

事前転置機構による VVC 向け高スループット IDCT アーキテクチャ

黒澤 流風^{1,a)} 丹羽 直也^{1,3,b)} 岩崎 裕江^{1,2,c)}

概要：動画コンテンツの増加により、通信に占める映像トラフィックの割合は非常に高いものとなっている。最新の国際標準映像符号化規格である H.266/VVC は、前規格比で映像帯域をおよそ半分に削減するが、一方で符号化/復号により多くの計算時間を要する。離散コサイン逆変換 (IDCT) は映像復号における主要な処理の一つであり、大量の行列演算を実行するため専用ハードウェアによる効率的な処理が必要となる。H.266/VVC では、新たな要素として MTS や長方形ブロック分割が追加され、それらにより符号化効率が向上している一方、効率的な処理が求められている。長方形変換ブロックを変換前に動的に転置することで、2 次元 IDCT におけるパイプラインストールを削減させ、スループットを向上する手法を提案する。本手法は、 4×4 から 16×16 までの正方形ブロックと 1:2 比率の長方形ブロックに IDCT を実行した場合において、事前転置を行わない比較モデルに対して 8.21% から 9.82% 総サイクル数を削減した。

A High-throughput IDCT Architecture for VVC Using Pre-transposition Mechanism

1. はじめに

映像コンテンツのデータ量は莫大であり、帯域の削減が急務となっている。2019 年には、全世界のトラフィックに占める動画ストリーミングの割合は 6 割を超えた [1]。こうした中 2020 年に、国際標準映像符号化規格 Versatile Video Coding (H.266/VVC) の策定が行われた。VVC は前規格の High Efficiency Video Coding (H.265/HEVC) と比べ、同等の映像品質を保ちつつおよそ 30% から 40% 帯域幅を削減する一方で、映像符号化/復号における計算量が増大している。

映像復号における主要な処理の一つに、逆変換 (IDCT) がある。これは映像信号を周波数領域から空間領域へ戻す処理であり、輝度や色差信号の行列と IDCT 係数行列の積を計算することによって行われる。VVC では 1 点から 64 点までの多様なサイズや比率の長方形ブロックが存在するため、IDCT を高スループットで行うためには、多様なサ

イズのブロックを連続して効率よく処理するアーキテクチャが必要となる。

本研究では、VVC 向け IDCT アーキテクチャの設計及び実装を目的とした。その際スループットを高めるため、長方形ブロックを変換前に動的に転置する事前転置アルゴリズムを設計し、提案アーキテクチャに組み込んだ。レジスタ転送レベル (RTL) 実装に対して IDCT をシミュレーション実行し、提案手法による総サイクル数の変化を定量化した。

2. VVC の逆変換 (IDCT) 処理

変換 (Transform) は、VVC の主要な処理の一つである。順変換は符号化時に実行されるのに対し、逆変換は主に復号時に実行される。映像ブロックに対する順変換は 2 次元変換となり、これはブロックの縦方向と横方向それぞれに 1 次元変換を適用することで実行される。1 次元変換は、ブロックの横方向もしくは縦方向のベクトルに、変換係数行列を掛けて行う。逆変換も同様に計算され、順変換で用いた係数行列を転置した行列を用いて逆変換を行う。順変換の計算を式 1 に示す。

¹ 東京農工大学

² 東北大学

³ 京都大学

a) ruka@st.go.tuat.ac.jp

b) naoya@go.tuat.ac.jp

c) hiroe@go.tuat.ac.jp

$$Y = (X \times C^T)^T \times C^T \quad (1)$$

逆変換の計算を式 2 に示す。

$$\hat{X} = (\hat{Y} \times C)^T \times C \quad (2)$$

変換に関連する VVC の新機能に、Multiple Transform Selection (MTS) がある。これは、従来変換に用いられてきた DCT-II に加え、DCT-VIII、DST-VII の 3 つの変換基底を動的に切り替えるものである。MTS は、DCT-II の代わりに DCT-VIII や DST-VII 基底を動的に利用することにより、より低周波に変換後の係数を集中させ、後続の量子化処理をより効果的にすることを目的としている。前規格の HEVC でも、 4×4 変換ブロックに対して DCT-II の代わりに DST-VII を適用する機能は存在したが、VVC では適用範囲が拡大しており、VVC では最大 32 点までの変換ブロックに対して、DCT-II、DCT-VIII、DST-VII のいずれかを選択して適用する。MTS の各変換基底を表 1 に示す。 N は変換を適用する方向のブロックサイズ、 i, j は行列の行および列を表す。

