九州大学大学院
システム情報科学研究科報告
第1巻　第1号　平成8年9月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.1, No.1, September 1996

文音声認識を目的としたCYK解析アルゴリズムの効率化

迫江博昭* ・片山喜規* ・内田誠一**

An Efficient CYK-based Algorithm for Continuous Speech Recognition

Hiroaki SAKOE , Yoshinori KATAYAMA and Seiichi UCHIDA

(Received June 24, 1996)

Abstract: An efficient implementation of CYK-based continuous speech recognition is investigated. First, the word level and the sentence level processes in Ney's algorithm were reorganized so that they proceed in synchronization with input frame. Then, beam search prunings were incorporated into the two processing levels. A new acceleration technique, beam data driven parsing, was successfully introduced. Considerable improvements in computational and memory efficiency were established through a sentence speech recognition experiment.

Keywords: Continuous speech recognition, CFG parsing, CYK algorithm, Dynamic programming, Beam search, Beam driven parsing

1. は じ め に

認識率の向上や処理の効率化を目的として文法制御を導入した連続音声認識法が研究されている．初期の研究では，文法として正規文法が用いられていたが[1)4)5)]，タスクの複雑化に伴い，より高度な文脈自由文法を用いた方法が研究されている[2)3)]．

文脈自由文法制御を導入した連続音声認識法の性能は，CYK法，Earley法，LR法など用いる構文解析アルゴリズムに大きく依存する．このうちCYK法は，他の方法に比べ単語予測機能が低く，計算量が多いという理由で，実用性が低いと言われている．しかしCYK法には，

- 原理が簡単である
- 書換規則がそのままの形でアルゴリズム中で取り扱われる
- 解析の途中結果が理解しやすい素直な形で与えられる

などの利点があり，小規模な音声認識システムや，音響レベルに重点をおいた実験システムに適した解析アルゴリズムであると考えられる．

CYK法を用いた連続音声認識の代表的なものに，Neyによる研究[2)]が挙げられる．Neyのアルゴリズムは単語レベル処理と文レベル処理に二分される．単語レベル処理では，現入力フレームを終端とするすべての部分区間にすべての単語の存在を仮定し，それぞれの区間の単語距離を始端フリーDPマッチングによって求める．文レベル処理では，単語距離を評価基準としたCYK法に基づく統語処理が，入力フレームに同期して行なわれている．CYK法が一種のDPであることを考えると，Neyのアルゴリズムは，正規文法制御2段DPマッチング[1)]における文レベル処理を自然な形で文脈自由文法制御に拡張したものと言える．

実時間動作を目標とした場合，Neyのアルゴリズムには以下の問題点がある．

- 文レベル処理の計算量がCYKアルゴリズムの$O(I^3)$ (Iは入力長)をそのまま受け継いでいる
- 単語レベル処理についても，認識対象語い数や入力長の増加に比例してDP起動回数が増加する
- ビームサーチなどの計算量低減手法について導入が検討されておらず，さらに単語レベル処理はそれらを効果的に導入できる形になっていない

本報告は，以上の問題点に対する方策を検討したものである．まず，Neyのアルゴリズムを出発点とし，ビームサーチなどの効率化手法を導入しやすくするために，すべての処理を完全にフレーム同期化する．次に実際に効率化手法を導入し，各レベルでの計算量および記憶量ともに大幅な低減を実現する．小規模なタスクを対象として評価した結果，処理時間は近似的に$O(I)$となることが確認された．

2. CYK解析に基づく連続音声認識アルゴリズムとその問題点

2.1 Ney のアルゴリズム

本研究の出発点となったNeyのアルゴリズムについて述べる．

- 文脈自由文法の書換規則(Chomsky標準形)

