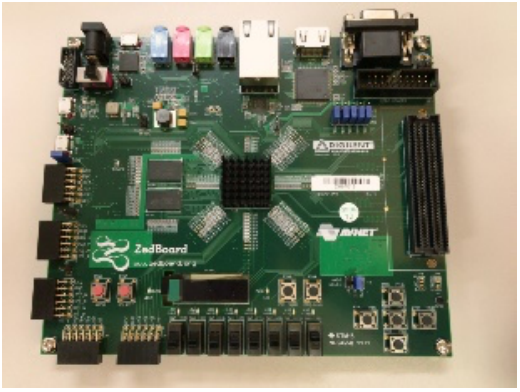

図 6　ZedBoard

カメラには OmniVision 社の OV7670[10]を使用し，カメラをボードに接続するため図 7 のアダプタを作成して ZedBoard に接続した．OV7670 は自動露出や自動ホワイトバランスなどの機能が組み込まれている安価なカメラで，設定によりカメラの動作や機能を制御することが可能である．

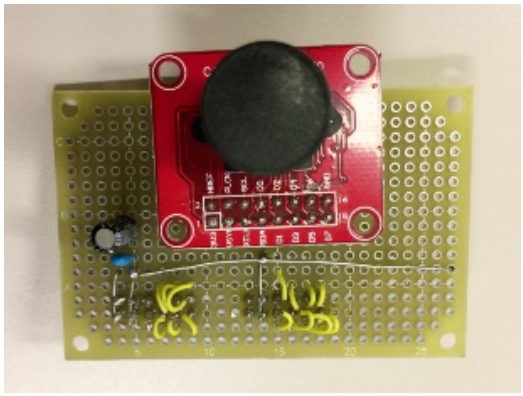

図 7　カメラモジュールとアダプタ

2.3　データの流れ

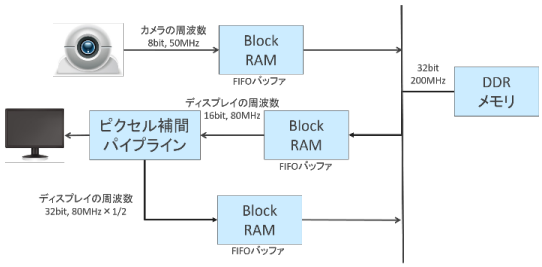

図 8　改良前のシステム概要

本システムでは，横 640 ピクセル縦 480 ピクセルの画像を 1 秒間に 60 枚入力し，それらにリアルタイムで ICBI アルゴリズムを適用してディスプレイに表示する．まず，カメラからの入力データを 2 ポート FIFO バッファである Block RAM で一時的に保存し，データ転送周波数を 50MHz から 200MHz に変更して DDR メモリに転送する．本システムでは，DDR メモリをフレームバッファとして用いることで BlockRAM の容量不足に対処している[4]．フレームバッファから転送されたデータは，別の 2 ポート FIFO バッファの Block RAM でディスプレイのドット周波数(80MHz)に変換して 1 ピクセルずつピクセル補間パイプラインに転送し，その結果をリアルタイムで出力し続けている．出力前にはデータの色形式を YUV422 から RGB へ変換する．また，入力データの読み出しや超解像処理に関するタイミングは画像表示タイミング回路によって制御している．

また，DDR メモリには，入力データとともに補間結果画像も保存する．補間結果画像を書き戻して再び処理を行い，補間結果が格納されたメモリアドレスのデータを読み出すと，その画像をさらに縦横 2 倍単位で補間し続ける．

縦横 4 倍以上に補間処理を行う場合，入力画像のすべてを補間するのではなく，横 640 ピクセル縦 480 ピクセル分抽出した範囲をピクセル補間パイプラインへ受け渡す．