表 1: N 点入力に対する DCT-II / DCT-VIII / DST-VII の変換基底 [2]

変換種類	基底関数 $C_i(j)$, $i, j = 0, 1, \dots, N-1$
DCT-II	$C_i(j) = \omega_0 \cdot \sqrt{\frac{2}{N}} \cdot \cos\left(\frac{\pi \cdot i \cdot (2j+1)}{2N}\right)$ $\text{where } \omega_0 = \begin{cases} \sqrt{\frac{2}{N}} & \text{if } i = 0 \\ 1 & \text{if } i \neq 0 \end{cases}$
DCT-VIII	$C_i(j) = \sqrt{\frac{4}{2N+1}} \cdot \cos\left(\frac{\pi \cdot (2i+1) \cdot (2j+1)}{4N+2}\right)$
DST-VII	$C_i(j) = \sqrt{\frac{4}{2N+1}} \cdot \sin\left(\frac{\pi \cdot (2i+1) \cdot (j+1)}{2N+1}\right)$

3. 関連研究

VVC 向け変換ハードウェア実装については、いくつかの提案がなされている。Garcia らは、 4×4 から 64×64 までの変換サイズをサポートする 1 次元逆変換アーキテクチャを提案した [3]。Hao らは、変換行列を商行列と余り行列に分解し、計算複雑性を低減する手法を提案した [4]。彼らは 32 点/サイクルの処理を実現したと報告している。Garrido らの実装は、1 次元アーキテクチャによるパイプライン処理を行っている [5]。彼らの実装は最大 64×64 までの変換サイズと、DCT-II、DCT-VIII、DST-VII の 3 つの変換タイプをサポートし、4K で 10FPS を達成している。Imen らは、DCT-II のみの変換をサポートする実装を提案した [6]。Kammoun らは、DCT-VIII および DST-VII の変換行列を、DCT-II の変換行列を利用した近似で表現する手法を提案した [7]。彼らの実装は 2K で 386FPS、4K

表 2: 各ブロックサイズ毎の、1 次元変換に要する乗算数

変換を適用する 辺の長さ	他方向の辺の長さ		
	4	8	16
4	64	128	256
8	256	512	1024
16	1024	2048	4096

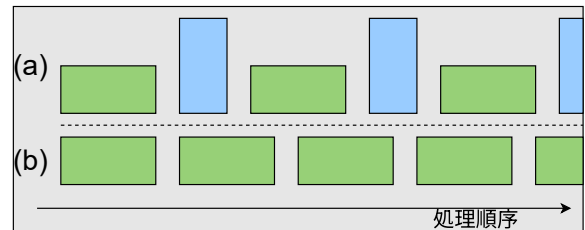


図 1: 長方形ブロックの想定入力ケース

で 96FPS のスループットを達成しつつ、ハードウェアリソースを削減している。Mert らの実装は VVC 策定前の初期のハードウェア実装であり、 4×4 と 8×8 の変換サイズのみをサポートしている [8]。Oztas らは ARM システム上で VVC デコーダを実装し、その中で 8 点及び 16 点の DCT-II 逆変換処理を FPGA にオフロードする処理を行っている [9]。Farhat らの実装は、 4×4 から 64×64 までの変換サイズをサポートし、4K で 48FPS のスループットを達成したと報告している [10]。

先行手法では Garrido らの実装 [5] に見られるように、2 つの 1 次元変換器を並列で動作させ、複数ブロックを同時に変換するパイプライン処理を行い、変換スループットを向上させている。パイプラインを効率的に動作させる上で課題となるのが、VVC で新たに追加された長方形ブロックの存在である。2 次元変換は、ブロックの縦方向および横方向にそれぞれ 1 次元変換を適用することで実現されるが、変換対象が長方形ブロックの場合は、2 つの 1 次元変換器で実行する乗算回数は異なる。実際の映像を処理する際は、横長ブロックと縦長ブロックの個数に偏りがあるため、既存の 2 次元変換アーキテクチャでは、片方の 1 次元変換ユニットに処理が集中する問題がある。