統語規則　　$U \xrightarrow{p} VW$　　$p = 1, 2, \ldots P$

平成8年6月24日受付

* 知能システム学専攻
** 知能システム学専攻博士後期課程

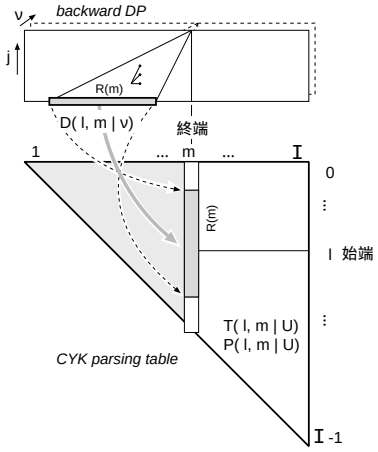

Fig.1　Ney のアルゴリズムにおける解析テーブルT, Pと単語レベル処理

単語辞書　　$U \xrightarrow{q} \nu$　　$q = 1, 2, \ldots Q$

英大文字は非終端記号，νは単語に対応する終端記号$(\nu = 1, 2, \ldots N)$を表す．またSは文記号とする．

- 入力音声およびその部分区間

$$\boldsymbol{A} = \boldsymbol{a}_1 \boldsymbol{a}_2 \cdots \boldsymbol{a}_i \cdots \boldsymbol{a}_I$$
$$A(l,\ m) = \boldsymbol{a}_{l+1} \boldsymbol{a}_{l+2} \cdots \boldsymbol{a}_m$$

- 単語標準パターン(J^νは単にJと略記)

$$\boldsymbol{B}^\nu = \boldsymbol{b}_1^\nu \boldsymbol{b}_2^\nu \cdots \boldsymbol{b}_j^\nu \cdots \boldsymbol{b}_J^\nu$$

- 単語距離 $D(l, m|\nu)$
  $A(l,\ m)$と$\boldsymbol{B}^\nu$間のDP距離．
- 累積距離 $T(l, m|U)$
  $A(l,\ m)$がUであるとしたときの，Uをルートノードとする部分解析木に含まれる単語列の単語距離の総和の最小値．
- バックポインタ $P(l,\ m \mid U)$
  $A(l,\ m)$がUであるとしたときの，Uをルートノードとする上記部分解析木を構成するための情報．具体的には，書換規則番号p(もしくはq)および，書換規則$U \to VW$におけるVとWの境界xの組$\langle p,\ x\rangle$

図-1に解析テーブルT, P，および単語距離Dが計算される様子を示す．

Neyのアルゴリズムでは，入力フレームmに同期しながら以下の単語レベル処理と文レベル処理が進行する．

単語レベル処理　　$\boldsymbol{B}^\nu$と，$A(l, m)$間の距離$D(l, m|\nu)$を，mを終端として固定した後向きDPマッチングにより求める．整合窓幅をwとすれば，1回のDPにより次の条件を満たす始端lに対して$D(l, m|\nu)$が求まる．

$$m - J - w/2 \le l \le m - J + w/2$$

以後この区間を$R(m)$と表記する．

後向きDPマッチングの漸化式は次式とする．

$$g(i, j) = d(\nu; i, j)\ +\ \min \begin{bmatrix} g(i+1, j) \\ g(i+1, j+1) \\ g(i+1, j+2) \end{bmatrix} \quad (1)$$

ここで

$$d(\nu; i, j) = \|\boldsymbol{a}_i - \boldsymbol{b}_j^\nu\|. \quad (2)$$

各m, νに関するDP終了毎に単語距離が次のように求まる．

$$D(l, m|\nu) = g(l+1, 1) \quad \text{for} \quad l \in R(m)$$

(1)式の漸化式計算そのものを，以後フレームレベル処理と呼ぶ．

文レベル処理**1**　　$U \to \nu$型の書換規則に関して，$T(l, m|U)$を決定する．次式をすべての$l \in R(m)$，Uについて計算する．

$$T(l, m|U) = \min_{U \xrightarrow{q} \nu} [D(l, m|\nu)] \quad (3)$$

同時に上式を最小化するqをバックポインタに保存する．

$$P(l, m|U) = \langle q,\ 0\rangle. \quad (4)$$

ここで$x = 0$は書換規則が単語辞書であることを意味する．

文レベル処理**2**　　$U \to VW$型の書換規則に関して，$T(l, m|U)$を決定する．次式をすべての$l < m - 1$，Uについて計算する．

$$T(l, m|U) = \min_{\substack{U \xrightarrow{p} VW \\ l\ <\ x\ <\ m}} [T(l, x|V) + T(x, m|W)] \quad (5)$$

同時に上式を最小化するpおよびxをバックポインタに保存する．

$$P(l, m|U) = \langle p,\ x\rangle. \quad (6)$$

以上の処理を$m = I$まで終了した後，$P(0, I|S)$をルートとする$P(l, m|U)$上でのバックトラックで，認識結果を得る．以上のアルゴリズムを図-2に示す．