3.　問題点

本システムにおける問題点は 2 つある．1 つは，補間結果を縦横 4 倍以上に拡大表示した際の拡大範囲が，画面左上の横 640 ピクセル縦 480 ピクセルに固定されている点である．改良前のシステムでは，スイッチを切り替えるたびに画面の左上が拡大されていくようになっており，ユーザが拡大したい部分を任意に指定することができない．縦横 2 倍に補間した図 9 の画像に対して補間処理を行うと，図 10 が出力される．これは，補間の開始場所を示すメモリアドレスが水平方向 0，垂直方向 0 の場所に固定されていたため，画面上の任意の場所を示すアドレスにアクセスする方法がなかったことが原因である．

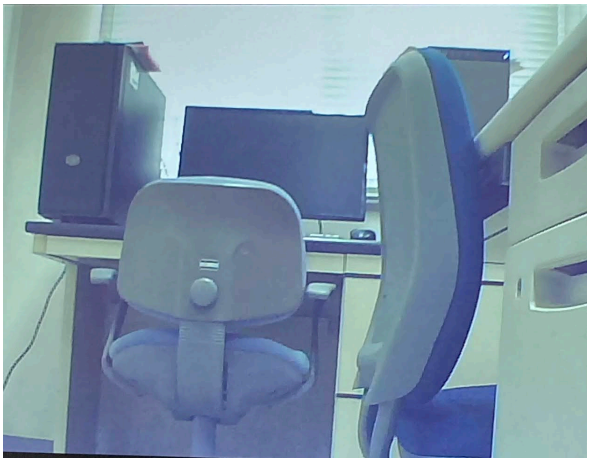

図 9　縦横 2 倍に補間した際の出力

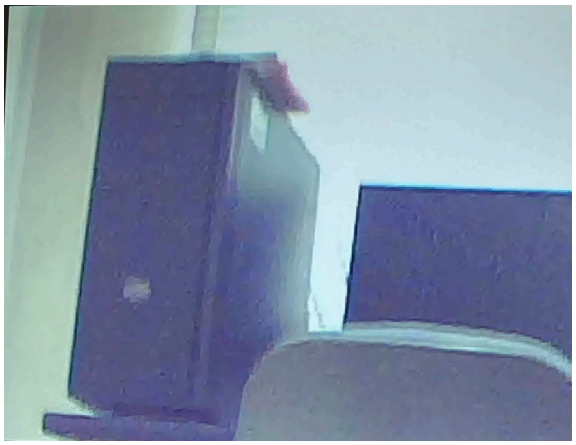

図 10　縦横 4 倍に補間した際の出力

2 点目は，縦横 4 倍以上の補間を行った際，出力が静止画になってしまう点である．これは，フレームバッファの補間結果を格納するアドレスから画像を読み出す際，クリックが押された時点において直近で格納されていた 1 枚のみが補間対象となっており，それ以降の入力画像が更新されていないためである．

本研究では，ハードウェアの改良及びポインティングデバイスの追加によってそれぞれの問題を解決する．

4. 改良方法

4.1 任意の範囲の表示

改良前のシステムは，補間処理も含めてすべてハードウェアで実装したものであるが，今回の改良では，これらに影響を及ぼさない範囲で補助的に Linux および Linux 上のソフトウェア処理を組み込む．システムに搭載する Linux は PetaLinux[11]で作成する．OS 領域とフレームバッファの衝突を防ぐため，OS が使用するメモリ量を 384MB に制限する．

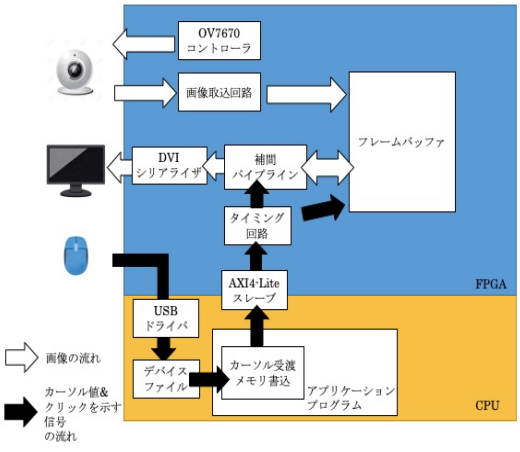

図 11　改良後のシステムの概要

(1) ソフトウェア処理

ソフトウェア部分では，Linux のデバイスファイルからマウスの挙動(左クリック・右クリック・水平方向のカーソル移動相対距離・垂直方向のカーソル移動相対距離)を取得し，それぞれに対応したレジスタへ受け渡す．