4. 提案手法

4.1 提案アルゴリズム

本研究では、条件付きの行列転置により、2 個の 1 次元逆変換器の計算サイクル数を均一化し、パイプラインストールを削減するアーキテクチャを提案する。

表 2 に示すように、長方形ブロックを処理する際、縦横それぞれの方向に必要な乗算回数は異なる。そのため、2 つの 1 次元逆変換器でパイプライン処理を行う場合、短辺側を処理するユニットは、長辺側ユニットの処理が完了す

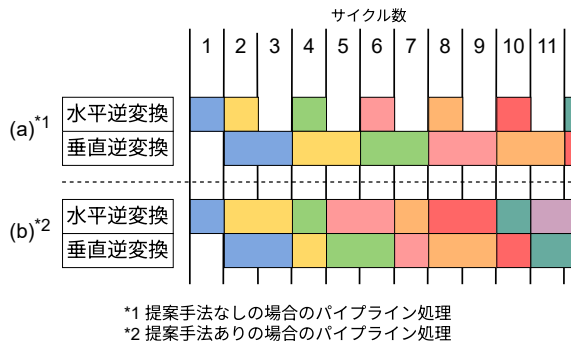


図 2: 長辺方向が同じ長方形ブロックを連続して処理する場合

Algorithm 1 . 事前転置の必要性を決定

```

1: if height > width and prev_height > prev_width or
   height < width and prev_height < prev_width then
2:   tr_flag = true
3:   prev_height = width
4:   prev_width = height
5: else
6:   tr_flag = false
7:   prev_height = height
8:   prev_width = width
9: end if

```

るまで待つ必要がある。図 1 に、想定する入力ケースを示す。(a) では十分なバッファサイズがあれば計算量の偏りはバッファにより解消され、ストールは発生しない。(b) では片方の 1 次元逆変換ユニットに乗算処理が集中し、ストールが発生する。提案手法では、本来 (b) の入力であったものを (a) のような形式になるように、長方形ブロックの一部を転置する。この転置処理により、図 2 に示すように、一定個のブロックを逆変換処理するのに必要な総サイクル数が減少することが期待される。

アルゴリズム 1 に、事前転置の必要性を判定する流れを示す。height および width は、現在処理しているブロックの高さおよび幅、prev_height および prev_width は、直前に処理したブロックの高さおよび幅を表す。tr_flag は判定結果を表し、判定は次のように行われる。現在処理中のブロックの長辺方向と、直前に処理したブロックの長辺方向が同じとき、tr_flag に true がセットされる。2 つの長辺方向が同じでないときは、tr_flag に false がセットされる。tr_flag のセットが完了した後、prev_height および prev_width の値を更新する。

4.2 提案アーキテクチャ

設計した 2 次元逆変換アーキテクチャを図 3 に示す。本

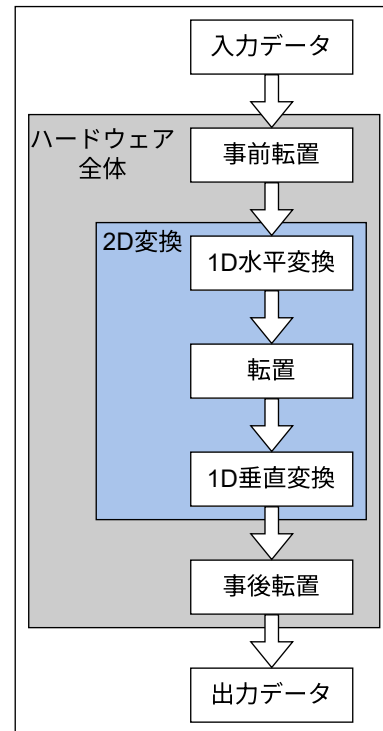


図 3: 提案アーキテクチャ

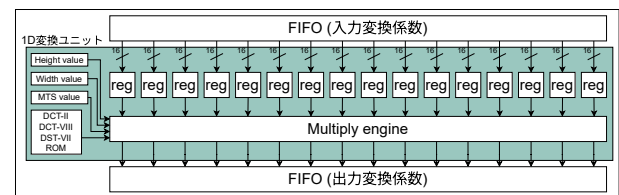


図 4: 1 次元変換ユニット

アーキテクチャは、2 つの 1 次元逆変換器と、それらを繋ぐ行列転置ユニットからなる。それに加え本アーキテクチャは、最初の 1 次元逆変換を実行する前に行列を転置する事前転置ユニットと、2 回目の 1 次元変換後に行列を転置する事後転置ユニットを備える。事後転置ユニットは、最終的な行列の転置方向が事前転置の有無に関わらず既存手法と等しくすることを目的としている。