2.2　Ney のアルゴリズムの問題点

Neyのアルゴリズムの文レベルにおける計算量および記憶量を表-1に示す．文レベル処理2に関して，m, l, xの3重のループより生じる$O(I^3)$の計算量は実時間システム実現の大きな障害となる．記憶量も$O(I^2)$となり，実験室レベルのシステムではタスクが制限されてしまう．

単語レベル処理についても，$O(INJ^2)$の漸化式計算が必要であり，各漸化式がベクトル演算(2)を内包することを考えると，見かけ以上の大量な計算が必要であると言える．

```
[Initialize]  T(l, m|U) = ∞ for m, l < m, U
for m = 1, ..., I
    [Word level processing]
    for ν = 1, ..., N
        for i = m − 1, ..., min R(m)
            [Frame level processing]
            for j = J, ..., 1
                Calculate g(i, j) using eq.(1)
        D(l, m|ν) ← g(l + 1, 1) for l ∈ R(m)
    [Sentence level processing 1]
    for l ∈ R(m)
        for q = 1, ..., Q
            ⟨U, ν⟩ ←− (U →^q ν)
            if D(l, m|ν) < T(l, m|U)
                T(l, m|U) ← D(l, m|ν)
                P(l, m|U) ← ⟨q, 0⟩
    [Sentence level processing 2]
    for l = m − 2, ..., 0
        for p = 1, ..., P
            ⟨U, V, W⟩ ←− (U →^p VW)
            for x = l + 1, ..., m − 1
                T = T(l, x|V) + T(x, m|W)
                if T < T(l, m|U)
                    T(l, m|U) ← T
                    P(l, m|U) ← ⟨p, x⟩
[Backtrack]  Obtain word sequence using P(l, m|U)
```

Fig.2 Ney のアルゴリズム

2.3 ビームサーチの導入

ビームサーチとは，並列的に進行する横型の最適解探索において, 各レベルで有望と判断されたものだけをそれ以降の処理に残す方法である．ビームサーチの導入によって解の最適性は保証されなくなるが，正規文法制御を用いた連続音声認識では，認識率をほとんど落さずに計算量および記憶量の大幅な低減がなされている[4]．

CYK法に基づく連続音声認識法へのビームサーチの導入を考える．具体的には3つの処理レベルそれぞれにビームサーチが考えられる．

フレームレベルビームサーチ　大きな値を持つ$g(i,j)$の除外(単語内DPパスの枝刈)

単語レベルビームサーチ　$T(l,m-1|U)$の値を元に，第mフレームを始端とする単語を制限する(単語予測)

文レベルビームサーチ　大きな値を持つ$T(l,m|U)$の除外(文レベルDPパスの枝刈)

文献[2]ではビームサーチによる効率化に関しては検討されていない．そこで，前述したビームサーチをNeyのアルゴリズムに直接導入するとして，その適合性を検討する．

まず，フレームレベルビームサーチについて，Neyのアルゴリズムでは各単語DP単位での枝刈しかできず，端点位置や単語が異なる$g(i,j)$を相互比較しながら枝刈をする方法[4]に比べて効率が低い．

また，文献[5]で指摘されているように，単語レベルのDPが後向きであることに起因して，ぼけた単語予測しかできない．

文レベルビームサーチは，ループ内での漸化式(5)を条件付きで実行するという形でも実現できるが，この簡単な方法では結局$O(I^3)$ 回のループおよびその条件判断によるオーバーヘッドのために大きな効果は期待できない．

なお, 伊藤他のアルゴリズム[3]には文レベルビームサーチが導入されているが，そのワークエリア構成およびループ制御は本質的にNeyの方法と同じで，ビームサーチの効果を極限まで追求したことにはなっていない．

3. 完全フレーム同期化とビーム駆動統語処理による高効率アルゴリズム

3.1 概要

本節では，ビームサーチが有効に動作するようにNeyのアルゴリズムを変形し, その後すべてのレベルに対して実際にビームサーチを導入したアルゴリズムを説明する．改良点を次にまとめる．

- 文レベルビームサーチの導入に伴う解析テーブルの縮小
- 統語処理漸化式(5)のビーム内データ駆動による計算回数低減，およびループ数低減によるオーバーヘッドの削減
- 前向きDPマッチングによる単語予測能力の高い単語レベルビームサーチの実現
- 単語レベル処理の完全なフレーム同期化と，フレームレベルビームサーチの導入による単語レベル処理の効率化