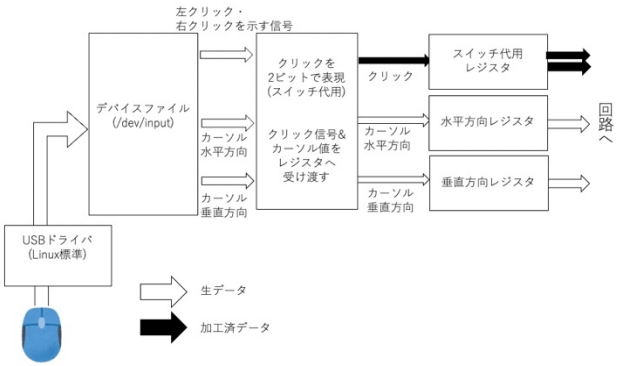

図 12　ソフトウェア処理の概要

(2) ハードウェア処理

ハードウェア部分では，マウスから得た値をタイミング回路に受け渡し，その値に対応する処理を回路に追加し，カーソルとしてシステムに反映する．

まず，フレームバッファを開始するアドレスを OS 領域と衝突しない位置に変更する．

次に，AXI(Advanced eXtensible Interface)のスレーブ IP を回路に追加し，マウスの挙動を記録したレジスタとタイミング回路に接続する．

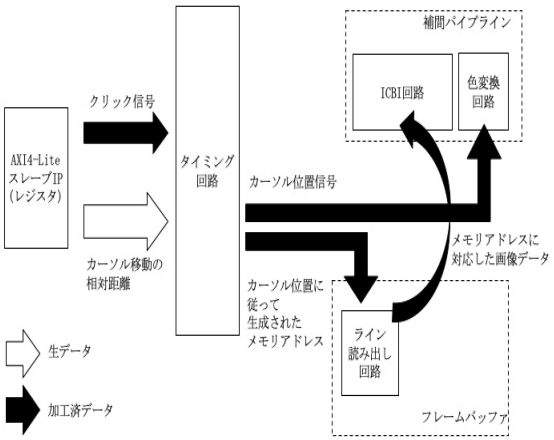

図 13　ハードウェア処理の概要

タイミング回路内でカーソルの初期値を設定し，マウスから得た相対移動距離を加算することでカーソルの絶対座標を求める．画面に表示する画素数を数えるカウンタとカーソルの絶対座標が一致した時，YUV422 から RGB へ色形

式を変換する回路へカーソル位置を示す信号を送ることで，カーソルとして示したい部分だけカメラからの入力の影響を受けないようにする．この結果，補間処理後の映像にカーソルとして四角形を表示できた．

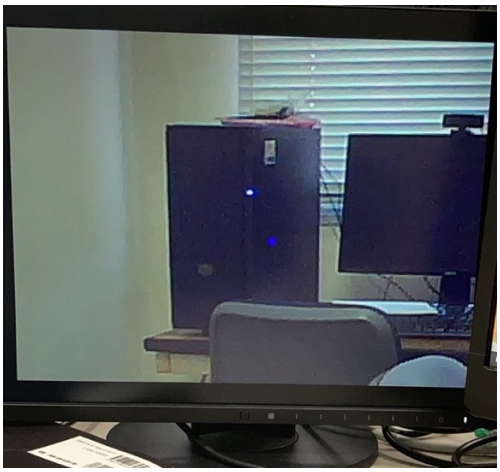

図 14　カーソルを表示させた結果

本システムでは，カーソルアイコンを中心にして横 640 ピクセル縦 480 ピクセル分を抽出し，再度補間して出力する．カーソル位置に合わせて画像を抽出する際，水平方向と垂直方向に分けて入力開始位置を調整する．水平方向の調整は，画像を読み出すメモリアドレスに水平方向座標を示す数値を入れ込み，画像の途中から出力できるようにする．垂直方向は，読み出しを開始するラインをカーソルの垂直方向座標によって指定する．