4.2.1 1 次元変換ユニットアーキテクチャ

1 次元変換アーキテクチャを図 4 に示す。全体の処理の流れは次の通りである。1 次元変換ユニットははじめに、ブロックの幅・高さ・MTS タイプを取得する。次に、係数

表 3: 各ブロックサイズ毎の、1 次元変換に要する乗算処理サイクル数

変換を適用する 辺の長さ	他方向の辺の長さ		
	4	8	16
4	1	2	4
8	4	8	16
16	16	32	64

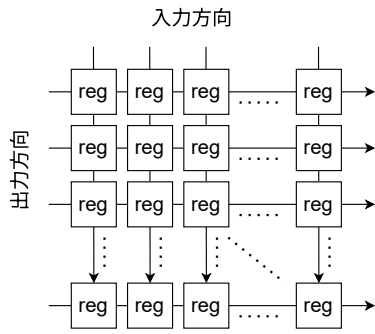


図 5: 転置ユニット

の 16 点を FIFO から読み出し、変換処理を行う。最後に、変換後の係数を FIFO へ出力する。本ユニットは 64 回/サイクルのレートで乗算を実行する。すなわち、 4×4 サイズでは 1 サイクル、 8×8 サイズでは 8 サイクル、 16×16 サイクルサイズでは 64 サイクルを乗算に要する。表 3 に、各変換サイズに必要な乗算サイクル数を示す。表の縦方向が、1 次元変換を適用する方向のブロックサイズ、表の横方向が、もう片方向のブロックサイズである。

4.2.2 転置ユニットアーキテクチャ

転置ユニットは、入力ブロックを転置する機能を有する。図 5 に示すように、本転置ユニットはレジスタによって構成される。1 次元変換ユニットと同様、転置ユニットはまず、FIFO からブロック 1 個分の係数を読み出す。そして行列が転置される向きに係数を出力することにより、入力されたブロックの転置が完了する。

5. 評価

本研究では、第 4 章に記述したアーキテクチャを C++ で実装し、高位合成により RTL 実装を行った。高位合成ソフトウェアには、Vitis HLS を用いた。

第 4 の各ユニットは、Vitis の提供する FIFO インタフェースにより構築された。各 FIFO サイズを次に示す。事前転置ユニットおよび 1 次元変換ユニットの前には、深さ 128 の FIFO が備わっているこの FIFO は、 4×4 ブロックを 128 個、もしくは 8×8 のブロックを 32 個、 16×16 のブロックを最大 8 個格納するのに十分な容量を有する。その他のユニット間には深さ 16 の FIFO が備わっており、 16×16 のブロック 1 個を格納するのに十分な容量を有する。

本研究では提案手法の有効性を評価するため、事前転置ユニットおよび事後転置ユニットを持たないアーキテクチャを比較用に用意し、回路規模とスループットを評価した。次に、実映像の逆変換シミュレーションを行うため、入力として用いるデータを VTM-20.2 を用いて用意した。映像は All Intra 構成で 1 フレームだけエンコードした後、VTM 上でデコードする際の逆変換前ブロックを入力として用いた。スループットを算出するには入力の全てのブ

ロックに対して逆変換を行うのが望ましいが、本アーキテクチャは一部のサイズのみサポートしているため、困難である。そのため本研究では次の式を用いて、FPS を算出した。

$$FPS = \frac{pixels \times freq}{cycles \times 1920 \times 1080} \quad (3)$$

式 3 において、FPS は推定される変換フレームレート、 $pixels$ はシミュレーションで変換した総ピクセル数、 $freq$ は実装で達成した周波数 (提案実装では 206.8MHz、比較実装では 213.04Mhz)、 $cycles$ は対応する全てのブロックを変換するのに要したサイクル数である。