3.2 アルゴリズム

まず，文レベルビームサーチ導入に伴い，$T(l,m|U)$，$P(l,m|U)$の代わりに以下のワークエリアを用いる．

- 累積距離$TW(l|U)$
  第mフレームに関する$T(l,m|U)$の値を保持する．
- バックポインタ$PW(l|U)$
  第mフレームにおける$P(l,m|U)$の値を保持する．
- ビーム内累積距離データ $PB(n) = \langle TV, p, x, l, m\rangle$
  第$m-1$フレームまでのビーム内に残った累積距離$TV = T(l,m|U)$，およびそれに付随するバックトラック情報$\langle p,x\rangle = P(l,m|U)$とその時刻 $\langle l,m\rangle$のセットを保持する．$PB(n)$内のデータ項目数を$NW(m)$でカウントする$(n = 1,2,\ldots,NW(m))$．

次にフレームレベルビームサーチ導入に伴い，$D(l,m|\nu)$の代わりに以下のワークエリアを用いる．

- ビーム内単語距離データ$DB(k) = \langle D,\nu,l\rangle$
  第mフレームにおける$D(l,m|\nu)$のうちビーム内に残ったものを，l，νとともに保持する$(k = 1,2,\ldots K(m))$．

他に単語レベルビームサーチで必要となる次の情報を用意する．

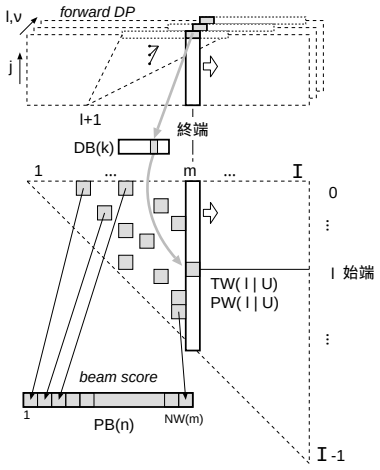

Fig.3　本手法におけるワークエリアTW, PW, PBと単語レベル処理

- 単語予測用集合$Z(\nu)$
  単語νの前に接続可能な非終端記号の集合．タスク設定時に書換規則を元に作成しておく．

図-3にテーブルTW, PW，ビーム内データPB, DB，および単語距離の計算の様子を示す．

図-4に示すように，フレームレベル処理を含めた全レベルの処理が入力フレームmに同期ながら進行する．

単語レベル処理　(1)式の漸化式計算を前向き処理とすることで，単語距離計算を完全に入力フレームmに同期させる．漸化式は次式で表される．

$$g(\nu, l; m, j) = d(\nu; m, j) + \min \begin{bmatrix} g(\nu, l; m-1, j) \\ g(\nu, l; m-1, j-1) \\ g(\nu, l; m-1, j-2) \end{bmatrix} \quad (7)$$

(7)式をフレームレベル処理と呼ぶ．第mフレームでは，$\boldsymbol{a}_m$の入力を受けて，$\min R(m) \leq l < m-1$，すべてのν，すべてのjに対して$g(\nu, l; m, j)$が(7)式によって計算される．

単語レベルビームサーチ　νの前に接続可能な非終端記号のスコアがビーム内に存在にすることを条件として，第mフレームより単語νについてのDPを開始する．すなわち，$U \in Z(\nu)$なるUに関して，一つでも$TW(l|U) \neq \infty$を満たす$TW(l|U)$があった場合に次式を計算する．