その結果，左クリックを 1 回認識すると，補間結果画像を格納するフレームバッファにアクセスし，カーソル位置から水平方向に 320 ピクセル左，垂直方向に 240 ピクセル上から画像の読み出しを開始する．システムの補間対象範囲が画面左上から画面全体に広がった．

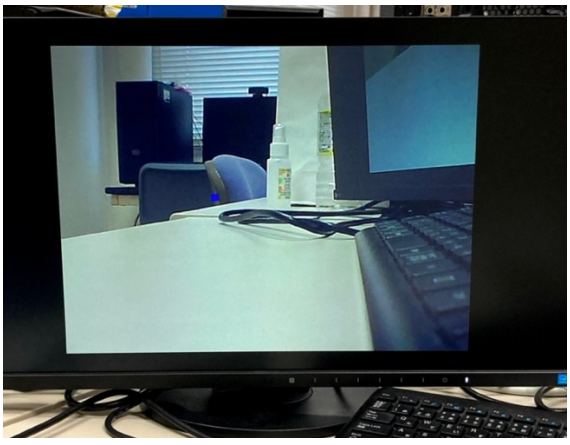

図 15　改良後の縦横 2 倍拡大画像

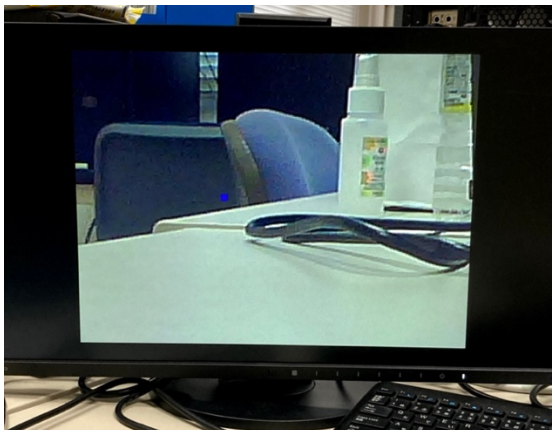

図 16　改良後の縦横 4 倍拡大画像．
図 11 におけるカーソル位置を中心にして拡大表示している．

出力画面上でマウスを操作する．拡大したい箇所に対して左クリックか右クリックをすると，そのカーソル値を中心に縦横 2 倍単位の画素補間が行われる．

4.2 出力倍率の可変化

本研究では，入力画像を縦横 2 倍，縦横 4 倍，縦横 8 倍に拡大した動画を出力できるように改良を行った．

1 秒間に 50 枚の画像表示を維持したまま 4 倍，8 倍…の補間処理を行うと，FPGA のハードウェア資源が不足する可能性が高い．これを解決する方法として，パイプラインを追加したり，本システムを複数台カスケードしたりする手法が考えられるが，前者は ZedBoard のリソース的観点，後者はコスト的観点からして現実的ではない．そこで，1 秒あたりの画像表示枚数を減らす代わりに複数回ピクセル補間パイプラインに通すことで，本システムの表示倍率を向上させる．拡大倍率を高めるほど補間パイプラインを経由する回数が多くなるので，1 秒間の表示枚数を反比例させていく必要がある．例えば，縦横 4 倍の画像を出力したい場合は 1 秒あたり 25 枚の画像，縦横 8 倍の画像の出力には 1 秒あたり 16.7 枚の画像の入力を必要とする．

また，改良前のシステムではディスプレイに表示する画像はピクセル補間パイプラインから直接出力していたが，安定して高倍率の動画を出力するために，今回の改良ではフレームバッファに書き戻した画像を直接出力している．