実験環境を表 4 に示す。

表 4: 実験環境

Item	Specification
Development env.	Vitis 2024.1
Target hardware	Virtex UltraScale+ XCVU13P-3

合成結果を表 5 に示す。提案アーキテクチャは比較アーキテクチャと比べ、FF および LUT の使用量がどちらも 6 割ほど増加した。この増加分は事前転置ユニットおよび事後転置ユニットによるものである。レジスタの代わりに転置メモリを用いた回路に変更することで、リソース使用量の増加を軽減できる可能性がある。

表 5: RTL 合成結果の比較

Method	Resource Utilization			
	BRAM	DSP	FF	LUT
Prop.	16	1,016	32,927	42,518
Conv.	16	1,016	20,558	26,332

表 6: 全ての変換ブロックに含まれる総ピクセル数、処理に要した総サイクル数、推定 FPS

Image	Pixels	Cycles		FPS	
		Prop.	Conv.	Prop.	Conv.
s201	1,413k	1,437k	1,593k	98.08	91.12
s210	1,129k	1,137k	1,241k	99.02	93.46
s214	869k	1,028k	1,120k	84.31	79.72
s218	367k	432k	477k	84.56	78.87

表 6 に、シミュレーションで処理した総ピクセル数、要した総サイクル数、算出した推定 FPS を示す。結果より、提案アーキテクチャは 8.21% から 9.82% サイクル数を削減

表 7: VVC 向け変換実装の比較

Item	[3]	[5]	[8]	[9]	[10]	Proposed
Platform	-	Cyclone V	-	Arty Z7-20	-	Virtex UltraScale+
Technology	40nm ASIC	28nm FPGA	40nm FPGA	28nm FPGA	28nm ASIC	16nm FPGA
Standard	VVC	VVC	FVC	VVC	VVC	VVC
Dimension	1D	2D	2D	2D	2D	2D
Block Size	4, 8, 16, 32	4, 8, 16, 32, 64	4, 8	8, 16	4, 8, 16, 32, 64	4, 8, 16
Frequency (MHz)	500	204	143	100	600	206.8
FF / Gate	668,690	-	4571	26,600	96,849	32,927
ALM	-	5,179	-	-	-	-
DSP	-	-	-	-	-	1,016
LUT	-	-	17173	28,012	-	42,518
Throughput	8K @60FPS	4K @10FPS	8K @35FPS	FHD @42.96FPS	4K @30FPS	FHD @84.31FPS
Transform Type	DCT-II, VIII DST-VII	DCT-II, VIII DST-VII	DCT-II, V, VIII DST-I, VII	DCT-II	DCT-II, VIII DST-VII	DCT-II, VIII DST-VII



図 6: s201 映像のブロック分割

するとともに、FHD 換算で 84.31fps から 99.02fps を達成した。本アーキテクチャは VVC で要求されるブロックサイズの一部のみサポートしているものの、VVC 映像のリアルタイムデコードに十分なスループットを達成する可能性を秘めている。

次に、提案実装と既存手法との比較を行う。表 7 に各実装の比較数値を示す。提案手法は一部の変換サイズのみ対応しているものの、MTS の全ての基底に対応している。32 点以上の変換サイズや、1:2 比率以外の長方形ブロックの変換対応については今後の課題とする。

最後に、結果より示された、サイクル数の削減について考察する。図 6 に、s201 映像の分割図を、表 8 に、s201 映像のサイズ毎のブロック個数を示す。長方形ブロックの個数に注目すると、 4×8 ブロックは 8×4 ブロックの約 4.9 倍、 8×16 ブロックは 16×8 ブロックの約 11 倍と、横長ブロックが縦長ブロックより多いことが分かる。ブロックの長辺方向に偏りがある映像は本提案手法が想定するケースに合致しており、総サイクル数を削減する結果につながったと考えられる。

6. まとめ

本研究では、VVC 向け 2 次元逆変換ハードウェアアーキテクチャを設計した。提案アーキテクチャは MTS で定

表 8: サイズ毎のブロック個数 (s201)