$$g(\nu, m-1; m, 1) = d(\nu; m, 1) \quad (8)$$

フレームレベルビームサーチ　第mフレームで計算された部分距離$g(\nu, l; m, j)$は，$(m-l)$で除して正規化すればそれらすべてのオーダーが揃う．ゆえに$g(\nu, l; m, j)$ に関してν, lに依らない横断的な相互比較が可能であり，効果的なフレームレベルビームサーチが実現できる．

```
[Initialize]  NW(1) = 0,  TW(0|U) = 0 for U
for m = 1, ..., I
  [Word level processing]
  for ν = 1, ..., N
    for l = min R(m), ..., m − 2
      [Frame level processing]
      for j = 1, ..., J
        Calculate g(ν, l; m, j) using eq.(7)
    [Word level beam search]
    if any TW(l|U) ≠ ∞ for U ∈ Z(ν), l < m
      Calculate g(ν, m − 1; m, 1)
  [Frame level beam search]
    Prune g(ν, l; m, j)
  for g(ν, l; m, J) in the beam
    DB(k) ⟵ ⟨g(ν, l; m, J), ν, l⟩
  [Initialize]  TW(l|U) ← ∞  for l < m, U
  [Sentence level processing 1]
  for k = 1, ..., K(m)
    ⟨D_k, ν_k, l_k⟩ ⟵ DB(k)
    for q which satisfies (U -q-> ν_k)
      ⟨U, φ⟩ ⟵ (U -q-> ν_k)
      if D_k < TW(l_k|U)
        TW(l_k|U) ← D_k
        PW(l_k|U) ← ⟨q, 0⟩
  [Sentence level processing 2]
  for n = NW(m), ..., 1
    ⟨TV_n, p_n, φ, l_n, x_n⟩ ⟵ PB(n)
    ⟨U', V', W'⟩ ⟵ (U -p_n-> VW)
    V_n ⟵ U'
    for p which satisfies (U -p-> V_n W)
      ⟨U, φ, W⟩ ⟵ (U -p-> V_n W)
      T = TV_n + TW(x_n|W)
      if T < TW(l_n|U)
        TW(l_n|U) ← T
        PW(l_n|U) ← ⟨p, x_n⟩
  [Sentence level beam search]
    Prune TW(l|U)
  for TW(l|U) in the beam
    ⟨p, x⟩ ⟵ PW(l|U)
    Append ⟨TW(l|U), p, x, l, m⟩
      to the tail of PB(n)
  Update NW(m)
[Backtrack]  Obtain word sequence from PB(n)
```

Fig.4　本手法のアルゴリズム

ビーム内に残った$g(\nu, l; m, j)$のうち，$j = J$のものがあれば，文レベル処理に受け渡すために次の代入を行なう

$$DB(k) = \langle g(\nu, l; m, J), \nu, l \rangle \quad (9)$$

文レベル処理1　$DB(k)$に含まれる単語距離データに対して，(3)式と同様の処理を行なう．すなわち，$DB(k) \longrightarrow \langle D_k, \nu_k, l_k \rangle$とすれば，

$$TW(l_k|U) = \min_{U \xrightarrow{q} \nu_k} [D_k] \quad (10)$$

をすべてのkを対象として計算する．同時に上式を最小化するqをバックポインタに保存する．

$$PW(l_k|U) = \langle q,\ 0 \rangle. \quad (11)$$

文レベル処理2(ビーム駆動統語処理) 文レベル処理2においても，前フレームまでの文レベルビームサーチの結果残った$TW(l|U)$，すなわち$PB(n)$内のデータによって(5)式を駆動することで，不要な条件判定やループのオーバーヘッドが回避できる．具体的には，$PB(n) \longrightarrow \langle TV_n, p_n, \phi, l_n, x_n \rangle$として($\phi$は該当項目の無視を意味する)，

$$TW(l_n|U) = \min_{U \xrightarrow{p} V_n W} [TV_n + TW(x_n|W)] \quad (12)$$

をすべてのnを対象として計算する．ここで，V_nは$(U \xrightarrow{p_n} VW) \longrightarrow \langle U', V', W' \rangle$とした時の$U'$に等しい．(12)式を最小化する$p$および$x$をバックポインタに保存する．

$$PW(l_n|U) = \langle p,\ x_n \rangle. \quad (13)$$

(12)式は，ワークエリア$TW(l|U)$上でのオーバーライトの形式となっているが，$PB(n)$を$n = NW(m), \ldots, 1$の順に参照することにより，矛盾なく最適値の計算ができる[6]．

文レベルビームサーチ $TW(l|U)$のオーダーが$(m-l)$であることから，フレームレベルビームサーチと同様に$TW(l|U)/(m-l)$と正規化し，相互比較もしくは閾値と比較することでビーム内に残すべき$TW(l|U)$を決定する．

最後にビーム内に残った$TW(l|U)$とその属性のセット$\langle TW(l|U), p, x, l, m \rangle$を$PB(n)$の最後尾に追加し，同時に$NW(m)$の更新を行なう．

3.3 効率化の評価

表-2に本手法の文レベル処理における記憶量および計算量を示す．

まず文レベル処理での記憶量低減効果について評価する．Neyのアルゴリズムでの記憶量が$3/2 \cdot |U| \cdot I^2$であるのに対し，本手法では$5NW(I) + 2I \cdot |U|$となっている．認識率を保証するためには$NW(I)$が$O(I^2)$となることを覚悟する必要があるが，現実的には$NW(I)$をI^2 より相当小さくできると期待できる．