各拡大倍率に応じて，出力までに経由するフレームバッファの数を操作する．縦横 4 倍の動画を出力する場合，まずカメラ画像を縦横 2 倍に補間し，補間後画像をフレームバッファに格納する．その後，その 2 倍拡大画像を入力として再度補間処理を行い，結果をディスプレイに出力する．この処理を繰り返す．ディスプレイには，出力の対象となる倍率画像を格納したフレームバッファのみを出力する．拡大画像を入力する範囲は，前述のカーソル値によるものとする．縦横 8 倍の動画を出力する場合は，4 倍の画像から 8 倍画像を出力するように分岐する．

拡大倍率は，マウスで拡大範囲と同時に指定する．図 8 でカーソル値と同時にレジスタに書き込んだ左クリック信号・右クリック信号によって縦横 4 倍・縦横 8 倍を指定する．

図 17　動画出力を間に合わせるための
フレームバッファ制御

5. おわりに

本研究によって，補間結果画像の出力範囲を任意に指定できるようになり，本システムの機能性が向上した．また，縦横の倍率を 4 倍，8 倍に高めても動画として出力できるようになり，ICBI アルゴリズムを適用させた画像をリアルタイムで動画表示する超解像システムを開発できた．その結果，人間の目に見やすい状態で任意の部分を拡大可能になった．この技術によって，監視カメラなどの低コストな遠隔監視機器の実用化に近づいた．

近年では，FPGA を用いてリアルタイムで画像認識を行う研究が進んでいる．本システムに画像認識アルゴリズムを実装し，FPGA を用いた研究開発を進めることを今後の目標としたい．

謝辞 本研究を行うにあたり，指導教官である松本尚教授には暖かいご指導をいただき，大変お世話になりました．また，主に FPGA の回路設計において助言をくださいました，株式会社フューチャーテクノロジーの相原治様にも深くお礼申し上げます．松本研究室の皆様には，日頃の研究室生活において大変お世話になりました．ありがとうございました．

参考文献

[1] 奥富正敏 他 “ディジタル画像処理” CG-ARTS 協会，2004.

[2] Andrea Giachetti Nicola Asuni (2011)“Real-Time Artifact-Free Image Upscaling” IEEE Transactions on Image Processing 20(10):2760 – 2768

[3] 松本 尚 山本 有紗 城 和貴(2016)“実時間超解像回路の試作—ICBI アルゴリズムの FPGA 実装—” 研究報告数理モデル化と問題解決（MPS）2016-MPS-109-11 p. 1-4

[4] Takashi Matsumoto, Arisa Yamamoto, Kazuki Joe (2016)“Real-Time Super Resolution: FPGA Implementation for the ICBI Algorithm” In Proceedings of 2016 International Conference on Parallel and Distributed Processing Techniques and Applications(PDPTA 2016),p. 415-422

[5] 松本 尚 眞田 麻代 安浪 涼花 城 和貴(2018) “FPGA 実装されたリアルタイム超解像回路の改良” 研究報告数理モデル化と問題解決（MPS）2018-MPS-119(13)

[6] 眞田 麻代 松本 尚(2020)”FPGA 実装された超解像回路の検証に向けて” 研究報告数理モデル化と問題解決（MPS) 2020-MPS-127(13)

[7] Xilinx,“ZedBoard” https://japan.xilinx.com/products/boards-and-kits/1-8dyf-11.html (最終閲覧日 2022/11/01)

[8] Xilinx, “Vivado Design Suite” https://japan.xilinx.com/products/design-tools/vivado/vivado-webpack.html (最終閲覧日 2022/11/01)

[9] M.Ali Altuncu,Taner Guven,Yasar Becerikli, and Suhap Sahin (2015) “Real-Time System Implementation for Image Processing with Hardware/Software Co-design on the Xilinx Zynq Platform”International Journal of Information and Electronics Engineering,Vol 5,No.6.November 2015:473-477

[10] EPFL,“Extension module with ov7670 CMOS camera” https://wiki.epfl.ch/prsoc/ov7670 (最終閲覧日 2022/11/01)

[11] Xilinx,”PetaLinux ツール” https://japan.xilinx.com/products/design-tools/embedded-software/petalinux-sdk.html (最終閲覧日 2022/11/01)