		幅		
		4	8	16
高さ	4	8198	10815	-
	8	2216	3300	3441
	16	-	313	677

められた 3 種類の変換基底をサポートし、 4×4 から 16×16 までの正方形ブロックと 1:2 比率までの長方形ブロックの変換に対応している。また提案アーキテクチャは、変換前に長方形ブロックを必要に応じて転置する事前転置機構を持ち、2 つの 1 次元変換器の計算量を均一化する機能を備えている。提案アーキテクチャは Vitis HLS を用いて RTL 実装された。提案アーキテクチャは比較アーキテクチャに対して 8.21% から 9.82% 総サイクル数を削減すると同時に、FHD 映像に含まれる対応サイズのブロックを 84.31FPS のレートで変換した。これにより、リアルタイムデコードに十分なスループットで変換処理を実行できる可能性を示した。今後はアーキテクチャの対応するブロックサイズをさらに増やし、提案手法の有効性を評価する予定である。

謝辞 本研究の一部は JSPS 科研費 JP23K24827、JP22H03571、JP24K20751、JST、BOOST、JPMJBY24F8 及び一般財団法人テレコム先端技術研究支援センター (SCAT)、公益財団法人放送文化基金、公益財団法人電気普及財団、公益財団法人立石科学技術振興財団の支援によるものです。

参考文献

- [1] Sandvine: The Global Internet Phenomena Report, https://www.sandvine.com/hubfs/Sandvine_Redesign_2019/Downloads/Internet%20Phenomena/Internet%20Phenomena%20Report%20Q32019%2020190910.pdf (2019).
- [2] Chen, J., Ye, Y. and Kim, S. H.: Algorithm description for Versatile Video Coding and Test Model 11 (VTM 11) (2020).
- [3] Garcia, B., Silveira, B., Diniz, C., Palomino, D. and Correa, G.: Multi-Size Inverse DCT-II Hardware Design for the VVC Decoder, *2023 IEEE 14th Latin America Symposium on Circuits and Systems (LASCAS)*, pp. 1–4 (online), DOI: 10.1109/LASCAS56464.2023.10108348 (2023).
- [4] Hao, Z., Sun, H., Xiang, G., Zhang, P., Zeng, X. and Fan, Y.: A Reconfigurable Multiple Transform Selection Architecture for VVC, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 31, No. 5, pp. 658–669 (online), DOI: 10.1109/TVLSI.2023.3245291 (2023).
- [5] Garrido, M. J., Pescador, F., Chavarrias, M., Lobo, P. J., Sanz, C. and Paz, P.: An FPGA-Based Architecture for the Versatile Video Coding Multiple Transform Selection Core, *IEEE Access*, Vol. 8, pp. 81887–81903 (online), DOI: 10.1109/ACCESS.2020.2991299 (2020).
- [6] Imen, W., Fatma, B., Amna, M. and Masmoudi, N.: DCT -II Transform Hardware-Based Acceleration for VVC Standard, *2021 IEEE International Conference on Design & Test of Integrated Micro & Nano-Systems (DTS)*, pp. 1–5 (online), DOI: 10.1109/DTS52014.2021.9498196 (2021).
- [7] Kammoun, A., Hamidouche, W., Philippe, P., Déforges, O., Belghith, F., Masmoudi, N. and Nezan, J.-F.: Forward-Inverse 2D Hardware Implementation of Approximate Transform Core for the VVC Standard, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 30, No. 11, pp. 4340–4354 (online), DOI: 10.1109/TCSVT.2019.2954749 (2020).
- [8] Mert, A. C., Kalali, E. and Hamzaoglu, I.: High performance 2D transform hardware for future video coding, *IEEE Transactions on Consumer Electronics*, Vol. 63, No. 2, pp. 117–125 (online), DOI: 10.1109/TCE.2017.014862 (2017).
- [9] Oztas, A. E., Ozteke, E., Demir, M. and Akgul, T.: Implementation of a Hardware Accelerated VVC Decoder on ARM and FPGA, *2023 30th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 1–4 (online), DOI: 10.1109/ICECS58634.2023.10382778 (2023).
- [10] Farhat, I., Hamidouche, W., Grill, A., Menard, D. and Déforges, O.: Lightweight Hardware Implementation of VVC Transform Block for ASIC Decoder, *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1663–1667 (online), DOI: 10.1109/ICASSP40776.2020.9054281 (2020).