次に文レベル処理におけるループ制御のためのオーバーヘッドおよび漸化式計算量の低減効果について検討する．文レベル処理1および文レベル処理2をビーム内のデータによって起動することで，ビーム外のデータに関する無意味な探索ループは無くなる．特に文レベル処理2ではその効果が大きく，まずl，xの2重ループがnの単ループとなり，さらにpのループも全てのpを参照する必要が無くなっている．前者については，事実上$NW(I) \ll I^2$であることから，大幅なオーバーヘッドの低減が実現される．後者のpのループ縮小に関して，その効果を引き出すためには，右辺左端にVを持つような統語規則を高速に引き出せるような工夫が必要である．これは文レベル処理1のqのループに関しても同様である．文レベル処理2の漸化式計算そのものもビーム内のデータによって起動されるので，文レベルビームサーチの効果が確実に漸化式計算回数低減に結び付いている．

単語レベルビームサーチに関しては，単語レベルDPを前向き処理としたことでぼけのない単語予測が実現され，接続不可能な単語のDP計算を確実に除去できる．

さらにフレーム同期化により単語レベルDPに関して横断的な枝刈が実現するので，単語単位の局所的な枝刈に比べビーム径を絞り込むことが可能となる．またこのビームサーチの導入により，文レベル処理の場合と同様に，$g(\nu, l; m, j)$に要する記憶量およびループのオーバーヘッドと漸化式計算量の大幅な低減が可能である[4][7]．

Table-1 Neyのアルゴリズムの文レベル計算量 / 記憶量

	計算量	記憶量
文レベル処理1	$3QIJ/2$ $3QJ/2$	$3\|U\|m^2/2$
文レベル処理2	$Pm^2/2$ (約)$PI^3/6$	$3\|U\|I^2/2$

上段:第 m フレーム / 下段:全体 / $|U|$:非終端記号数

Table-2 本手法の文レベル計算量 / 記憶量

	計算量	記憶量
文レベル処理1	$K(m)Q$ $\sum K(m)Q$	$5NW(m) + 2\|U\|m$
文レベル処理2	$NW(m)P$ $\sum NW(m)P$	$5NW(I) + 2\|U\|I$

上段:第mフレーム / 下段:全体 / $|U|$:非終端記号数

4. 評価実験

4.1 実験条件

効率化の評価のために小規模な実験を行なった．男性話者1名による離散発声日本語単語を標準パターンとし，同一話者による3 ~ 9単語からなる文章100文を入力パターンとした．文章は，統語規則数$P = 9$，単語辞書数$Q = N = 43$，非終端記号数$|U| = 9$の文脈自由文法に従う．分析はフレーム周期10msecで16次元メルスペクトラムとした．最大フレーム数は入力パターンで384，標準パターンで74であった． 使用計算機はSun Ultra1 model 140 (SPECint92:215,SPECfp92:303)である．

フレームレベルおよび文レベルのビームサーチにおける枝刈のための閾値は，余裕定数θ_1，θ_2を用いて，各m毎に次式で決定した．

$$\min [g(\nu, l; m, j)/(m-l)] + \theta_1$$
$$\min [TW(l|U)/(m-l)] + \theta_2$$

なお，今回の実験では単語レベルビームサーチは組み込んでいない．

Table-3 タスク 100 文に対する認識結果

余裕定数 θ_1	余裕定数 θ_2	認識率 (%)	$NW(I)$ (平均:%)	平均認識時間 (秒) 単語距離	平均認識時間 (秒) 統語処理
	0.05	68	1.01		0.14
	0.07	92	1.21		0.15
0.27	0.10	98	1.46	1.16	0.17
	0.15	100	1.83		0.19
	0.20	100	2.22		0.22
∞	∞	100	100	71.77	4.72

記憶量($NW(I)$)はNeyのアルゴリズムの場合を100%として計算

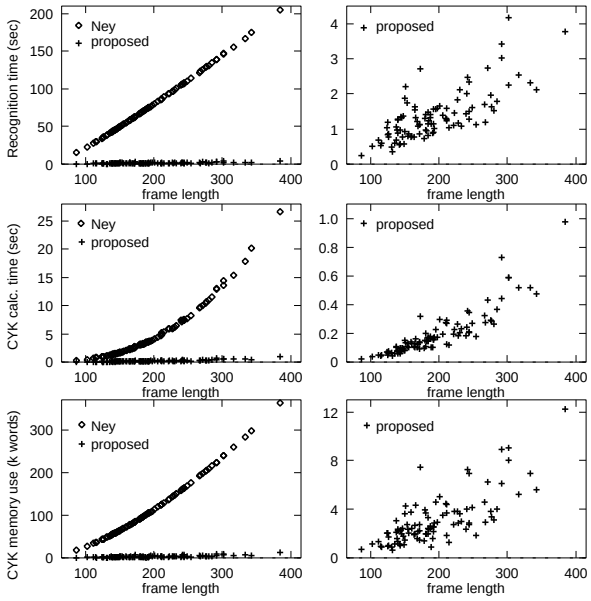

Fig.5 入力フレーム長に対する認識時間 (上), 文レベル処理時間 (中), 文レベル処理記憶量 (下) の変化 ((左) Ney との比較, (右) 本手法のみ)

4.2 実験結果と考察

最初に余裕定数に対する，認識時間，記憶量および認識率の変化を測定した．フレームレベル余裕定数θ_1は事前に求めた最適値で固定し，文レベル余裕定数θ_2のみを変化させた．結果を表-3に示す．また$\theta_1 = \theta_2 = \infty$とした場合をNeyのアルゴリズムの近似として併せて示した．

まずフレームレベルビームサーチの導入によって，単語距離計算に要する時間を1 ～ 2桁程度低減できていることがわかる．

文レベル処理についても，誤認識を誘発することなく記憶量をNeyのアルゴリズムの場合の2%程度まで小さくできていることがわかる($\theta_2 = 0.15$の時)．また3.3節で述べた通り，この記憶量の低減は直接文レベル計算量の低減につながっており，文レベル処理に要する認識時間も平均的に1 ～ 2桁程度低減されている．

次に入力フレーム長に対する認識時間，文レベル記憶量の変化を測定した．結果を図-5に示す．余裕定数は$\theta_1 = 0.27$，$\theta_2 = 0.15$ とした．$\theta_1 = \theta_2 = \infty$とした場合も併せて示した．

このグラフより，本手法では入力フレーム長に対しほぼ線形に近い認識時間特性が得られていることがわかる．またタスクとしてより長い文章を選んだ場合，文レベル処理に要する認識時間短縮効果は，さらに大きくなると考えられる．

5. ま　と　め

CYK法をベースとする連続音声認識アルゴリズムの計算量および記憶量の低減を図り，その評価を行なった．主な改良点は，

- 文レベル処理へのビームサーチ導入および統語処理DPのビーム駆動による不要なループの除去
- 単語レベル処理のフレーム同期化およびビームサーチの導入

である．評価の結果，文レベル処理に関して大幅な探索空間の絞り込みが可能となり，解析テーブルのサイズも2桁程度圧縮できることが明らかになった．また，実行時間も全体として1 ～ 2桁程度短縮されており，入力長に対する認識時間の依存傾向は，ほぼ線形となった．

参　考　文　献

1) 迫江博昭: “単語を単位とした連続音声認識の一手法”, 信学技報 **PRL80**-19(1980-7).
2) H.Ney: “Dynamic programming speech recognition using a context-free grammar”, Proc. ICASSP97, pp. 3.2.1-4(1987).
3) 伊藤彰則; 牧野正三; 城戸健一: “機能語予測CYK 法による日本語文音声の統語処理”, 信学論, **J74-DII**, No. 9, pp. 1147–1155(1991-9).
4) 迫江博昭; 藤井浩美; 吉田和永; 亘理誠夫: “フレーム同期化, ビームサーチ, ベクトル量子化の統合によるDP マッチングの高速化”, 信学論, **J71-D**, No. 9, pp. 1650–1659(1988-9).
5) H. Sakoe: “A generalized two-level DP-matching algorithm for continuous speech recognition”, *Trans. IECE Japan*, **E65**, No. 11, pp. 649–656(1982-11).
6) 片山喜規; 迫江博昭: “ビームサーチ CYK 法による連続音声認識”, 音講論, 3-8-13, pp. 133–134(1994-10).
7) H. Ney; D. Mergel; A. Noll; A. Paeseler: “A data-driven organization of the dynamic programming beam search for continuous speech recognition”, Proc. ICASSP87, pp. 20.10.1–4(1987).